



Joanna Goode
University of Oregon

Gail Chapman
Computer Science Equity Alliance

Acknowledgements

Contributing Writers

George Benainous, Hollywood High School, Los Angeles, California
 Robb Cutler, Tutor Crossing, Inc., Santa Clara, California
 Judy Hromcik, Arlington High School, Arlington, Texas
 Michelle Hutton, The Girl's School, Mountain View, California
 John Landa, South East High School, South Gate, California

Curriculum Design Team Members

Joanna Goode, University of Oregon
 Gail Chapman, Computer Science Equity Alliance
 Jane Margolis, UCLA
 David Bernier, UCLA
 Todd Ullah, Los Angeles Unified School District
 Diane Watkins, Los Angeles Unified School District
 Chris Stephenson, Computer Science Teachers Association

Sponsors & Supporters

This curriculum was created under the auspices of the Broadening the Participation in Computing National Science Foundation grant, "Into the Loop: An University K-12 Alliance to Increase and Enhance the Computer Science Learning Opportunities for African-American, Latino/a, and Female Students in the Second Largest School District in the Country". Principal Investigator: Jane Margolis (UCLA); Co-Principal Investigators Joanna Goode (University of Oregon), Todd Ullah (LAUSD), Deborah Estrin (UCLA).



CONTENTS

Course Overview	5
Goals	5
Standards	5
Hardware.....	5
Software	5
Prerequisites.....	5
The Instructional Philosophy of <i>Exploring Computer Science</i>	6
Introduction to Curricular Approach	6
Concrete Instructional Strategies	10
Assessment.....	11
Overview of the Instructional Materials	12
Unifying Themes and Practices	13
Scope and Sequence	14
Overview Chart	17
Topic Descriptions and Objectives	21
Unit 1: Human Computer Interaction (~4 weeks)	21
Unit 2: Problem Solving (5 weeks)	22
Unit 3: Web Design (6 weeks)	23
Unit 4: Introduction to Programming (7 weeks)	24
Unit 5: Robotics (8 weeks).....	25
Unit 6: Computing Applications (6 weeks)	26
Unit 1: Human Computer Interaction.....	28
Introduction.....	29
Daily Overview Chart	30
Daily Lesson Plans.....	31
Final Project.....	67

Unit 2: Problem Solving.....	69
Introduction.....	70
Daily Overview Chart	71
Daily Lesson Plans.....	72
Final Project.....	95
Unit 3: Web Design	97
Introduction.....	98
Daily Overview Chart	99
Daily Lesson Plans.....	100
Final Project.....	122
Flash Animation Supplement	125
Javascript Supplement	130
Unit 4: Introduction to Programming.....	132
Introduction.....	133
Daily Overview Chart	134
Daily Lesson Plans.....	135
Final Project.....	187
Unit 5: Robotics.....	191
Introduction.....	192
Daily Overview Chart	193
Daily Lesson Plans.....	194
Final Project.....	230
Unit 6: Computing Applications	238
Introduction.....	239
Daily Overview Chart	240
Daily Lesson Plans.....	241
Final Project.....	285

Course Overview

Goals

Exploring Computer Science is designed to introduce students to the breadth of the field of computer science. The goal of *Exploring Computer Science* is to develop in students the computational thinking practices of algorithm development, problem solving and programming within the context of problems that are relevant to the lives of today's students. Students will also be introduced to topics such as interface design, limits of computers and societal and ethical issues of software engineering.

This curriculum has been developed for a culturally, linguistically, and socially diverse group of students in Los Angeles Unified School District. District-wide, student ethnicities include .3% American Indian, 3.7% Asian, .4% Pacific Islander, 2.3% Filipino, 73.0% Latino, 10.9% African American, 8.8% White, and .6% Other or multiple responses. Over 38% of students are English-language learners, with most English language learners students speaking Spanish as their primary language. Furthermore, 74% of students qualify for free or reduced lunches.

Standards

The standards used for the *Exploring Computer Science* curriculum are based on the topics and goals outlined in *A Model Curriculum for K-12 Computer Science* developed by the ACM K-12 task force curriculum committee. Most of the objectives in the course align with the Level III course, *Computer Science as Analysis and Design*, while some objectives are necessarily aligned with the Level II course, *Computer Science in the Modern World*, in order to provide appropriate background knowledge for the more advanced topics.

Hardware

An ideal laboratory environment for this course would include one computer for each student in the class. These computers can be either Macintosh or PC depending on availability. A networked system would make installation of software easier for the teacher.

Software

Each computer in the classroom should have a web browser installed that allows students to perform searches and make use of a variety of websites and internet tools. Teachers will need to download and install the Scratch programming language available at <http://www.scratch.mit.edu> and the Python programming language available at <http://www.python.org>.

Prerequisites

This course will be considered a college preparatory elective for California students, geared towards 11th and 12th graders, and will require Algebra as a course prerequisite. Thus, the course should provide a rigorous, but accessible, introduction to computer science. No previous computer science course is required to take this course.

The Instructional Philosophy of *Exploring Computer Science*

Introduction to Curricular Approach

Exploring Computer Science teaches the creative, collaborative, interdisciplinary, and problem-solving nature of computing with instructional materials which feature an inquiry-based approach to learning and teaching. As part of this curriculum, students will delve into real-world computing problems that are culturally-relevant and address social and ethical issues while delivering foundational computer science knowledge to students. Students will engage in several in-depth projects to demonstrate the real-world applications of computing.

This curriculum builds off of learning theories that view learning as a social and cultural process that does not only occur in a vacuum at school; that is, students bring to school bodies of knowledge from their lives, culture, and communities. Building from students' prior knowledge, the collection of problem solving skills, everyday "algorithmic thinking", and social and ethical knowledge of computer-related problems will result in a more student-centered curriculum. Each unit connects students' informal knowledge, technology skills, and beliefs about computing to the theoretical and foundational tenets of computer science. Students will become members of a "computing community of practice" in the classroom where they will be introduced to the behavior, language, and skills of computer scientists. Furthermore, the interdisciplinary nature of computing allows for the incorporation of subject-matter topics across disciplines into the computing curriculum.

The Nine Principles of Learning from the Institute for Learning provide the theoretical foundation of research-based instructional practices that provide the foundation for the Secondary Redesign Comprehensive Plan. These nine principles underscore the beliefs of the Los Angeles Unified School District; they are integrated throughout and explain the pedagogy used in the course.

1. Organizing for Effort

An effort-based school replaces the assumption that aptitude determines what and how much students learn with the assumption that sustained and directed effort can yield high achievement for all students. Everything is organized to evoke and support this effort, to send the message that effort is expected and that tough problems yield to sustained work. High minimum standards are set and assessments are geared to the standards. All students are taught a rigorous curriculum aligned to the standards, along with as much time and expert instruction as they need to meet or exceed expectations. This principle is one of the guiding beliefs common in every school in the Los Angeles Unified School District.

2. Clear Expectations

If we expect all students to achieve at high levels, then we need to define explicitly what we expect students to learn. These expectations need to be communicated clearly in ways that get them "into the heads" of school professionals, parents, school communities and, above all, students themselves. Descriptive criteria and models of work that meets standards should be publicly displayed, and students should refer to these displays to help them analyze and discuss their work. With visible accomplishment targets to aim toward at each stage of learning, students can participate in evaluating their own work and setting goals for their own efforts.

3. Fair and Credible Evaluations

If we expect students to put forth sustained effort over time, we need to use assessments that students find fair, and that parents, community, and employers find credible. Fair evaluations are ones that students can prepare for: therefore, tests, exams and classroom assessments as well as the curriculum must be aligned to the standards. Fair assessment also means grading against absolute standards rather than on a curve, so students clearly see the results of their learning efforts. Assessments that meet these criteria provide parents, colleges, and employers with credible evaluations of what individual students know and can do.

4. Recognition of Accomplishment

If we expect students to put forth and sustain high levels of effort, we need to motivate them by regularly recognizing their accomplishments. Clear recognition of authentic accomplishment is the hallmark of an effort-based school. This recognition can take the form of celebrations of work that meets standards or intermediate progress benchmarks en route to the standards. Progress points should be articulated so that, regardless of entering performance level, every student can meet real accomplishment criteria often enough to be recognized frequently. Recognition of accomplishment can be tied to an opportunity to participate in events that matter to students and their families. Student accomplishment is also recognized when student performance on standards-based assessments is related to opportunities at work and in higher education.

5. Academic Rigor in a Thinking Curriculum

Thinking and problem solving will be the "new basics" of the 21st century, but the common idea that we can teach thinking without a solid foundation of knowledge must be abandoned, so must the idea that we can teach knowledge without engaging students in thinking. Knowledge and thinking are intimately joined. This implies a curriculum organized around major concepts that students are expected to know deeply. Teaching must engage students in active reasoning about these concepts. In every subject, at every grade level, instruction and learning must include commitment to a knowledge core, high thinking demand, and active use of knowledge.

6. Accountable Talk

Talking with others about ideas and work is fundamental to learning but not all talk sustains learning. For classroom talk to promote learning it must be accountable to the learning community, to accurate and appropriate knowledge, and to rigorous thinking. Accountable talk seriously responds to and further develops what others in the group have said. It puts forth and demands knowledge that is accurate and relevant to the issue under discussion. Accountable talk uses evidence appropriate to the discipline (e.g., proofs in mathematics, data from investigations in science, textual details in literature, documentary sources in history) and follows established norms of good reasoning. Teachers should intentionally create the norms and skills of accountable talk in their classrooms.

7. Socializing Intelligence

Intelligence is much more than an innate ability to think quickly and stockpile bits of knowledge. Intelligence is a set of problem-solving and reasoning capabilities along with the habits of mind that lead one to use those capabilities regularly. Intelligence is equally a set of beliefs about one's right and obligation to understand and make sense of the world, and one's capacity to figure things out over time.

Intelligent habits of mind are learned through the daily expectations placed on the learner by calling on students to use the skills of intelligent thinking, and by holding them responsible for doing so, educators can "teach" intelligence. This is what teachers normally do with students from whom they expect achievement; it should be standard practice with all students.

8. Self-management of Learning

If students are going to be responsible for the quality of their thinking and learning, they need to develop and regularly use an array of self-monitoring and self-management strategies. These meta- cognitive skills include noticing when one doesn't understand something and taking steps to remedy the situation, as well as formulating questions and inquiries that let one explore deep levels of meaning. Students also manage their own learning by evaluating the feedback they get from others; bringing their background knowledge to bear on new learning; anticipating learning difficulties and apportioning their time accordingly and judging their progress toward a learning goal. These are strategies that good learners use spontaneously and that all students can learn through appropriate instruction and socialization. Learning environments should be designed to model and encourage the regular use of self-management strategies.

9. Learning as Apprenticeship

For many centuries most people learned by working alongside an expert who modeled skilled practice and guided novices as they created authentic products or performances for interested and critical audiences. This kind of apprenticeship allowed learners to acquire complex interdisciplinary knowledge, practical abilities, and appropriate forms of social behavior. Much of the power of apprenticeship learning can be brought into schooling by organizing learning environments so that complex thinking is modeled and analyzed, and by providing mentoring and coaching as students undertake extended projects and develop presentations of finished work, both in and beyond the classroom.

The units in *Exploring Computer Science* contain individual lessons that taken together as a unit fit the construct for inquiry-based learning outlined in the following chart adapted from the "5 E Model".

The Inquiry-Based Learning Cycle

(Adapted from the 5 E Model", R. Bybee)

Stage of Inquiry in an Inquiry-Based Science Program	Possible Student Behavior	Possible Teacher Strategy
Engage	Asks questions such as, Why did this happen? What do I already know about this? What can I find out about this? How can I solve this problem? Shows interest in the topic.	Creates interest. Generates curiosity. Raises questions and problems. Elicits responses that uncover student knowledge about the concept/topic.
Explore	Thinks creatively within the limits of the activity. Tests predictions and hypotheses. Forms new predictions and hypotheses. Tries alternatives to solve a problem and discusses them with others. Records observations and ideas. Suspends judgment. Tests idea	Encourages students to work together without direct instruction from the teacher. Observes and listens to students as they interact. Asks probing questions to redirect students' investigations when necessary. Provides time for students to puzzle through problems. Acts as a consultant for students.
Explain	Explains their thinking, ideas and possible solutions or answers to other students. Listens critically to other students' explanations. Questions other students' explanations. Listens to and tries to comprehend explanations offered by the teacher. Refers to previous activities. Uses recorded data in explanations.	Encourages students to explain concepts and definitions in their own words. Asks for justification (evidence) and clarification from students. Formally provides definitions, explanations, and new vocabulary. Uses students' previous experiences as the basis for explaining concepts.
Elaborate	Applies scientific concepts, labels, definitions, explanations, and skills in new, but similar situations. Uses previous information to ask questions, propose solutions, make decisions, design experiments. Draws reasonable conclusions from evidence. Records observations and explanations	Expects students to use vocabulary, definitions, and explanations provided previously in new context. Encourages students to apply the concepts and skills in new situations. Reminds students of alternative explanations. Refers students to alternative explanations.
	Checks for understanding among peers. Answers open-ended questions by using	Refers students to existing data and evidence and asks, What do you know?

Stage of Inquiry in an Inquiry-Based Science Program	Possible Student Behavior	Possible Teacher Strategy
Evaluate	observations, evidence, and previously accepted explanations. Demonstrates an understanding or knowledge of the concept or skill. Evaluates his or her own progress and knowledge. Asks related questions that would encourage future investigations.	Why do you think...? Observes students as they apply new concepts and skills. Assesses students' knowledge and/or skills. Looks for evidence that students have changed their thinking. Allows students to assess their learning and group process skills. Asks open-ended questions such as, Why do you think...? What evidence do you have? What do you know about the problem? How would you answer the question?

Concrete Instructional Strategies

There are several concrete instructional strategies that are included in each unit to implement this culturally relevant, student-centered, and inquiry-based vision.

- Each unit begins with a description of the topic, an explanation of the importance of this topic, possible social applications of this topic, and objectives/standards for the unit.
- Whenever possible, units begin with kinesthetic activity to get students involved in the unit topic. Students are more engaged when they go beyond seatwork to gain familiarity with the scope of a topic. Acting out computing concepts is one way to have students actively engaged in the curriculum.
- Whenever possible, units present the final unit project at the beginning of the unit so students understand what type of project they will engage in at the end of the unit. Daily assignments help scaffold their knowledge towards gaining the knowledge needed to complete a particular project. The final project represents a culmination of their new knowledge and provides an opportunity to expand their understandings to a particular socially-relevant problem.
- Computing terms and definitions are explicit and part of the instruction. The curriculum avoids unnecessary jargon which might distract learning of the critical content. Students have opportunities to use writing to reinforce the literacy component behind these computing terms and definitions.
- Foundational computing topics are connected to the 'pop-technology' students have likely encountered: cellular phones, iPods, MySpace / Facebook, blogs, Internet browsing, etc.

- Whenever possible, real-world problems are presented in the context of socially-relevant issues impacting urban communities (housing, safety, poverty, health care, access to equal rights, educational opportunities, improving social services, translation services, transportation, etc.)
- Students have opportunities to work on problems that they help define and can individualize – i.e. selecting their own content for Web sites; creating original, not pre-scripted, problem-solving strategies, etc.
- Activities are designed to encourage students to work in a variety of collaborative settings: peer-programming, group research projects, etc. which encourage conversations around computing topics.
- Students will experience a variety of ways to communicate their answers – academic writing, writing a letter to a friend or companion, using presentation software, developing graphics or animation, listing algorithms, drawing illustrations, oral presentations, etc.
- Units incorporate examples of careers in computing as they arise in the curriculum. Students will be given hypothetical opportunities to act as a professional to take on the behavior and skills to solve a given problem.
- Though using technology is a core component of this curriculum, using computers is not necessarily embedded in the curriculum on a daily basis.

All of these strategies contribute to developing the problem solving skills and algorithmic thinking processes that are emphasized throughout the course.

It is important to note that each unit focuses on different instructional strategies; this is purposeful. In some cases, it is because the particular subject matter lends itself more successfully to a particular set of strategies, but this was also done to highlight the wide variety of possible strategies that can be used effectively in teaching this course. We encourage teachers to experiment by trying strategies that work well for them in a variety of different places in the curriculum. For example, the peer review process utilized in Unit 4 could be adapted for use in other units; the idea of an “elbow” partner can be used in all units. Journal responses and blog entries can be used by students to communicate about their work in any of the units. There are many other possibilities to consider.

Assessment

With the exception of the final projects, there are no specific assessments listed in the lesson plans. There are also very few specific “homework” assignments. Differences in grading policies, types of assessments required, and student schedules make it difficult to gauge the best combination of assessment tools to use in a particular environment. Teachers are encouraged to determine which class activities might lend themselves to some research outside of class and which might make useful assessments. Additional assessment instruments can be developed by individual teachers or teacher teams. All forms of assessment should meet the criteria outlined in the Nine Principles of Learning.

Overview of the Instructional Materials

The pages that follow contain the core of the materials teachers will need in order to plan and deliver *Exploring Computer Science*. The materials begin with the unifying themes and practices that are woven throughout the course followed by a Scope and Sequence chart that details the various topics included in the course, along with the unit in the course where each is introduced and reinforced. Teachers should continue to refer back to previous units where appropriate. For example, Unit 3 builds on many of the Unit 1 concepts by taking students from discussing and viewing websites to actually using and developing them. The approximate time allotment noted in the chart includes all activities from introduction through application.

Following the Scope and Sequence is an overview of each unit that includes the unit description and overall objectives of the unit. There is also a table that indicates the topics for each instructional day of the course.

Finally, are the daily lesson plans with detailed student activities and teaching strategies for each day. Each lesson has been built on a 55 minute class period. In schools where class periods are shorter or longer (or on varying block schedules) adjustments will need to be made; such adjustments may include combining lessons (for longer class periods) or assigning parts of the lesson for homework (for shorter class periods).

An attempt was made to provide enough detail to the teaching strategies sections to give teachers clear guidance as to the activities involved and the types of questions that might need to be asked to prompt discussion. At the same time, an effort was made not to be prescriptive.

Each unit includes supplementary materials, a final project, and a sample rubric for the final project.

Unifying Themes and Practices

The individual lessons in this course were developed to reinforce the unifying themes and support the use of the computational practices that we expect students to employ.

The three themes are

- The creative nature of computing
- Technology as a tool for solving problems
- The relevance of computer science and its impact on society

There are many technological tools that enable people to explore concepts and create exciting and personally relevant artifacts that impact society. In this course, programming is used as one of the tools, but not the only tool. Students are asked to be creative in designing and implementing solutions as they translate ideas into tangible forms. As students actively create, they will also discuss the broader implications of computing technologies.

Throughout the course students will gain experience in employing the following computational practices.

- Analyze the effects of developments in computing
- Design and implement creative solutions and artifacts
- Apply abstractions and models
- Analyze their own computational work and the work of others
- Connect computation with other disciplines
- Communicate thought processes and results
- Work effectively in teams

As students design and implement solutions using abstractions and models, they will analyze the processes they and their peers use to arrive at solutions, study the effects of their creations and learn how computing concepts connect explicitly and implicitly to other disciplines. Students will learn about the collaborative nature of computer science by working in teams and communicate the results of their work in writing and orally supported by graphs, visualizations and computational analysis.

Scope and Sequence

Topic	Focus	HCI	PS	WEB	PR	ROB	CA
1. Principles of Computer Organization (~1 week)	1. Terminology	I		R	R	A	A
	2. Hardware components	I		R	R	A	A
	3. Software components	I		R	R	A	A
	4. Interaction of components	I		R	R	A	A
	5. Purchasing a computer	I					
2. Internet concepts/Web Design and Development (~1 week)	1. Internet elements/Terminology	I		R			
	2. Search engine fundamentals	I		R			
	3. Search engines and directories	I		R			
	4. Refining search parameters	I		R			
	5. Evaluating Web sites	I		R			
	6. Security on the Internet	I		R			
3. Models of intelligent behavior (~1 week)	1. What is intelligence?	I				R,A	
	2. Natural language	I				R,A	
	3. Knowledge-based systems	I				R,A	
	4. Machine learning	I				R,A	
	5. Game playing, searching	I					
	6. Myth of intelligent behavior	I				R,A	
4. Interdisciplinary Utility of Computers and Problem Solving (~2 weeks)	1. How are computers used?	I					
	2. Information storage and retrieval	I				R,A	R,A
	3. Decision-making support	I				R,A	R,A
	4. Data visualization	I				R,A	R,A
	5. Communications	I				R,A	R,A
	6. Modeling and design	I				R,A	R,A
	7. Art, music, video	I					
	8. Education and training	I					
	9. E-commerce	I					
	10. Embedded systems	I					
5. Design for Usability (~6 weeks)	1. Fundamental HCI concepts	I		R			
	2. Identify elements of user-friendly Web sites	I		R			
	3. HTML tags			I			
	4. Styles and markup			I			
	5. Design a user-friendly Web site	I					
	6. Create a user-friendly Web site			I			
	7. Design a user-interface for a program			I	R	A	A
	8. Documentation techniques			I	R	A	A
	9. Web development tools			I			
6. Problem Solving and Program Design (~6 weeks)	1. Problem-solving process		I	R	R	A	A
	2. Understanding the problem		I	R	R	A	A

	3. Exploring problems : problem-solving heuristics and strategies		I	R	R	A	A
	4. Design creation and representation		I	R	R	A	A
	5. Problem data		I	R	R	A	A
	6. Solution accuracy		I	R	R	A	A
	7. Program coding and testing		I	R	R	A	A
	8. Design Re-evaluation and refinement		I	R	R	A	A
	9. Decomposing the complex		I	R	R	A	A
	10. From Source to Execution			I	R	A	A
	11. Abstraction		I	R	R	A	A
	12. Communicate results		I	R	R	A	A
7. Discrete Mathematics—Connections between mathematics and computer science (~3 weeks)	1. Logic		I		R	A	A
	2. Binary number system		I				
	3. Basic Sets		I		R	A	A
	4. Concepts of Functions		I		R	A	A
	5. De Morgan's Laws		I		R	A	A
	6. Graphs		I		R	A	A
8. Programming Languages (~8 weeks)	1. Terminology				I	R,A	R,A
	2. Representation of text				I	R,A	R,A
	3. Representation of numbers				I	R,A	R,A
	4. Data types				I	R,A	R,A
	5. Programming style				I	R,A	R,A
	6. Objects				I	R,A	R,A
	7. Input and Output				I	R,A	R,A
	8. Expressions				I	R,A	R,A
	9. Selection				I	R,A	R,A
	10. Iteration				I	R,A	R,A
	11. Interactive programming				I	R,A	R,A
	12. Methods (functions) and parameters				I	R,A	R,A
	13. Properties				I	R,A	R,A
9. Fundamentals of Hardware Design (~1 week)	1. Conversion between decimal and binary number systems		I				
	2. Binary counting and switching		I				
	3. Representation of numbers		I		R	R	R
10. Limits of Computing (~2 weeks)	1. Computers vs. humans		I		R	R	R
	2. Algorithm efficiency		I			R	R
	3. Computationally intensive problems		I			R	R
	4. Parallel processing					I	R
	5. Unsolvable problem for the computer					I	R
	6. Computationally hard problems.		I			R	R
11. Principles of Software Engineering (~5 weeks)	1. Software design team					I	R

	2. Break a problem statement into specific requirements		I	R	R	R,A	R,A
	3. Design a solution to a problem		I	R	R	R,A	R,A
	4. Code a solution from a design			I	R	R,A	R,A
	5. Test a solution to identify bugs		I	R	R	A	A
	6. Pair programming						I
	7. Team Oral Presentations	I	R	R	R	R	R
12. Ethical Issues and Social Issues (weave throughout)	1. Terminology						
	2. How technology has changed						
	3. The effect of technology						
	4. Privacy and sharing of information						
	5. Intellectual property						
	6. Responsible use of software						
	7. Software licensing agreements						
	8. Digital rights management (DRM)						
	9. Intellectual property/fair use conflicts						
	10. Current legislation and/or litigation						
13. Careers in Computing (weave throughout)	1. List careers related to computers						
	2. Personal career choices						
	3. Skills and academic background						
	4. Choose a computing career						

Overview Chart

Human Computer Interaction Unit Overview	
Instructional Day	Topic
1-2	Explore the concepts of <i>computer</i> and <i>computing</i> .
3-4	“Demystify” and learn the function of the parts of a personal computer by dissecting a real computer. Learn the terminology of hardware components necessary for the purchase of a home computer.
5-6	Computer Buying Project
7-9	Explore the world wide web and search engines. Experiment with a variety of search techniques, internet resources, and Web 2.0, applications. Evaluate websites.
10	Explore how computers are used for communications.
11-14	Explore how computers are used as a tool for visualizing data, modeling and design, and art in the context of culturally situated design tools.
15	Field trip
16-17	Introduce the concept of a computer program as a set of instructions.
18-20	Explore the idea of intelligence—especially as it relates to computers. Explore what it means for a machine to “learn”. Discuss whether computers are intelligent or whether they only behave intelligently.
21-22	Final projects and presentations
Problem Solving Unit Overview	
Instructional Day	Topic
1	Introduce the four steps of the problem solving process.
2-4	Apply the problem solving process. Use different strategies to plan and carry out the plan to solve several problems.
5-7	Reinforce the four steps of the problems solving process.
8-9	Count in the binary number system.
10-11	Convert between binary and decimal numbers in the context of topics that are important to computer science.
12-13	Introduce the linear and binary search algorithms.
14-16	Explore sorted and unsorted lists and various sorting algorithms.
17-18	Introduce minimal spanning trees and how graphs can be used to help solve problems.

19-22	Final projects and presentations
Web Design Unit Overview	
Instructional Day	Topic
1-2	Issues of social responsibility in web use are explored as well as the relative merits of the influence of the web on society, personal lives, and education.
3-4	Introduce the use of basic html.
5	Introduce basic formatting in html.
6-7	Explore image editing for the web using Photoshop or an image editor of choice.
8-10	Introduce basic css.
11-13	Explore the concept of separating style from structure by keeping separate html and css files.
14	Add hyperlinks to other websites.
15-16	Introduce a variety of page layout styles.
17-19	Practice using style tables, lists and css in the context of a web page creation project.
20-21	Introduce several web user interface elements combining javascript, html, css, and Photoshop.
22	Implement advanced functionality with javascript libraries. Create accordion menus based on the mootools implementation.
23-24	Further explore the use of javascript library effects, including lightbox slideshow and sliding image puzzles.
25-28	Final projects and gallery walk
Introduction to Programming Unit Overview	
Instructional Day	Topic
1	Introduce the Scratch programming language, including the basic terms utilized in the language.
2-3	Practice using the basic features of Scratch in the context of creating a simple program.
4	Create a dialogue between two sprites.
5-6	Introduce the methods of moving sprites in Scratch.
7-8	Practice the concept of event driven programming through the creation of an alphabet game.

9	Introduce the concept of broadcasting via role play.
10	Develop a story to be used in a Scratch program.
11-15	Write Scratch stories and present them to the class. Peer reviews are conducted.
16	Introduce the concept of variable.
17	Introduce the concept of conditionals.
18-19	Introduce And, Or and randomness.
20	Apply knowledge of conditionals to develop a Rock Paper Scissors program in Scratch.
21	Build on previous programming concepts to create a timer.
22-26	Create a timing game in Scratch and present it to the class. Peer reviews are conducted.
27	Investigate two types of games that may provide ideas for the final project.
28	Explain final project and the rubric for the final project.
29-33	Write Scratch programs for either My Community or Game project. Peer reviews will be conducted.
34-35	Presentations of final projects
Robotics Unit Overview	
Instructional Day	Topic
1	What is a robot? Identify the criteria that make an item a robot.
2-3	Evaluate robot body designs and create algorithms to control robot behavior.
4	Set up LEGO® Mindstorms® kit.
5	Build robot base.
6-7	Introduce the features of NXT Brick—the “brain” of the robot.
8-9	Introduce the features of the Mindstorms NXT software.
10-14	Program the robot using the Mindstorm Robot Educator Software tutorials.
15	Introduce RoboCup real life robotic competition and write instructions for tic-tac-toe.
16	RoboTic-Tac-Toe Tournament and introduction to RoboCupJunior Dance Challenge.

17-20	Build, program, and present a dancing robot.
21-25	Build program and present a rescue robot.
26-30	Design, build and program a robot that solves a stated problem. (Optional—time permitting)
31-40	Final projects and presentations
Computing Applications Unit Overview	
Instructional Day	Topic
1	Introduce the Python programming environment and the Pen class.
2	Introduce drawing in Python by using coordinates .
3-5	Create a program to draw a dream house or car using the concept of pair programming.
6	Introduce the use of Dialogs in Python.
7-10	Introduce the concepts of software development activities, models and design teams. Practice dialogs and working in teams to create an order form program.
11	Introduce numerical types and math in Python.
12	Introduce functions in Python.
13	Practice the use of functions through programs to exchange currencies and calculate measurements.
14	Introduce conditionals in Python.
15-17	Practice the use of conditionals and functions through the creation of a Choose Your Own Adventure program.
18	Introduce while loops in Python.
19	Introduce the for loop in Python.
20	Introduce the concept of lists.
21-25	Practice the use of loops, conditionals, and list through the creation of an opinion poll program.
26-30	Final project

Topic Descriptions and Objectives

Unit 1: Human Computer Interaction (~4 weeks)

Topics to be addressed:

- Principles of Computer Organization
- Internet Concepts
- Models of Intelligent Behavior
- Interdisciplinary Utility of Computers and Problem Solving in the Modern World

Topic Description:

The student will be introduced to the major components of the computer, including: input, output, memory, storage, processing, software, and the operating system. Students will consider how Internet elements (e.g. email, chat, WWW) are organized and will engage in effective searching. They will explore a variety of web applications. Fundamental notions of Human Computer Interaction (HCI) and ergonomics are introduced. Students will learn that “intelligent” machine behavior is not “magic” but is based on algorithms applied to useful representations of information. Students will learn the characteristics that make certain tasks easy or difficult for computers, and how these differ from those that humans characteristically find easy or difficult. Students will gain an appreciation for the many ways (types of use) in which computers have had an impact across the range of human activity, as well as for the many different fields in which they are used. Examples illustrate the broad, interdisciplinary utility of computers and algorithmic problem solving in the modern world.

Objectives:

The student will be able to:

- Identify the various functional components of a computer.
- Match a list of computer terms and definitions/functions.
- Describe the interaction of the various functional components of the computer.
- Make appropriate decisions when purchasing a computer for home use.
- List at least three strengths and weaknesses of each of three Internet elements and at least one use for each.
- Use at least two Internet elements.
- Use appropriate tools and methods to execute Internet searches which yield requested data.
- Develop and use a rubric to evaluate the results of web searches and reliability of information found on the web.
- Given a list of tasks from several application areas of artificial intelligence, indicate whether or not computers can do those tasks, using current technology.
- Find (in newspapers, magazines, through interviews, or on the Internet) and describe three examples of the use of technology in non-computer fields.
- Choose the appropriate category for each item in a list of technology applications.

Unit 2: Problem Solving (5 weeks)**Topics to be addressed:**

- Problem Solving and Program Design
- Discrete Mathematics—Logic and Functions
- Connections between Mathematics and Computer Science

Topic Description:

This unit covers the basic steps in algorithmic problem-solving, including the problem statement and explanation, examination of sample instances, design, coding, testing, and verification. Tools for expressing design will be used. This unit also focuses on the connections between mathematics and computer science. Students will be introduced to selected topics in discrete mathematics including (but not limited to) Boolean logic, functions, and graphs. Students will be introduced to the binary number system. Students are also introduced to searching and sorting algorithms and graphs. Suitable exercises are presented that illustrate the value of mathematical abstraction in solving computing problems.

Objectives:**The student will be able to:**

- Name and explain the steps they use in solving a problem.
- Solve a problem by applying various problem solving techniques.
- Express a solution using standard design tools.
- Determine if a given algorithm successfully solves a stated problem.
- Write algorithms that use simple and complex logic statements (relational operators and Boolean operators).
- Count in binary and convert between decimal and binary numbers.
- Describe selected searching and sorting algorithms.
- Analyze selected searching and sorting algorithms.
- Write an algorithm that uses mathematical functions.
- Apply simple graph concepts in problem solving.

Unit 3: Web Design (6 weeks)**Topics to be addressed:**

- Web Page Design and Development
- Design for Usability
- Hierarchy and Abstraction in Computing

Topic Description:

This section prepares students to take the role of a developer by expanding their knowledge of programming and web page design and applying it to the creation of web pages, programs, and documentation for users and equipment. Students will explore issues of social responsibility in web use. They will learn to plan and code their web pages using a variety of techniques and check their sites for usability. Students learn to create user-friendly Web sites. Students will apply fundamental notions of Human Computer Interaction (HCI) and ergonomics.

Objectives:**The student will be able to:**

- Correctly use HTML tags to create web pages, apply styles to HTML documents to control presentation, and express the design of a web site using standard tools.
- Create user-friendly and functional web sites that apply good HCI practices.
- Prepare documentation.

Unit 4: Introduction to Programming (7 weeks)**Topics to be addressed:**

- Program design
- Programming constructs

Topic Description:

Students will be introduced to some basic issues associated with program design and development. Students design algorithms and programming solutions to a variety of computational problems using Scratch. Programming problems include control structures, functions, parameters, objects and classes, structured programming and event-driven programming techniques.

Objectives:**The student will be able to:**

- Code, test, and execute a program that corresponds to a set of specifications.
- Convert a word problem into code using top-down design.
- Select appropriate data types.
- Write structured program code.
- Draw a series of diagrams showing the scope and values of variables during execution of a simple program.

Unit 5: Robotics (8 weeks)**Topics to be addressed:**

- Fundamentals of Hardware Design
- Applications of Computing

Topic Description:

This unit introduces robotics as an advanced application of computer science. Students explore how to integrate hardware and software in order to solve problems. Students will see the effect of software and hardware design on the resulting product. Students will apply previously learned topics to the study of robotics.

Objectives:**The student will be able to:**

- Identify the criteria that describe a robot and determine if something is a robot.
- Describe the steps that happen when a computer processes an instruction.
- Match the actions of the robot to the corresponding parts of the program.
- Build, code, and test a robot that solves a stated problem.
- List and explain ways in which different hardware designs affect the function of a machine.
- Identify multiple ways to program the robot to achieve a goal and explain why one is better than another.

Unit 6: Computing Applications (6 weeks)**Topics to be addressed:**

- Limits of Computing
- Principles of Software Engineering
- Applications of Computing

Topic Description:

This unit provides an elementary introduction to computational complexity theory to encourage an appreciation for the relative efficiency of various algorithms. Students are introduced to examples of computationally “hard” problems, computationally unsolvable problems, and problems that are made difficult by the complexity of the realities they attempt to model (air traffic control, human intelligence, weather). Students are introduced to software engineering concepts and team-oriented approaches for solving problems. They learn the essential methods of the software development life cycle and use these methods in one or more group projects involving large data sets.

Objectives:**The student will be able to:**

- List activities in which humans excel over computers and activities in which computers excel over humans.
- Calculate the number of steps required to execute a given algorithm.
- Describe and run computationally intensive problems.
- Describe at least one problem computers cannot solve.
- Describe at least one computationally hard (NP) problem.
- Name the different phases of the software development process.
- Use a software process model (such as the waterfall, RAD, incremental, or XP) to solve a problem.
- Complete a project as a software design team with assigned roles and responsibilities for each member.
- Complete programs using pair programming.

The following topics should be woven throughout the course as appropriate:

- Ethical Issues and Social Issues
- Careers in Computing

Topic Description:

The proliferation of computers and networks raises a number of ethical issues. Technology has had both positive and negative impacts on human culture. Students will be able to identify ethical behavior and articulate both sides of ethical topics. Students study the responsibilities of software users and software developers with respect to intellectual property rights, software failures, and the piracy of software and other digital media. They are introduced to the concept of open-source software development and explore its implications. Students identify and describe careers in computing and careers that employ computing. Information is provided about the required technical skill set, soft skills, educational pathways, and ongoing training required for computing careers. Students also explore how computers are used in other career choices. Finally, students are made aware of which additional secondary-level courses might be needed in preparation for various careers.

Objectives:

The student will be able to:

- Distinguish between ethical and legal issues in a case study by listing the issues that can be resolved through the legal system and those issues that cannot be legally resolved.
- Defend an ethical stance given a controversial or ethically ambiguous situation in a debate.
- List and explain at least two positive and negative effects of one technological innovation on human culture.
- Define intellectual property and state the impact of provisions to protect it.
- Identify at least two benefits and two drawbacks of using commercial, public domain, open source, and shareware.
- Demonstrate behavior in the use of technology that conforms to school and local code.
- Define intellectual property, explain the rights of owners and end users, and provide rationale for the need to protect owners and end users.
- Define software piracy and discuss its effect on software company profits and the price of software to the consumer.
- List at least two ways in which software (and other digital media) is protected and state at least one current law to protect software and the makers of software.
- Describe the responsibilities of software professionals to society and to each other.
- List the advantages and disadvantages of open-source software.
- List five careers related to computers.
- List three or more skills needed to succeed in at least three computer-related careers.
- State the level of education and ongoing training needed for at least three careers.

Unit 1:

Human Computer Interaction



Introduction

Computers and other computing technology have become an integral part of our society. Because of that it is easy to forget that not every student will come to this course with the same background and skills. For many students this course will be the first opportunity they have to become facile with using the keyboard and navigating the internet.

The topics in this unit are designed to allow all students to gain familiarity with computers and computing in the context of activities that give students an opportunity to work at their own pace, work in groups where they can learn from each other and generally gain an overview of the many and varied ways in which computers and computing are used.

The unit is divided into three main sections.

- Exploring the concepts of computer and computing by investigating computer hardware components and a variety of internet resources (Days 1-9)
- Exploring the use of computers in a variety of fields (Days 10-15)
- The computer as a machine that needs to be “told what to do” (Days 16-20)

The goal is for all students in the class to reach a level of comfort in using the computer and understand that the computer is not magic. The fact that for a computer to accomplish its tasks it needs to be given precise instructions motivates the need for the problem solving techniques that will be addressed in Unit 2.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1-2	Explore the concepts of <i>computer</i> and <i>computing</i> .
3-4	“Demystify” and learn the function of the parts of a personal computer by dissecting a real computer. Learn the terminology of hardware components necessary for the purchase of a home computer.
5-6	Computer Buying Project
7-9	Explore the world wide web and search engines. Experiment with a variety of search techniques, internet resources, and Web 2.0, applications. Evaluate websites.
10	Explore how computers are used for communications.
11-14	Explore how computers are used as a tool for visualizing data, modeling and design, and art in the context of culturally situated design tools.
15	Field trip
16-17	Introduce the concept of a computer program as a set of instructions.
18-20	Explore the idea of intelligence—especially as it relates to computers. Explore what it means for a machine to “learn”. Discuss whether computers are intelligent or whether they only behave intelligently.
21-22	Final projects and presentations

Daily Lesson Plans

Instructional Days: 1-2

Topic Description: What is a computer? In this lesson the concepts of *computer* and *computing* are explored through examples of each.

Objectives:

The student will be able to:

- Explain and give examples of the concepts of *computer* and *computing*.

Outline of the Lesson:

- Journal Entry. (10 minutes)
- Exploring computers (60 minutes)
- Classification of computing groups (30 minutes)
- Definition of the terms *computer* and *computing* (10 minutes)

Student Activities:

- Complete journal entry.
- Groups of students create posters of their ideas of what a computer is.
- Participate in a gallery walk and complete group presentations.

Teaching/Learning Strategies:

- Journal Entry: What is a computer?
 - Have students write responses to the question in their journals and then share the response with their elbow partner.
- Exploring computers
 - Divide students into groups of 3 or 4. Ask the students to discuss examples of computers (or things containing computers). As they find examples of computers, have them form a list and create a poster that highlights their ideas. (Examples of computers include: Macintosh, Windows PC, cell phone, mp3 player, most appliances (television, coffee maker, washer, dishwasher, etc.), cars, medical equipment, planes, watches, cash registers, ATMs, traffic lights, scoreboards, humans, and calculators.)
 - Have student groups present their posters and share their ideas. After each presentation, give the other students an opportunity to suggest why any particular example seems not to be a computer (or is not obviously a computer). If necessary, ask questions to draw out the student questions and responses. (For example, if the student says “dishwasher,” you might ask, “why is a dishwasher a computer.”)

- Classification of computing groups.
 - Ask students to do a gallery walk of the posters and put group labels on items that appear in the posters.
 - Following the gallery walk, create a new list with the various items listed under a group classification.
- Definition of the terms *computer* and *computing*.
 - Revisit the question “What is a computer?” and ask the possibly more pertinent question, “What is *computing*?”
 - Have the students use their list of “computers” and their classifications to help formalize their answers.
 - Note that there is no “correct” answer. These definitions will be revisited and possibly modified throughout the course of the unit.
 - Reinforce the idea of different types of computers and classifications by reviewing the lists and groups created by the students.
- Assignment
 - Ask students to find and bring in to class a print advertisement (from a newspaper, magazine, or the internet) for a personal computer.

Resources:

- No additional resources needed

Instructional Days: 3-4

Topic Description: Students explore the functionality of various parts of a personal computer by dissecting a real computer. The correct terminology for computer hardware components is discussed.

Objectives:

The student will be able to:

- Use the correct terminology for computer hardware components
- Describe the functionality of the parts of a personal computer.

Outline of the Lesson:

- Dissection of a computer (95 minutes)
- Review of components and uses (15 minutes)

Student Activities:

- Work in teams to take apart a computer and label the parts.

Teaching/Learning Strategies:

- Dissection of a computer.
 - Have the students work in teams (the number of students per team depends on the number of computers you have) to carefully take apart a computer.
 - Emphasize the importance of safety when working with electronics and tools of any type.
 - During the dissection, students should label the parts. One student in each group should search the web for the part based on the Computer Components Webquest.
 - Circulate the room and answer questions.
 - Note: If there are no computers available for dissection, there are other options. For example,
 - Have one computer for the entire class to dissect
 - Students complete the Computer Components Webquest with their elbow partner and label the advertisements that they brought to class and then present to the remainder of the class.
- Review the main parts of the hardware of a personal computer.
 - Ask the students if they better understand the “technical jargon” that appears in the ads. Challenge them to think about what features they might want if they were purchasing a home computer (i.e., large screen, fast, attractive color, etc.).

Resources:

- Several old personal computers (working or non-working) that can be taken apart by the students. These computers can often be found at garage sales or may be destined for the garbage heap by a school or

local business. If possible, get several different types of computers that have a variety of components.

Do not use school computers or computers which need to remain in working condition!

- Various screwdrivers (both slotted-head and phillips-head)
- Computer Components Webquest
- Wikipedia has more detail about all of the hardware terminology listed: <http://www.wikipedia.org>

Computer Components Webquest

Your job is to use the internet to investigate the different components (parts) of a computer. Rather than search for each part individually, you may find it easier to do a search for computer hardware components. There are many sites with images and descriptions.

Below are the different components for you to investigate:

- Processor
- Operating System
- Memory
- Hard Drive
- Optical Drive
- Monitor
- Video Card
- Sound Card
- Speakers
- Keyboard
- Mouse
- Modem

For each of the components write down:

- The name
- What it is used for
- What are the different options or sizes for the components

Instructional Days: 5-6

Topic Description: Students complete a project related to choosing appropriate components for a personal computer.

Objectives:

The student will be able to:

- Describe the uses for computer hardware components.
- Choose hardware components for various types of users.

Outline of the Lesson:

- Research and development of computer buying project (85 minutes)
- Project presentations by student teams (25 minutes)

Student Activities:

- Student teams research and complete projects.
- Student teams present projects.

Teaching/Learning Strategies:

- Distribute project information and rubric.
 - Have students work in the same teams as they did for dissecting the computer.
 - Explain project and rubric; answer questions; assign each team a scenario.
- Project presentations by student teams
 - Have each team discuss the scenario that they were assigned, display their comparison chart and explain which computer was chosen and why.

Resources:

- Computer Buying Project
- Computer Buying Project Sample Scenarios
- Computer Buying Project Sample Rubric

Computer Buying Project

You will be given a scenario for someone that wants to buy a new computer. Your task is to give them at least 4 options and then give them advice on which one to buy. Your project will be presented to the class.

The final product can be a:

- Powerpoint
- Skit
- Video
- Poster
- Other approved product

Your final product should have:

- A title with group members' names
- The scenario that you are given
- Computer comparison chart
- Which computer is chosen
- Justification for choosing that computer

Example computer comparison chart (more information can be added):

	Dell Inspiron 530s	Macbook
Laptop or Desktop		
Processor (CPU)		
Operating System		
Memory (RAM)		
Hard Drive (storage)		
Optical Drive		
Monitor or Screen Size		
Video Card		
Sound Card		
Other Accessories		
Cost		

Student Grouping:

You will be in a group of up to 4 students.

Sample Scenarios for Computer Buying Project

Scenario #1

I play lots of games. I play them in my room so I can have the tv on while I play. Whenever a new game comes out, I buy it. I need the games to run as fast as possible with the best graphics and sound. My parent's are buying the computer so I don't care how much it costs.

Scenario #2

I take the bus to work everyday. I need a computer that I can use on the bus. I make financial reports and charts using Powerpoint and Excel. My budget is \$1500.

Scenario #3

I am a writer. I need to write books wherever inspiration hits me. Sometimes I write at the beach. Other than that, I use the computer for the internet sometimes. I haven't sold a book yet, so I only have about \$900 to spend on a computer.

Scenario #4

I am dad that takes lots of videos of my kids. I want to be able to save them on my hard drive. I also want professional software that will help me edit the videos. I want to be edit the movies quickly and make dvds for my entire family. My budget is \$3000.

Scenario #5

I am teacher that needs a computer that I can use at home and take to work. I've used a lot of Apples in the past and like the fact that they are really easy to use. I only need basic office software. My budget is \$1500.

Scenario #6

I collect music from all kinds of bands. I need a computer that can store all the music and videos I have of my favorite bands. I only want to use my computer at home. Other than that I just surf the internet. My budget is \$1000.

Scenario #7

I am a graphic designer. I use Adobe Photoshop and Illustrator. I need a computer that can allow me to quickly edit large pictures. My budget is \$2000. I only work at home.

Computer Buying Project Sample Rubric

Group Members Names: (up to 4)

_____	_____
_____	_____

Do you have?	Points Possible	Yes	No	Points Earned
Product				
Title with group members' names	10			
Scenario is described	10			
Computer comparison chart with at least 4 options	5			
Chart has specifications of options	5			
Options on chart fit your scenario	10			
Show justification for your computer choice	10			
Your choice fits the scenario	10			
Visuals of your choices (pictures or video of choices)	10			
Presentation				
Present your project	15			
Present all required parts of project	15			
Extra Credit				
Project exhibits creativity above and beyond	Up to 10			
TOTAL:	100			

Instructional Days: 7-9

Topic Description: Search engines and how they work are explored through trying various internet search techniques. A selection of Internet resources that are useful for finding information are introduced as well as a selection of Web 2.0 applications. Several websites are evaluated by using a rubric to determine if they are “good” websites.

Objectives:

The student will be able to:

- Perform searches and explain how to refine searches to retrieve better information.
- Identify resources for finding information in addition to ranking based search engines.
- Differentiate between ranking based search engines and social bookmarking (collaborative) search engines.
- Use a variety of Web 2.0 applications.
- Develop and use a rubric to evaluate websites.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Experiment with refining searches (15 minutes)
- Discussion of other resources for finding information (15 minutes)
- Experimentation with these resources (20 minutes)
- Jigsaw activity involving Web 2.0 applications (65 minutes)
- Web site evaluation criteria. (20 minutes)
- Hands-on evaluation of web sites (25 minutes)

Student Activities:

- Complete journal entry.
- Perform internet searches using varying levels of refinement.
- Identify other resources for finding information.
- In groups use the other resources to find relevant information.
- In groups complete jigsaw Web 2.0 activity and presentations.
- Identify evaluation criteria and work in groups to evaluate websites using the rubric.

Teaching/Learning Strategies:

- Journal Entry: List at least three ways in which you currently use the internet.
 - Have students share their responses with their elbow partner.
- Have the students explore by trying out and refining various searching techniques.
- Other resources for finding information
 - Have the students work with their elbow partner to identify at least three other resources (other than search engines) that they use to find information on the internet along with advantages (or disadvantages) over a general search engine.

- Ask students to post their results. Some examples might be:
 - Sites such as Google Maps or Mapquest to get directions or see satellite or street view images of anywhere in the country.
 - Address and telephone number lookup sites such as Switchboard or Yellow Pages to get personal and business information.
 - Sites such as the Internet Movie Database to get information on movies and television shows.
 - Sites such as *Dictionary.com* and *Thesaurus.com* to look up the meaning or spelling of a word or to find a synonym of a word.
 - Encyclopedic sites such as Wikipedia, Encyclopedia Britannica, or How Stuff Works to find an overview of a particular topic.
 - The Wayback Machine which stores snapshots of websites on various dates so that you can “go back in time” to see a site as it used to be.
- Experimentation with these resources
 - Have the students work in groups to use the resources identified above in ways that are relevant to them. For example,
 - Use Google maps and StreetView to find and display where they live or the location of the school.
 - Use Wikipedia and Encyclopedia Britannica to find information on a topic they’re studying in another class. Have them compare the two articles and decide which provides more information.
 - Use the Wayback Machine to view an early version of the school website. Compare how much it has changed from the school’s current website.
- Jigsaw activity involving Web 2.0 applications
 - Divide students into groups to work on each of three different web 2.0 applications. (Depending on the size of the class, more than one group may need to work on each application.) Applications should include a social bookmarking site (delicious.com or stumbleupon.com), a word cloud site (wordle.net) and a list creation site (tadalist.com). Each group should:
 - Set up an account in the application.
 - Explore the site and its features.
 - Prepare a presentation on their site for the remainder of the class.
 - During the student presentations, ensure that the following questions/issues are addressed:
 - What are the differences between ranking based and social bookmarking search engines?
 - Why would you want to create word clouds?
 - What are the advantages of using tadalist? Disadvantages?
 - What issues might there be with creating accounts online? (Lead into a discussion on privacy—what information should be kept private and why? Discuss encryption.)
- Web site evaluation criteria
 - Display or distribute a copy of the front page to <http://www.martinlutherking.org> This is a website which purports to be a “True Historical Examination” of the life of Martin Luther King, Jr., but is, in reality, a hateful site run by a white nationalist organization.

- This particular site is obviously biased. However, it is important to be able to tell when a site is more subtly biased.
- Journal Entry: How might you be able to evaluate a site to determine whether or not it is “good.” What criteria do you use to evaluate them?
- Ask students to volunteer their criteria.
- Hands-on evaluation of web sties
 - An excellent resource with examples of different types of websites appropriate for evaluation can be found at the New Mexico State University Library website: <http://lib.nmsu.edu/instruction/eval.html>.
 - Have the students work in groups and ask them to use the Website Evaluation Rubric to evaluate the example sites found at NMSU.
 - Discuss the results of their evaluations.

Resources:

- The Wayback Machine: <http://www.archive.org>
- Google Maps (including StreetView): <http://maps.google.com>
- Wikipedia: <http://www.wikipedia.org>
- Encyclopedia Britannica: <http://www.britannica.com>
- Mapquest: <http://www.mapquest.com>
- Internet Movie Database: <http://www.imdb.com>
- Switchboard: <http://www.switchboard.com>
- Yellow Pages: <http://www.yellowpages.com>
- How Stuff Works: <http://www.howstuffworks.com>
- <http://www.delicious.com>
- <http://www.stumbleupon.com>
- <http://wordle.net>
- <http://tadalist.com>
- The white nationalist site on Martin Luther King, Jr.:
<http://www.martinlutherking.org>
- Website Evaluation Rubric

Website Evaluation Rubric			
Authority			
Is the author identified?	Yes	No	Unsure
Does the author have appropriate qualifications with respect to the information being presented?	Yes	No	Unsure
Purpose			
Is the purpose to inform or give factual information?	Yes	No	Unsure
Coverage			
Is the information primary or secondary in nature?	Yes	No	Unsure
Is the information presented comparable to information on the same topic presented by other sites?	Yes	No	Unsure
Accuracy			
Is the information free of factual errors?	Yes	No	Unsure
Do the conclusions appear to be well-reasoned and supported by the facts presented?	Yes	No	Unsure
Is the information properly referenced?	Yes	No	Unsure
Objectivity			
Is the information free from obvious bias?	Yes	No	Unsure
Does the author avoid the use of emotional or inflammatory language?	Yes	No	Unsure
Does the author avoid trying to sell something or persuade the reader of a particular viewpoint?	Yes	No	Unsure
Currency			
Is the information up-to-date?	Yes	No	Unsure
Are there creation and revision dates?	Yes	No	Unsure
Appearance			
Does the site have a professional appearance?	Yes	No	Unsure
Does it use proper grammar, spelling, and composition?	Yes	No	Unsure

Instructional Day: 10

Topic Description: The use of computers for communications and the impact this has had on society is discussed.

Objectives:

The student will be able to:

- Explain how computers are used for communications.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Identification of communications mechanisms (10 minutes)
- Impact of changes to communications on society (40 minutes)

Student Activities:

- Complete journal entry.
- Identify communications mechanisms.
- In groups, students discuss the impact of changes to communications on society.
- Groups share a summary of their discussions with the class.

Teaching/Learning Strategies:

- Journal Entry: List as many computer-based communications mechanisms as you can.
- Identification of communications mechanisms
 - Volunteers provide examples from their journal entry. Post the responses.
 - Prompt students as necessary with examples.
 - Internet-based communication (email, chat, facebook, Internet telephony)
 - Telephone-based communication (cell phones, texting, “landline” telephone service)
 - News and information “on demand”
- Impact of changes to communications on society
 - Divide class into groups of 3-4. Assign each group 1 or 2 of the mechanisms on the list depending on the size of the class and number of items on the list. Ask the groups to do the following:
 - Imagine life without some or all of the computer-based communications mechanisms that we now take for granted.
 - List some of the consequences of an absence of technology (for example, without cell phones, the ability to instantly reach anyone goes away).
 - Based on these consequences, draw conclusions about the impact of the presence of the communications mechanism. (For example, if the absence of cell phones means the absence of the ability to instantly contact anyone, then the presence of cell phones

means that we now have the ability to instantly contact anyone. One conclusion we can draw is that we have less privacy than we used to.) Consider each of the following broad categories of societal change:

- Privacy
- Safety
- Globalization
- Connectivity (keeping in touch with people)
- Permanence of historical information
- Discuss whether these consequences have a positive or negative impact on society and, if negative, how these consequences can be minimized.
- Groups share a summary of their discussions with the class.
 - Challenge the students to predict what communications will be like in 5 years, 10 years, and 25 years.

Resources:

- No additional resources needed

Instructional Days: 11-14

Topic Description: In this lesson, students learn how computers can be used as a tool for visualizing data, modeling and design, and art in the context of culturally situated design tools. Connections between the design of the tools and mathematics will be explored.

Objectives:

The student will be able to:

- Explain how computers can be used as tools for visualizing data, modeling and design, and art.
- Identify mathematical connections in the output of the tools.
- Edit an image using Photoshop Express.

Outline of the Lesson:

- Research on the cultural background associated with the design tool. (25 minutes)
- Design tool tutorials (30 minutes)
- Creation of designs using the design tools (65 minutes)
- Online presentation on how to get started using Photoshop Express (15 minutes)
- Design editing (30 minutes)
- Preparation of presentations (40 minutes)
- Group presentations (15 minutes)

Student Activities:

- Groups do research on the cultural background information associated with the design tools they are assigned.
- Groups prepare and deliver brief presentations on the cultural aspects of their design tools.
- Students complete design tool tutorials.
- Groups create designs using the design tools.
- Watch an online presentation on how to get started using Photoshop Express.
- Edit images created with the design tools.
- Groups prepare presentations.
- Groups deliver presentations.

Teaching/Learning Strategies:

- Post the possible design tools:
 - Virtual Bead Loom
 - Pacific Northwest Basket Weaver
 - Navajo Rug Weaver
- Display the first page of each tool in order to give students an idea of what each does.
(<http://csdt.rpi.edu>)

- Students divide into groups to work on the tool of their choice. Group sizes will depend on the size of the class. You may need to have more than one group per tool.
- Each member of the group should go through the entire cultural background section individually.
 - Answer any questions posed in the section in their journal.
 - Look for and write down the mathematical connections.
- All group members discuss the section.
 - Resolve answers to questions and mathematical connections.
- Each member of the group completes the tutorial.
 - Students should go through the tutorial at their own pace, but discuss with other member as questions arise. (Note: The bead loom tutorial is online; the other two are not. The print versions included here have been adapted from the bead loom tutorial.)
 - Encourage students to record in their journal points that they want to remember.
- Groups create designs using the design tool software.
 - Each person should choose one of the goal pictures for practice and discuss any issues with the other group members.
 - Groups decide whether they want to create one design as a group or have multiple designs for their presentation.
 - Groups work on design/designs—these should be their own creations rather than a mimic of one of the preloaded designs.
- Edit designs with Photoshop Express.
 - Have students watch the online tutorial and create an account.
 - Edit the design.
- Prepare presentations to include:
 - Culture
 - Math connections
 - Demo of software
 - Display of designs
- Groups deliver presentations.

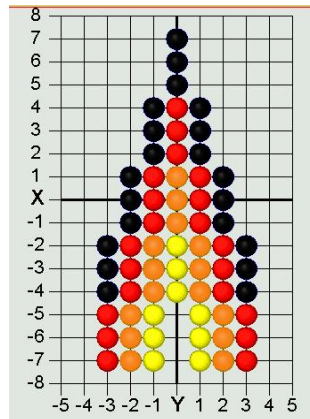
Resources:

- Culturally Situated Design Tools—<http://www.csdt.rpi.edu> (site and adaptations of tutorials courtesy Ron Eglash)
- Virtual Bead Loom Tutorial
- Pacific Northwest Basket Weaver Tutorial
- Navajo Rug Weaver Tutorial
- Culturally Situated Design Tools Project Sample Rubric
- <http://www.photoshopexpresstechniques.com>

Virtual Bead Loom Tutorial

Tutorial 1

The Virtual Bead Loom simulates the same grid pattern as the traditional bead loom. Users place colored circles in columns (the Y-axis) and rows (the X-axis).



There are several tools for placing beads on the virtual loom. In each case you use the "tab" key or the mouse to move your cursor to the field for entering the coordinates, then you enter them, and then press the button for the shape tool. The point tool places a single bead:

x = 2	y = 2
-------	-------

The line tool places lines of beads. You specify the two endpoints of the line. Diagonal lines tend to be jagged, but resizing the grid can help that (see "Options menu" on next page).

x1 = -3	y1 = -3
x2 = 3	y2 = 3

The rectangle tool fills in a rectangle of beads. You specify two vertices (lower right and upper left). The rectangles of this tool are always aligned with the axes.

x1 = -3	y1 = -3
x2 = 3	y2 = 3

Tutorial 2

The triangle tool fills in a triangle of beads. You specify the three vertices.

x1 =	-3	y1 =	1
x2 =	0	y2 =	4
x3 =	3	y3 =	1

The iterative triangle tool: Our first triangle tool made jagged edges, while traditional beadwork has beautifully regular edges. We interviewed some native beadworkers, and found that their algorithms were iterative. The triangle iteration tool reflects this tradition of indigenous mathematics. For example, the triangle in the beadwork at the top of this page was made by adding one bead on each side of the row, every three rows, as you go in the -Y direction.

Starting at x=0 y=0

after every 3 rows

add 1 to both ends

for 9 rows in total

in the ☐ +y ☒ -y direction.

☐ -x ☒ +x

- "Direction" -- determines in which direction your rows will accumulate
- Starting at X,Y -- that is the center of the starting row.
- "After every ___ rows" -- lets you determine how many rows you go through before adding more beads to the end.
- "Add ___ to both ends" -- the number of beads that will be added on each side of the center each time.
- "For ___ rows in total" -- how many rows you will bead in this triangle.

Note that this tool has two colors -- some traditional bead work shifts color in each iteration. This allows you to select the starting color and ending color; the software does the shifting for you.

Tutorial 3

There are also controls that apply to all the tools. "Clear" deletes everything. Normally "Create" is selected, so that your tools will fill their specified shape with beads. "Remove" will erase all beads in the specified shape, so if you make an error use "undo" not "remove." The color button allows you to select the bead color. Clicking on the little square in the upper right of the screen will give you a list of all the colors you have selected so far. The "Save" menu allows you to save the work on your hard drive and edit the design later. Make sure your file name is only letters, not spaces or numbers, and that you go back to the same computer when you want to edit your work.

The "Options" menu allows you to resize the grid smaller or larger -- maximum size is 150 by 150. You can also change the location of the coordinate values, hide the grid, or create a title or notes about your design. You can also switch to Wampum beads, using either traditional 1X2 Wampum or a 1X1 Wampum (which is easier for math teaching--special thanks to Joyce Lewis of the Onondaga Nation for that concept!).

Printing: after you have your design completed, do a screen capture. In windows you can do that by pressing the "print screen" button on your keyboard, usually located at the upper right above the F10 key. On a Macintosh press shift + apple + 3 at the same time (also shift + apple +4 to select just a portion). That screen capture will save an image of the entire screen to your clipboard. You can then paste the clipboard image into a blank canvas in Word, Photoshop, Imaging (comes free in the "Accessories" folder in Windows) or other image editor.



Tutorial 1

The Virtual Basket Weaver simulates the same grid pattern as the traditional basket weaving loom. Users place colored circles in columns (the Y-axis) and rows (the X-axis).



There are several tools for placing wefts on the virtual loom. In each case you use the "tab" key or the mouse to move your cursor to the field for entering the coordinates, then you enter them, and then press the button for the shape tool. The point tool places a single weft:

x =	2	y =	2
-----	---	-----	---

The line tool places lines of wefts. You specify the two endpoints of the line. Diagonal lines tend to be jagged, but resizing the grid can help that (see "Options menu" on next page).

x1 =	-3	y1 =	-3
x2 =	3	y2 =	3

The rectangle tool fills in a rectangle of wefts. You specify two vertices (lower right and upper left). The rectangles of this tool are always aligned with the axes.

x1 =	-3	y1 =	-3
x2 =	3	y2 =	3

Tutorial 2

The triangle tool fills in a triangle of wefts. You specify the three vertices.

x1 =	-3	y1 =	1
x2 =	0	y2 =	4
x3 =	3	y3 =	1

The iterative triangle tool: Our first triangle tool made jagged edges, while traditional basket work has beautifully regular edges. The triangle iteration tool reflects the tradition of indigenous mathematics. For example, a triangle can be made by adding one bead on each side of the row, every three rows, as you go in the -Y direction.

Starting at x=0 y=0

after every 3 rows

add 1 to both ends

for 9 rows in total

in the ☐ -x ☐ +x direction.

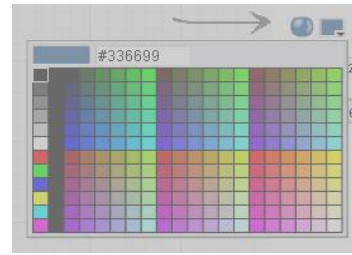
☒ -y

- "Direction" -- determines in which direction your rows will accumulate
- Starting at X,Y -- that is the center of the starting row.
- "After every ___ rows" -- lets you determine how many rows you go through before adding more wefts to the end.
- "Add ___ to both ends" -- the number of wefts that will be added on each side of the center each time.
- "For ___ rows in total" -- how many rows you will weft in this triangle.

Note that this tool has two colors -- some traditional basket work shifts color in each iteration. This allows you to select the starting color and ending color; the software does the shifting for you.

Tutorial 3

There are also controls that apply to all the tools. "Clear" deletes everything. Normally "Create" is selected, so that your tools will fill their specified shape with wefts. "Remove" will erase all wefts in the specified shape, so if you make an error use "undo" not "remove." The color button allows you to select the weft color. Clicking on the little square in the upper right of the screen will give you a list of all the colors you have selected so far. The "Save" menu allows you to save the work on your hard drive and edit the design later. Make sure your file name is only letters, not spaces or numbers, and that you go back to the same computer when you want to edit your work.



The "Options" menu allows you to hide the grid, or create a title or notes about your design.

Printing: after you have your design completed, do a screen capture. In windows you can do that by pressing the "print screen" button on your keyboard, usually located at the upper right above the F10 key. On a Macintosh press shift + apple + 3 at the same time (also shift + apple +4 to select just a portion). That screen capture will save an image of the entire screen to your clipboard. You can then paste the clipboard image into a blank canvas in Word, Photoshop, Imaging (comes free in the "Accessories" folder in Windows) or other image editor.

Navajo Rug Weaver Tutorial

Tutorial 1

The Virtual Rug Weaver simulates the same grid pattern as the traditional rug loom. Users place colored circles in columns (the Y-axis) and rows (the X-axis).



There are several tools for placing wefts on the virtual loom. In each case you use the "tab" key or the mouse to move your cursor to the field for entering the coordinates, then you enter them, and then press the button for the shape tool. The point tool places a single weft:

x =	2	y =	2
-----	---	-----	---

The line tool places lines of wefts. You specify the two endpoints of the line. Diagonal lines tend to be jagged, but resizing the grid can help that (see "Options menu" on next page).

x1 =	-3	y1 =	-3
x2 =	3	y2 =	3

The rectangle tool fills in a rectangle of wefts. You specify two vertices (lower right and upper left). The rectangles of this tool are always aligned with the axes.

x1 =	-3	y1 =	-3
x2 =	3	y2 =	3

Tutorial 2

The triangle tool fills in a triangle of wefts. You specify the three vertices.

x1 =	-3	y1 =	1
x2 =	0	y2 =	4
x3 =	3	y3 =	1

The iterative triangle tool: Our first triangle tool made jagged edges, while traditional rug work has beautifully regular edges. The triangle iteration tool reflects the tradition of indigenous mathematics. For example, a triangle can be made by adding one bead on each side of the row, every three rows, as you go in the -Y direction.

Starting at x=0 y=0


after every 3 rows

add 1 to both ends

for 9 rows in total

in the ☐ -x ☐ +x direction.

☒ -y

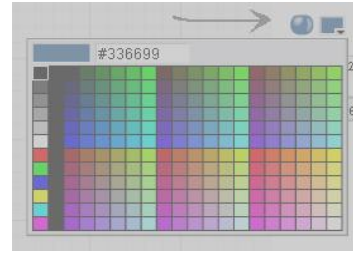


- "Direction" -- determines in which direction your rows will accumulate
- Starting at X,Y -- that is the center of the starting row.
- "After every ___ rows" -- lets you determine how many rows you go through before adding more wefts to the end.
- "Add ___ to both ends" -- the number of wefts that will be added on each side of the center each time.
- "For ___ rows in total" -- how many rows you will weft in this triangle.

Note that this tool has two colors -- some traditional rug work shifts color in each iteration. This allows you to select the starting color and ending color; the software does the shifting for you.

Tutorial 3

There are also controls that apply to all the tools. "Clear" deletes everything. Normally "Create" is selected, so that your tools will fill their specified shape with wefts. "Remove" will erase all wefts in the specified shape, so if you make an error use "undo" not "remove." The color button allows you to select the weft color. Clicking on the little square in the upper right of the screen will give you a list of all the colors you have selected so far. The "Save" menu allows you to save the work on your hard drive and edit the design later. Make sure your file name is only letters, not spaces or numbers, and that you go back to the same computer when you want to edit your work.



The "Options" menu allows you to hide the grid, or create a title or notes about your design.

Printing: after you have your design completed, do a screen capture. In windows you can do that by pressing the "print screen" button on your keyboard, usually located at the upper right above the F10 key. On a Macintosh press shift + apple + 3 at the same time (also shift + apple +4 to select just a portion). That screen capture will save an image of the entire screen to your clipboard. You can then paste the clipboard image into a blank canvas in Word, Photoshop, Imaging (comes free in the "Accessories" folder in Windows) or other image editor.

Culturally Situated Design Tools Project Sample Rubric

Group Members Names: (up to 4)

Do you have?	Points Possible	Yes	No	Points Earned
Product				
Title with group members' names	10			
Information related to the culture	10			
Mathematical connections	10			
Demo of software	5			
Display of designs	5			
Design is original	10			
Design has been edited with Photoshop Express	10			
Visuals of your choices (pictures or video of choices)	10			
Presentation				
Present your project	15			
Present all required parts of project	15			
Extra Credit				
Project exhibits creativity above and beyond	Up to 10			
TOTAL:	100			

Instructional Day: 15

Topic Description: Students participate in a field trip to a local business or host a guest speaker.

Objectives:

The student will be able to:

- List and describe a variety of uses for computers.

Outline of the Lesson:

- Field trip or guest speaker (55 minutes)

Student Activities:

- Participate in the field trip or listen to the guest speaker.

Teaching/Learning Strategies:

- A field trip may not be able to be arranged this early in the year, but at some point the teacher should arrange for students to visit a local business that uses computers or invite a guest speaker.
- Encourage students to ask questions.

Resources:

- No additional resources necessary

Instructional Days: 16-17

Topic Description: This lesson introduces the concept of a computer program within the context of a set of instructions for completing a common activity.

Objectives:

The student will be able to:

- Explain the concept of a computer program.

Outline of the Lesson:

- Following directions (55 minutes)
- Designing a program (15 minutes)
- Running a program (25 minutes)
- Being more precise with instructions (15 minutes)

Student Activities:

- Complete the Following Directions quiz.
- Complete the Drawing Pictures Activity.
- Write the instructions for making a peanut butter and jelly sandwich.

Teaching/Learning Strategies:

- Following directions
 - Distribute copies of Following Directions Quiz to each student face down in front of them. Each student should have a blank piece of paper and a pencil as well.
 - Give the students five minutes to do the quiz. Make note of how many students stand up and shout “hooray.”
 - Collect the papers when time has expired.
 - Point out that a perfect paper is one which has only the word “December” written in the top left corner. (The directions said to read all parts of the test before doing anything and step 14 says to only complete step #3.)
 - Give students about 10 minutes to complete the Drawing Pictures Activity.
 - Ask volunteers to show their pictures and explain why they drew the pictures as they did.
 - After the first volunteer, ask if someone drew it differently.
 - Ask the students what following directions has to do with computers. Prompt them as necessary that a computer follows a specific set of directions (or instructions) called a computer program and must follow all of the directions precisely.
- Designing a program

- Ask the students to write down a set of instructions for a computer to make a peanut butter and jelly sandwich. Give them 5-10 minutes to write down these instructions.
- Collect the instructions.
- Running a program
 - Take out the bread, peanut butter, jelly, and knife and put them on your desk. Pick a set of instructions for making a sandwich (best to pick one which is not too detailed).
 - Read each instruction and carry it out—literally. For example, if the first instruction is “put the peanut butter on the bread,” take the jar of peanut butter and put it on the loaf of bread. If an instruction says to “spread the peanut butter on the bread,” use your fingers rather than a knife. If an instruction says to “cut the sandwich in half,” be creative and cut it between the two slices of bread. In other words, your goal is to show that instructions need to be very precise.
 - Repeat the process with another set of instructions.
- Being more precise with instructions
 - Clearly, no matter how precise they tried to be, the instructions for making a peanut butter and jelly sandwich were open to interpretation. Ask the students to brainstorm how we could overcome this problem so that a computer could follow the instructions and make a perfect sandwich each time.
 - Guide the students toward the idea that we need a better “language” than English for describing the instructions. This is, in fact, the idea behind many computer programs. There is a limited set of instructions which define very precisely what the computer does. For example, we can have a computer turn on a “dot” of a specific color in a specific location on the screen. By having the computer turn on many different dots in different colors, we can have the computer draw a picture. Note though that we don’t have an instruction for the computer to “draw a picture of a house” as that’s much too general and too open for interpretation.

Resources:

- <http://www.justriddlesandmore.com/direct.html>
The basis for the “following directions” quiz (the quiz was modified slightly.)
- Following Directions Quiz
- Drawing Pictures Activity
- Bread, peanut butter, jelly, and a knife.

Following Directions Quiz

Directions: You have a 5 minute time limit to complete the parts of this test. Carefully read all of the parts of the test before doing anything. In order to ensure the accuracy of this exam, you should not use more than the allotted time of 5 minutes. Good Luck!!

You may begin now!!

1. Write today's date--month-day-year in the top right hand corner of your test paper.
2. Write the answer to the following multiplication problem directly underneath the date on your test paper--6 X 5 = ?
3. Write the name of the month that begins with the letter "D" in the top left hand corner of your test paper.
4. Add 15 to the answer you got in part #2, and write this new total directly underneath your answer for part #3.
5. In the lower left hand corner of your test paper, write the names of your favorite singer and your favorite group.
6. Just above your answer to part #5, write "This test is very easy."
7. In the lower right hand corner of your test paper, draw a rectangle and inside the rectangle draw a five pointed star. The size of these drawings is not important.
8. Directly above your answer to part #7, draw a row of three small circles. Once again, size is not important.
9. Write the name of the first president of the United States on the back of your test paper anywhere you choose. If you don't know who this is, write your own name instead.
10. Write the name of any country that begins with the letter "I" directly underneath you answer to part #2.
11. Stand up, shout "hooray!", and sit down.
12. Take the number of dwarfs in the Snow White story and add it to the number of bears in the Goldilocks story. Divide by 2. Write this total in the approximate center of your test paper.
13. Think of a number between 1 and 50. Double that number. Add 20. Add 6. Subtract 17. Subtract 9. Divide by 2. Write this number on your test paper directly underneath your answer to part #11.
14. Now that you have carefully read all of the parts so far, and you have not carried out any of the actual work, skip the next 2 parts and go back and only complete part #3.
15. The name of the first president of the United States is George Washington. He was president from 1789 until 1797. Add the 2 dates together to see if the total is less than 5000.
16. You should not be reading the end of the exam before the beginning of the exam, but now that you are here, you have just wasted some of the time you may need to complete the test.

Drawing Pictures Activity

1. Draw a picture of a house in the middle of the page.
2. Draw a picture of a stick figure father, mother and daughter.
3. Draw a picture of a mustang next to the house.
4. Draw a picture of the sun in the sky.

Instructional Days: 18-20

Topic Description: The question “What is intelligence?” is addressed through discussion of the differences between humans and computers. Various models of machine learning are investigated along with the concept of natural language understanding.

Objectives:

The student will be able to:

- Explain the idea of intelligence – especially as it relates to computers.
- Explain what it means for a machine to “learn”.
- Discuss whether computers are intelligent or whether they only behave intelligently.

Outline of the Lesson:

- Journal Entry (15 minutes)
- Differentiation between humans and computers (95 minutes)
- A simple model of machine learning (55 minutes)

Student Activities:

- Complete journal entry.
- Complete CS Unplugged Activity 20: Conversations with Computers—The Turing Test.
- Interact with web-based chatterbots (Part I of The Computer Intelligence Activity).
- In groups, play several rounds of a guessing game (Part II of The Computer Intelligence Activity).

Teaching/Learning Strategies:

- Journal Entry: What is intelligence? Are computers intelligent? Why or why not?
 - Volunteers share their responses.
- Differentiating between humans and computers.
 - CS Unplugged Activity 20: Conversations with Computers—The Turing Test
 - This activity can be downloaded from <http://csunplugged.com> From the menu, click on Activities, click on Turing Test, and then download the pdf for Activity 20. Note there are many additional resources listed that you may wish to explore.
 - It will be helpful for you to read through the entire activity before beginning it with your students. In addition to the explanation of the activity, it provides good background information that you will want to ensure is part of the discussion you have with students.
 - Based on the directions under “What to Do” (p. 214), assign and explain roles to 4 students.
 - Follow the remaining directions under “What to Do” (p.214-215).
 - Have students complete Part I of Computer Intelligence Activity.

- Assign each pair of students (students work with their elbow partner) two of the questions from the Turing Test Activity.
- Discuss the results.
- A simple model of machine learning.
 - Have students complete Part II of the Computer Intelligence Activity.
 - Assign students to groups of 3 or 4 and assign each group 2 of the games in the activity.
 - Discuss the results.

Resources:

- Computer Science Unplugged Activity 20: Conversations with Computers—The Turing Test (<http://csunplugged.com>), pp. 213-226
- Computer Science Unplugged Activity 20: Conversations with Computers—The Turing Test, p. 225—questions (one copy for each pair of students)
- Computer Science Unplugged Activity 20: Conversations with Computers—The Turing Test, p. 226—answers (one copy to post or display)
- Computer Intelligence Activity

Computer Intelligence Activity

Part I

A program passes [The Turing Test](https://en.wikipedia.org/wiki/Turing_Test) (en.wikipedia.org/wiki/Turing_Test) if a person can have a conversation with both it and a person and not be able to tell which one is the computer.

Try each of these chatterbots with the questions you were assigned.

1. Try to chat with [Eliza](http://ai.ijs.si/eliza/eliza.htm) (ai.ijs.si/eliza/eliza.htm). How realistic is she? Would she pass the Turing Test?
2. Try to chat with [Athena](http://Athena.blueinfos.com) (Athena.blueinfos.com). How realistic is she? Would she pass the Turing Test?
3. Try to chat with [Friend4U](http://virtualentities.com/friend4u) (virtualentities.com/friend4u). How realistic is she? Would she pass the Turing Test?
4. Try to chat with [InteliAvatar](http://inteliwise.com) (inteliwise.com). How realistic is she? Would she pass the Turing Test?
5. Which of the above chatterbots was the most like a real person?
6. What is the [Chatterbox Challenge](http://chatterboxchallenge.com) (chatterboxchallenge.com)?

Part II

1. Go to 20q.net. Choose your language (Think in American is recommended). Choose one of the games from the bottom that was assigned to your group. You are supposed to think of something in that category and answer the computer's questions by clicking them. The computer will try to guess what you chose in 20 questions or less. Play the game several times addressing each of the following:
 - Pick an item and see how many questions are required.
 - Choose the same item and see if you can make it require more questions
 - Repeat this with another item.
 - How intelligent is this? Would this pass the Turing Test?

Play the second game you were assigned and repeat the process above.

2. The Turing test is a person checking to see if it is talking to a computer. Can you think of any occasions that a computer might want to know if it is talking to *another* computer or a real life person?

Instructional Days: 21-22

Topic Description: Complete final unit project.

Objectives:

The student will be able to:

- Incorporate the objectives of the unit into the final project.

Outline of the Lesson:

- Explanation of final project (10 minutes)
- Research and development of final project (85 minutes)
- Gallery walk of projects (15 minutes)

Student Activities:

- Student teams research and complete final projects.
- Participate in a gallery walk of final projects.

Teaching/Learning Strategies:

- Distribute final project information and rubric.
 - Set up teams and answer initial questions.
 - Note that the scenario can be revised to meet the needs of your students.
 - An example of how tools might be used might be to illustrate the impact computers have on our lives with a word cloud from several articles on the internet supplemented by images edited in Photoshop. There are many other possibilities for students to be creative in their presentations.
 - This project might be a good sample for a student portfolio as it could be extended later in the course as students have more knowledge and have facility with more tools.
 - Circulate the room and monitor student work.
- Gallery walk of projects
 - Facilitate by providing an order in which students should visit the various projects.
 - Wrap up with a discussion of the answers to the questions in the project.

Resources:

- Final Project
- Final Project Sample Rubric

Final Project

In your group you will create a response to the following scenario that answers several questions about computers and computing and utilizes some of the tools that you learned about in this unit.

Scenario:

The school board has decided to cut computer classes starting next year because not enough students signed up for the courses this year. Create a recruitment campaign to encourage students to enroll in computer science classes. Include an explanation of the need for learning about computer science that answers the following questions.

- What is a computer/computing?
- How do computers and the internet work?
- How do computers impact our lives (individually, socially, economically, culturally, etc.)?

Create a visual representation to accompany your written responses that makes use of at least two of the tools that you learned about in this unit. Depending on the tools you use, your written responses can be included within the visual.

Final Project Sample Rubric

Group Members Names: (up to 4)

Do you have?	Points Possible	Yes	No	Points Earned
Overall explanation of the need to study computer science?	10			
Answers to questions				
What is a computer/computing?	5			
How do computers and the internet work?				
Have at least three examples	15			
How do computers impact our lives				
Have at least three examples	15			
Each example has a different kind of reason (social, economic, cultural, etc.)	10			
Additional examples				
Of how computers and the internet work	5			
Of how computers impact our lives	5			
Visual representation				
Uses at least 2 tools	20			
Representation supports the answers to the questions	15			
TOTAL:	100			

Unit 2:

Problem Solving



Introduction

In order for students to become “computational thinkers” they need experience with solving a wide range of problems and the opportunity to experiment with a variety of solution strategies. This unit begins with an introduction to the problem solving process. Students are asked to solve problems with which they may not be familiar by planning a strategy, designing and producing solutions, and then reflecting on their solutions and strategies.

Throughout the unit the emphasis should be on the process rather than the solution. Most of the world’s problems today do not have single simple solutions. In order to contribute effectively to the solution of these problems, students need to be comfortable in a collaborative environment where multiple approaches are valued and encouraged and where failure is seen as part of the process toward solution. Students must learn to think abstractly and apply known algorithms where appropriate, but also create new algorithms that can be applied to complex problems.

As students reflect on their solution processes and solutions and share those reflections with their peers, it is an opportunity to pull out instances where one strategy might be preferred over another and problems for which there are “standard” solutions versus those where there are many possible solutions.

Many of the problems presented have a mathematical basis and can serve to provide connections between mathematics and computer science. Common computer science topics such as searching, sorting, and graphing are introduced. Although programming the solutions to many of these problems is beyond the scope of this course, students will gain a basic understanding of the algorithms and be able to analyze them. In particular, it is important to emphasize that the models used for solving computational problems are the underpinnings of computer science and as such remain largely the same even as we add new tools and languages.

A key point of emphasis throughout the unit is the connection between the solution “process” and the discussion toward the end of unit 1 related to how computers are programmed. It is also important to emphasize that not all problems are easily solved by computers.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1	Introduce the four steps of the problem solving process.
2-4	Apply the problem solving process. Use different strategies to plan and carry out the plan to solve several problems.
5-7	Reinforce the four steps of the problems solving process.
8-9	Count in the binary number system.
10-11	Convert between binary and decimal numbers in the context of topics that are important to computer science.
12-13	Introduce the linear and binary search algorithms.
14-16	Explore sorted and unsorted lists and various sorting algorithms.
17-18	Introduce minimal spanning trees and how graphs can be used to help solve problems.
19-22	Final projects and presentations

Daily Lesson Plans

Instructional Day: 1

Topic Description:

This lesson introduces the four main phases of the problem-solving process as defined by G. Polya in [How to Solve It](#).

Objectives:

The students will be able to:

- Name and explain the steps in the problem-solving process.
- Solve a problem by applying the problem-solving process.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Candy bar activity (25 minutes)
- Discussion of solutions (10 minutes)
- Introduction of the steps in the problem-solving process (15 minutes)

Student Activities:

- Complete journal entry.
- In groups, participate in the candy bar activity.
- Participate in discussion of solutions.
- Reflect on the candy bar activity as it relates to the problem-solving process.

Teaching/Learning Strategies:

- Journal Entry: What are the steps you use to solve a problem?
- Candy bar activity
 - Divide the students into groups of 2 or 3. Give each group a candy bar.
 - Explain that their task is to determine how many "breaks" it will take to break the candy bar into 12 equal pieces. One break of one piece of the candy bar will result in that one piece being divided into two pieces. Demonstrate a "break" by breaking the bar into two pieces. Then stack the two pieces together and break or cut the two pieces into four.
 - At this point, have each student write in their journal the number of breaks they think it will take to break the bar into 12 equal pieces. This should be done without talking to their partner or group members.
 - Working together with their partner or group, have the students discuss and then write their plan for solving the problem. They may revise their guess at this point.
 - Once this is completed, the students should implement the plan by opening the candy, breaking the candy, and counting the number of breaks it takes to get 12 equal pieces.
- Discussion of solutions

- Choose a group to present their plan to the class.
 - Sample questions to ask—Was your guess correct? What process did you use to come up with your guess? Did working with your group and creating your plan change your guess?? How many breaks did it take (11 is the answer)? Did your plan work?
- How do the steps they used match what they wrote in their journal?
- Introduction to the steps in the problem-solving process
 - How do the steps they used relate to the “formal” steps of the problem solving process?
 - Understand the problem – read or listen to the problem statement.
 - Make a plan to solve the problem – use pictures, charts, graphs, systematic lists, objects, or act out the solution to help you devise a plan to solve the problem
 - In Computer Science we call this plan an **algorithm**.
 - Carry out the plan – once the plan is conceived and understood, follow the plan. If you have planned well, this is the easy part.
 - Review and reflect on how the problem was solved – Once the problem is solved, reflect on the plan that was used.
- Extend breaking the candy into N pieces.
 - Post the Number of Pieces/Number of Breaks Chart (without solutions), including N and have students give you the # of breaks needed for each number of pieces.
 - Ask questions as you go through the chart that lead students to the following understandings.
 - One problem-solving strategy used in solving a problem is to solve a problem for specific values, find the pattern and then generalize the solution. In this case, they are generalizing the solution for an unknown positive number of pieces.
- Reflections on the candy bar problem: Ask the students to reflect on the candy bar problem. Why is this problem an important problem to solve for: a carpenter, a chef, a teacher?

Resources:

- Polya, G. How to Solve It. 2nd. Princeton, NJ: Princeton University Press, 2004.
Candy bar problem suggested by Dr. Manuel Blum, Carnegie Mellon University
- Candy bars for student groups to use
- Number of Pieces/Number of Breaks Chart

Number of Pieces/Number of Breaks Chart

Number of Pieces	Number of Breaks
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11
N	<i>N-1</i>

Instructional Days: 2-4**Topic Description:**

Students will apply different strategies to help them make a plan and carry out the plan to solve several problems. These strategies may include (but are not limited to): draw a diagram or picture, make systematic lists, divide and conquer, find the pattern, and guess and check.

Objectives:

The students will be able to:

- Solve a problem by applying the problem-solving process.
- Express a solution using standard design tools.
- Determine if a given solution successfully solves a stated problem.

Outline of the Lesson:

- Handshake problem #1 and Fence Post problem (20 minutes)
- Explanation of solutions (15 minutes)
- Handshake problem #2 and reflections (75 minutes)
- Presentations of Handshake problem (40 minutes)
- Discussion of reflections (15 minutes)

Student Activities:

- Work individually on Handshake problem #1 and the Fence Post problem.
- Volunteers present solutions to problems.
- Work in groups to complete Handshake problem #2.
- Groups give presentations of their problem solutions.
- Discuss reflections on the process.

Teaching/Learning Strategies:

- Handshake problem #1 and Fence Post problem
 - Students work individually on Handshake problem #1 and the Fence Post problem.
- Explanation of solutions
 - Have some students volunteer their solutions to the problems.
 - Reinforce each step of the problem-solving process by asking questions similar to those from the candy bar problem. You want students to understand that
 - Diagrams can be very useful in problems like this to look at a smaller version of the problem before trying to solve for N.
 - The fencepost problem is a variation of the candy bar problem or the handshake problem.
- Handshake problem #2 and Reflections

- In groups of 3 or 4, have students discuss, plan, execute, and reflect on Handshake problem #2. Students should follow the directions given in the activity document and write their group's thoughts on paper.
- Encourage students to make drawings or charts and/or act out the solution. Chart paper can be given to students to display pictures, charts, or graphs. Their job is to explain the process and the solution so that everyone understands.
- Student Presentations
 - Each group should be given about 5-10 minutes (depending on the size of the class) to present their plan and solution to the class. Be sure the students show all 4 steps in the problem-solving process.
 -
- Discussion of reflections

Resources:

- Polya, G. How to Solve It. 2nd. Princeton, NJ: Princeton University Press, 2004.
- Handshake and Fencepost Activity
- Handshake Activity #2 Sample Solution

Handshake and Fencepost Activity

For each problem, complete the following information.

Understanding the problem:

- What data or information is known?
- What is unknown?
- What are the conditions?

Plan the solution: Show your plan for solving this problem.

Carry out the plan: Using your plan, show your work and your solution.

Review and discuss your solution: Reflect on your solution.

Complete problems #1 and #2 individually.

1. Handshake Problem #1: Assume there are 20 people in a room, including you. You must shake hands with everyone else in the room. How many hands will you shake? If there are N (where $N > 0$) people in the room, how many hands will you shake?
2. Fence Post Problem: You need to build one side of a fence that is 12 yards long. This fence will be built with fence posts and rails that connect one fence post to another. If each fence post is 1 yard away from the next fence post, how many fence posts will be needed for this side of the fence? How many fence posts will be needed for a side of a fence that is N (where $N > 0$) yards long?

Read and begin planning your solution for problems #3 and #4. These problems will be completed in class tomorrow with your group. Each group will present their solutions to the class.

3. Handshake Problem #2: Assume there are 10 people in a room, including you. Each person in the room must shake hands one time, and only time, with all the other people in the room. How many handshakes will occur? If there are 20 people in the room, how many handshakes will occur? If there are N (where $N > 0$) people in the room, how many handshakes will occur?
4. Reflections: Why are problems like these important to learn how to solve? How could this type of solution be of benefit to a carpenter, a chef, a teacher?

Handshake Activity #2 Sample Solution

The sample solution is only one possibility. Student groups may have a wide variety of strategies. Ask questions that probe their understanding of the steps of the problem-solving process they used.

Understanding the problem:

- What data or information is known? *There are 10 people or N people in the room*
- What is unknown? *Total number of handshakes*
- What are the conditions? *Each person must shake hands only one time with all others in the room. All of the handshakes must be added together.*

Plan the solution: *A sample plan could be to describe the plan in words or use a chart or draw a picture and then act it out.*

Have the people line up in the room. The first person in the line walks down the line and shakes hands with all of the people in the line and then leaves the room. Count the number of handshakes and add to the total.

The next person in line walks down the line and shakes hands with all of the people left in the line and then leaves the room. Count the number of handshakes and add to the total.

This continues until there are only 2 people left. They shake hands and leave together. Increase the total by one.

Once the answer is known for 10 people, look for a pattern. Try the process for 5 people, 2 people. See if the pattern holds.

Carry out the plan: Using your plan, show your work and your solution.

Person	Shakes Hands with # of people left in line
A	9
B	8
C	7
D	6
E	5
F	4
G	3
H	2
I	1
J	0 (last person in line— no one left to shake hands with)

Now add up the number of handshakes: $9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0 = 45$

For 10 people, the answer is the sum of the numbers from 1 to 9, which is 45. 9 is $10 - 1$.

For 5 people, the answer is the sum of the numbers from 1 to 4, which is 10. 4 is $5 - 1$.

For 2 people, the answer is the sum of the numbers from 1 to 1, which is 1. 1 is $2 - 1$.

For N people, the answer is the sum of the numbers from 1 to $(N-1)$.

Review and discuss your solution: Each person shakes hands with $N - 1$ other people. The answer is not $N(N-1)$, though, because each handshake counts as the one handshake for each person, but only one handshake for the total. The Hershey Bar problem helped to start the plan for this problem, but I needed to adjust the plan to only allow one handshake between each pair in the room.

So the 10 people make 9 handshakes each, but each handshake happens between 2 people, and can only be counted once. I could "divide" the handshake and let each person count the handshake as a $1/2$ handshake. So 10 people make 9 half-handshakes each = 45 handshakes.

N people make $N-1$ half-handshakes each = $N(N-1)/2$

The sum of the numbers from 1 to $N-1$ = $N(N-1)/2$

Instructional Days: 5-7**Topic Description:**

This lesson reinforces the four main phases in the problem-solving process.

Objectives:

The students will be able to:

- Name and explain the steps in the problem-solving process.
- Solve a problem by applying the problem-solving process.
- Express a solution using standard design tools.
- Determine if a given solution successfully solves a stated problem.

Outline of the Lesson:

- Cultural background of cornrow braiding (15 minutes)
- Group discussion on cultural background of cornrow braiding (15 minutes)
- Cornrow curves design tool tutorial (80 minutes)
- Cornrow curves project (50 minutes)
- Gallery walk (5 minutes)

Student Activities:

- Work individually to review the history of cornrow braiding.
- Work in groups to answer reflection question and share with the remainder of the class.
- Work with elbow partner to complete the tutorial.
- Work individually to complete cornrow curves project.
- Participate in gallery walk.

Teaching/Learning Strategies:

- Cultural background of cornrow braiding
 - Students read the cultural background and how to braid sections (csdt.rpi.edu, Cornrow Curves).
- Group discussion on cultural background of cornrow braiding
 - Divide students into groups of 3-4 and ask each group to reflect on one of the following sections:
 - African Origins
 - Middle Passage
 - Civil War to Civil Rights
 - Hip Hop
 - Each group shares their response with the rest of the class.
- Cornrow curves design tool tutorial
 - Individual students complete Part I of the tutorial following all instructions and checking their work with their elbow partner.

- Discuss any issues as a class before proceeding to Part II.
 - Complete Part II of the design tutorial.
 - Stress mathematics and structured inquiry.
- Cornrow curves project
 - Students create their own design.
 - Describe each step of the problem-solving process used.
 - Highlight the mathematical concepts used and where used.
 - Reinforce the strategy of finding a similar problem that has already been solved to help solve the new problem.
- Gallery walk of designs
 - Students share their solutions.

Resources:

- Culturally Situated Design Tools Cornrow Curves—csdt.rpi.edu (courtesy Ron Eglash)

Instructional Days: 8-9

Topic Description: This lesson introduces the binary number system and how to count in binary.

Objectives:

The students will be able to:

- Count forward and backward in binary.

Outline of the Lesson:

- Journal Entry (5 minutes)
- CS Unplugged Activity 1: Count the Dots—Binary Numbers (counting in binary) (50 minutes)
- CS Unplugged Activity 1: Count the Dots—Binary Numbers (binary number system) (50 minutes)
- Revisit journal entry (5 minutes)

Student Activities:

- Complete journal entry.
- Participate in the Count the Dots activities.
- Revisit journal entry.

Teaching/Learning Strategies:

- Journal Entry: How high can you count with your ten fingers?
- Use the CS Unplugged: Count the Dots activity to introduce binary representation and counting in binary.
 - Start with the introductory activity on p. 4 of the activity. (The activity can be downloaded from <http://csunplugged.com>) It will be helpful to read through the entire activity in advance, so that you can revise questions, add your own questions, and think about how you might want to structure each part of the activity. The goal is for students to be actively involved in some way and for all students to be able to represent numbers and count in binary. What follows is the minimal suggestion.
 - Have 5 students come to the front of the room and demonstrate as you follow the instructions and ask the questions. (Each student should receive a large card with one of the numbers of dots—1, 2, 4, 8, 16)
- Use the CS Unplugged: Count the Dots activity to explain the binary number system and have the students practice counting forward and backward.
 - Complete the Binary Numbers activity on p. 5 and Working with Binary activity on p 7.
 - Have 5 students come to the front of the room and try counting as you call out the numbers. (Each student should receive a large card with one of the numbers of dots—1, 2, 4, 8, 16)
 - Have different groups of 5 students at a time come to the front and have the other students provide counting and representation challenges. You could also have a competition with multiple teams of students each trying to get the answer. There are many other possibilities. Be creative!!
- Revisit Journal Entry.

Resources:

- Bell, Tim, Ian Witten and Mike Fellows. Computer Science Unplugged. Canterbury, New Zealand: 2002.
- Computer Science Unplugged Activity 1: Count the Dots—Binary Numbers, pp. 3-13
- Binary number cards for each student
- Large binary number cards for the demonstrations

Instructional Days: 10-11

Topic Description: Students will learn how to convert between binary and decimal numbers in the context of topics that are important to computer science.

Objectives:

The students will be able to:

- Explain why binary numbers are important in computer science.
- Use binary digits to encode and decode messages.
- Use binary digits to create binary art.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Discussion of why binary numbers are important in computer science (15 minutes)
- First 3 minutes of the video "The Primes" (10 minutes)
- Explanation of binary art project (10 minutes)
- Binary art project (55 minutes)
- Gallery walk and presentations of binary art projects (15 minutes)

Student Activities:

- Complete journal entry.
- Participate in a discussion of why binary numbers are important in computer science.
- View the first three minutes of the video "The Primes".
- Work on binary art project.
- Participate in a gallery walk and presentations of binary art projects.

Teaching/Learning Strategies:

- Journal Entry: Complete the Sending Secret Messages activity on p. 8 of the CS Unplugged: Count the Dots activity. (Solution is on p. 13.)
- Discussion of why binary numbers are important in computer science
 - CS Unplugged: Count the Dots activity pp. 11-12 is a good summary.
- Video "The Primes"
 - View the first 3 minutes from the video "The Primes" that explains how Frank Drake sent 1679 bits of information into space in the search of intelligent life.
 - The video can be found at: <http://www.learner.org/resources/series210.html?pop=yes&pid=2283>
 - You will need to sign up for a free account at <http://www.learner.org> first.
 - If you cannot view and display the video, the book that goes with the video explains the transmission. It can be found at: <http://www.learner.org/channel/courses/mathilluminated/units/1/textbook/01.php>
 - Also discuss SETI and what it does
- Explanation of Binary Art project

- Distribute the instructions for the binary art project.
 - Use the project included (Students should create art that represents something about them using 0's and 1's or white (0) and black (1) or some other 2 color system.) or create your own project for them to do.
 - As an alternate to the binary art project, you may choose to complete the remaining activities in CS Unplugged: Count the Dots. (Email and Modems—p. 9, Counting Higher than 31—p. 10, and/or More on Binary Numbers—p. 11)
- Binary art project
 - Allow students time to work on their projects
- Gallery walk and student presentations of projects
 - Depending on what project you end up assigning, you may want to ask for a few volunteers to describe their formulas and/or have other students try to determine the formulas.

Resources:

- Bell, Tim, Ian Witten and Mike Fellows. Computer Science Unplugged. Canterbury, New Zealand: 2002.
- Computer Science Unplugged Activity 1: Count the Dots—Binary Numbers, pp. 3-13
- Binary Art Project

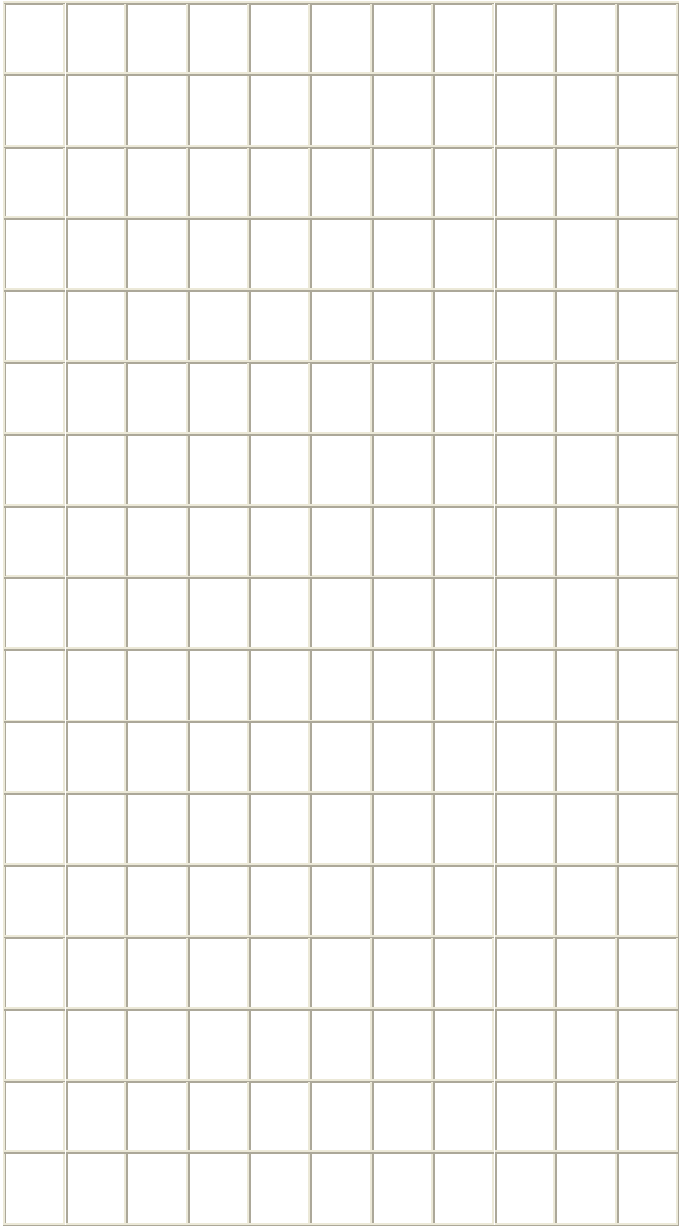
Binary Art Project

Create 341 bits of information about you that could be sent into space. If intelligent life out there receives this information and decodes your information, the resulting pictogram should display information about you.

341 is a semi-prime, only divisible by 11 and 31. The table below has 31 rows and 11 columns.

Use 0's and 1's or two colors to encode your pictogram.

[illegible]



Instructional Days: 12-13

Topic Description: This lesson introduces the linear and binary search algorithms.

Objectives:

The students will be able to:

- Describe the linear search algorithm.
- Describe the binary search algorithm.

Outline of the Lesson:

- Tower Building Activity (55 minutes)
- Model tower building algorithm. (25 minutes)
- Model binary search (15 minutes)
- Comparison of linear and binary search (15 minutes)

Student Activities:

- In pairs complete the Tower Building Activity.
- Model the tower building algorithm.
- Students participate in the activity modeling binary search.

Teaching/Learning Strategies:

- Tower Building Activity
 - Have students complete the Tower Building Activity with their elbow partner and write their solutions in their journals.
- Model tower building activity.
 - Have students share their solutions with another elbow partner pair.
 - Have one set of students use 10 legos (or checkers or some other easily manipulated piece) to model the algorithm for solving the problem in front of the entire class.
- Model binary search.
 - Use 2 copies of the same dictionary. Hand one dictionary to 2 students and have them pick out a word in the dictionary.
 - Choose 2 other students to count the number of times you choose a word from the dictionary to search for the students' word.
 - Start by using a linear search. It should not take long for students to suggest that this is not a good strategy. Ask them to provide a better strategy.
 - Guide them to binary search.
 - Discuss the number of guesses required and how this is similar to the tower building problem.
- Comparison of linear and binary search.
 - Linear—start at the beginning, look at each item until you find it or there is no more data. Data can be sorted or not.
 - Binary—look at middle item, eliminate the half where the value is not located. Find the new middle element and continue the process until you find it, or there is no more data. Ask students to describe what is necessary in order to use a binary search—the list must be sorted.

- Have students provide examples of where each type of search is appropriate and why.

Resources:

- Shasha, Dennis. The Puzzling Adventures of Doctor Ecco. Mineola, New York: Dover Publications, Inc., 1998.
- Tower Building Activity

Tower Building Activity

Donald Trump wants to build a 100 meter high tower as quickly as possible. He has unlimited resources and an unlimited budget and is willing to spend any amount to get the job done.

He has chosen to build the tower with blocks that are 100 meters long and 100 meters wide, but only 1 meter tall. The blocks interlock on top and bottom (like legos). They cannot be stacked sideways.

Using special lifters, putting one block on top of another block takes one week regardless of how high the stacks are.

What is the shortest amount of time that it will take to build the tower?

Suggestions:

- Use legos to help solve this problem.
- Start with a smaller tower of 10 or 20—solve a smaller problem.
- Extend that knowledge to the larger problem.

Instructional Days: 14-16

Topic Description: In this lesson the concept of a list (sorted and unsorted) and sorting algorithms will be explored.

Objectives:

The students will be able to:

- Define sorted and unsorted lists.
- Describe various sorting algorithms.
- Compare various sorting algorithms.

Outline of the Lesson:

- Journal Entry (15 minutes)
- CS Unplugged Activity 7: Lightest and Heaviest—Sorting Algorithms (explore sorting) (40 minutes)
- CS Unplugged Activity 7: Lightest and Heaviest—Sorting Algorithms (discover and describe sorting algorithms) (55 minutes)
- CS Unplugged Activity 7: Lightest and Heaviest—Sorting Algorithms (compare sorting algorithms) (55 minutes)

Student Activities:

- Complete journal entry.
- Groups participate in the various parts of the CS Unplugged: Lightest and Heaviest activity.

Teaching/Learning Strategies:

- Journal Entry: List examples of where it matters whether items are in order (sorted).
 - Have volunteers provide examples from their lists and explain why it matters that they are sorted; in other words, what are the consequences if the list is not sorted?
- CS Unplugged: Lightest and Heaviest activity
 - The activity can be downloaded from <http://csunplugged.com>. It will be helpful to read through the entire activity in advance, so that you can revise questions, add your own questions, and think about how you might want to structure each part of the activity. The goal is for students to be actively involved in some way and for all students to be able to describe the various types of sorting. What follows is the minimal suggestion.
 - Divide students into groups of 3-4 and give each group a set of weights and a balance scale as described in steps 1 and 2 on p. 66 of the Sorting Weights Activity. (There are many possible ways to make the weights. One would be to use bags with varying numbers of pieces of candy. If you don't have balance scales, you can help students come up with a strategy that will simulate a scale. For example, if you make the weights clearly different in weight, they could do this by feel.)
 - Have students complete #3 and #4 on p. 66 and then discuss their answers as indicated.
 - Have students complete #5 on p. 66.
 - At this point in the activity, students should present their findings to the class and discuss. Point out the selection sort information on p.66.

- Have students complete the Divide and Conquer activity on p.67. Throughout, guide students as necessary and have them keep track of the processes they use.
- If time permits, have students try both sorting methods to sort cards that have 50 random numbers on them and analyze the number of comparisons required for each.

Resources:

- Bell, Tim, Ian Witten and Mike Fellows. Computer Science Unplugged., New Zealand: 2002.
- Computer Science Unplugged Activity 7: Lightest and Heaviest—Sorting Algorithms, pp. 64-70
- Containers of the same size with different weights
- Balance scales

Instructional Day: 17-18

Topic Description: Minimal spanning trees and graphs will be explored. Students will learn how graphs can be used to help solve problems.

Objectives:

The students will be able to:

- Solve a minimal spanning tree.
- Draw a graph to solve a problem.

Outline of the Lesson:

- CS Unplugged Activity 9: The Muddy City—Minimal Spanning Trees (55 minutes)
- CS Unplugged Activity 9: The Muddy City—Minimal Spanning Trees (extension) (55 minutes)

Student Activities:

- Participate in the various parts of the CS Unplugged: The Muddy City activity.
- Participate in the various parts of the CS Unplugged: The Muddy City activity extension.

Teaching/Learning Strategies:

- CS Unplugged: The Muddy City activity
 - The activity can be downloaded from <http://csunplugged.com>. It will be helpful to read through the entire activity in advance, so that you can revise questions, add your own questions, and think about how you might want to structure each part of the activity. The goal is for students to be actively involved in some way and for all students to be able to describe shortest path strategies. What follows is the minimal suggestion.
 - Follow the directions in The Muddy City Problem on p. 78.
 - Have students work with their elbow partners.
 - Have students share their solutions and lead the follow-up discussion p.77.
- CS Unplugged: The Muddy City activity extension
 - Have students repeat the Muddy City Problem with the abstract representation in the figure on p. 79 and answer the questions on p 79.
 - Discuss various applications of this problem in anticipation of the final project (p.80).

Resources:

- Bell, Tim, Ian Witten and Mike Fellows. Computer Science Unplugged. Canterbury, New Zealand: 2002.
- Computer Science Unplugged Activity 9: The Muddy City—Minimal Spanning Trees, pp. 76-80

Instructional Days: 19-22

Topic Description: Students work on final unit project.

Objectives:

The students will be able to:

- Incorporate all unit objectives into the final project.

Outline of the Lesson:

- Explanation of final project (15 minutes)
- Completion of final projects (150 minutes)
- Presentations of final projects (55 minutes)

Student Activities:

- Groups work on final projects.
- Groups present final projects.

Teaching/Learning Strategies:

- Explanation of final project
 - Distribute final project explanation.
 - Note: You may wish to modify the scenario of the problem to address student interests and abilities. Another possible example might be a variation on finding the cheapest route between locations based on the price of gasoline. This could be in the context of a family vacation, carpool routes, running errands, etc.
 - Divide students into groups of 3-4 and distribute a list of 10 US cities to each group.
- Completion of final projects
 - Monitor student work, answering questions as necessary.
- Presentations of final projects
 - Have each group present the information in their final project.

Resources:

- Road atlases
- Final Project (This project is adapted from MathmaniaCS Lesson 13 (<http://www.mathmaniacs.org/lessons>)
- Final Project Sample Rubric

Final Project

Your task is to create the most inexpensive plan for constructing new high speed roadways in place of existing roads between 10 US cities and present your plan to the class.

Scenario:

The Department of Transportation is considering the construction of new high speed roadways in place of existing roads between the 10 US cities you have been given. Their goal is to connect all these cities as cheaply as possible. The cost of changing an existing road to a high speed roadway is 1 (where 1 represents whatever the base cost per mile is) for every mile of interstate, 2 (twice the cost per mile as interstate) for every mile of state highway, and 3 (three times the cost per mile as interstate) for every county highway, etc.

You will need a road atlas to find the various distances and types of roads.

Your presentation can be given as a poster, a powerpoint, a video or other pre-approved product.

Your presentation should include:

1. The names of people in your group
2. A picture (graph) representing all the cities with all roads between them labeled with mileages and costs.
3. A detailed plan of your solution
4. A written explanation of the strategies you used to find the least expensive solution
 - The solution on the graph and the total cost of the project.

Final Project Sample Rubric

Group Members Names: (up to 4)

Do you have?	Points Possible	Yes	No	Points Earned
Detailed Plans				
Overall plan to solve the problem	10			
Explanation of strategies used to solve the problem	15			
Other parts of your project:				
Graph labeled with cities and mileage	10			
Graph labeled with costs	15			
Solution labeled on graph	20			
Total cost	10			
Presentation				
Present your solution	20			
TOTAL:	100			

Unit 3:

Web Design



Introduction

The Web Design unit builds on the concepts presented in the previous units by having students apply problem solving strategies to web design; thus, it also serves as a bridge to the Introduction to Programming unit as students move from user to creator. The unit also provides an opportunity to expand upon the issues of ethics and privacy related to the internet that were introduced in the first unit.

The basics of html and css are introduced as a method for describing features of web pages that students can use to design and develop web pages based on their own culture, interests and unique experiences.

The html and css lessons are scaffolded in order to provide all students an entry point, but it is likely that as students explore they will encounter features they wish to add for which they do not yet know the correct tags. Many students will be able to figure these out on their own and should be encouraged to do so.

Resist the temptation to provide lists of appropriate font and color palettes and/or best layout designs. As students experiment and share their work, challenge them to explain why they chose the features they did and encourage peers to comment.

Example projects are provided as a starting point, but students should be encouraged to work on projects that are authentic for them. They may choose to create web pages on different topics for each assigned project or build on a prior one as appropriate.

There are two supplements—Flash animation and Javascript—at the end of the unit for use if there is additional time and interest.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1-2	Issues of social responsibility in web use are explored as well as the relative merits of the influence of the web on society, personal lives, and education.
3-4	Introduce the use of basic html.
5	Introduce basic formatting in html.
6-7	Explore image editing for the web using Photoshop or an image editor of choice.
8-10	Introduce basic css.
11-13	Explore the concept of separating style from structure by keeping separate html and css files..
14	Add hyperlinks to other websites .
15-16	Introduce a variety of page layout styles.
17-19	Practice using style tables, lists and css in the context of a web page creation project.
20-21	Introduce several web user interface elements combining javascript, html, css, and Photoshop.
22	Implement advanced functionality with javascript libraries. Create accordion menus based on the mootools implementation.
23-24	Further explore the use of javascript library effects, including lightbox slideshow and sliding image puzzles.
25-28	Final projects and gallery walk

Daily Lesson Plans

Instructional Days: 1-2

Topic Description: This lesson engages students in a discussion of the web as social experience. Issues of social responsibility in web use are explored as well as the relative merits of the influence of the web on society, personal lives, and education.

Objectives:

The student will be able to:

- Set up a blog.
- Explain basic security issues on the internet.
- Identify web applications which influence society and education.
- Identify appropriate vs. inappropriate use of social websites.

Outline of the Lesson:

- Set up a blog (25 minutes)
- Discussion of online security issues (20 minutes)
- Blog entry of students' online experiences (10 minutes)
- Discussion of parts 1-3 of *Growing Up Online* from the PBS series *Frontline* (40 minutes)
- Blog entry reflecting on *Growing Up Online*. (15 minutes)

Student Activities:

- Set up a blog.
- Participate in discussion of online security.
- Create a blog entry on online experiences.
- Participate in a discussion about online experiences with social networking sites, blogs, email, online chatting and the kind of impact it has had on their lives?
- View and discuss *Growing Up Online*.
- Create a Blog entry reflecting on *Growing Up Online*.

Teaching/Learning Strategies:

- Set up a blog
 - Complete the portions of the setup that were not done prior to class.
 - Guide a discussion of online security as students work on setting up their blogs.
- Create a blog entry on online experiences.
 - Show students how to create a blog entry using the blogging tool chosen for the class.
 - Have students create an entry that describes some of their current online experiences.

- Guide a discussion regarding student use of social networking applications. (Note: This discussion may be a review of discussions from Unit 1) Ask questions such as:
 - Which social networking applications do you use? (blogging, facebook, myspace)
 - How often? How many of your friends use them?
 - How important are these web applications to your lives? How have they changed your lives?
 - Living their lives essentially online
 - A revolution in classrooms and in social life
 - Self expression, trying on new Identities
- Display parts 1-3 of *Growing Up Online* from the PBS series *Frontline*.
 - After viewing the video, lead a discussion on the content.
- Blog entry reflecting on *Growing Up Online*
 - Have students create a blog entry reflecting on the video. Did any of their thoughts change after viewing the video?

Note: Helping each student set up a blog will require some time outside of class and should be completed in advance of the lesson. For example, students need to get a free google gmail account before signing up for blogger. This can only be done outside of the LAUSD firewall.

Resources:

- <http://www.pbs.org/wgbh/pages/frontline/kidsonline/>

Suggested blogging tools:

- <http://www.blogger.com>
- <http://www.wordpress.com>
- <http://www.tumblr.com>

Instructional Days: 3-4

Topic Description: An introduction to the use of basic html.

Objectives:

The student will be able to:

- Navigate an html editor.
- Create an html page with a title and a body.
- Create an html page with paragraph tags, headings, line breaks, and horizontal lines.

Outline of the Lesson:

- Demo of html editor (15 minutes)
- Html page with a title and body (20 minutes)
- Html page with paragraphs and headings (30 minutes)
- Html page with line breaks and horizontal lines (45 minutes)

Student Activities:

- Follow along during the demo of the html editor.
- Create an html page with a title and body.
- Create an html page with paragraphs and headings.
- Create an html page with line breaks and horizontal lines.

Teaching/Learning Strategies:

- Demo of html editor
 - Display the html editor that you have chosen for the class. Point out the following html tags.

Tag	Description	End Tag
<html>	Defines an HTML document	</html>
<head>	Defines information about the document	</head>
<title>	Defines the title of the document	</title>
<body>	Defines the main part of the document	</body>

- Enter a title and a one sentence body. Demonstrate how to save the document as an html file and how to view the output page in a browser. Point out that the title appears in the bar at the top of the window. Also point out that the end tag is necessary in order to tell the computer when to stop doing a particular thing.
- Html page with a title and a body
 - Have students write a paragraph in the body section and give it a title. (Students can choose to write about themselves or another topic of interest.)

- Demonstrate the creation of a basic html/css page in the html editor of choice.
- Html page with paragraphs and headings
 - Have students add a second paragraph to their web page and note what happens.
 - Then have them add two lists related to their topic (favorite movies, music, hobbies, etc.) and note what happens.
 - Guide students to notice that everything runs together no matter how they type it.
 - Explain the following html tags.

Tag	Description	End Tag
<p>	Defines a paragraph	</p>
<h1> to <h6>	Defines headings at levels 1-6	</h1> to <h6>

- Have students try inserting these new tags into their web page and note what happens.
- Remind students that they need the end tag.
- This is a good place to point out that html is one language that can be used to give the computer instructions as discussed in Unit 1.
- Html page with line breaks and horizontal lines
 - Explain the following html tags.

Tag	Description	End Tag
 	Defines a single line break	
<hr/>	Defines a horizontal line	

- Have students try inserting these new tags into their web pages and note what happens.
- Give students time to experiment and determine what combination of tags will allow them to put their lists in a column, with each list having its own heading.
- Note that you can retrieve an html reference from <http://www.w3schools.com/html/>

Resources:

html editors

- <http://www.tacosw.com> (mac only)
- <http://www.barebones.com/products/TextWrangler/index.shtml> (mac only)
- <http://smultron.sourceforge.net/> (mac only)
- <http://www.alleycode.com/download.htm> (windows only)

html tutorial

- <http://www.w3schools.com/html/>

Instructional Day: 5

Topic Description: An introduction to basic formatting in html.

Objectives:

The student will be able to:

- Create an html page that includes underlined, italicized, and boldface text.

Outline of the Lesson:

- Review of tags learned to date (5 minutes)
- Html pages that include underlined, italicized, and boldface text (50 minutes)

Student Activities:

- Participate in review of tags.
- Create an html page that includes underlined, italicized and boldface text.

Teaching/Learning Strategies:

- Review of tags
 - Have students open their files; then lead a quick review of the tags.
- Html pages that include underlined, italicized, and boldface text
 - Explain the following html tags.

Tag	Description	End Tag
	Defines where you want text to be bold	
<i>	Defines where you want text to be italics	</i>
<u>	Defines where you want text to be underlined	</u>

- Have students try inserting these new tags into their web pages and note what happens.
- Give students time to experiment
- Ask students to think about what things they would like to be able to do with web pages that they have not done already. Answers will vary: pictures, different types of fonts, colors, etc.

Resources:

html tutorial

- <http://www.w3schools.com/html/>

Instructional Days: 6-7

Topic Description: Explore image editing for the web using Photoshop or an image editor of choice.

Objectives:

The student will be able to:

- Identify the standard image resolution for the web (72 dpi).
- Resize and crop images for the web.
- Identify and differentiate between the various image formats used in web sites: jpg, gif, png.
- Create an html page that includes images.

Outline of the Lesson:

- Discussion of various web image formats (5 minutes)
- Demo of resizing and cropping images (15 minutes)
- Selecting and cropping a images (35 minutes)
- Html pages that include images (55 minutes)

Student Activities:

- Participate in discussion and follow along with demo.
- Select and crop a few images.
- Create an html page that includes images.

Teaching/Learning Strategies:

- Discussion of various web image formats
 - Explain that image properties are relevant to web use.
 - It is important to check the size when preparing an image for use on the web. Resolution can be set under image size.
- Selecting and cropping an image
 - Demonstrate how to crop and resize images in Photoshop or image editor of choice. Part of this will be review from Unit 1.
 - Have students choose a few images that they will add to their web page and crop them.
 - Explain that students should save their images for use in this project and later projects.
- Html pages that include images
 - Explain the following html tag.

Tag	Description	End Tag

- Point out that the correct syntax for defining an image is

- xxxx is the name of the image file. The image should be in the same folder as the html file.
- Have students insert their image into their html page.
- They can resize the photo on the screen with: ``
- They can add a title by: ``
- Give students time to experiment with placement, sizes, headings, and additional images.
- For students who finish early, you can have them view the filters and effects section of <http://www.georgebenainous.com/web> and try modifying their images.

Resources:

- <http://www.georgebenainous.com/web> (photoshop—filters/effects)
- <http://morph.cs.st-andrews.ac.uk/>
- <https://www.photoshop.com/express>

Instructional Days: 8-10

Topic Description: An introduction to the use of basic css.

Objectives:

The student will be able to:

- Create inline styles with css.
- Create a web page that uses inline styles.
- Create an internal style sheet with css.
- Create a web page that uses an internal style sheet.

Outline of the Lesson:

- Overview of css (10 minutes)
- Sample inline styles (15 minutes)
- A web page that uses inline styles (15 minutes)
- Sample internal style sheet (15 minutes)
- Movie review html/css page (45 minutes)
- Share student work (10 minutes)
- A second html/css page (45 minutes)
- Share student work (10 minutes)

Student Activities:

- Examine sample web content.
- Create a web page that uses inline styles.
- Examine sample web content.
- Complete movie review html/css project.
- Share completed projects.
- Complete second html/css project.
- Share completed projects.

Teaching/Learning Strategies:

- Overview of css
 - CSS stands for Cascading Style Sheets.
 - CSS provides the formatting and style for a web page, while html provides the content.
 - There are three methods for inserting styles.
 - Inline styles
 - Internal style sheet
 - External style sheet
 - The basic format for a style is:

Selector	Declaration	Declaration
h1	{color:blue;	Font_size:12px;}

- The selector is the element you want to style; each declaration consists of a property and a value; the property is the attribute you want to change and each property has a value.
- To make it more readable you can put each declaration on a separate line.
- Demonstrate creating a header with the inline style listed above.
- Note that you can retrieve a css reference from <http://www.w3schools.com/css/>.
- Display http://www.w3schools.com/tags/ref_colornames.asp and http://www.w3schools.com/tags/ref_colorpicker.asp as sources for choosing colors.
- Have students suggest a few different declarations and demonstrate the results.
- Create a web page that uses inline styles.
 - Have students add a few styles to their web page.
- Sample internal style sheet.
 - Point out that inline styles should be used sparingly because they defeat the purpose of separating the style from the content.
 - Have students view <http://www.georgebenainous.com/web> (html/css--basic markup). They should view the page before and after the styling is added.
 - Point out what each piece of the styling does to the original page. Point out the format and that the internal style sheet is included in the <head>. Also note that the style applies to the entire page unless a specific inline style is added.
- Movie review html/css page
 - Students create a website with one or more movie reviews. The html page will contain the following paragraphs for each review: title, director, synopsis, review. The css stylesheet will have corresponding classes. The page will also include:
 - At least one picture
 - The name of at least one of the actors in italics
 - Give the background and text colors
- Share student work
 - Guide students in sharing their work either by a gallery walk, volunteers, etc.
- Complete second html/css project
 - Note: you may choose to have students continue working on their movie project rather than start a second one if time is short.
 - Some examples of projects from which to have students choose are provided in the tutorial or have students create their own project.
- Share results of student work.
 - Guide students in sharing their work either by a gallery walk, volunteers, etc.

Resources:

- <http://www.georgebenainous.com/web> (html/css--basic markup)

- http://www.w3schools.com/tags/ref_colornames.asp
- http://www.w3schools.com/tags/ref_colorpicker.asp

Instructional Days: 11-13

Topic Description: Explore the concept of separating style from structure by keeping separate html and css files.

Objectives:

The student will be able to

- Create an html page which links to a separate css file.
- Use html tags and css styling elements to separate style from structure.

Outline of the Lesson:

- Review of html/css concepts and description of how to link to a separate css file (15 minutes)
- Sample web content (20 minutes)
- Creation of separate html and css pages for the previous movie review project (40 minutes)
- Creation of separate html and css pages for a second project (75 minutes)
- Share student work (15 minutes)

Student Activities:

- Review html/css concepts.
- Examine sample web content.
- Complete html/css movie review project.
- Complete html/css project 2.
- Share completed projects.

Teaching/Learning Strategies:

- Review of html/css concepts
 - Guide a discussion of the highlights of the previous lesson.
 - Using the text editor, demonstrate how to create an external file for the styles. Emphasize that whatever the name of the style sheet, it needs to have a .css extension. Save the file in the web folder.
- Sample web content
 - Answer questions as students view and read the html/css reusable code section at <http://www.georgebenainous.com/web> (html/css—reusable code)
 - Demonstrate how to add the appropriate link to the html file.
- Complete html/css movie review project.
 - Have students revise their previous movie review project to use an external css file.
- Complete html/css project 2.
 - One example of a project might be to create a website with information on their favorite band.
 - A Paragraph with the name of the band in large bold print.
 - At least one picture
 - The genre of the band in italics (i.e. Rock, Rap, etc)

- A list of some of the songs from the band in a paragraph in regular print
 - A separate section that explains why the band is your favorite
 - The background and text in different colors
 - Other examples can be found in the tutorial
- Share student work
 - Guide students in sharing their work either by a gallery walk, volunteers, etc.

Resources:

- <http://www.georgebenainous.com/web> (html/css—reusable code)

Instructional Day: 14

Topic Description: This lesson explores the use of links to other websites.

Objectives:

The student will be able to:

- Create an html page that includes hyperlinks.

Outline of the Lesson:

- Explanation of how to add hyperlinks (15 minutes)
- Addition of hyperlinks to web pages (40 minutes)

Student Activities:

- Participate in discussion of hyperlinks.
- Add hyperlinks to web page.

Teaching/Learning Strategies:

- Html pages that include hyperlinks
 - Explain the following html tag.

Tag	Description	End Tag
<a href = "url"	Defines what is to be displayed.	< /a>

- Point out that the correct syntax for defining a hyperlink is Link text
- The start tag contains information about the link address.
- What is to be displayed can be text, an image, etc..
- Give students time to experiment with adding hyperlinks to their previous project, including placement and sizes.

Resources:

- No additional resources needed

Instructional Days: 15-16

Topic Description: In this lesson a variety of page layout styles are introduced.

Objectives:

The student will be able to:

- Use table, row, and column tagging in an html page.
- Add css styling to an html table.
- Use ordered and unordered list tagging in an html page.
- Add css styling to an html list.
- Use grid elements in css div placement.
- Add a menu to an html page.
- Create a web page that includes layout styles.

Outline of the Lesson:

- Explanation of how to create an html table (15 minutes)
- Examples where data lends itself to being presented in a table (5 minutes)
- Explanation of how to create html ordered and unordered lists and how to add styling to list elements. (15 minutes)
- Examples where data lends itself to being presented in a list. (5 minutes)
- Preliminary css positioning and opacity exercise (15 minutes)
- Explanation of how to create a menu (15 minutes)
- Creation of a web page that includes layout styles (40 minutes)

Student Activities:

- Participate in the discussion of creating an html table.
- View examples where data lends itself to being presented in a table.
- Participate in the discussion of creating ordered and unordered lists.
- View examples where data lends itself to being presented in a list.
- Complete css positioning exercise.
- Participate in the discussion of menu creation.
- Create a web page that includes layout styles.

Teaching/Learning Strategies:

- Use the tutorial (<http://www.georgebenainous.com/web/html/css—tables>) to demonstrate how to create a table, how to add rows and columns and how to add css styling to table, row, and column elements. Have students view the example code and predict the results prior to viewing.
- Demonstrate examples where data lends itself to being presented in a table

- Use the tutorial (<http://www.georgebenainous.com/web> html/css—styled lists) to demonstrate how to create ordered and unordered lists and how to add css styling to list elements. Have students view the example code and predict the results prior to viewing.
- Demonstrate examples where data lends itself to being presented in a list.
- Use the tutorial (<http://www.georgebenainous.com/web> html/css—page layout) to demonstrate div positioning using css. Have students view the example code and predict the results prior to viewing.
- Use the tutorial (<http://www.georgebenainous.com/web> html/css—menus) to demonstrate how to create a menu. Have students view the example code and predict the results prior to viewing.
- Have students create a web page which serves as an advertisement for a product of their choice.
 - Explain project requirements.
 - The page must make use of at least one of the page layout methods discussed.
 - Circulate room and help students with projects.

Resources:

- <http://www.georgebenainous.com/web> (html/css—tables, styled lists, page layout, menus)

html and css tutorials

- <http://www.w3schools.com/html/>
- <http://www.w3schools.com/css/default.asp>

Instructional Days: 17-19

Topic Description: Practice the use of various design elements.

Objectives:

The student will be able to:

- Create web pages which incorporate design elements previously studied.

Outline of the Lesson:

- Explanation of project (10 minutes)
- Design and creation of a web page that links to at least 5 other websites (135 minutes)
- Share student work (20 minutes)

Student Activities:

- Design and create a 3 page website about their future that links to at least 5 other websites and includes a variety of design elements.
- Share completed work.

Teaching/Learning Strategies:

- Design and create a web page about their future that links to at least 5 other websites and includes a variety of design elements.
 - The three pages may either scroll or link to each other.
 - The project should include images related to their future.
- Share student work.

Resources:

- <http://www.georgebenainous.com/web> (html/css)

Instructional Days: 20-21**Topic Description:**

This lesson introduces several web user interface elements combining javascript, html, css, and Photoshop.

Objectives:

The student will be able to:

- Apply a Photoshop filter, effect, or image adjustment using the lasso tool.
- Create a rollover button using javascript.
- Implement a menu using a styled list (with pre-implemented css).
- Create a multi-page web site.

Outline of the Lesson:

- Review of the use of filters, effects, and image adjustments in Photoshop and demonstrate the creation of rollover buttons. (10 minutes)
- Creation of several javascript rollover buttons (10 minutes)
- Creation of a menu (10 minutes)
- Creation of a single template html file. (10 minutes)
- Explanation of the requirements for the continental mapping project and the use of an html template. (15 minutes)
- Continental mapping project (40 minutes)
- Share student work (10 minutes)

Student Activities:

- View the rollover section of the javascript tutorial on the creation of rollover buttons.
- Create several rollover buttons.
- Review the section of the tutorial website dealing with using the lasso for inverted selections.
- Create a single template html file with a menu.
- Implement the continental mapping project.
- Share completed work.

Teaching/Learning Strategies:

- Review of the use of filters, effects, and image adjustments in Photoshop.
 - View the rollover section of the javascript tutorial on the creation of rollover buttons.
- Creation of several rollover buttons.
 - Guide students in the creation and choice of visual themes for the buttons.
- Review the section of the tutorial website dealing with using the lasso for inverted selections
- Review the section of the tutorial website dealing with menus.

- Creation of a single template html file.
 - Circulate room and help students.
- Continental mapping project
 - Review the requirements for the continental mapping project.
 - Respond to questions on the use of an html template.
 - Use the template created the previous day to implement the continental mapping project as outlined in the tutorial website.
 - Demonstrate how to extend the code samples provided in the tutorial website.
- Share student work.
- Consider creating a second project using the same techniques as the continental mapping project. (See the sample projects described in the tutorial website.)

Resources:

- <http://www.georgebenainous.com/web> (photoshop—filters/effects, javascript—rollover buttons, photoshop—project 2)
- <https://www.photoshop.com/express/index.html>

Instructional Day: 22

Topic Description: Use Mootools to create a Web 2.0 style navigation called the accordion menu.

Objectives:

The student will be able to:

- Include a javascript library.
- Create a website which uses the accordion menu based on the mootools implementation.

Outline of the Lesson:

- Demos from mootools.net (10 minutes)
- Use of the accordion menu (5 minutes)
- Creation of a website in which content is displayed using the accordion menu. (30 minutes)
- Share student work (10 minutes)

Student Activities:

- View demos from mootools.net on how to include a javascript library in html code.
- View the mootools section of the tutorial website.
- Create a website in which content is displayed using the accordion menu.
- Share completed work.

Teaching/Learning Strategies:

- Demonstrate how to include a javascript library.
- Guide students through the mootools section of the tutorial website.
- Guide students through mootools.net, demonstrating other possible ajax effects.
- Delineate the technical and content requirements for an accordion menu driven website.
- Explain project
 - Circulate room and help students with projects.
- Review/evaluate student work.

Resources:

- <http://www.georgebenainous.com/web> (javascript—mootools)
- <http://www.mootools.net>
- <http://www.wikipedia.net>

Instructional Day: 23-24

Topic Description: This lesson further explores the use of javascript library effects, including the lightbox slideshow and jquery.

Objectives:

The student will be able to:

- Implement advanced interactive web functionality through the inclusion of javascript libraries.
- Create a lightbox slideshow.
- Create a sliding image puzzle.

Outline of the Lesson:

- Demonstration of lightbox slideshow (5 minutes)
- Creation of a website in which content is displayed using a lightbox slideshow (20 minutes)
- Share student work (5 minutes)
- Demonstration of a sliding image puzzle (5 minutes)
- Creation of a sliding image page (15 minutes)
- Share student work (5 minutes)

Student Activities:

- View the use of a lightbox slideshow as implemented in the tutorial website.
- Create a website in which content is displayed using a lightbox slideshow.
- Share completed work.
- View the use of a sliding image puzzle as implemented in the tutorial website.
- Create sliding image page.
- Share completed work.

Teaching/Learning Strategies:

- Guide students through the lightbox slideshow as implemented in the tutorial website.
- Create a website in which content is displayed using a lightbox slideshow.
 - Have students create a slideshow with at least five pictures and include captions for each picture.
- Share student work.
- Guide students through the sliding image puzzle as implemented in the tutorial website.
- Create sliding image page
 - Students can use an image of their choice.
 - Have students use Photoshop to resize /crop images as needed.
 - Share puzzles with elbow partners.

Resources:

- <http://www.georgebenainous.com/web> (javascript—light box, javascript—jquery)
- <http://www.lokeshdhakar.com/projects/lightbox2/>
- <http://www.bennadel.com/blog/1009-jQuery-Demo-Creating-A-Sliding-Image-Puzzle-Plug-In.htm>

Instructional Days: 25-28

Topic Description: Students complete final projects.

Objectives:

The students will be able to:

- Incorporate all objectives of the unit into the final project.

Outline of the Lesson:

- Explanation of final project (15 minutes)
- Final project (135 minutes)
- Gallery walk and vote on final projects (15 minutes)

Student Activities:

- Complete final project.
- Participate in gallery walk to view and vote on completed projects.

Teaching/Learning Strategies:

- Final project
 - Explain final project choices.
 - Help students with projects as necessary.
- Gallery walk
 - Encourage students to ask each other questions as they view the websites.
 - Have students vote on their favorite.

Resources:

- Final Project
- Final Project Sample Rubric

Final Project

Your task is to create a website that includes

- Images and text with references to sources
- Pages with headers, navigation and content
- An external css file to define layout and styling

You may choose any of the following topics

- An ethical dilemma
- A career
- A worldwide or community problem
- A topic of your choice that has been approved

Ethical Dilemma Web Site

Your task is to analyze an ethical dilemma. Choose one of the four dilemmas listed below or get approval for a different one. You must consider the alternatives and give reasons for the why and the why not you should do what is described. Then you must choose what you would do and explain why. The website should include pages that

1. Describe the dilemma you have chosen.
2. Give 3 reasons why you should do what is described.
3. Give 3 reasons why you should NOT do what is described.
4. Describes what you will do and explains why.

Ethical Dilemmas:

1. People illegally download music over the internet. Although it's free, it is still illegal. What do you choose to do? Why?
2. Your parent loses his/her job. You could help out by selling illegal dvds on the streets. What should you do?
3. You have the ability to hack into the school computer system. You can change people's grades. Would you change your own? Why or why not? What if you could change the grade for a basketball player who has a scholarship to play for a big university?
4. Someone you know works at a store that sells iPods. He steals some and asks if you want to buy one for half the price the store sells it for? Should you buy it? Why or why not?

Career Website

Research a career and create a website that provides information about it.

The website should include pages that

- Provide a brief description of the career
- Explain the education required
- Describe tasks performed in the career, salaries and how computer science is used in the career.

Worldwide or Community Problem Website

Research a worldwide or community problem and create a website that provides information about it.

The website should include pages that

- Provide a brief description of the problem.
- Explain how the problem is affecting people.
- Describe possible solutions to the problem and what people reading the website can do to help solve it.

Final Project Sample Rubric:

Do you have?	Points Possible	Yes	No	Points Earned
Website Content				
A home page with an image and a brief description of your topic	5			
3 or more additional pages on your site	15			
Images that support your topic	10			
Cite the source(s) of you images	5			
Complete information for your topic	10			
Cite the source(s) of your information	5			
Website Design				
Have a background color or image.	5			
Incorporate one of the layout styles into your website	5			
Include rollover images	5			
Links to all the pages of your website on each page.	10			
Integrate the lightbox slideshow, sliding images, or mootools accordion into your website.	5			
Use one shared external .css file for your site.	5			
Peer Grading	15			
Your project is voted best by your peers. (EXTRA CREDIT)	up to 10			
Total	100			

Flash Animation Supplement

(These activities can be used as the last days prior to the final project if students finish other projects prior to the time allotted.)

Instructional Day: 1

Topic Description: Adobe Flash (formerly Macromedia Flash) is a proprietary web animation platform. The introductory lesson demonstrates how to use stop action photography and Flash to create a flipbook effect.

Objectives:

The student will be able to:

- Use stop action photography in animated flip books.
- Create a simple flash animation by importing a series of images.

Outline of the Lesson:

- Preview the stop action photography study of the galloping horse (5 minutes)
- Demonstration of how to clip each image in Photoshop (5 minutes)
- Demonstration of how to import a series of images into Flash and how to play the movie (10 minutes)
- Practice of the import procedure (5 minutes)
- Creation of a movie from stop action photography (25 minutes)
- Share student work (5 minutes)

Student Activities:

- Preview the stop action photography study of the galloping horse from the Flash section of the tutorial website.
- View how to clip each image in Photoshop.
- View how to import a series of images into Flash and how to play the movie.
- Create a movie.
- Share completed work.

Teaching/Learning Strategies:

- Preview of the the stop action photography study of the galloping horse from the Flash section of the tutorial website
 - Discuss the historical significance of Eadweard Muybridge and stop action photography.
 - Preview various Eadweard Muybridge photographic studies.
- Demonstration of how to clip each image in Photoshop
 - Create eleven separate images..
 - Follow a numerical naming convention: 01.jpg, 02.jpg...11.jpg.
- Demonstration of how to import a series of images into Flash and how to play the movie
 - Guide students as they follow the procedure after it is demonstrated.

- Creation of a movie
 - Suggest students download another Eadweard Muybridge stop action photographic study and follow the same procedure or have them photograph their own stop action study.
 - Circulate room and help students choose and complete projects.
- Share student work

Resources:

- <http://www.georgebenainous.com/web> (flash—flipbook)
- <http://www.adobe.com/cfusion/designcenter/search.cfm?product=Flash&go=Go>

Instructional Day: 2

Topic Description: An animation technique called tweening is explored in Adobe Flash.

Objectives:

The student will be able to:

- Use frame/timeline based animation.
- Use an automatic frame based animation technique called tweening.
- Create several examples of tweened animations.

Outline of the Lesson:

- Demonstration of tweening techniques (15 minutes)
- Creation of a visual composition (30 minutes)
- Share student work (10 minutes)

Student Activities:

- View tweening techniques (outlined in the tutorial website).
- Create a visual composition.
- Share completed work.

Teaching/Learning Strategies:

- Demonstration of tweening techniques (outlined in the tutorial website)
 - motion, size, rotation, color
 - Discuss elements of design as they pertain to objects in motion.
- Creation of a visual composition
 - Explain the requirements for one or more of the following sample projects and guide students as they create their versions.
 - Create a visual composition using Flash tweening. (Remember to put each tween on a separate layer). Study the ideas of symmetry (balance) and asymmetry (imbalance) in motion.
 - Create a visual composition using Flash tweening applied to initials. Students can use their own initials. (Type can be tweened in Flash.)
- Share student work

Resources:

- <http://www.georgebenainous.com/web> (flash—tweening)
- <http://www.adobe.com/cfusion/designcenter/search.cfm?product=Flash&go=Go>

Instructional Day: 3

Topic Description: The movie clip is the basic unit of Flash animation which allows for reusability and scripting. This lesson is an introduction in the creation of movie clips.

Objectives:

The student will be able to:

- Explain the concept of a Flash movie clip.
- Differentiate between a movie clip and an instance of a movie clip.
- Create a movie clip based on keyframed animation.

Outline of the Lesson:

- Demonstration of creating a movie clip (10 minutes)
- Demonstration of how to reuse multiple instances of a movie (5 minutes)
- Creation of a horse movie clip (20 minutes)
- Creation and implementation of movie clips (20 minutes)

Student Activities:

- View examples of Flash movie clips.
- Create various Flash movie clips based on the galloping horse study as outlined in the Flash section of the tutorial website.
- Design and create follow-up movie clips.
- View the creation of a movie clip.
- View how to reuse multiple instances of a movie clip.
- Create the horse movie clip as outlined in the tutorial website.
- Create and implement movie clips.

Teaching/Learning Strategies:

- Demonstration of examples of Flash movie clips
 - Explain how to create a keyframed animation based on the galloping horse example from the Flash section of the tutorial website.
 - Guide students as they create various Flash movie clips based on the galloping horse study as outlined in the Flash section of the tutorial website.
 - Guide students as they design and create follow-up movie clips.
- Demonstration of creating a movie clip.
- Demonstration of how to reuse multiple instances of a movie clip
 - including a secondary tweening
- Creation of a horse movie clip as outlined in the tutorial website
- Creation and implementation of movie clips

- Guide students as they create their own ideas and then implement.

Resources:

- <http://www.georgebenainous.com/web> (flash—movie clips)
- <http://www.adobe.com/cfusion/designcenter/search.cfm?product=Flash&go=Go>

Javascript Supplement

Instructional Day: 1

Topic Description: Introduce basic javascript; add interactivity to web pages.

Objectives:

The student will be able to:

- Add a javascript to an html page.
- Link to an external javascript file.
- Create alerts and prompts in javascript.
- Write basic math statements in javascript.

Outline of the Lesson:

- Demonstration of javascript basic markup code samples (5 minutes)
- Creation of javascripts (10 minutes)
- Extension of code samples provided in tutorial website (40 minutes)

Student Activities:

- View code samples from tutorial website.
- Create initial javascripts.
- Extend code samples provided in tutorial website.

Teaching/Learning Strategies:

- Demonstrate how to create and link to a javascript file.
- Guide students in the creation of initial javascript files.
- Extension of code samples provided in tutorial website
 - Demonstrate how to extend the code samples.
 - Students extend the code samples.

Resources:

- <http://www.georgebenainous.com/web> (javascript—basic scripting)

Javascript tutorial

- <http://www.w3schools.com/JS/default.asp>

Instructional Day: 2

Topic Description: Introduce javascript functions. Create modular, reusable code and use javascript to learn fundamental programming concepts.

Objectives:

The student will be able to:

- Use the correct syntax rules for creating functions in javascript.
- Create javascript math functions.
- Create javascript functions which apply css styling to a div.

Outline of the Lesson:

- Demonstration of javascript function code samples (10 minutes)
- Creation of a javascript function (10 minutes)
- Extension of code samples provided in tutorial website (35 minutes)

Student Activities:

- View javascript code samples from tutorial website.
- Create a simple javascript function.
- Extend code samples provided in tutorial website and create math functions.

Teaching/Learning Strategies:

- Demonstrate how to create a javascript function.
- Guide students in the creation of initial javascript functions.
- Extension of code samples provided in tutorial website
 - Demonstrate how to extend the code samples provided in the tutorial website
 - Students extend the code samples to create their own math functions a

Resources:

- <http://www.georgebenainous.com/web> (javascript—functions)

Javascript tutorial

- <http://www.w3schools.com/JS/default.asp>

Unit 4:

Introduction to Programming



Introduction

Programming is one of the creative processes that can transform ideas into reality. The intention of this unit is to highlight what can be created by using programming as a tool. As with the previous unit, students will create projects that reflect the diversity of interests in the classroom and that are personal to individual students.

Scratch provides an environment that lends itself to “tinkering”. The drag and drop nature of the blocks moves the focus away from messy syntax and allows for making modifications quickly. As students work through the unit, they should be encouraged to reflect on their tinkering and the thought processes that go into it. Engage in discussions of why a particular set of instructions didn’t work the way they thought they would and in discussions of “what if” scenarios. It is through these discussions that you can help students connect mathematics and logic to computation in programs and highlight the various abstractions they are using in creating their projects.

The projects listed are examples only and represent a minimal set of activities. When students complete the assigned projects, they should expand their work by adding features, collaborating with other students, and adding more personalization. There are many more projects on the Scratch website (<http://scratch.mit.edu>). The website is also a vibrant community. Students should be encouraged to become part of the community where they can collaborate beyond their classroom and get additional ideas for projects.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1	Introduce the Scratch programming language, including the basic terms utilized in the language.
2-3	Practice using the basic features of Scratch in the context of creating a simple program.
4	Create a dialogue between two sprites.
5-6	Introduce the methods of moving sprites in Scratch.
7-8	Practice the concept of event driven programming through the creation of an alphabet game.
9	Introduce the concept of broadcasting via role play.
10	Develop a story to be used in a Scratch program.
11-15	Write Scratch stories and present them to the class. Peer reviews are conducted.
16	Introduce the concept of variable.
17	Introduce the concept of conditionals.
18-19	Introduce And, Or and randomness.
20	Apply knowledge of conditionals to develop a Rock Paper Scissors program in Scratch.
21	Build on previous programming concepts to create a timer.
22-26	Create a timing game in Scratch and present it to the class. Peer reviews are conducted.
27	Investigate two types of games that may provide ideas for the final project.
28	Explain final project and the rubric for the final project.
29-33	Write Scratch programs for either My Community or Game project. Peer reviews will be conducted.
34-35	Presentations of final projects

Daily Lesson Plans

Instructional Day: 1

Topic Description: This lesson introduces the Scratch programming language, including the basic terms utilized in the language.

Objectives:

The students will be able to:

- Name the basic terms used in Scratch.
- Create the beginning of a simple program in Scratch.

Outline of the Lesson:

- Journal Entry (5 minutes)
- KWL chart about programming/Scratch (15 minutes)
- Scratch introductory video (10 minutes)
- Model of how to start name assignment (25 minutes)

Student Activities:

- Complete journal entry.
- Complete KWL chart about programming/Scratch.
- Groups take turns sharing out their K's and W's orally.
- Watch Scratch introductory video.
- Follow along with Scratch open as teacher models how to start name assignment.

Teaching/Learning Strategies

- Journal Entry: How do you think programs like Microsoft Word, Internet Explorer and Windows are made?
- KWL chart
 - Students meet with groups and each group completes a KWL chart. (Know, Want to Learn, Learned)
 - Groups take turns sharing out their K's and W's orally. They are encouraged not to repeat anything that has already been said.
 - Put KWL charts up in the classroom; tell students that they will finish the L towards the end of the unit.
- Scratch introductory video
 - Played with sound. Can be played over a projector.
- Model of how to start name assignment
 - Address how sound will be handled in the classroom.
 - Scratch lends itself to playing sounds so it can get noisy. The teacher needs to decide how to address this. Headsets with microphones are probably the best solution.
 - Build a name project similar to name.sb.
 - Emphasize
 - Every character in Scratch is called a Sprite.

- Although Scratch is programming, it is not used in industry. Point out a few languages that are used in industry—Java, C, C++. Throughout the unit, you will want to reinforce that the basic constructs used in Scratch are also used in “industrial strength” languages.
- How to choose a Sprite from a file
- How to paint your own sprite
- Each sprite has its own scripts.
- You can right click any block and select help to get more information on how to use it.
- How to change the language in Scratch (for your English Learners)
- How to go to full screen mode and back
- How to switch back and forth between sprites by clicking on them
- X and Y coordinates on the screen are shown on the bottom right below the stage
- How to save in the proper location (the default is to save in the Scratch Projects folder (C:\\Program Files\\Scratch\\Projects))
- The following blocks should be modeled:
 - Move _ steps
 - If on edge, bounce
 - Turn _ degrees
 - Forever
 - Change color effect by _
 - When the green flag is clicked
- Students should not be scared to just try and experiment. They can’t break the computer by dragging the wrong block.
- Show students where they can access ScratchGettingStarted.pdf. (It would probably be useful to have printed copies for each student.)
- Show students Name Rubric.

Resources:

- KWL Graphic Organizer Chart.pdf (UCLA SMP)
- ScratchIntro.wmv (scratch.mit.edu)
- ScratchGettingStarted.pdf (scratch.mit.edu)
- name.sb
- Name Sample Rubric
- <http://scratch.mit.edu>

Name Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
Have a separate sprite for each letter of your name.	5			
Have at least 3 different interesting behaviors for the letters in your name.	5			
All the letters have a behavior	4			
Use the “when green flag clicked” block	3			
Use the “forever” block	3			
Extra Credit				
Have your name reinitialize itself when the green flag is clicked. In other words, all the letters will start off in the right location facing the correct way.	2			
TOTAL:	20			

Instructional Days: 2-3

Topic Description: This lesson provides students an opportunity to practice using the features of Scratch outlined on Day 1 in the context of creating a simple program.

Objectives:

The students will be able to:

- Complete a simple Scratch program.
- Utilize the green flag feature.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Class discussion of journal entry (15 minutes)
- Name programs (90 minutes)

Student Activities:

- Complete journal entry.
- Share journal entry responses with the entire class.
- Write programs based on their own names.

Teaching/Learning Strategies:

- Journal Entry: What do you remember about Scratch from yesterday? What do some of the blocks do?
- Class discussion of journal entry
 - Allow students to share their responses.
 - In the process, make sure to review concepts needed to finish the name project.
 - Review rubric for name project.
 - Tell students that they will do a gallery walk of the projects at the beginning of tomorrow.
- Name programs
 - Students write programs based on their own names.
 - Teacher circulates room checking progress and answering questions.
 - Before time is up, remind students to save their work.
 - Remind students that Scratch is free to download at scratch.mit.edu.

Resources:

- ScratchGettingStarted.pdf (scratch.mit.edu)
- name.sb
- Name Sample Rubric
- <http://scratch.mit.edu>

Instructional Day: 4

Topic Description: This lesson describes how to create a dialogue between two sprites by first creating a written dialogue.

Objectives:

The students will be able to:

- Develop a dialogue between two or more Scratch sprites.
- Explain the reasoning behind how their dialogue works.

Outline of the Lesson:

- Gallery walk of name projects (10 minutes)
- Assignment introduction (5 minutes)
- Develop dialogue (30 minutes)
- Student presentations (10 minutes)

Student Activities:

- Participate in a gallery walk of name projects.
- Participate and listen to assignment introduction.
- Develop a dialogue.
- Present dialogues.

Teaching/Learning Strategies:

- Gallery walk of name projects
 - Facilitate gallery walk by giving the students an order or pattern to follow in walking around the room (dependent on lab).
- Assignment introduction
 - Tell the students that they'll be making a dialogue between two or more sprites.
 - Have a sample dialogue with a student; for example,
 - Teacher: Knock-Knock
 - Student: Who's there?
 - Teacher: Juana
 - Student: Juana who?
 - Teacher: Juana go write a program in Scratch!
 - Student: Ha ha!
 - Make a sample program using only "say _ for _ secs" blocks.
 - Ask the students what was the difference between the live dialogue and the sample program? (Answer is that in the program they are talking at the same time.)
 - Have the students help you find the "wait _ sec" block. Add a few so the students can see the sprites taking turns.
 - Show the students Dialogue Rubric and tell them that they can create their own dialogue. They can do their own knock-knock joke, or they can use their creativity.
- Develop Dialogue
 - Circulate room and help students.

- Student presentations
 - Have students volunteer to present their dialogues for the entire class.

Resources:

- Dialogue Sample Rubric

Dialogue Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
Have 2 or more sprites talking in dialogue.	4			
Have 3 or more sprites talking in dialogue.	5			
All the sprites are polite and they take turns talking	4			
Each sprite says at least 3 things.	3			
The conversation starts “when green flag clicked”	4			
Extra Credit				
Have 4 or more sprites talking in dialogue	2			
TOTAL:	20			

Instructional Days: 5-6

Topic Description: This lesson describes the methods of moving Sprites in Scratch.

Objectives:

The students will be able to:

- Explain the 3 major ways to move sprites.
- Choose the appropriate method of moving to make a cat circle the bases.

Outline of the Lesson:

- Journal Entry (5 minutes)
- moving.sb (20 minutes)
- Discussion of answers to questions (15 minutes)
- baseball.sb (70 minutes)

Student Activities:

- Complete journal entry.
- Examine moving.sb.
- Discuss answers to questions.
- Complete baseball.sb.

Teaching/Learning Strategies:

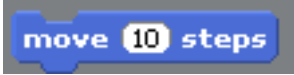


- Journal Entry: Briefly describe how you would graph in your Algebra class (The x-y coordinate plane, etc)
 - Time the students so they work 4 minutes individually and 1 minute sharing with their elbow partners.
- moving.sb
 - Circulate the room and help students answer the questions.
- Discussion of answers to questions
 - Emphasize that the “repeat” block will do whatever is inside it n times. This behavior can be called iteration or looping.
 - Point out that iteration is a construct that is used in other programming languages.
 - Emphasize the differences between the 3 ways to move.
 - Emphasize how the sprites will reinitialize themselves when the green flag is clicked.
- baseball.sb
 - Circulate the room and help students finish baseball.sb.
 - After a student can get the cat around the bases, encourage them to use the “point in direction” block to get the cat to turn the correct way when running.
 - If students need a hint for the extra credit, show them the “next costume” and “switch to costume” blocks under the “Looks” tab.

Resources:

- Moving Project
- Moving Project Solutions
- moving.sb
- baseball.sb
- baseball solution.sb

Moving Project

There are basically 3 ways to move sprites in Scratch. Try the file moving.sb and answer the questions below:

1. Click the green flag. What do the three animals do?
2. Look at the scripts for each of the 3 sprites. What 3 blocks do all three sprites use?
3. What blocks does the cat use to move?
4. What block does the dog use to move?
5. What block does the monkey use to move?
6. Describe in your own words how the move block works.
 
7. Describe in your own words how the go to xy block works.
 
8. Describe in your own words how the glide block works.
 
9. Some of the blocks require x: and y: coordinates. Place the mouse over the white window and look at the mouse x: and mouse y: numbers underneath the bottom. How are the x: and y: coordinates determined in Scratch?
10. Use what you've learned about moving to get the cat to run the bases (as realistically as possible – bases are run counter clockwise) in baseball.sb. Make sure that when you click the green flag, the cat starts at home plate again.
11. Extra Credit: Make the cat change costumes so that it looks like it is running as it circles the bases.

Moving Project Solutions

There are basically 3 ways to move sprites in Scratch. Try the file moving.sb and answer the questions below:

12. Click the green flag. What do the three animals do?

They move across the screen.

13. Look at the scripts for each of the 3 sprites. What 3 blocks do all three sprites use?

When green flag clicked, go to x: _ y: _ and wait _ sec

14. What blocks does the cat use to move?

Repeat _ and move _ steps

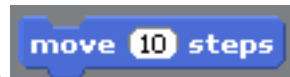
15. What block does the dog use to move?

go to x: _ y: _

16. What block does the monkey use to move?

glide

17. Describe in your own words how the move block works.



Move the sprite n steps. If n is positive, the direction will be to the right.

18. Describe in your own words how the go to xy block works.



Move immediately to that (x,y) position.

19. Describe in your own words how the glide block works.



Take n seconds to move from my current position to (x,y).

20. Some of the blocks require x: and y: coordinates. Place the mouse over the white window and look at the mouse x: and mouse y: numbers underneath the bottom. How are the x: and y: coordinates determined in Scratch?

It is just like the 2 dimensional x y graphs from Algebra. (0,0) is in the exact middle of the stage.

21. Use what you've learned about moving to get the cat to run the bases (as realistically as possible – bases are run counter clockwise) in baseball.sb. Make sure that when you click the green flag, the cat starts at home plate again.

22. Extra Credit: Make the cat change costumes so that it looks like it is running as it circles the bases.

Instructional Days: 7-8

Topic Description: This lesson introduces the concept of event driven programming and provides practice through the creation of an alphabet learning game.

Objectives:

The students will be able to:

- Explain event driven programming.
- Write a program that responds to user created events from the mouse and keyboard.

Outline of the Lesson:

- Presentation of solution for baseball extra credit (10 minutes)
- Journal Entry (10 minutes)
- Event lecture/description of Alphabet Learning Game (20 minutes)
- Alphabet Learning Game (60 minutes)
- Student presentations (10 minutes)

Student Activities:

- Present solution for baseball extra credit.
- Complete journal entry.
- Develop an Alphabet Learning game.
- Volunteers complete presentations.

Teaching/Learning Strategies:

- Presentation of solution for baseball extra credit
 - A student may present while others watch. If no student completed the extra credit, teacher presents. See baseball solution.sb from previous lesson.
- Journal entry: How do the programs on the computer know what the user wants to do next? In other words, if you are surfing the web, how does the computer know what page to go to next?
- Event lecture/description of Alphabet Learning Game
 - Allow some students to share journal entry with class. Steer them towards the idea of user events (clicks, typing) driving the program and causing it to respond.
 - Scratch provides some blocks that allow you to write programs that respond to user events relatively easily.
 - When green flag clicked (we've already seen this)
 - When Sprite clicked
 - When _ key pressed
 - Tell the students that they will be making an alphabet learning game.
 - Share Alphabet Sample Rubric with the students.
 - Create the first letter in front of the class with the students helping you. See alphabet learning.sb.
 - Show how to create new costumes.
 - Explain that students may bring in pictures from the internet.
 - Download a .gif or .jpg.
 - Use import or paint to make it the second costume for your letter.
 - Show how to change costumes.

- Use a “switch to costume _” block.
 - Show students how to output in talk bubbles.
 - Use a “say _ for _ sec” block.
 - Remind the student that they may pick the theme of alphabet game (animals, food, etc).
- Alphabet Learning Game
 - Circulate room and answer questions.
- Voluntary student presentations
 - Facilitate students in presenting.

Resources:

- alphabet learning.sb
- Alphabet Sample Rubric

Alphabet Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
Have at least 10 different letters.	4			
Have a theme for your letter game (i.e. animals, food, etc.)	3			
Sprites change costume when clicked on.	4			
Sprites change costume when letter is typed on keyboard	4			
Use the “say _ for _ sec” to output what the letter stands for (i.e. “E is for Elephant”)	3			
Sprites all turn to letters when the “when green flag clicked”	2			
Extra Credit				
Use a microphone to record sounds for all the letters and play the sound when the letter is clicked or typed (i.e. “E is for Elephant”)	2			
TOTAL:	20			

Instructional Day: 9

Topic Description: This lesson introduces the concept of broadcasting through role play and then provides students an opportunity to complete a broadcast event in Scratch.

Objectives:

The students will be able to:

- Broadcast their own events.
- Listen to and respond to their own events.
- Change the background of the stage.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Discussion of journal entry (2 minutes)
- Role Play (20 minutes)
- Scratch Summer Story (28 minutes)

Student Activities:

- Complete journal entry.
- Participate in discussion of journal entry.
- Participate in role play.
- Create a Scratch summer story.

Teaching/Learning Strategies:

- Journal Entry: What does it mean to broadcast something (example the radio station is broadcasting music right now)? If a radio or tv station is broadcasting something, does that mean that everyone is listening to it?
- Discussion of journal entry
 - Have a few students share their responses.
 - Stress that even though a lot of things are being broadcast, not everyone is listening to every thing that is being broadcast.
- Role Play
 - Solicit Volunteers to be the various characters.
 - Give the performers a paper with ONLY their part. See Scratch Broadcast Role Play.
 - Pass out the chart that shows all the parts to students that are not performing. See Scratch Broadcast Role Play Interwoven.
 - The students can think of it as a three act play where the scenes change. The difference here is that there are no curtains so they will see everything change.
 - The teacher will be the director and will make sure everything and everyone is in place during each scene. The teacher can yell action before the scene starts to signify that everything checks out.
 - Each performer's paper is broken into scripts for the various scenes.
 - One performer will be in charge of setting the stage. They can do this by erasing and drawing pictures on the white board behind the stage.
 - The Cat's first two scripts end with broadcasts. The cat will tell the director (teacher) that it is time to go on to the next scene.

- You might want to have different students perform the role play a second time. This time the teacher will only yell out when the green flag is clicked. The students can check themselves to make sure that everything is okay.
- Interesting Questions
 - Why do The Crab and the Date have only two scripts? (Answer: They remain hidden during the other scene.)
 - Instead of using broadcast, do you think you could just use “wait _ secs” blocks? (Answer: yes.)
 - What might be an advantage to using broadcast instead? (Answer: One advantage is that if your entire program is based on waits, if you edit something in scene 1, it could possibly throw the timing off for the rest of the program. Using broadcast can be simpler in the long run.)
- Discussion
 - In Scratch, any sprite can broadcast their own event.
 - One reason why The Cat is doing the broadcasts is because he is the last one to act in the first two scenes. Therefore he knows when the scene is over.
 - Other sprites (including the one that broadcasts the event) can receive the event and perform a script
- Scratch Summer Story
 - Show students
 - Directions: Summer Story Project
 - File to edit: summer.sb
 - Rubric: Summer Story Project Sample Rubric
 - Circulate the room and answer questions.

Resources:

- Scratch Broadcast Role Play
- Scratch Broadcast Role Play Interwoven
- Summer Story Project
- summer.sb
- Summer Story Project Sample Rubric

Scratch Broadcast Role Play

This is meant to be performed in front of a white board. This can also be done using more elaborate props. Each character's parts are broken down by events that are broadcast out to everyone. Select characters and give them their parts of the scripts. There is also a script so that observers can see the flow of the entire program.

Useful props: sunglasses, a basketball, and a bag of popcorn or chips

Characters:

The Cat: our main character

The Crab:

The Opponent:

The Date:

Stage: in charge of drawing the background of the scene on the board

Scripts for the individual actors:

The Cat

when GREEN FLAG clicked:

switch to costume: sunglasses

say: Hello!

say: I'm going to tell you about my summer.

say: I spent some time at the beach.

broadcast BASKETBALL SCENE (tell everyone it's time for the next scene)

when I receive BASKETBALL SCENE:

switch to costume: basketball

say: I played lots of ball.

broadcast MOVIE SCENE (tell everyone it's time for the next scene)

when I receive MOVIE SCENE:

switch to costume: bag of popcorn or chips

say: I went on a date. We went to the movies.

The Crab

when GREEN FLAG clicked:

show: (Go up on stage. You might want to pose like a crab by making your hands into claws.)

when I receive BASKETBALL SCENE:

hide: (Disappear from the stage)

The Opponent

when GREEN FLAG clicked:

hide: (Disappear from the stage)

when I receive BASKETBALL SCENE:

show: (Go up on stage. You might want to pose like a basketball player.)

when I receive MOVIE SCENE:

hide: (Disappear from the stage)

The Date

when GREEN FLAG clicked:

hide: (Disappear from the stage)

when I receive MOVIE SCENE:

show: (Go up on stage.)

wait 2 secs:

say: I can't wait to see Get Smart. Would you believe I spent my entire check from work on these tickets? No? Would you believe that it cost me \$15 for both using my student discount? No? How about a stick of gum and a nickel I found on the floor?

Stage

when GREEN FLAG clicked:

Switch to background BEACH: (Draw a picture of the beach on the white board. A sun in one corner and a wavy line for sand is fine.)

when I receive BASKETBALL SCENE:

Switch to background BASKETBALL COURT: (Draw a picture of a basketball court. Drawing the backboard and rim should be fine.)

when I receive MOVIE SCENE:


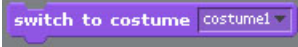


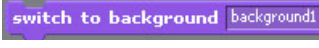
Switch to background MOVIES: (Draw a picture of a movie theater. Drawing a sign that says movies should be ok.)

Scratch Broadcast Role Play Interwoven

	when GREEN FLAG clicked:	when I receive BASKETBALL SCENE:	When I receive MOVIE SCENE:
The Cat	switch to costume: sunglasses say: Hello! say: I'm going to tell you about my summer. say: I spent some time at the beach. broadcast BASKETBALL SCENE	switch to costume: basketball say: I played lots of ball. broadcast MOVIE SCENE	switch to costume: bag of popcorn or chips say: I went on a date. We went to the movies.
The Crab	show: (Go up on stage. Pose like a crab.)	hide: (Disappear from the stage)	
The Opponent	hide: (Disappear from the stage)	show: (Go up on stage. Pose like a basketball player.)	hide: (Disappear from the stage)
The Date	hide: (Disappear from the stage)		show: (Go up on stage.) wait 2 secs: say: I can't wait to see Get Smart. Would you believe I spent my entire check from work on these tickets? No? Would you believe that it cost me \$15 for both using my student discount? No? How about a stick of gum and a nickel I found on the floor?
Stage	Switch to background BEACH: (Draw the beach.)	Switch to background BASKETBALL COURT: (Draw a basketball court.)	Switch to background MOVIES: (Draw a movie theater.)

Summer Story Project

Finish a story about what the cat did over summer. Answer questions 1, 2, and 6 on paper.

1. Open the file [summer.sb](#). Click the flag. What does it do so far?
2. Click on the cat and look at his script. What does the cat broadcast in the last block?
3. We'll make a basketball scene (a second script)
 - a. Drag a  block into the script section.
 - b. Click the empty box and choose "basketball scene".
 - c. Under looks, drag a  block into your script
 - d. Change costume1 to costume3
 - e. Give the cat something to say about playing basketball over summer.
 - f. Drag a  block to the end of this second script.
 - g. Click the empty box.
 - h. Choose new
 - i. Type in movie scene and hit ok.
4. We'll make the background change as well.
 - a. Click on the stage
 - b. Choose scripts
 - c. Drag a  block into the script section.
 - d. Click the empty box and choose "basketball scene".
 - e. Under looks, drag a  block into your script
 - f. Change background1 to basketball-court
5. Now add a third scene about going to the movies..
6. Summarize how you can use broadcast to change scenes in a story. Get your work checked off.
7. Now add in another character into each scene like in the role play (i.e. The Crab, The Opponent and The Date). These characters should show and hide.
8. Feel free to add in additional scenes.

Summer Story Project Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
Answer question 1, 2 and 6	5			
Add in the second scene (basketball)	5			
Add in the third scene (movies)	5			
Add in additional characters into each scene that show and hide	5			
Extra Credit				
Add in additional scenes	1			
TOTAL:	20			

Instructional Day: 10

Topic Description: Students will begin to develop their own stories and write them in Scratch.

Objectives:

The students will be able to:

- Start to develop a Scratch story.

Outline of the Lesson:

- Introduction of project (5 minutes)
- Journal Entry (5 minutes)
- Review of brainstorming (10 minutes)
- Scratch story (35 minutes)

Student Activities:

- Complete journal entry.
- Participate in discussion of brainstorming.
- Develop a Scratch story.

Teaching/Learning Strategies:

- Introduction of project
 - Show rubric: Story Project Sample Rubric.
 - Emphasize that they will make a small presentation along with showing their story.
 - Emphasize that there is extra credit for the best stories.
 - Show example: cat story.sb
- Journal Entry: Brainstorm some ideas for your story.
- Review of brainstorming
 - Split students into groups of three.
 - Have students rotate so that each student will share brainstorms and receive feedback/suggestions from the other students.
- Scratch story
 - Circulate room and help students with stories.
 - If students don't know where to start
 - Have them first develop their story on paper.
 - If they have their story and don't know where to start
 - Have them make a title screen or a first scene.

Resources:

- cat story.sb
- Story Project Sample Rubric

Story Project Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
The Story				
Have at least 3 scenes	10			
Have at least 4 scenes	5			
Have at least 3 different sprites	10			
Have at least 8 say or think boxes	10			
Animate the movement of your characters	5			
Use broadcast to change scenes in your story	10			
Have the characters take turns speaking to each other	5			
Have at least one conversation between characters	5			
Have a title scene with your name on it	10			
Story initializes itself when the flag is clicked	4			
The entire story plays once you click the flag	4			
The Presentation				
Explain an example from your program of how events (broadcast) were used to transition from one scene to another.	10			
Peer Grading	12			
Extra Credit				
Have the best project as voted on by peers	Up to 10			
TOTAL:	100			

Instructional Days: 11-15

Topic Description: Students will review how to broadcast events by developing a Scratch story and presenting it to the class.

Objectives:

The students will be able to:

- Broadcast events.
- Complete a Scratch story.
- Develop a Scratch story project.
- Assess their peers to help them gauge their progress.
- Complete a rubric.
- Prepare and make a presentation of a Scratch story to the class.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Review of broadcasting (5 minutes)
- Whole class review of broadcasting (5 minutes)
- Scratch story (40 minutes)
- Scratch story project (55 minutes)
- Peer review and discussion (15 minutes)
- Completion of Scratch story project (95 minutes)
- Presentation of stories (55minutes)

Student Activities:

- Complete journal entry.
- Review broadcasting with partner.
- Participate in whole class review of broadcasting
- Continue developing story.
- Develop Scratch story project.
- Participate in peer review and discussion.
- Complete Scratch story project.
- Present stories.

Teaching/Learning Strategies:

- Journal Entry: Write down everything you remember about using the “broadcast” block in Scratch and changing scenes in a story.
 - Have students do this silently on paper.
- Review of broadcasting
 - Have students pair up with elbow partners. One student shares first. The other student then shares their response adding to the first student’s response.
- Whole class review of broadcasting
 - Have a few students volunteer. Clear up any misconceptions. Refer back to the previous day’s role play as an example.

- Continue developing the story
 - Circulate room and help students with projects.
 - If students don't know where to start
 - Have them first develop their story on paper.
 - If they have their story and don't know where to start
 - Have them make a title screen or a first scene.
- Scratch story project
 - If students get stuck, break the project into smaller parts. Have them focus on the next part that they can complete.
 - Refer them to the rubric to make sure they know what they need to complete in order to earn the grade that they want.
- Peer review and discussion
 - Circulate the room and make sure students understand the rubric and what they still need to accomplish to finish their project.
- Completion of Scratch story project
 - Circulate room and help students with projects.
 - Collect projects and rubrics.
 - Help students prepare their presentations.
- Presentation of stories
 - Have students complete the Peer Grading sheet.
 - To help students vote on the best, you may need to do a quick recap of the stories, i.e. Bob's story about Poodles.
 - If the class does not finish presentations in one day, the voting will be done the next day.

Resources:

- Story Project Sample Rubric
- Peer Grading

Name _____ Computer # _____

VOTINGFrom **ALL** the projects, choose1ST Place _____2nd Place _____**PEER GRADING**For **EACH** of the following give the student a score from 1 to 4.

Use the rubric online to decide the score.

4 – Student has everything on the rubric: A**3** – Student has most things on the rubric: B**2** – Student has some things on the rubric: C**1** – Student turned in project, but is missing many items: D

Student Name	Score (1-4)		Student Name	Score (1-4)

Instructional Day: 16

Topic Description: This lesson provides an introduction to the concept of variables.

Objectives:

The students will be able to:

- Explain the concept of variables.
- Create examples of variables.

Outline of the Lesson:

- Finish Presentations (25 minutes)
- Journal Entry (5 minutes)
- Make Variable Example (15 minutes)
- Enhance Variable Example (10 minutes)

Student Activities:

- Finish Presentations.
- Complete journal entry.
- Participate in a discussion of the Make Variable example.
- Enhance the variable example.

Teaching/Learning Strategies:

- Finish Presentations
 - Have students fill out Peer Grading sheet.
 - To help students vote on the best, you may need to do a quick recap of the stories. You may also want to quickly replay some of the better ones.
 - Another option is to have students pick the best of each day and do a run off just replaying the top 3 from each day.
- Journal Entry: What does the word variable mean in both mathematical and English terms?
 - Time the students so they work 3 minutes individually and 2 minutes sharing with their elbow partners.
- Make Variable Example
 - Give two math examples. $x + 3 = 5$, $2x = 12$
 - Ask: What is the name of the variable here? (Answer: x)
 - Although you have x in both equations, its value **varies**: it is 2 in one equation and 6 in another.
 - The notion is the same in a program—a variable is a name that represents a value that can be changed. In the math example, the name was x .
 - Make the variable example with the students (variable example.sb) having the students help you and build their own at the same time. A possible sequence might be
 - Start by explaining that you want to make a game where you earn points for picking healthy foods and lose points for picking unhealthy ones.
 - What do you think the variable will be? If no answer, ask what name will represent a number that will change? (Answer: Points (or Good Nutrition Points in the example))
 - Add the sprites for the banana, cheesie poofs and text that says “Click on food to eat it”.
 - Ask: What tab do you think you should click on to make a variable? (Answer: Variables.)
 - Click “make a variable” calling it Good Nutrition Points.

- Clicking the checkbox next to the variable will show or hide it.
- Ask: If I want to make my points increase by 1 every time I choose the banana, how would I write that script? (See example.)
- As: How about making the points decrease by one when I click on the cheesie poofs? (See example.)
- Ask: What do you think should happen when the green flag is clicked? (Answer: reset the points to 0.)
 - Ask: How do you think we should do that? (In example the script for this is under Sprite4)
 - Ask: Does it matter which script the “when green flag clicked” is under? (Answer: No.)
- Enhance Variable Example
 - Have Students enhance the variable example by
 - Adding a food that is worth 2 points when clicked on.
 - Adding a food that is –3 points when clicked on.

Resources:

- Peer Grading

Instructional Day: 17

Topic Description: This lesson provides an introduction to the concept of conditionals.

Objectives:

The students will be able to:

- Explain the concept of conditionals.
- Enhance a variable program with conditionals.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Conditional lecture (15 minutes)
- Age program (10 minutes)
- Age solutions (5 minutes)
- Enhance variable example (20 minutes)

Student Activities:

- Complete journal entry.
- Participate in discussion of conditions.
- Develop an Age program.
- Review Age solutions.
- Enhance the variable example.

Teaching/Learning Strategies:

- Journal Entry: What comes to mind when you hear the word “if”? What are some ways we use the word “if” in English?
 - Time the students so they work 3 minutes individually and 2 minutes sharing with their elbow partners.
- Conditional lecture
 - Have a few students share their responses for the “if” parts and use that as a springboard.
 - Basically in English, if is used to state a condition where something might happen if the condition is true. Hence this topic is called conditionals. Point out that this is a common computer science construct.
 - An example from computing is when a program like Microsoft Word asks you if you want to save your work when you hit close. If you click yes, it saves your changes. If you click no, it discards your changes.
 - if (some condition)
 - then do this
 - Show students “if” block in Scratch.
 - Notice that only hexagon shaped blocks can fit within it.
 - Notice that if the condition is true, it will do anything that is enclosed within the top and bottom of the “if” block.
 - Show the students age.doc and age.sb.
 - Remind students that since we are using integers (whole numbers) > 15 it means people that are over 15 not including 15.
 - Show them how to use the slider to change the age.
- Age Program
 - For solution, see age solution.sb.

- Age solutions
 - Show a solution like age solution.sb.
 - Show an alternate solution.
 - Since numbers are integers (whole numbers) we can do “age > 2” to mean “age >= 3”.
 - To do >= in scratch, you need to use the “or” block. See age – greater-equal.sb.
- Enhance variable example
 - Instruct students to go back into their variable example about nutrition and add:
 - A message about being nutritious if the amount of points becomes greater than 9.
 - A message about eating healthier food if the points becomes less than –4.
 - They can either have a sprite say the message or use broadcast to change the sprites/stage to convey the message.

Resources:

- Age Project
- age.sb
- age solution.sb
- age – greater-equal.sb
- variable example.sb

Age Project

You are going to finish a program that will tell you what you can do depending on your age. Use the slider to set the age.

1. Currently, it only does the first condition. Your task is to finish the program so that the cat will tell you the rest:

If you are older than 2 "you don't need diapers"

If you are older than 15 "you can drive"

If you are older than 16 "you can see an R rated movie"

If you are older than 17 "you can vote"

If you are older than 20 "you can gamble"

If you are older than 24 "you can rent a car"

If you are older than 49 "you can retire"

2. If the age is less than 3, make the code tell you:
"Sorry, you are not old enough for anything yet"

3. Feel free to add more conditions.

Instructional Days: 18-19

Topic Description: This lesson introduces And , Or and randomness. Students have an opportunity to practice utilizing these features in the context of programs.

Objectives:

The students will be able to:

- Use conditionals with And and Or to write a grade program.
- Use a random number generator to write a dice program.

Outline of the Lesson:

- Journal Entry (5 minutes)
- And/Or discussion (15 minutes)
- Grades program (35 minutes)
- Random lecture (20 minutes)
- Dice (35 minutes)

Student Activities:

- Complete journal entry.
- Participate in And/Or discussion.
- Develop Grades program.
- Participate in discussion of Random.
- Complete Dice program.

Teaching/Learning Strategies:

- Journal Entry: What's the difference between And and Or? What does the word random mean in English?
 - Teacher times the students so they work 4 minutes individually and 1 minute sharing with their elbow partners.
- And/Or Discussion
 - Start with a few journal entries about And and Or.
 - Kinesthetic And/Or Activity (Following is a possible set of conditions.)
 - Tell the students to stand up if the condition is true.
 - Say: If (you are a girl AND you are wearing blue) stand up.
 - Find a girl that is not wearing blue and is sitting. Ask her why she is sitting if she's a girl? (Answer: she's not wearing blue)
 - Ask: How many parts of the condition must be true for you to stand up if it is an AND? (Answer: both)
 - Say: If (you are a boy OR you are wearing blue) stand up.
 - Find a boy that is standing but is not wearing blue. Ask: Why are you standing if you are NOT wearing blue? (Answer: I'm a boy)
 - Ask: How many parts of the condition must be true for you to stand up if it is an OR? (Answer: at least one)
 - Ask: If both parts of the condition are true for an OR, is it ok to stand? (Answer: YES!)

- Show the students the “and” and “or” blocks in Scratch.
 - Emphasize how they are hexagon shaped and take two other hexagons.
- Show the students Grades Project.
- Grades
 - Circulate and help students with projects.
 - If many students are stuck, build the “B” part of the code together as a class.
 - In the last minute, have students share their solutions with their elbow partners.
- Random lecture
 - Have a few students share their journal entries about what random means.
 - Ask: if I roll a pair of dice, will the numbers come out in order (2, then 3, then 4 the next roll, etc.) (Answer: Most likely not)
 - Roll a pair of dice a few times to prove it.
 - This unpredictability is called randomness.
 - Randomness can make games more exciting.
 - For example, how many spaces will I get to move this turn?
 - Walk students through dice.sb showing them the “pick random _ to _” block.
 - Explain that the numbers create the range that the random integer can fall under. The block works inclusively. Therefore 1 to 6 will produce numbers 1,2,3,4,5,6.
- Dice
 - Instruct students to finish dice.sb so that it creates a pair of dice. They can create their own look for the dice.
 - Circulate and help students with projects.
 - In the last minute, have students share their solutions with their elbow partners.

Resources:

- Grades Project
- grades solution.sb
- dice.sb
- dice solution.sb

Grades Project

Your task is to make a Scratch program that will tell you the letter grade based on the percentage.

1. Create a variable grade.
2. Double click grade to display the scroll bar.
3. When the green flag is clicked, the program should look at the value of grade and the sprite should respond with a letter as follows:
 - A: greater than 90
 - B: greater than 79 and less than 90
 - C: greater than 69 and less than 80
 - D: greater than 59 and less than 70
 - F: less than 60

At Crazy High School, students only qualify for tutoring if they have a B OR a D. After it says the grade, make your program say “You qualify for tutoring” if the grade is a B or D.

Instructional Day: 20

Topic Description: This lesson requires students to apply their knowledge of conditionals to develop a Rock Paper Scissors program in Scratch.

Objectives:

The students will be able to:

- Apply knowledge of conditionals to complete a Rock Paper Scissors program.

Outline of the Lesson:

- Review of Rock Paper Scissors rules (5 minutes)
- Rock Paper Scissors discussion (10 minutes)
- Rock Paper Scissors project (40 minutes)

Student Activities:

- Review Rock Paper Scissors rules.
- Participate in Rock Paper Scissors discussion.
- Complete Rock Paper Scissors project

Teaching/Learning Strategies:

- Review of Rock Paper Scissors rules
 - Lead a class discussion—students volunteer to share the rules for Rock, Paper Scissors.
- Rock Paper Scissors discussion
 - Give students a tour of rps starter.sb.
 - Show them how there are variables for ROCK, PAPER and SCISSORS.
 - Ask: Why might it be easier to work with the variables instead of just using numbers? (Answer: It makes the code easier to read.)
 - Show students variables for player and computer.
 - Ask how does the computer determine if they will choose rock, paper, or scissors? (Answer: It randomly chooses one using “pick random 0 to 2”.)
 - Closely examine the computer’s “when I receive showPick” script
 - Explain how the else part works if the condition of the if is false.
 - Ask: Why don’t you need a statement that says “if computer = scissors”? (Answer: You asked if it was = to rock and that was false, then you asked if it was equal to paper and that was false so the only thing left was for it to equal scissors. Hence, you can just put the “switch to costume scissors” in the else.)
 - Instruct students that they only need to change the script that starts with “When I receive determine winner” under the computer sprite. (They may change more to make the program more fancy.)
 - Facilitate them in writing some pseudo code to handle all the cases for the computer choosing ROCK.
 - Create two versions, one like rps solution.sb and one like rps solution b.sb. This way students can choose the method that they understand better.
 - Show students a working example in presentation mode (so they can’t see the blocks).
- Rock Paper Scissors project

- Circulate room and help students with projects.
- Allow students to try various approaches to solving the problem.
- If students finish, offer them extra credit for keeping score of the wins for the computer and player.

Resources:

- rps starter.sb
- rps solution.sb
- rps solution b.sb

Instructional Day: 21

Topic Description: This lesson builds on previous concepts to create a timer.

Objectives:

The students will be able to:

- Create a timer.

Outline of the Lesson:

- Review of Rock Paper Scissors solutions (10 minutes)
- Creation of a timer (15 minutes)
- Review of Timer solutions (5 minutes)
- Introduction of Timing Game (15 minutes)
- Timing Game theme (10 minutes)

Student Activities:

- Review Rock Paper Scissors solutions.
- Create a timer.
- Review Timer solutions.
- Choose Timing Game theme.

Teaching/Learning Strategies:

- Review of Rock Paper Scissors solutions
 - Review rps solution.sb and rps solution b.sb.
 - Allow students to share their own unique solutions.
- Creation of a timer
 - Explain to students that they will make a timer that will count down from 10 to 0.
 - Show students Timer Project.
- Review of Timer solutions
 - Allow students to share their own unique solutions.
 - Review timer solution a.sb and timer solution b.sb.
- Introduction of Timing Game
 - Have students help build an example. (See timing.sb.)
 - Review Timing Game Sample Rubric.
- Timing Game theme
 - Circulate room and help students pick the theme of their timing game.

Resources:

- rps solution.sb (modified version of Jesse Moya's solution)
- rps solution b.sb (modified version of Jesse Moya's solution)
- Timer Project
- Timing Game Sample Rubric
- timer solution a.sb
- timer solution b.sb

- `timing.sb`

Timer Project

How to make a timer in Scratch:

1. Create a variable called timer.
2. When the flag is clicked, initialize the timer to 10.
3. Continually, wait 1 second and check if the timer = 0
 - a. output the current time either with a sprite or just show the variable
 - b. If the timer = 0 make either the background or a huge sprite say “Time’s Up”
4. When the flag is clicked, everything should start over.
5. Be creative as to what you want your program to look like.
6. Make sure the timer stops at 0 and does not continue into negatives.

Timing Game Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
The Game				
Have 3 or more “timed” sprites	10			
Have 4 or more “timed” sprites	5			
Use a timer for your game	5			
Keep score (points)	10			
Give the user feedback as to how well they timed their button pressing	10			
Have a help screen with directions	5			
Does the game reset when the flag is clicked	10			
Does the game stop when it is over	5			
Does the game notify the user when it is over	10			
Does the game keep track of how many “perfects” in a row	5			
Does the game get harder as you keep playing	5			
Peer Grading	20			
Extra Credit				
Have the best project as voted on by peers	Up to 10			
TOTAL:	100			

Instructional Day: 22-26

Topic Description: Students create a timing game in Scratch and participate in an Arcade Day during which they display their games.

Objectives:

The students will be able to:

- Create a timing game.
- Assess their peers to help them gauge progress.
- Complete their rubrics and submit their timing games.
- Prepare a presentation of a Scratch program.
- Evaluate their peers' timing games.

Outline of the Lesson:

- Timing game (110 minutes)
- Peer Review and discussion (15 minutes)
- Completion of timing game (95 minutes)
- Arcade walk (55 minutes)

Student Activities:

- Work on timing game.
- Participate in peer review and discussion.
- Continue working on and complete timing game.
- Participate in arcade walk.

Teaching/Learning Strategies:

- Work on timing game
 - Circulate room and help students with projects.
- Peer review and discussion
 - Circulate the room and make sure students understand the rubric and what they still need to accomplish to finish their project.
- Completion of timing game
 - Circulate room and help students with projects.
 - Collect projects and rubrics.
 - Help students prepare their presentations.
- Arcade Walk
 - Have students rotate through the room playing each other's games and giving each one a score on their Peer Grading sheet. Use a timer to indicate the amount of time that each student has at each computer.
 - Have students vote for the top two games out of the entire class.

Resources:

- Timing Game Sample Rubric
- Peer Grading

Instructional Day: 27**Topic Description:** Investigating Games**Objectives:**

The students will be able to:

- Investigate two different types of games.
- Get ideas for their final projects.

Outline of the Lesson:

- Monkey game (25 minutes)
- Review of answers (5 minutes)
- Pinball game (25 minutes)

Student Activities:

- Complete Monkey game.
- Review answers.
- Complete Pinball game.

Teaching/Learning Strategies:

- Monkey game
 - Have students answer the questions in Monkey Game Project.
 - Have students enhance monkey game.sb.
- Review answers
 - See Monkey Game Project Solutions and monkey game solution.sb.
- Pinball game
 - Have students answer questions in Pinball Project.
 - Have students enhance pinball.sb.

Resources:

- Monkey Game Project
- Monkey Game Project Solutions
- monkey game solution.sb
- monkey game.sb
- Pinball Project
- Pinball Project Solutions
- Pinball.sb (An example that comes with Scratch)

Monkey Game Project

Answer these questions on paper:

1. Play the game by using the arrow keys. What blocks make the monkey respond to the keys?
2. Does the banana always appear in the same place?
3. What blocks do you think decide what x and y the banana should change to?
4. What are the names of the orange blocks under Variables?
5. What block(s) are used to change the score?

Make these changes to the file:

6. Customize the sprites in the game (make the characters be who you want).
7. Add another sprite that gives you 2 points if you touch it.
8. Get the game to stop at 10 points or more by telling you that you win.

Monkey Game Project Solutions

Answer these questions on paper:

9. Play the game by using the arrow keys. What blocks make the monkey respond to the keys? "when _ key pressed"
10. Does the banana always appear in the same place? No, it's random.
11. What blocks do you think decide what x and y the banana should change to? "set x to _" and "set y to _" combined with "pick random _ to _"
12. What are the names of the orange blocks under Variables? "change points by _", "set points to _", and "points"
13. What block(s) are used to change the score? "set points to 0" when the green flag is clicked and "change points by 1" when the monkey touches the banana.

Make these changes to the file: [see monkey game solution.sb](#)

14. Customize the sprites in the game (make the characters be who you want).
15. Add another sprite that gives you 2 points if you touch it.
16. Get the game to stop at 10 points or more by telling you that you win.

Pinball Project

Open pinball.sb and answer the questions below on paper:

1. Look at the scripts for the pinball. How did the author simulate gravity?
2. How does the ball know when to “bounce” off of something?
3. Does the ball always bounce the same way when it hits something?
4. How do you think the ball determines which direction to bounce?
5. What’s the purpose of the purple line at the very bottom of the game?
6. Modify the game to keep track of points and get it checked off. Write down what changes you made.
7. What other features do think would make this game better?

Pinball Project Solutions

Open pinball.sb and answer the questions below on paper:

1. Look at the scripts for the pinball. How did the author simulate gravity? There is a variable called gravity that is constantly affecting the direction of the ball.
2. How does the ball know when to “bounce” off of something? If it touches something that is green, orange or red, it will “bounce” off of it.
3. Does the ball always bounce the same way when it hits something? No, the amount of the turn is random.
4. How do you think the ball determines which direction to bounce? It uses “pick random _ to _” to vary the amount of the turn.
5. What’s the purpose of the purple line at the very bottom of the game? If the ball touches purple, you lose.
6. Modify the game to keep track of points and get it checked off. Write down what changes you made. See pinball solution.sb
7. What other features do think would make this game better? Answers will vary.

Instructional Day: 28

Topic Description: Introduce the final project.

Objectives:

The students will be able to:

- Make an appropriate choice of which final project they will do.

Outline of the Lesson:

- Review of answers for Pinball Project (5 minutes)
- Introduction of projects (15 minutes)
- Final projects (35 minutes)

Student Activities:

- Review answers for Pinball project.
- Participate in discussion of introduction to projects.
- Start Final Projects.

Teaching/Learning Strategies:

- Review of answers for Pinball project.
- Introduction of projects
 - Review both sets of descriptions and rubrics.
- Final projects
 - Circulate the room
 - Help students decide which project to do.
 - If students are making the game, tell them to start off small and add features as they go along to ensure that they will finish.

Resources:

- Pinball Project
- Pinball Project Solutions
- Game Project Sample Rubric
- My Community Project
- My Community Sample Rubric

Instructional Day: 29-32

Topic Description: Complete final projects.

Objectives:

The students will be able to:

- Incorporate all objectives in the unit into the final project.

Outline of the Lesson:

- Work on final project (110 minutes)
- Peer review and discussion (15 minutes)
- Completion of final project (40 minutes)

Student Activities:

- Work on final project.
- Participate in peer review and discussion.
- Complete final project.

Teaching/Learning Strategies:

- Work on final project
 - Circulate room and help students with projects.
- Peer Review and discussion
 - Pair students with someone that has a similar project if possible (communities together and games together).
 - In pairs, students take turns using a rubric to grade the other person's project. Students should be able to get an idea of where their grade stands now, and what else they need to do to complete their project.
 - Circulate the room and make sure students understand the rubric and what they still need to accomplish to finish their project.
- Completion of final project
 - Circulate room and help students with projects.

Resources:

- Game Project
- My Community Project
- Game Project Sample Rubric
- My Community Sample Rubric

Instructional Day: 33

Topic Description: Complete final projects.

Objectives:

The students will be able to:

- Finish their KWL charts.
- Complete their rubrics and turn in their final projects.
- Prepare their presentations.

Outline of the Lesson:

- KWL chart (15 minutes)
- Completion of final projects (40 minutes)

Student Activities:

- Complete KWL chart.
- Groups take turns sharing their L's orally.
- Finish final projects.

Teaching/Learning Strategies:

- KWL chart
 - Refresh memory on activity.
 - They already did the K (Know) and W (Want to learn), now they must fill in the L (Learned) section of their chart.
 - They should meet in their original groups.
 - Students fill in the L portion of their chart.
- Groups take turns sharing out their L's orally. Encourage them not to repeat anything that has already been said.
- Completion of final projects
 - Collect projects and rubrics.
 - Help students prepare their presentations.

Resources:

- KWL Graphic Organizer Chart.pdf (UCLA SMP)

Instructional Day: 34

Topic Description: Complete My Community presentations.

Objectives:

The students will be able to:

- Complete a presentation on the My Community project.

Outline of the Lesson:

- My Community Presentations (55 minutes)

Student Activities:

- Complete My Community presentations.

Teaching/Learning Strategies:

- My Community Presentations
 - Have students that elected to write programs about their community take turns presenting. They will present their community as well as explain how they created the projects.
 - Help guide students by asking questions if the student does not fully explain how they wrote their program.
 - Have students in audience fill out Peer Grading.
 - Have students vote for first and second place.

Resources:

- Peer Grading
- My Community Project Sample Rubric

Instructional Day: 35

Topic Description: Complete presentations of Game projects.

Objectives:

The students will be able to:

- Complete a presentation on the Game project.

Outline of the Lesson:

- Games Presentations (35 minutes)
- Arcade Walk (20 minutes)

Student Activities:

- Complete Games presentations
- Participate in an arcade walk.

Teaching/Learning Strategies:

- Games Presentations
 - Have students take turns presenting their games to the rest of the class. They must walk the class through how they created their game.
 - Help guide students by asking questions if the student does not fully explain how they created their game.
 - Have students in audience fill out Peer Grading.
- Arcade Walk
 - Have students rotate through and play the games that were presented
 - For example, teacher might allocate 1 minute per game and have the students rotate to the next one.
 - Have students vote for first and second place.

Resources:

- Peer Grading
- Game Project Sample Rubric

Final Project

Choose one of the following:

My Community Project

You will use Scratch to make a project about your community. You will use blocks like broadcast in your project. You should have at least three different pages or scenes in this project.

Decide on one positive thing that you want to highlight and one thing you want to improve about your community. Then find at least one statistic to backup your conclusions. Also include at least one personal comment/recording and one picture. Lastly, you should have at least one observation from someone else in the class about the topic – this means you will have to ask them what they think and either record it or write it down.

Use these websites to find statistics about Los Angeles and California:

For Education Data go to:

<http://dq.cde.ca.gov/dataquest/>

<http://www.ed-data.k12.ca.us/welcome.asp>

For information about other concerns in LA, like hunger and homelessness, go to:

<http://www.unitedwayla.org/GETINFORMED/RR/Pages/default.aspx>

<http://www.unitedwayla.org/getinformed/rr/datalinks/Pages/default.aspx>

For information about health and health care go to (if you need a password, let me know):

<http://www.chis.ucla.edu/>

For Los Angeles Sponsored web sites go to:

<http://www.ci.la.ca.us/>

Community Project Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
The Content				
3 or more scenes	10			
1 or more positive things in the community that you want to highlight	5			
1 or more things in the community that you would like to improve	5			
Statistics to back up your conclusions	5			
A personal comment/recording	5			
A personal picture	5			
A comment/recording from someone else in the class	5			
The Program				
Use broadcast	10			
Scenes only show sprites that are supposed to be in that scene	10			
Program starts and restarts when green flag is clicked	10			
The Presentation				
Show and explain the contents of each scene in your project	9			
Explain how each scene changes (how the program works)	9			
Peer Grading	12			
Extra Credit				
Have the best project as voted on by peers	Up to 10			
TOTAL:	100			

Create A Game Project

Create your own game that meets the specifications outlined in the rubric. Be creative!

Game Project Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
The Game				
Let the player know if they win	10			
Keep score	10			
Have a timer	10			
Have a help screen with directions	10			
The game resets when the green flag is clicked	10			
The game stops when it is over	10			
Does the game get harder as you keep playing (more than one level)	10			
The Presentation				
Brief description of the game explaining how it was programmed using Scratch (what blocks you used, etc.)	10			
Peer Grading	20			
Extra Credit				
Have the best project as voted on by peers	Up to 10			
TOTAL:	100			

Unit 5:

Robotics



Introduction

Robotics provides a physical application of the programming and problem solving skills acquired in the previous units. The LEGO® Mindstorms NXT software uses drag and drop programming which will provide a natural transition from Scratch. Robots are shared by several students which will emphasize the collaborative nature of computing. In order to design, build and improve their robots, students will need to apply effective team practices and understand the different roles that are important for success.

Discussing the features of robots provides an opportunity to emphasize how computing has far-reaching effects on society and has led to significant innovation. Students can discuss such topics as:

- The effects innovations in robotics have had on people.
- The significance of processes that have been automated because of robots.
- How innovations in robotics have spurred additional innovations.

The unit consists of three main sections:

- The features of robots (Days 1-3)
- Familiarization with the robot and the software (Days 4-14)
- Robotics projects (Days 15-40)

Throughout the unit the similarities and differences between Scratch and the programming needed to move the robot can be highlighted.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1	What is a robot? Identify the criteria that make an item a robot
2-3	Evaluate robot body designs and create algorithms to control robot behavior.
4	Set up LEGO® Mindstorms® NXT® kit.
5	Build robot base.
6-7	Introduce the features of NXT Brick—the “brain” of the robot.
8-9	Introduce the features of the Mindstorms NXT software.
10-14	Program the robot using the Mindstorm Robot Educator Software tutorials.
15	Introduce RoboCup real life robotic competition and write instructions for tic-tac-toe.
16	RoboTic-Tac-Toe Tournament and introduction to RoboCupJunior Dance Challenge.
17-20	Build, program, and present a dancing robot.
21-25	Build program and present a rescue robot.
26-30	Design, build and program a robot that solves a stated problem. (Optional—time permitting)
31-40	Final projects and presentations

Daily Lesson Plans

Instructional Day: 1

Topic Description: “What is a Robot”? Identify the criteria that make an item a robot.

Objectives:

Students will be able to:

- List and explain the criteria that describe a robot.
- Determine if something is a robot, using the criteria.

Outline of the Lesson:

- Brainstorm about robot definition (10 minutes)
- “Kismet” video (5 minutes)
- Elements of a robot (10 minutes)
- Am I a Robot? Activity 1 (15 minutes)
- Student group work—Are we Robots? (15 minutes)

Student Activities:

- Brainstorm what they think of when they hear “robot” and then identify common features of robots.
- Participate in whole class activity determining if common items are robots.
- Work in small groups to complete “Are we Robots?” activity.

Teaching/Learning Strategies:

- Brainstorm: Ask students what they think of when they hear “robot”. Display responses. Responses may include the following:
 - Movie and TV robots such as Wall-E, iRobot, Robots, Rosie from The Jetsons
 - Modern industrial robots such as those involved in assembly-line factory work
 - Mars Rovers
 - iRobot robots, both the vacuum cleaner and the robots built for military use, other robots such as bomb detection and detonation
- View the video “Kismet” from Teachers Domain.
- Ask students if they can identify common features of the robots they have identified. What do all those robots have in common? What tasks are easy for robots? What tasks are hard for robots? (Answers: robots are often used for dangerous or repetitive tasks such as recovering bombs, search and rescue operations in dangerous conditions where the robots search and the humans rescue, factory work. They are replaceable, unlike humans, and don’t get bored or make

mistakes when doing the same thing over and over. Tasks that require judgment or human-like interaction such as recognizing when there is a problem or walking and seeing like humans are hard for robots. The two articles listed in the resource section provide more information and would be interesting for students to read.)

- Use the What is a Robot? handout to guide a discussion of robots.
- Hand out copies of Am I a Robot? activity, with the pictures of a basic stove and a fancy microwave. Check with students to make sure they recognize the items in the two pictures. Based on student input, display the five criteria for whether something is a robot: body, input, program, output, behavior. Note that what distinguishes a robot from a programmable device is the ability to respond to changes in the environment and adapt; robots respond to. Explain to the class that as a group you will figure out whether each of the two machines shown is a robot. Go through the stove first. Ask students to figure out whether the stove meets the criteria for a robot:

Body—yes

Input—yes (dials to turn the burners off and on, set oven temp)

Programmable—yes, in the sense that oven temperature tells a sensor what temperature the oven needs to be heated.

Output—yes (heat!)

Behavior—yes, the oven responds by stopping at the desired temperature. It also adapts to changes as in opening the oven door, adding a frozen item, etc. by adding more heat to get back to the desired temperature.

Next go through the microwave in a similar way:

Body—yes

Input—yes (buttons)

Programmable—yes (buttons set time, set mode, microwave can be programmed by the user, for example “cook 3 minutes 50% power, hold 1 minute, cook 1 minute 90% power)

Output—yes (microwaves in chamber, light comes on)

Behavior—yes (cooks food, makes popcorn, boils water...)

Question: Does a microwave adapt?

- Hand out copies of Are we Robots? activity two. Explain the directions. Either have students brainstorm machines as a group to complete the table or have them think of machines on their own. Have students work in small groups to complete the table, determining whether each machine is a robot according to the criteria.
- Optional Extra Credit—have students research Isaac Asimov’s three Laws of Robotics. What are the three laws? What is law Zero? Why did he come up with these laws and how do they think these laws affect our thinking about robots today?
Law Zero: A robot may not injure humanity, or through inaction, allow humanity to come to harm
Law One: A robot may not injure a human being, or through inaction, allow a human being to

come to harm

Law Two: A robot must obey the orders given to it by human beings, except where such orders conflict with Law One

Law Three: A robot must protect its own existence, as long as such protection does not conflict with Laws One and Two.

Resources:

- Jennifer Casper and Robin Murphy, Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center, IEEE Transactions on Systems, Man and Cybernetics 33:3, 2003, pp. 367-85.
- Robin Murphy, J. Kravitz, S. Stover and R. Shoureshi, Mobile robots in mine rescue and recovery, IEEE Robotics & Automation Magazine 16:2, June 2009, pp. 91-2003.
- What is a Robot? Handout (Based on handouts from The Big Picture “Robotics Teacher Guide 1” (Item #29852 from LEGO Dacta))
- Am I a Robot? Activity
- Are we Robots? Activity (Based on handouts from The Big Picture, “Robotics Teacher Guide 1” (Item #29852 from LEGO Dacta))
- http://www.teachersdomain.org/resources/eng06/sci/engin/design/lp_robot/index.html specifically
<http://www.teachersdomain.org/resources/eng06/sci/engin/design/kismet/index.html> (may require free registration)
- Asimov’s three laws of robotics: http://en.wikipedia.org/wiki/Three_Laws_of_Robotics,
http://www.asimovonline.com/asimov_FAQ.html#series13 , essay at
<http://www.sfwriter.com/rmasilaw.htm>

What is a Robot? Handout

There are many different kinds of robots, from ones designed to build cars to ones that vacuum to ones that explore other planets. To be a robot, a machine must meet certain criteria. A machine is only a robot if it has all the elements listed below:

Body

The body is a physical substance and shape of some type. The body will be designed based on the function – some look like vehicles, some like an arm, some like a person. If you can touch it, that's the body.

Control

Control is a program to control the robot. Robots must be told what to do. To control a robot we need:

Input

Input is the information that comes from the robot's sensors. Robots have sensors that they use to get information from the robot's environment. For example, a smoke detector can detect smoke. (In other words, sensing the robot's environment). Robots typically have external and internal sensors.

Programmable

The program is a set of instructions or rules that the programmer gives the robot. For example, a smoke detector has a program to make a sound if it senses smoke. To be a robot, a machine must be programmable.

Output

The output is the action a robot takes, often involving motors, lights, or sounds. For example, a smoke detector makes a loud sound and might flash lights. (In other words, effecting change in the robot's environment—adapting.)

Behavior

Behavior is the combination of outputs that result in the task or job the robot does. For example, the behavior of a smoke detector is to “go off” in the presence of smoke. “Going off” is a combination of making noise and flashing lights, and may also involve calling the fire department.

Am I a Robot? Activity

Image 1: Basic Stove

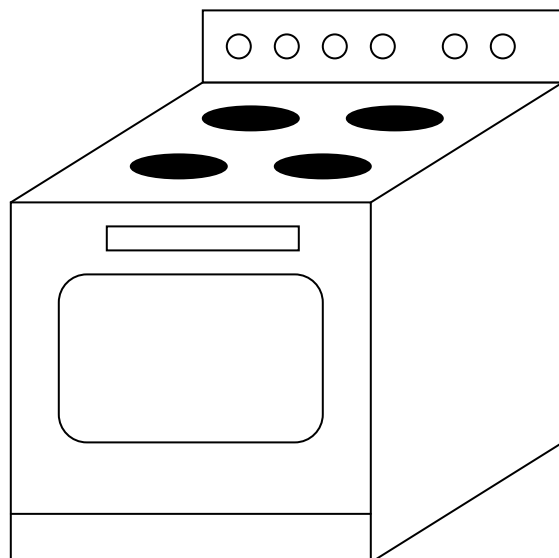
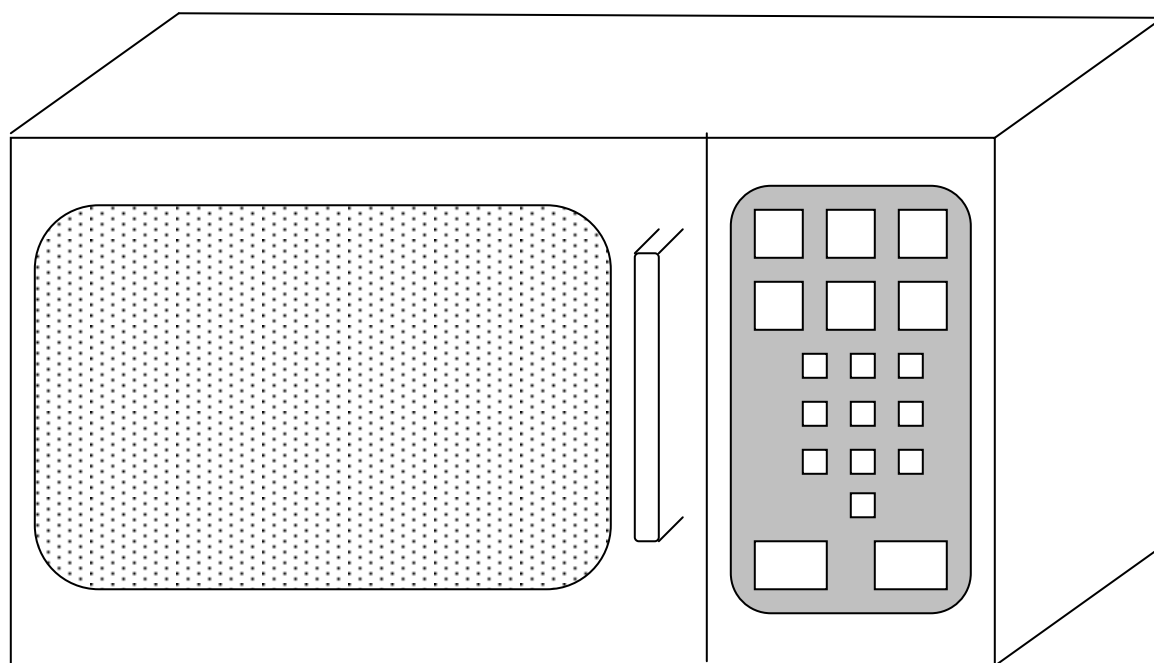


Image 2: New Microwave



Are we Robots? Activity

Instructions: Below is a list of machines that you may encounter in your daily life. Add machines to the bottom. Complete the table by deciding if the machine meets the criteria for being a robot. Then determine if the machine is a robot.

Body—physical form of some kind

Control—

Input—gets information from sensors, buttons, etc.

Program—Is programmable, follows a set of instructions you give it

Output—an action it takes

Behavior—what it does; the function it performs

	Body	Input	Program	Output	Behavior	Is it a robot?
Stove						
Microwave						
Radio						
iPod						
Flashlight						
Bicycle						
Car						
Alarm clock						
Traffic light						
Photocopier						
Computer						
Mars Rover						

Instructional Days: 2-3

Topic Description: Evaluate robot body designs and create algorithms to control robot behavior.

Objectives:

Students will be able to:

- Evaluate how the design of a robot’s body affects its behavior.
- Create an algorithm to direct a human “robot” from one part of the room to another.

Outline of the Lesson:

- “Are we Robots” activity (15 minutes)
- The effect of changing design (20 minutes)
- Student group work—Can a robot tie your shoes? (40 minutes)
- Student group work—Walk like a robot (35 minutes)

Student Activities:

- Participate in discussion of “Are we robots” activity.
- Discuss how changing the design of an item affects the item.
- Students work in pairs to try tying a shoe in several robot-esque situations including with closed eyes, with tongue depressors, pliers, and with another person.
- Students work in small groups to direct a person to move along a path given a limited list of commands.

Teaching/Learning Strategies:

- Revisit “Are we robots” activity. Go through the list of items, asking students to indicate if they thought each item was a robot or not. Occasionally, especially if there is disagreement, ask students to defend their answer.
- Discuss the effect of changing design. (This can be done as a warm up writing exercise with students sharing their responses.)
 - Ask students, “If you could change the body of the printer (or another device in the room) what would you change? How would that affect other things like the behavior or function of the printer, price, cost to build, or popularity? Have students share their ideas.
- Explain that there are limits to what robots can do because robots are limited by their bodies. For example, it is difficult to create a robotic hand that can grasp small or delicate items – it would require many motors (simulating all the muscles in the hand) and many sensors to detect the item (simulating the neurons in the hand).
 - Make sure each pair of students has a shoe that can be tied.

- Direct students to first try tying the shoe blindfolded or with eyes shut. Discuss how it went—Was it hard? What was hard about it? How was it like a robot tying the shoe?
- Direct students to tie the shoe with heavy gloves on. Discuss the experience. How was it like a robot tying the shoe? What made it hard?
- Direct students to tie the shoe with tongue depressors taped onto thumbs and forefingers or just holding tongue depressors. Discuss the experience. How was it like a robot tying the shoe? What made it hard?
- Direct students to tie the shoe with pliers. How was it like a robot tying the shoe? What made it hard?
- Direct the students to work with their partner to tie the shoes using the pliers, each person holding one pair. Discuss the experience. How was it like two robots working together? What made it hard?
- Activity: Walk like a robot
 - Choose one student to be a “robot” or tell students that you will be the robot. Choose a starting point and an ending point between which the “robot” must navigate. Make sure the path is not direct.
 - Tell the class that they must direct the robot from the starting point to the ending point using only five commands:
 - Turn left 90 deg.
 - Turn right 90 deg.
 - Take a step forward with the left foot.
 - Take a step forward with the right foot.
 - Stop.
 - Students can take turns or work as a group. The robot should only follow those five commands and not respond to other commands. Tell students to be careful with the robot and not walk it into walls or barriers. (The robot should stop before it hits a barrier such as a wall.).
 - At some point, remind students about loops. They can tell the robot to repeat a command or a block of commands such as “repeat: take a step forward with the left foot, take a step forward with the right foot until you are at the wall”
 - Point out that this is frequently what is done in dancing and choreography—sequences of steps are repeated.
 - If there is time, show the video of the “macarena” reference in the resource section.
 - Conclude by pointing out that these kinds of commands are what they will be programming their robots to execute.

Resources:

- Activity: Can a robot tie your shoes? (From www.thetech.org/robotics/activities/page05.html)
- Materials: shoes that tie, tongue depressors, masking tape, heavy gloves, pairs of pliers, blind folds

- Walk like a robot activity from LEGO Materials.
- <http://www.cs.colorado.edu/~lizb/chaotic-dance/macarena-orig.mpeg.gz>
- Explanation of video: <http://www.cs.colorado.edu/~lizb/chaotic-dance.html>

Instructional Day: 4

Topic Description: Set up LEGO® trays.

Objectives:

Students will be able to

- Distinguish between the LEGO parts for building a robot.

Outline of the Lesson:

- Distribution of LEGO kits (10 minutes)
- Separation of LEGO parts into the appropriate compartments of the trays (45 minutes)

Student Activities:

- Student groups work together to set up their LEGO kits for use in building robots.

Teaching/Learning Strategies:

- Give each pair (or group of three) a LEGO® Mindstorms® NXT® kit. Point out the picture that shows where each item should be placed in the tray.
- Ask students to set up their trays so that they will be ready for use in building robots.

Resources:

- LEGO Mindstorms NXT kit

Instructional Day: 5

Topic Description: Build the base of the robot.

Objectives:

Students will be able to

- Assemble the base of the robot.

Outline of the Lesson:

- Explanation of LEGO Mindstorms manual (10 minutes)
- Assembly of base of robot (45 minutes)

Student Activities:

- Student groups assemble the base of the robot.

Teaching/Learning Strategies:

- Have students get out their kits and the manual that comes with the kit. Go through step 1 on p. 8 with the students to make sure they understand the format of the manual.
- Ask student groups to assemble the base of their robot according to the instructions on pp. 8-21. (Batteries should be charged in advance.)

Resources:

- LEGO Mindstorms manual

Instructional Days: 6-7

Topic Description: Introduce the features of the NXT Brick—the “brain” of the robot.

Objectives:

Students will be able to

- Distinguish between the parts of the NXT brick.
- Hook up input and output devices correctly.
- Use built-in NXT Brick programs.

Outline of the Lesson:

- Observation of the NXT brick (20 minutes)
- Plug in sensors, motors, and light, and run ‘View’ programs (30 minutes)
- Try Me’ built in programs (40 minutes)
- NXT Brick programs (20 minutes)

Student Activities:

- Articulate what they observe about the the NXT brick while the teacher explains each part.
- Test sensor data using the ‘View’ programs and report observations.
- Run ‘try me’ programs and describe what the programs do.

Teaching/Learning Strategies:

- Have students get out their robot base, sensors, lights, motors, and three wheels. Explain that the NXT is the brain of the robot. Have students describe the parts they see and make sure the following parts get identified. (The NXT User Guide pp. 9-12 can be used as support.)
 - Ports number 1-4: these are input ports. You use wires to plug sensors into the NXT brick. There are four kinds of sensors: touch sensors (detect touch/obstacles), sound sensor (detect the sound levels), light sensor (detects light level), ultrasonic sensor (detects movement and distance to an object). Reminder: input means sensing something in the robot’s environment.
 - Ports A-C: these are output ports. You use wires to connect devices for output. The devices are lamps and motors. Also, note that the speaker is an output port. The output is that the light can go on or off and that the motor can turn or stop turning. Reminder: output means effecting change in the robot’s environment.
 - Buttons:
 - Orange button: On/Enter
 - Light grey arrows: Navigation, left and right

Dark Grey button: Clear/Go back. Keep pressing this to turn off until prompt, then hit orange

- Lines in the right side: speaker. This is where noise comes out of the robot.
- If the rechargeable battery is in, there will be a power plug and LCD lights.
- Tell students to turn on the robot by pressing the orange button. What happens? (It makes a happy little song. LOUD.) What do they see now?
 - NXT at the top—name of the brick. Can be changed in the software
 - Battery level top right
 - Running icon—next to the battery icon. As long as it is spinning, the NXT is turned on and working correctly. If it freezes, the NXT has frozen and must be reset.
 - There are three icons on the screen. The one that is highlighted by default looks like two floppy disks and has the label above “My Files”. If they start hitting buttons, they can scroll through several menu options by using the arrow keys or go into My Files by hitting the orange button. The menu options are:
 - My Files—where programs will be stored.
 - NXT Program—allows you to build small programs using only the NXT without the need for a computer.
 - View—you can do a quick test of your sensors and motors and see the current data for each. You have to select the test you want to do and which port the sensor or motor is on. Only one test can be run at a time. The data will display on the screen.
 - Bluetooth—you can set a wireless connection between the NXT and other Bluetooth devices including other NXTs, phones, and computers.
 - Settings—you can change settings such as the speaker volume and the sleep time.
 - Try Me—built-in programs
 - Explain that in order for the robot to really do anything, you have to hook up input and output devices. Ask students to try to identify the devices in the kit. Make sure they can identify the touch sensor, sound sensor, light sensor, ultrasonic sensor, servo motor and lamps. Reinforce that the sensors are all input devices and the motors and lamps are output devices.
- Demonstrate and have students carefully plug the devices into the NXT. Sensors can be plugged into any input port numbered 1-4 but these default settings are used for the test and sample programs. See pp. 5-6 and 9 of the NXT User Guide for more information. Make sure students know to support the weight of the devices and the NXT brick without pulling on the wires.
 - Port 1: Touch sensor
 - Port 2: Sound Sensor
 - Port 3: Light Sensor
 - Port 4: Ultrasonic Sensor.
 - Port A: Light

- Port B & C: Servo motor
- Have students navigate to the View menu. They should test each of the sensors and see what the displays do. Make sure they also use the Motor rotations and motor degrees program. See NXT User Guide pp. 23-33 for more information. After a few minutes with students experimenting, ask what they noticed. What kind of data does each of the sensors provide? How could a robot use this in a program?
- Have students navigate to the Try Me menu by using the dark gray button to move up the menus and using the light gray arrows and orange button to enter the Try Me menu.
 - Select one of the programs and have all the students try it. Once they have tested it, ask them what it did. See if they can flowchart what the program does.
 - Try sound—moves the motors faster as more sound is detected.
 - Try touch—changes display and makes noise when button is touched.
 - Try light—makes noise based on how much light is detected.
 - Try ultrasonic—changes noise based on distance detected.
 - Try motor—changes sound based on motion of motor on port A.
- Finally, have students navigate to the Program menu and follow the directions in the LEGO Mindstorms manual on pp. 22-45, trying the programs indicated. They should then test the programs and make sure each one works as expected. (Also see the NXT User guide pp. 15-16 for more information.)

Resources:

- NXT User Guide

Instructional Days: 8-9

Topic Description: Introduce the features of the Mindstorms NXT Software.

Objectives:

The students will be able to

- Recognize the parts of the Mindstorms NXT software.
- Explain the different types of icons in the common palette and how to use them.
- Explain the different types of icons in the complete palette and how to use them.

Outline of the Lesson:

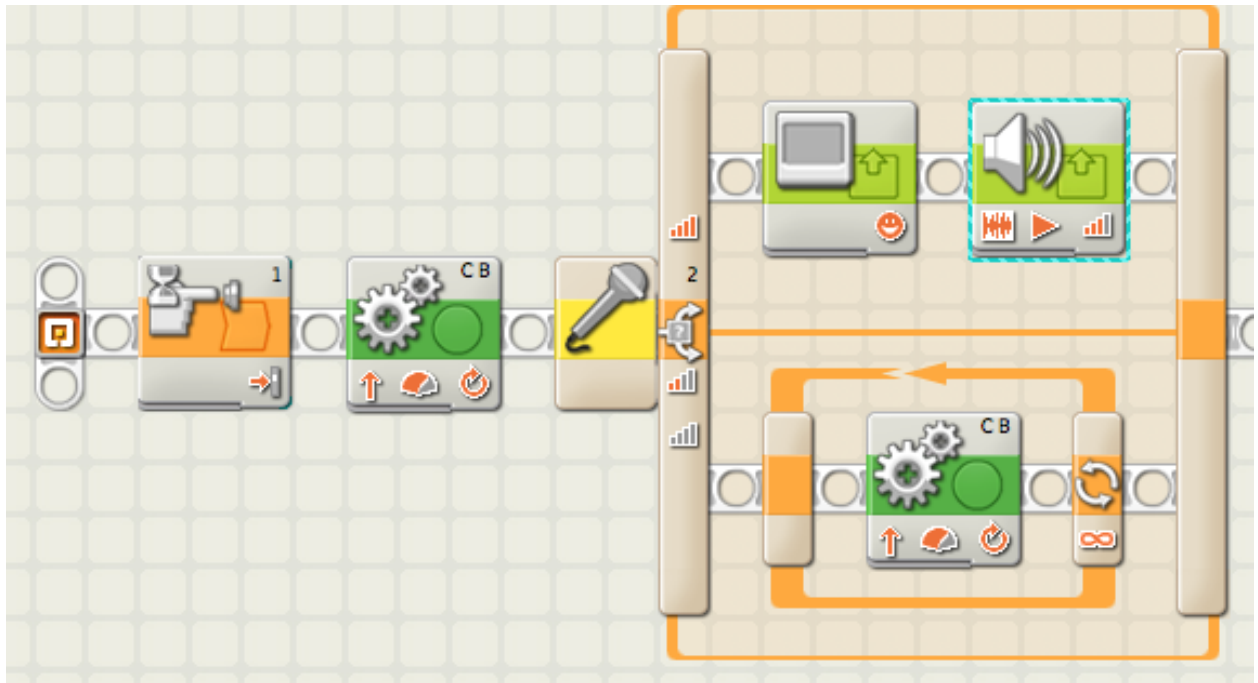
- Review of Program activity from Day 7 (20 minutes)
- Interface: the parts of the Mindstorms NXT software (10 minutes)
- A simple program from the common palette (30 minutes)
- A simple program from the complete palette (40 minutes)
- How to use the tutorials (10 minutes)

Student Activities:

- Discuss how the programs were created in the NXT brick and how they behaved compared to expectations.
- Listen to explanation of Mindstorms NXT software and respond to questions.
- Give ideas to teacher as s/he writes small programs in the software.
- Listen to explanation of how to use the tutorials.

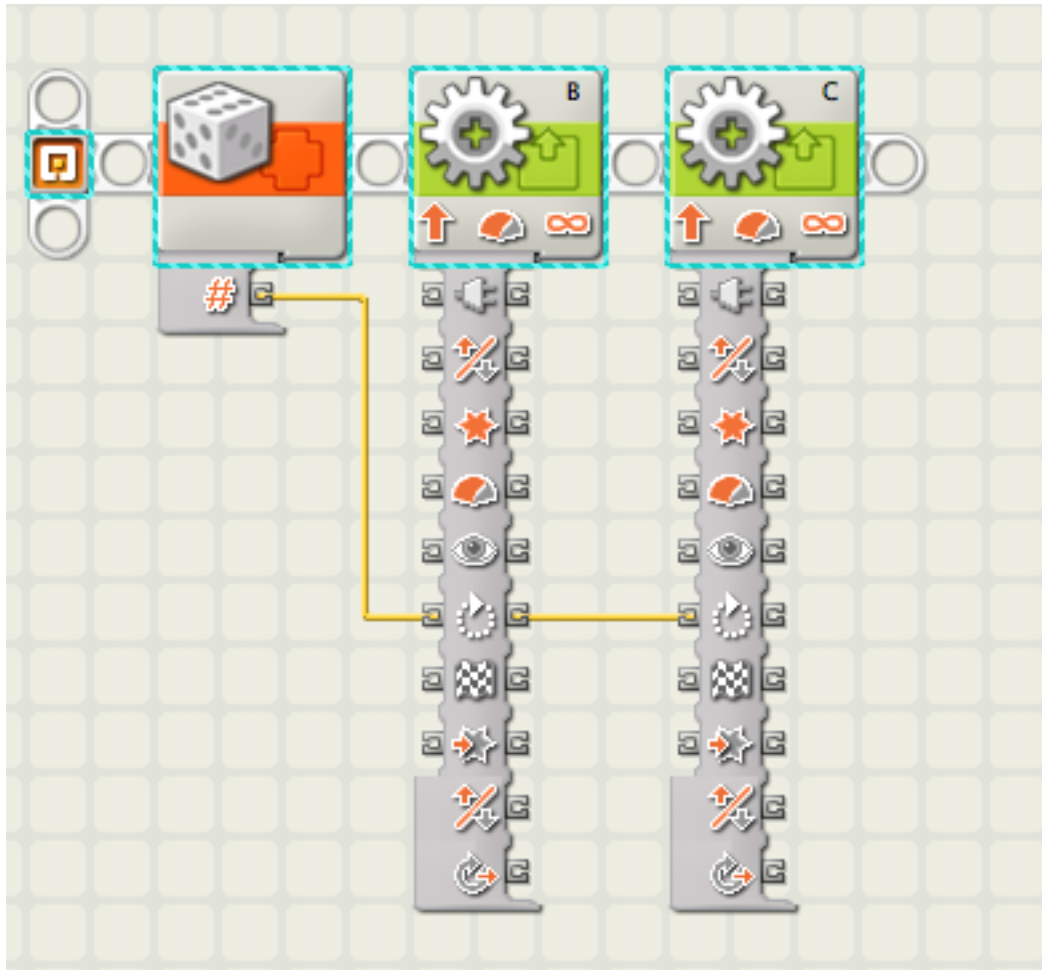
Teaching/Learning Strategies:

- Ask students what they programmed the robot to do. Get several answers. Did it do what they expected? Why or why not? Would it be a good idea to use the NXT Program interface to write all their programs? Why not? (It can only take 5 commands in a program.)
- Projecting the teacher's screen, launch the Mindstorms NXT software. Show the students where the tutorials are in the Robot Educator section and how to open a new program. Using the User Guide pp. 48-49, describe all the parts of the interface.
- With student input, use the common palette to build a small program. Ideally, use a variety of the blocks of the common palette, explaining what each one does as you use it. For example, if you wanted to build a program that told the robot to wait until the touch sensor was touched, then move forward for one rotation then listen and if a loud sound occurs, then display a smiley face and play a sound otherwise move forward, it would look like this:



- Save the program and download it to an NXT brick. Make sure the brick is set up to do the actions—have one built with the driving base and any necessary sensors. Demonstrate the running of the program.
- Modify the program and download it again. Try to make mistakes during this period and show how to debug the program by frequently testing it, downloading extra blocks, and also making mistakes such as having disconnected blocks. During this part have students try to work with the software themselves and follow along with you.
- Open a new program and switch to the complete palette. Show the differences in the two palettes. With student input, write a new program using the blocks of the complete palette. Show the differences in controlling the program. Make sure to show how to wire things in the data hub. For example, a program that runs the motors for a random amount of time would

look like this:



- Make sure to make mistakes and demonstrate how to solve problems with the software such as mis-wiring ports. Have students try these features at their seats as you do it. Point out the similarities between programming the NXT software and what they did in the last unit with Scratch.
- This is a good point at which to discuss
 - Software vs. hardware errors—in robotics the programming may be correct, but the robot configured incorrectly.
 - Syntax vs. logical errors—the program may compile, but the logic can still be incorrect.
- Tell students that the next five days will be spent going through the tutorials in order to learn how to build and program the NXT system.

Resources:

- NXT Robot Educator

Instructional Days: 10-14

Topic Description: Program the robot using the Mindstorm Robot Educator Software tutorials.

Objectives:

The students will be able to:

- Use the building blocks of the common palette to program the robot.
- Build robots that can execute the functions programmed through the Robot Educator Software.
- Program the robot using some or all of the complete palette of blocks.

Outline of the Lesson:

- Description of the assessment model (10 minutes)
- How to use the tutorials (10 minutes)
- Build and program robots according to tutorials (255 minutes)

Student Activities:

- In groups of 2-4, students follow tutorials to build and program small robots.

Teaching/Learning Strategies:

- Explain assessment model for tutorials. (Recommended: observe some but not all robots, such as those for tutorials 8, 16, and 20 in the common palette along with several from the complete palette; look at robot construction and the program as well as execution to determine grade.)
- All students should complete the tutorials for the common palette before moving on to the complete palette. It will be helpful for the future projects if students complete most, if not all, of the tutorials for the complete palette as well.
- Circulate throughout class to answer questions, help troubleshoot, and assess robots.
- If some groups finish early, have them assist other groups.

Resources:

- NXT User guide pp. 50-53 explain the tutorials

Instructional Day: 15

Topic Description: Introduce RoboCup real life robotic competition and write instructions for tic-tac-toe.

Objectives:

Students will be able to

- Explain how a sequence of game moves can be expressed in simple statements.
- Describe the RoboCup challenge and examine how robots have been programmed to play soccer.
- Develop if-then statements and use Boolean operators to direct a robot to play tic-tac-toe.

Outline of the Lesson:

- Tic-tac-toe (10 minutes)
- “Robot Competitors Meet on a Soccer Field of Dreams” (25 minutes)
- Instructions for a “robot” to play tic-tac-toe. (20 minutes)

Student Activities:

- In pairs, students play a game of tic-tac-toe; then they discuss and write answers to the posted questions.
- Read and discuss the article, “Robot Competitors Meet on a Soccer Field of Dreams”.
- In pairs, students write a series of clear instructions for a “robot” to play tic-tac-toe.

Teaching/Learning Strategies:

- Before students enter the classroom, write the following on the board or chart paper: “Play a game of tic-tac-toe with your partner. Then think about these questions together, and write your answers: What are the rules of tic-tac-toe? What decisions does a player need to make before taking a turn? How would you verbally describe each of these decisions? What is the action a robot would need to take based on the decisions?”
- After a few minutes, have students share some of their responses. Make a list of the rules of tic-tac-toe on the board. Ensure students remember that if statements and conditionals are required to describe the moves of the game. Collect the written responses to the warm up activity.
- Distribute the article “Robot Competitors Meet on a Soccer Field of Dreams” and have students read it.
- Lead a discussion about the article.
- Explain to students that they will be working in pairs to write an application for robots to enable them to play tic-tac-toe. The following day will be the RoboTicTacToe Challenge. Remind them

of the earlier discussion of tic-tac-toe. What goals does each player have? Who starts the game? Is there a “best place” to put the first X? What are some winning strategies for the next move? For example, If the X is in the center, then where should an O be placed? Why is “if-then” logic a good way to explain strategy for a simple game like tic-tac-toe? How can Boolean operators, and/or/not, help simplify the commands?

- Demonstrate the opening move for a game of tic-tac-toe on the board. Draw a nine-space grid and label the squares one through nine. Then ask students where to place the first X. Depending on where it is placed, have students create an if-then statement that determines the next move. For example, “If the first X is in the center, place an O in a corner square.”
- Ask students to complete the instructions. Each instruction in the entire sequence will cover every possible combination of moves the students can think of until a game is completed. Students need to remember that there are multiple options for each move (including the beginning move). They should consider all of the possibilities in developing their code. They also need to consider what the behavior the robot will exhibit based on the instructions provided.
- Note that the focus in this lesson is really a reinforcement of programming as a set of instructions in the context of something most students understand. The game of tic-tac-toe is not a natural example of robotics because robot environments are generally dynamic with infinite possible states of the environment.

Resources:

- Lesson plan from NY Times Lesson Plan Archive:
<http://www.nytimes.com/learning/teachers/lessons/20010802thursday.html>
- Copy of article
http://www.nytimes.com/learning/teachers/featured_articles/20010802thursday.html
- Dictionary

Instructional Day: 16

Topic Description: RoboTic-Tac-Toe Challenge and Introduction to RoboCupJunior Dance Challenge.

Objectives:

Students will be able to

- Debug conditional statements by testing them and compete as teams in a RoboTic-Tac-Toe Challenge.
- Describe dancing robots that have competed in the RoboCupJunior Dance Challenge.

Outline of the Lesson:

- Debugging of robotic-tac-toe statements (5 minutes)
- RoboTic-Tac-Toe challenge (35 minutes)
- Introduction to RoboCupJunior Dance challenge (15 minutes)

Student Activities:

- Complete debugging tic-tac-toe statements by testing that they work correctly in several games.
- Participate in RoboTic-Tac-Toe challenge.
- Listen to an explanation of RoboCupJunior Dance Challenge and watch videos of dancing robots from RoboCupJunior challenges.

Teaching/Learning Strategies:

- Ask students to quickly test their tic-tac-toe instructions to make sure they are complete and correct. They should play tic-tac-toe following only the instructions they have written.
- Explain the challenge: each team will be acting as a single robot programmed by the application they developed. One student will read a command from their application and the other student will execute the command. Teams play against each other, testing how successful their code is. Each game should be observed by the rest of the class and monitored to ensure the teams only execute the commands read.
- At the conclusion of the challenge, celebrate the winning team. Ask the students to describe why that team won? What have they learned? How would they improve their programs? (Remind students that precise instructions are required in programming.)
- Explain that RoboCup is a research initiative founded in 1997 by an international group of scientists interested in defining a common problem that could be addressed by researchers in robotics, engineering, and artificial intelligence. Most participants are university and industry research labs. RoboCupJunior (RCJ) was founded in 2000, with a focus on education. The RCJ Rescue challenge was piloted in 2001 and adapted in 2003. RCJ is open to students up to age 19. There are two divisions: primary, which is up to age 14, and secondary, which is age 14 to

19. The first two robot projects will be based on the RoboCupJunior program. The first one is the dancing robot which is the introductory level of the RoboCupJunior program. Students will build and program a robot that dances. Show videos of dancing robots in competition.

Resources:

- RoboCupJunior videos: <http://rcj.robocup.org/videos.html>
- More videos available through YouTube such as <http://www.youtube.com/watch?v=25sZr3u-WwU>

Instructional Day: 17-20

Topic Description: Build, program, and present a dancing robot.

Objectives:

The students will be able to:

- Use the NXT and output devices to build and program a robot that dances in time to music.

Outline of the Lesson:

- Explanation of project guidelines and show dance floor (15 minutes)
- Design, build, and program dancing robot (150 minutes)
- Dance challenge (40 minutes)
- Reflection and Clean up (15 minutes)

Student Activities:

- Agree on ideas and music for robot.
- Build robot.
- Write a program in Robot Educator software.
- Test robot and refine program and hardware.
- Participate in dance challenge.
- Complete project reflection. Take robots apart and put materials away.

Teaching/Learning Strategies:

- Hand out requirements and rubric. Explain guidelines and answer questions.
- Circulate and make sure students are on task; answer questions as needed.
- Before the dance challenge, assign one student as timekeeper and another as DJ. Collect each group's program as they compete and immediately assess the robot using the rubric, while the next group gets set up. You may declare a winner or have the students vote for the best robot.
- At the end of the challenge, have each student complete the project reflection and submit it, then clean up the robots.

Resources:

- Official RoboCupJunior Dance Challenge rules (2008): <http://rcj.robocup.org/dance.html>
- Dancing Robot Activity
- Dancing Robot Sample Rubric
- Project Reflection
- Recommendations for dance floor: Create a large square on one or more pieces of butcher paper.

Dancing Robot Activity

The dancing robot assignment is based on the first level of RoboCupJunior, an international competition. More information about RoboCupJunior is available at <http://rcj.robocup.org>

Task:

Build a robot that dances to music for 1-2 minutes.

Requirements:

- The robot should not take any input, only have output in the form of various dance moves.
- Dance must be 1-2 minutes long. You have a total of 5 minutes to get set up, have the robot dance, and get out of the way for the next group.
- The robot must stay in the marked space.
- The robot must be autonomous. Other than hitting the start button, no human can touch it while it performs.
- The dance should be choreographed to the music you provide. The music must be appropriate for playing at school – no obscenities, etc.
- Teams may restart the robot up to 2 times at the discretion of the teacher. Any re-started, unless due to a problem not the fault of the team, will result in a grade penalty.
- Teams are encouraged to be as creative and entertaining as possible! Props, costumes, and varied dance moves are encouraged. You may dance alongside your robot.
- Each team must print out its program and hand it in at the same time that they compete.
- Fair play is an important part of the RoboCup competition. Teams are expected to help other teams as needed and not deliberately interfere with or damage other teams' work. All students are expected to respectfully watch all other teams compete.

Process:

1. Brainstorm ideas about how your robot should look, how it should work (wheels? Arms?) and how you'll build it. Select music.
2. Start building your robot.
3. Build a program that directs the robot to do your dance moves.
4. Test and revise the program. Make sure it runs for 1-2 minutes. Make sure it matches the music. Make sure it won't fall apart!
5. Show off the robot during the dance in class.

You will have two class periods to build and program the robot, then you will present it on the third day.

Performance will be judged on

- Programming (eg: use of loops, jumps, sub-routines, type of programming language used, etc)
- Choreography (eg: robots to move in time with music, and change actions as music changes tempo or rhythm. Choreography of humans and robots will be scored separately, etc)

- Construction (i.e., robots should be of sound construction, components should not fall off , appropriate use of gearing, smooth and reliable operation, interesting movements, effective use of mechanics to achieve a purpose, etc.)
- Entertainment Value (i.e.,How much does the performance entertain or delight the audience? Originality and creativity of the presentation, etc.)
- Costume (Costume of humans and robots will be scored separately.)
- Cooperation between teams

Dance stage will be a flat area. Official RobocupJunior stage size is 6X4 m.

Dancing Robot Sample Rubric

	Extra Credit	A	B	C	F
Programming	Program uses advanced techniques including blocks from the complete palette, flow blocks, etc.	Program is straightforward and efficient, using loops and parallel sequences as necessary. Program directs attached output devices to dance.	Program is straightforward and easy to understand. Program is inefficient and could use constructs such as loops.	Program is poorly written or difficult to understand. Program has unused parts or does not correctly control robot.	Program does not work.
Choreography	Dance has at least 10 different dance moves. Dance matched music precisely. Robot changed actions as music changed tempo or rhythm	Dance has at least 6 different dance moves. Dance is varied and entertaining. Dance is choreographed to match music	Dance has at least 4 different dance moves. Dance is repetitive. Dance lasted for 45-60 seconds or 120-150 seconds.	Dance has 3 different dance moves. Dance lasted for 30-45 seconds or 150-210 seconds. Dance did not match music.	Robot did not move or did not appear to dance.
Construction	Robot constructed using advanced gearing or other advanced construction techniques. Robot demonstrates extraordinary creativity.	Robot is of sound construction: nothing falls off, robot works as intended. Mechanics used well to achieve dance moves desired.	Robot dances as intended, but some extraneous parts fall off.	Robot does not work as intended, but does move. Robot falls apart. Very simple construction – mechanics not used well.	Robot falls apart or does not move at all. Construction appears careless or haphazard.
Entertainment Value	Presentation is unusually creative. Humans dance with robot. Costume, props, etc enhance robot.	Audience is entertained by robot, presentation, etc. Robot runs correctly the first time.	Presentation is not smooth: robot must be restarted.	Problems occur but robot does eventually run mostly correctly.	Robot does not compete.
Cooperation	Student(s) helped other groups	Student worked well with group. Student participated actively in all parts of project.	Student worked somewhat well with group. Student participated in most parts of project.	Student had trouble working with group. Student participated in few parts of project.	Student did not participate in project. Student sabotaged others' work.

Robot Project Reflection

For each member of your group, evaluate their performance as a team member:

<u>Name:</u>	<u>Circle one word to describe his/her performance</u>
_____	Excellent Good Average Poor

Why?

_____	Excellent Good Average Poor
Why?	

_____	Excellent Good Average Poor
Why?	

What was your favorite thing about this project?

If you could do this project over, what would you do differently?

Instructional Days: 21-25

Topic Description: Build, program and present a rescue robot.

Objectives:

Students will be able to:

- Build and program a robot that uses input and output devices to count simulated people by following a black line and counting “people” on the path.

Outline of the Lesson:

- Explanation of project guidelines and floor (15 minutes)
- Design, build, program robot (195 minutes)
- Rescue Robot challenge (50 minutes)
- Reflection and clean up (15 minutes)

Student Activities:

- Brainstorm how to build and program the robot.
- Build the robot.
- Write a program in Robot Educator software.
- Test the robot frequently and refine program and hardware.
- Participate in rescue challenge.
- Complete project reflection. Take robots apart and put materials away.

Teaching/Learning Strategies:

- Hand out requirements and rubric. Explain guidelines and answer questions. Show students the arena with the victims laid out. Explain that they must use sensors so that the robot will follow the black line and will sense when it has encountered a victim or a gap.
- Circulate the room and make sure students are on task; answer questions as needed.
- During the rescue challenge, assign one student as timekeeper and one to keep track of victims found. Collect each group’s program as they compete and immediately assess the robot using the rubric, while the next group gets set up.
- At the end of the challenge, have each student complete the project reflection and submit it, then clean up the robots.

Resources:

- Rescue Robot Activity
- Rescue Robot Sample Rubric
- Project Reflection

- Official RoboCup Jr Rescue Competition Rules (2008): <http://rcj.robocup.org/rescue.html>
- Instructions for building modules are available at <http://rcj.sci.brooklyn.cuny.edu/rcj2004/rescue-field-plans-2004.html>. Alternatively use white butcher paper on the floor with black electrical tape as a path. Use green electrical tape to indicate victims.

Rescue Robot Activity

The rescue robot assignment is based on the second level of RoboCupJunior, an international competition. More information about RoboCupJunior is available at <http://rcj.robocup.org>. This robot simulates robots sent to rescue people during natural disasters. It must find “victims” along the path through each “room” and avoid obstacles. The goal is to program a robot that uses sensors to respond to different stimuli.

Task:

Build a robot that follows a black line on a white background, counts green or metallic “people” and avoids obstacles.

Requirements:

- The robot must follow the black line and attempt to complete the course through the entire arena. The robot will begin at the starting location in the doorway of the first “room”
- The robot should stop and flash a light for at least two seconds to indicate it has found a victim. For extra credit, count the number of victims and display the count.
- The robot should be able to avoid items of debris blocking the black line.
- If a robot has been stuck or lost the black line for more than 20 seconds, the teacher may pick it up and put it back onto the black line a little beyond where it ran into problems. The 20-second rule allows it to try to find its way back to the line without intervention. A team may decide to quit if the robot is faulty or repeatedly loses the line.
- Robots must be controlled autonomously except for being started by a member of the team.
- The robot will have 10 minutes to complete the course and identify all victims.
- Each team must print out its program and hand it in at the same time that they compete.
- Fair play is an important part of the RoboCup challenge. Teams are expected to help other teams as needed and not deliberately interfere with or damage other teams’ work. All students are expected to respectfully watch all other teams compete.

Process:

6. Brainstorm ideas about how your robot should work: what sensors will you need? What motors and lights? What programming constructs will you need?
7. Start building your robot.
8. Build a program that controls the robot
9. Test frequently and revise the program. Make sure it correctly detects victims and that it can follow the line. Check if it can navigate gaps.

You will have three and a half class periods to build and program the robot; then you will present it in class.

Official Rules available

- <http://rcj.robocup.org/rescue.html>

Official RoboCupJunior. Rescue Challenge

5.1. Victims:

5.1.1. Ten (10) points are awarded for each victim located by the robot. The robot indicates that it has found a victim by stopping and flashing a lamp for at least two (2) seconds.

5.1.2. Extra points are NOT awarded for the same victim being located more than once.

5.2. Gaps in the black line:

5.2.1. Ten (10) points are awarded for each gap in the black line that the robot successfully negotiates (i.e. recovers the line on the far side of the gap).

5.3. Debris blocking the black line:

5.3.1. Ten (10) points are awarded for each item of debris blocking the black line that the robot successfully avoids (i.e. moves around the debris and recovers the line).

5.4. Rooms:

5.4.1. Ten (10) points are awarded for each room that the robot navigates successfully (i.e. enters through one doorway and exits through the other doorway).

5.5. Ramp:

5.5.1. Thirty (30) points are awarded for the robot successfully negotiating a ramp without any assistance.

5.6. Penalties:

5.6.1. Two (2) points are deducted for each false victim identification (i.e. whenever a robot indicates that it has found a victim at a location where there isn't one).

5.6.2. Five (5) points are deducted for each lack of progress (i.e. whenever human intervention is required to enable a robot to resume progress along the black line).

Official Rules available

<http://rcj.robocup.org/rcj2008/china-rescue-rules-page.pdf> (Note: This references the RoboCupJunior 2008 Rescue rules. The committee members were Ashley Green, Maverick Luk, Eli Kolberg and Bill Freitas. You may wish to work with the most up to date version.)

Rescue Robot Rubric

	Extra Credit	A	B	C	F
Victims	Found victims are counted and count is displayed	All victims correctly identified	Most victims correctly identified	Some victims correctly identified	No victims correctly identified
Gaps		All gaps navigated correctly	Most gaps navigated correctly	Some gaps navigated correctly	No gaps navigated correctly
Debris		Robot avoided all debris	Robot avoided most debris	Robot avoided some debris	Robot unable to avoid debris
Rooms		Robot entered all rooms through one door and exited through the other	Robot entered most rooms through one door and exited through the other	Robot entered one room and was unable to exit	Robot did not enter the first room
Construction	Robot constructed using advanced gearing or other advanced construction techniques. Robot demonstrates extraordinary creativity.	Robot is of sound construction: nothing falls off, robot works as intended.	Parts of robot fall off. Very simple construction – mechanics not used well.	Robot does not work as intended, but does move. Robot falls apart. Robot is unable to navigate due to construction	Robot falls apart or does not move at all. Construction appears careless or haphazard.
Programming	Program uses advanced techniques including blocks from the complete palette, flow blocks, etc.	Program is straightforward and efficient, using loops and parallel sequences as necessary. Program uses sensors and strong logic to navigate challenges and find victims.	Program is straightforward and easy to understand. Program uses inefficient logic to navigate challenges and find victims.	Program is poorly written or difficult to understand. Program has unused parts or does not correctly control robot. Program does not correctly use sensors to control motion.	Program does not work.
Cooperation	Student(s) helped other groups. Managed own role & helped group members.	Student worked well with group. Student participated actively in all parts of project.	Student worked somewhat well with group. Student participated in most parts of project.	Student had trouble working with group. Student participated in few parts of project.	Student did not participate in project. Student sabotaged others' work. Made it difficult for group to work.

Instructional Days: 26-30

Topic Description: Build, program and present a robot project of choice. (This lesson is optional, depending on the time remaining to complete the unit.)

Objectives:

Students will be able to:

- Design, build, and program a robot that solves a stated problem.

Outline of the Lesson:

- Explanation of project guidelines (15 minutes)
- Design, build, program robot (195 minutes)
- Demonstration of robot projects (50 minutes)
- Reflection and Clean up (15 minutes)

Student Activities:

- In groups, decide on a project.
- Plan the robot.
- Design, build, program, and refine a robot which meets the challenge.
- Demonstrate the various robots.
- Complete project reflection. Take robots apart and put materials away.

Teaching/Learning Strategies:

- Provide resources to students so that they can choose their projects. Some projects may require use of the materials in the extension kit. (Note that <http://www.nxtprograms.com> has building instructions, video clips and coding solutions.)
- Approve planning documents as students finish plan and prepare to build and program robot.
- Circulate and make sure students are on task; answer questions as needed.
- During the demonstrations, fill out each rubric as you observe the robot. If possible, videotape (or have a volunteer videotape) the running of each robot.
- At the end of the competition, have each student complete the project reflection and submit it, then clean up the robots.

Resources:

- **ENTER NAME OF SUPPLEMENTAL BOOK**
- <http://www.nxtprograms.com>

Instructional Days: 31-40

Topic Description: Complete Design Challenge final project.

Objectives:

Students will be able to:

- Design, build, and program a robot that solves a stated problem.

Outline of the Lesson:

- Explanation of project guidelines (15 minutes)
- Distribution of challenges (10 minutes)
- Design, build, and program robot (~7.5 class periods)
- Design challenge gallery walk (1 class period)
- Clean up (1 class period)

Student Activities:

- In groups, determine who will complete each of the four roles.
- Use the planning document to plan the robot.
- Design, build, program, and refine a robot which meets the challenge.
- Set up their robot and participate in a gallery walk.
- Disassemble the robots and carefully organize all the robotics equipment.

Teaching/Learning Strategies:

- Hand out requirements, planning document, and rubric. Explain guidelines and answer questions.
- Hand out challenges. Allow students to trade challenges as necessary. You may choose to have each group working on a different challenge or have them overlap.
- Approve planning documents as students finish plan and prepare to build and program robot.
- Circulate and make sure students are on task; answer questions as needed. At the end of each day, remind information specialists to fill out paperwork and remind groups to clean up the space. Optionally, have students fill out the daily group evaluation.
- During the design challenge, fill out each rubric as you observe the robot. If possible, videotape (or have a volunteer videotape) the running of each robot.
- On the final day of the unit have students disassemble the robots and organize the equipment.

Resources:

- Design Challenge Sample Rubric
- Information Specialist Report

- Project-Reflection
- Daily Group Evaluation
- Challenges:
 - Option 1: Challenges from Design Challenges for computer-controlled LEGO products by Len Litowitz. (Litowitz-challenges.doc) Some of these challenges are more appropriate than others.
 - Option 2: Gary Stager's LEGO Challenges available from <http://www.stager.org/LEGO/challenges.pdf> (stager-challenges.pdf) Not all of these challenges are appropriate.
 - Option 3: Webquest

Final Project

Design Challenge Planning

STEP #1 TASK DEFINITION

Determine the purpose of your challenge – What are we supposed to do?

Criteria – list the specifications the robot needs to meet

1.

2.

3.

4.

5.

STEP #2 TASK BREAK-DOWN

List the steps the robot will need to go through to accomplish the task

1.

2.

3.

4.

5.

6.

7.

8.

9.

STEP #3 BRAINSTORMING

List some possible solutions to the challenge:

1.

2.

3.

4.

5.

6.

7.

8.

STEP #4 ROBOT DESIGN

Use scratch paper to sketch ideas for the robot, then choose the “best” design idea and illustrate it NEATLY below. Include any labels or explanations necessary to make your design understandable.

STEP #5 PROGRAM FLOWCHARTING

Outline the programming steps for your robot to accomplish the task. This should be in the form of a flowchart.

STOP!!! – GET TEACHER APPROVAL BEFORE MOVING ON: _____

STEP #6 ROBOT BUILDING AND PROGRAMMING

Build the robot and program it according to your plan!

Design Challenge Rubric

	Extra Credit	A	B	C	F
Successful Solution	Meets criteria and one or more super challenge criteria	Solution clearly solves the problem but not super challenges.	Solution solves problem inelegantly or inefficiently.	Solution does not completely solve problem.	No reasonable attempt made to solve problem.
Programming	Program uses advanced techniques including Boolean logic, Complete palette blocks, etc. Program demonstrates extraordinary creativity or unique way of solving problem	Program is straightforward and efficient, and uses appropriate programming constructs. Program has a reasonable algorithm for solving problem and uses good logic.	Program is straightforward and easy to understand. Program is inefficient. Program has a reasonable algorithm for solving problem.	Program is poorly written or difficult to understand. Program has unused parts or does not correctly control robot. Algorithm is strained.	Program does not work. Program does not solve problem effectively.
Construction	Robot constructed using advanced gearing or other advanced construction techniques. Robot demonstrates extraordinary creativity.	Robot is of sound construction: nothing falls off, robot works as intended. Mechanics used well to achieve desired outcome. Robot can solve problem repeatedly.	Robot works as intended, but some extraneous parts fall off. Moderate degree of repeatability: robot will run again but must be adjusted or fixed.	Robot does not work as intended, but does move. Robot falls apart. Very simple construction – mechanics not used well. Robot cannot run repeatedly.	Robot falls apart or does not move at all. Construction appears careless or haphazard.
Documentation	Documentation goes beyond required paperwork.	Ample and accurate documentation. Documentation kept consistently and thoroughly.	Good documentation: documentation kept consistently but not as thorough as it could be.	Fair documentation: documentation kept inconsistently and missing parts.	Little or no documentation
Cooperation	Student(s) helped other groups	Student worked well with group. Student participated actively in all parts of project.	Student worked somewhat well with group. Student participated in most parts of project.	Student had trouble working with group. Student participated in few parts of project.	Student did not participate in project. Student sabotaged others' work.

Daily Group Evaluation

Date: _____

List each member of your group (including yourself) and assess each area with:

3 = strongly agree (s/he was very good at this)

2 = agree (about right)

1 = disagree (this was a problem)

<i>Name</i>	<i>Listened respectfully to group members</i>	<i>Was focused and on-task</i>	<i>Did his/her share of work</i>
(self)			

Comments:

Information Specialist Report

You are responsible for reporting the status of the project to the Team Manager every day. How has the team progressed? Address the following questions:

1. What did your team accomplish today?
2. What problems did the team find today?
3. What solutions did the team try?
4. Other comments?

	Tasks	Report
Per 1	Get Challenge Begin brainstorming & Designing	
Per 2	Finish Design & get approval Begin building test parts – try different ideas	
Per 3	Finish building test parts & begin assembling robot from successfully tested parts	
Per 4	Continue assembling robot from parts Create program for robot	
Per 5	Continue building & programming robot – test regularly	
Per 6	Continue to refine robot – test regularly with the program	
Per 7	Finish refining robot- make sure it completes challenge!	
Per 8	Finish or enhance robot	
Per 9	Design Challenge: Show off robot!	
Per 10	Clean up: Take apart robot, return materials to original state	

Names: _____

Unit 6:

Computing Applications



Introduction

Managing and interpreting large amounts of data is part of the foundation of our information society and the economy. The ability to analyze, visualize and draw conclusions from large data sets is critical to computing. This unit ends with a project based on earthquake data collected by the Center for Embedded Network Sensing. There are a variety of tools and programming languages that can be used to work with large sets of data. Python was chosen for this unit because it had the capabilities necessary (which Scratch did not), but would provide a fairly easy transition from the drag and drop environments used in Scratch and robotics.

Python syntax is introduced only to the extent necessary to solve problems that utilize features required for data analysis. Projects require students to work in pairs or larger teams and principles of software engineering and pair programming are discussed in that context.

Specific topics for each instructional day are listed in the overview chart on the next page.

Daily Overview Chart	
Instructional Day	Topic
1	Introduce the Python programming environment and the Pen class.
2	Introduce drawing in Python by using coordinates .
3-5	Create a program to draw a dream house or car using the concept of pair programming.
6	Introduce the use of Dialogs in Python.
7-10	Introduce the concepts of software development activities, models and design teams. Practice dialogs and working in teams to create an order form program.
11	Introduce numerical types and math in Python.
12	Introduce functions in Python.
13	Practice the use of functions through programs to exchange currencies and calculate measurements.
14	Introduce conditionals in Python.
15-17	Practice the use of conditionals and functions through the creation of a Choose Your Own Adventure program.
18	Introduce while loops in Python.
19	Introduce the for loop in Python.
20	Introduce the concept of lists.
21-25	Practice the use of loops, conditionals, and list through the creation of an opinion poll program.
26-30	Complete final project.

Daily Lesson Plans

Instructional Day: 1

Topic Description: This lesson introduces the Python programming environment and the Pen class.

Objectives:

The students will be able to:

- Navigate the Python environment.
- Start drawing with the Pen class.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Introduction of Python (5 minutes)
- Introduction of final project (5 minutes)
- Introduction of Pen class (10 minutes)
- Use of the Pen class to make a very simple house (30 minutes)

Student Activities:

- Complete journal entry.
- Use the Pen class to make a very simple house.

Teaching/Learning Strategies:

- Journal Entry : Write down detailed instructions for drawing a square on a piece of paper using a pen.
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- Introduction of Python
 - Python can be downloaded for free at <http://www.python.org>.
 - Python is an interpretive language that is used by companies like Google, YouTube, and Pixar.
 - Introduce the environment.
 - To run Python, go to the start menu and run IDLE (Python GUI). This opens the Python shell.
 - You could type your code directly into Python, but it's easier to just keep your work in a file.
 - Use File->New Window to open up the editor.
 - Type: `print('Hello World')`
 - Save your file as `hello.py` (File->Save)
 - When saving Python files the first time, you need to type in the `.py` extension. Your files will still run without it, but the `.py` files are syntax highlighted.

- To run the file, Run->Run Module or F5
 - The file will run in the Python Shell window.
- Introduction of final project
 - Show Final Project.
 - Final Project Description
 - Final Project Sample Rubric
- Introduction of Pen class
 - On the board or chart paper, draw a rectangle to be your canvas.
 - Ask the students to guide you in drawing a square using only the following:
 - `forward(100)`—draw a line forward
 - `left(degrees)`—turn left so many degrees
 - Explain that the Pen starts in the middle with the direction facing toward the top of the screen.
 - Keep track of the current direction as you draw.
 - Students write the code to draw the square in Python.
 - The file must start with: `from turtle import *`
 - The next line is: `pen = Pen()`
 - The commands must start with `pen.` (i.e. `pen.forward(100)`)
- Students use the Pen class to make a very simple house.
 - Circulate room and help students.
 - See House Sample Rubric.
 - Provide students with Drawing Reference.

Resources:

- Drawing Class Reference
- House Sample Rubric
- Final Project
- Final Project Sample Rubric
- <http://www.python.org>

Drawing Class Reference

class Pen()

Define a pen. All functions below can be called as a method on the given pen. The constructor automatically creates a canvas to be drawn on. The pen is either up (off the canvas) or down (on the canvas and drawing). It is also pointed in a given direction. The pen starts in the middle (0,0) of the canvas pointed straight up. It keeps track of its current location

forward(*distance*)

Go forward *distance* steps.

Example: `pen.forward(100)`

left(*angle*)

Turn left *angle* degrees.

Example: `pen.left(90)`

right(*angle*)

Turn right *angle* degrees.

Example: `pen.right(90)`

up()

Move the pen up -- stop drawing.

Example: `pen.up()`

down()

Move the pen up -- draw when moving.

Example: `pen.down()`

width(*width*)

Set the line width to *width*.

Example: `pen.width(5)`

color(*s*)

color(*r, g, b*)

Set the pen color. In the first form, the color is specified as a Tk color specification as a string. The second form specifies the color as a tuple of the RGB values, each in the range [0..1]. For the third form, the color is specified giving the RGB values as three separate parameters (each in the range [0..1]).

Example: `pen.color('blue')`

Example: `pen.color(0,1,0.75)`

write(*text*)

Write *text* at the current pen position.

Example: `pen.text('Hello')`

circle(*radius* [, *extent*])

Draw a circle with radius *radius* whose center-point is *radius* units left of the current position. *extent* determines which part of a circle is drawn: if not given it defaults to a full circle.

If *extent* is not a full circle, one endpoint of the arc is the current pen position. The arc is drawn in a counter clockwise direction if *radius* is positive, otherwise in a clockwise direction. In the process, the direction of the pen is changed by the amount of the *extent*.

Example: `pen.circle(10)`

goto(*x, y*)

Go to co-ordinates *x, y*.

Example: `pen.goto(100,0)`

begin_fill()

Switch pen into filling mode; Must eventually be followed by a corresponding `end_fill()` call. Otherwise it will be ignored.

end_fill()

End filling mode, and fill the shape.

House Sample Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
Your house has a pointed roof	4			
There is a line that separates the roof from the rest of the house (the house looks like a square with a triangle on top of it)	4			
Your house has a door	2			
Extra Credit				
Add color to your house	1			
TOTAL:	10			

Instructional Day: 2

Topic Description: This lesson provides an introduction to drawing in Python using coordinates.

Objectives:

The students will be able to:

- Use up() and down() to move the pen without drawing.
- Draw a happy face using goto() and circle().

Outline of the Lesson:

- Journal Entry (5 minutes)
- Introduction of drawing using coordinates (15 minutes)
- Use of the Pen class to make a happy face (35 minutes)

Student Activities:

- Complete journal entry.
- Make a happy face using coordinates.

Teaching/Learning Strategies:

- Journal Entry: Write down detailed instructions for drawing a happy face on a piece of paper using a pen.
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- Introduction of drawing using coordinates
 - Coordinates:
 - The pen class uses an x-y coordinate plane just like in Algebra.
 - The origin (0,0) is in the middle of the canvas.
 - You can use goto(x,y) to move the pen to a specific point instead of using forward(), left() and right(). It is similar to playing connect the dots.
 - Have students try this example on their own and then explain it:


```
pen.goto(0,200)
pen.goto(100,300)
pen.goto(200,200)
pen.goto(200,0)
pen.goto(0,0)
```
 - Ask: What if I wanted to put in a window on the house that didn't touch the wall? Say: We'll come back to that in a moment.
 - Ask the students to volunteer to share their journal response for drawing a happy face as you follow their directions and draw on the board.

- DO NOT lift the marker off the board unless they tell you to. In other words, go ahead and draw lines that connect the mouth to the eyes, etc.
- The main point of the happy face is that sometimes, you want to lift the marker off the board in order to move without actually drawing the connecting lines.
- Introduce `up()` and `down()`
 - `up()`—lift pen off canvas (don't draw)
 - `down()`—put pen down on canvas (will draw)
 - The pen starts down on the canvas.
- Have students use `up()` and `down()` to make a window on the simple house that does not touch the walls. Go over an example with class. Example:


```
pen.up()
pen.goto(50,150)
pen.down()
pen.goto(75,150)
pen.goto(75,175)
pen.goto(50,175)
pen.goto(50,150)
```
- To draw circles, students have to use `pen.circle(radius)`.
 - The current location will be a point on the circle with the center radius units to the left.
 - Show the students `ballon.py`.
- Students use the Pen class to make a happy face.
 - Have students follow directions in Happy Face Project to make their own happy face in Python.
 - See `happy solution.py`.

Resources:

- Drawing Class Reference
- `balloon.py`
- Happy Face Project
- `happy solution.py`

Happy Face Project

Write code that will draw the happy face in the example picture. Have fun!

Square Head:

The lower left corner is at the point (0,0), and the length of each side of the square is 200 units. Therefore, the other points are (200,0), (200,200) and (0,200).

Eyes:

The eyes each have a radius of 10 units and are centered at (60,150) and (140,150).

Mouth:

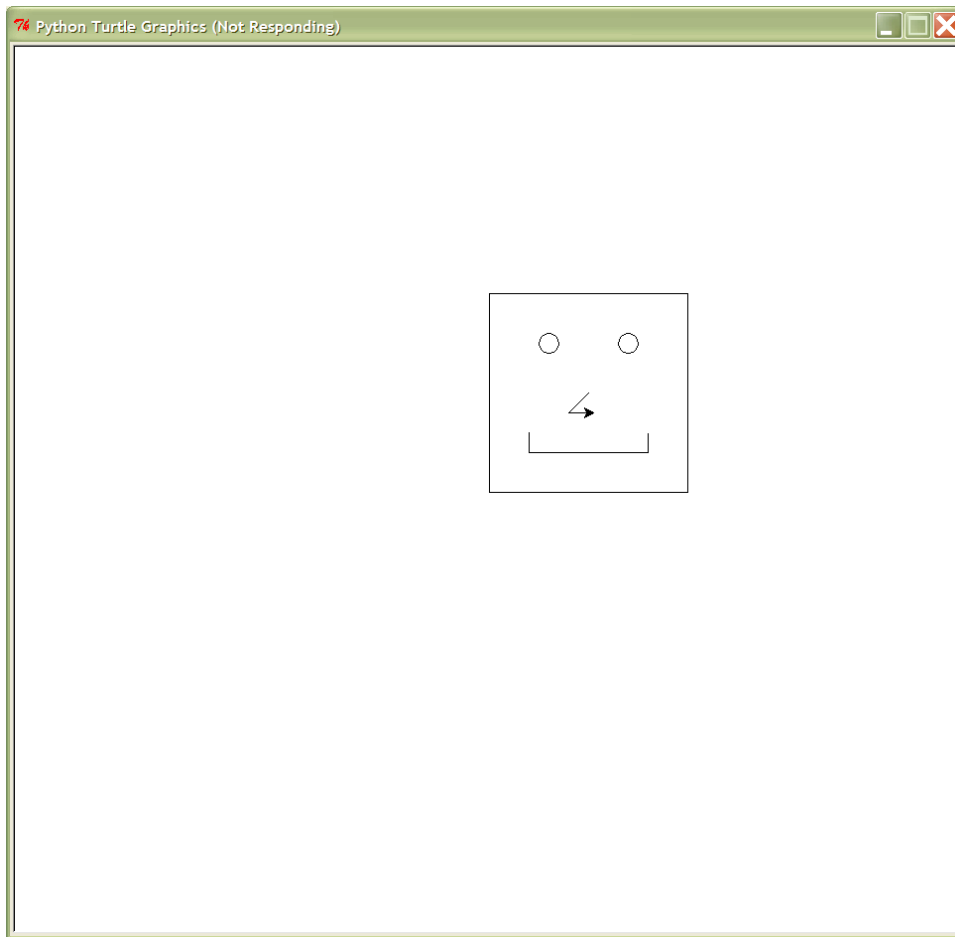
The upper left part of the mouth begins at (40,60). The other points are up to you. Hint, try a y value smaller than 60 for the bottom left of the mouth. For the right side, try x values that are larger.

Nose:

The upper part of the nose begins at (100,100). The other key points are left for you to decide.

Extra credit:

Add hair or a body.



Instructional Days: 3-5

Topic Description: In this lesson students are introduced to the concept of pair programming.

Objectives:

The students will be able to:

- Develop a program using pair programming.
- Use the Pen class to draw their dream house or car.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Review of drawing with Pen class (5 minutes)
- Discussion of pair programming (10 minutes)
- Journal Entry (5 minutes)
- Description of project (5 minutes)
- Dream House/Car project (55 minutes)
- Peer review (15 minutes)
- Completion of Dream House/Car project (40 minutes)
- Gallery walk (15 minutes)
- Discussion of the advantages and disadvantages of pair programming (10 minutes)

Student Activities:

- Complete journal entry.
- Review drawing with Pen class.
- Participate in discussion on pair programming.
- Complete journal entry.
- Work on Dream House/Car project.
- Participate in peer review.
- Complete work on Dream House/Car project.
- Participate in gallery walk.
- Participate in a discussion on the advantages and disadvantages of pair programming.

Teaching/Learning Strategies:

- Journal Entry: Summarize the two ways to draw using the pen class.
 - Monitor students as they summarize the two ways to draw using the pen class.
 - Have students share responses with elbow partner.
- Review of drawing with Pen class
 - Have a few students share their journal entries. Take their responses and sort them into two columns. Try to reinforce some of the following:

- Drawing with forward(), left(), right()
 - You need to keep track of current direction.
 - You don't have to know the coordinates.
 - It can be tricky to connect a line to a previously drawn point (i.e. connecting the roof to the actual house).
 - Drawing using goto()
 - You don't have to worry about the angles.
 - You need to know the coordinates.
 - It can be easier if you can keep track of the points or if you draw your figure labeled on graph paper.
- Introduction of pair programming
 - Describe the process of one person in the pair being the driver (typing) and one being the navigator (reviews code as it is typed) and that the pair switches roles every 30 minutes or so.
 - Point out that there are advantages and disadvantages to this procedure.
 - Have students write what they think some of those might be and tell them that you will discuss those after they have an opportunity to actually participate in pair programming.
- Journal Entry: Summarize the problem solving process from Unit 2.
 - Monitor students as they summarize the problem solving process.
 - Have students share responses with elbow partner.
- Description of project
 - Introduce project (Dream House/Car Project and Dream House/Car Sample Rubric).
 - They will do the programming using the pair programming process.
 - They should use the problem solving process to design their house and think about the code before they begin typing; they can use graph paper to draw their house before coding.
- Dream House/Car project
 - Circulate room and help students.
- Peer Review
 - Have programming pairs examine another pair's project. The examiner pair uses the rubric to give the other pair a progress grade (i.e. , so far you have this many points).
- Continued work on Dream House/Car
 - Circulate room and help students.
- Gallery Walk
 - Facilitate students in circulating the room and completing the Peer Grading form.
- Discussion of the advantages and disadvantages of pair programming.
 - Ask students to report on their experiences.

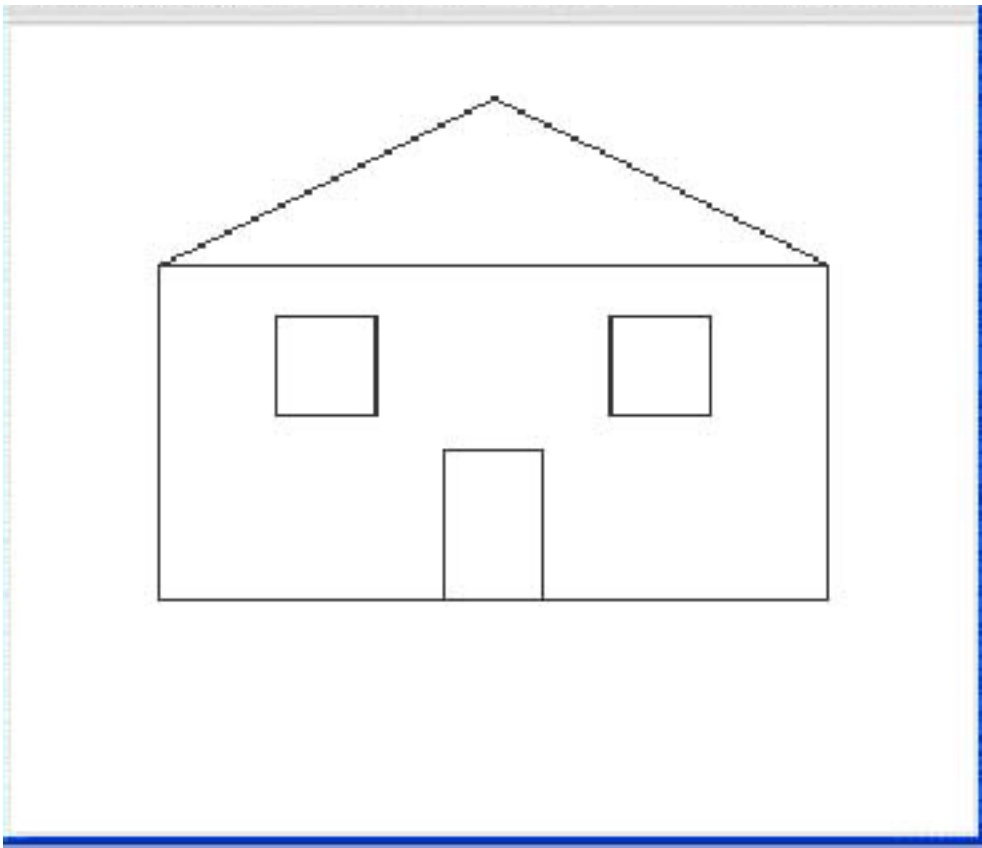
Resources:

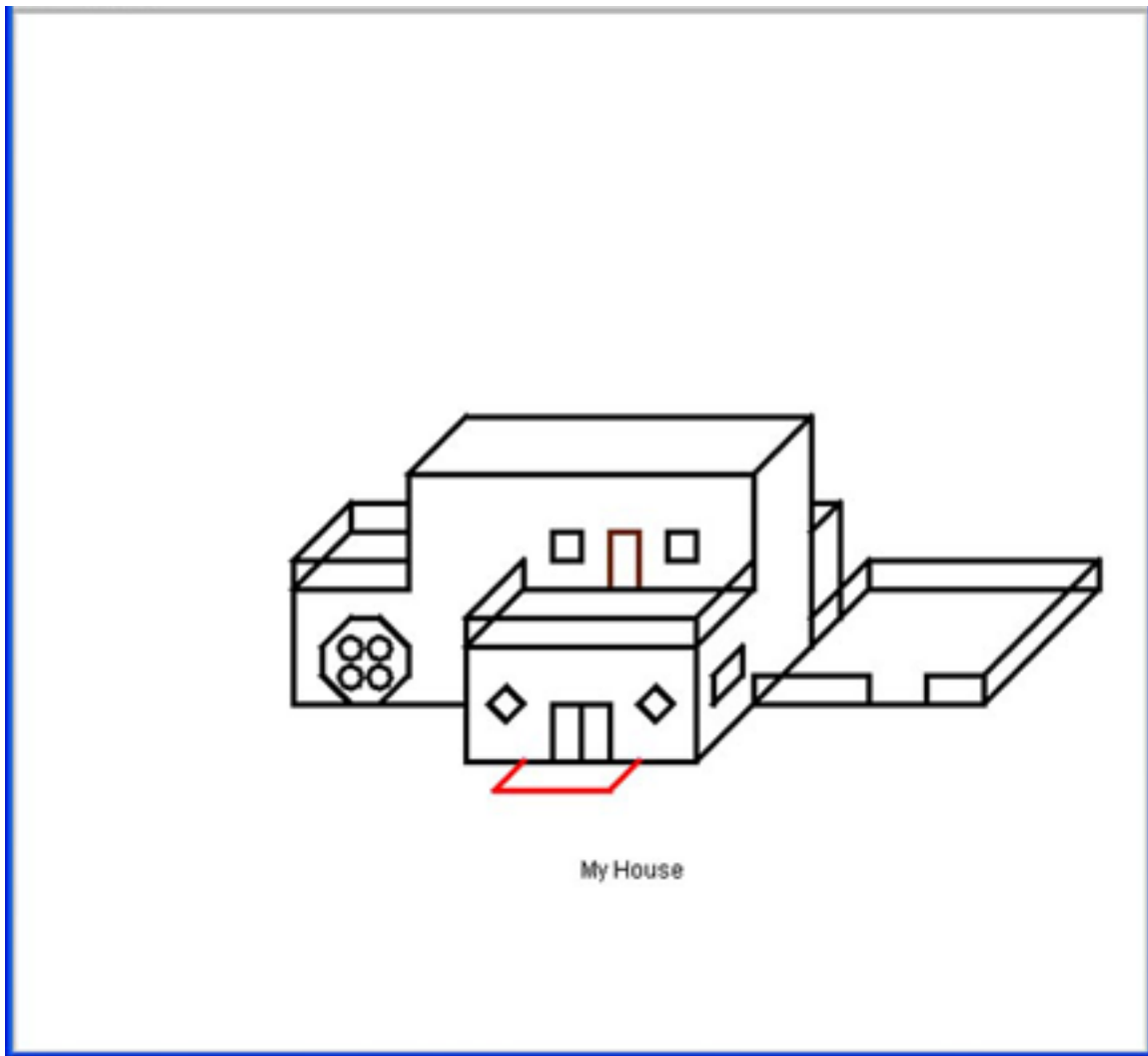
- Dream House/Car Project
- Dream House/Car Sample Rubric
- Drawing Class Reference
- Peer Grading

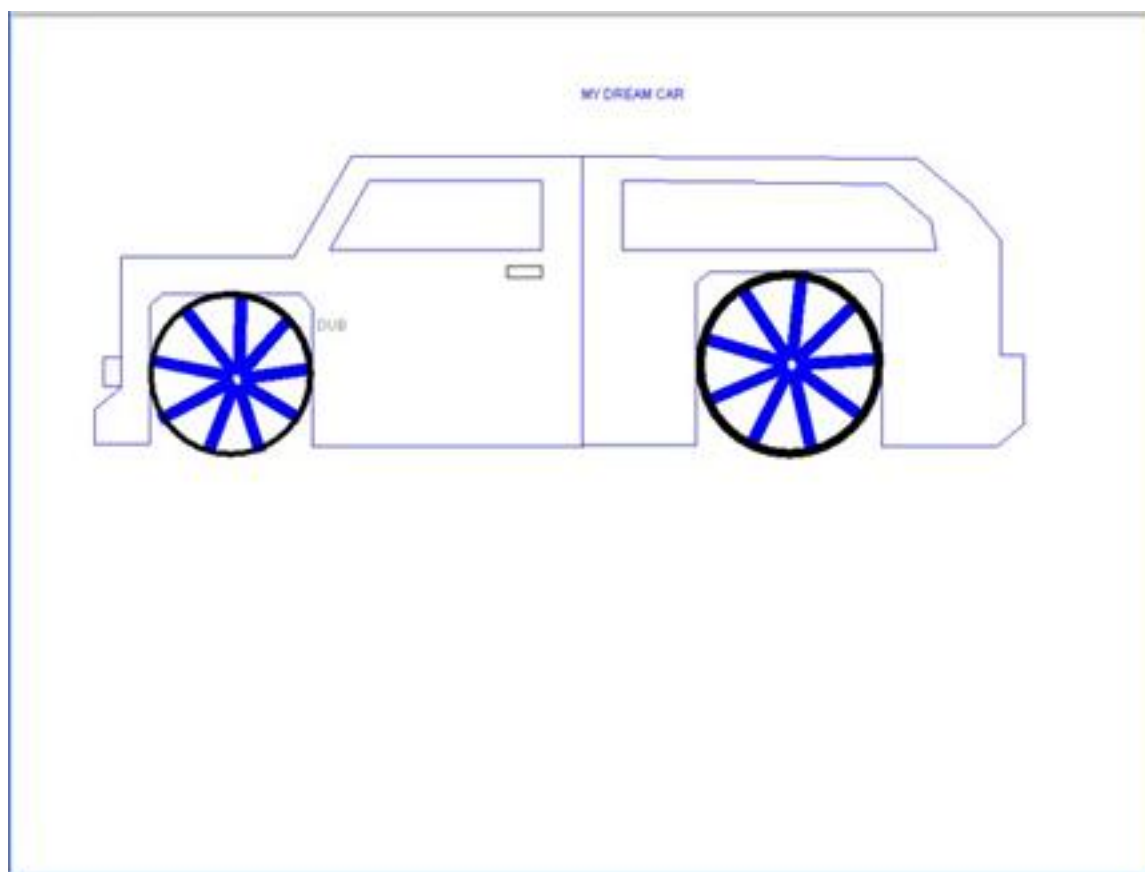
Dream House/Car Project

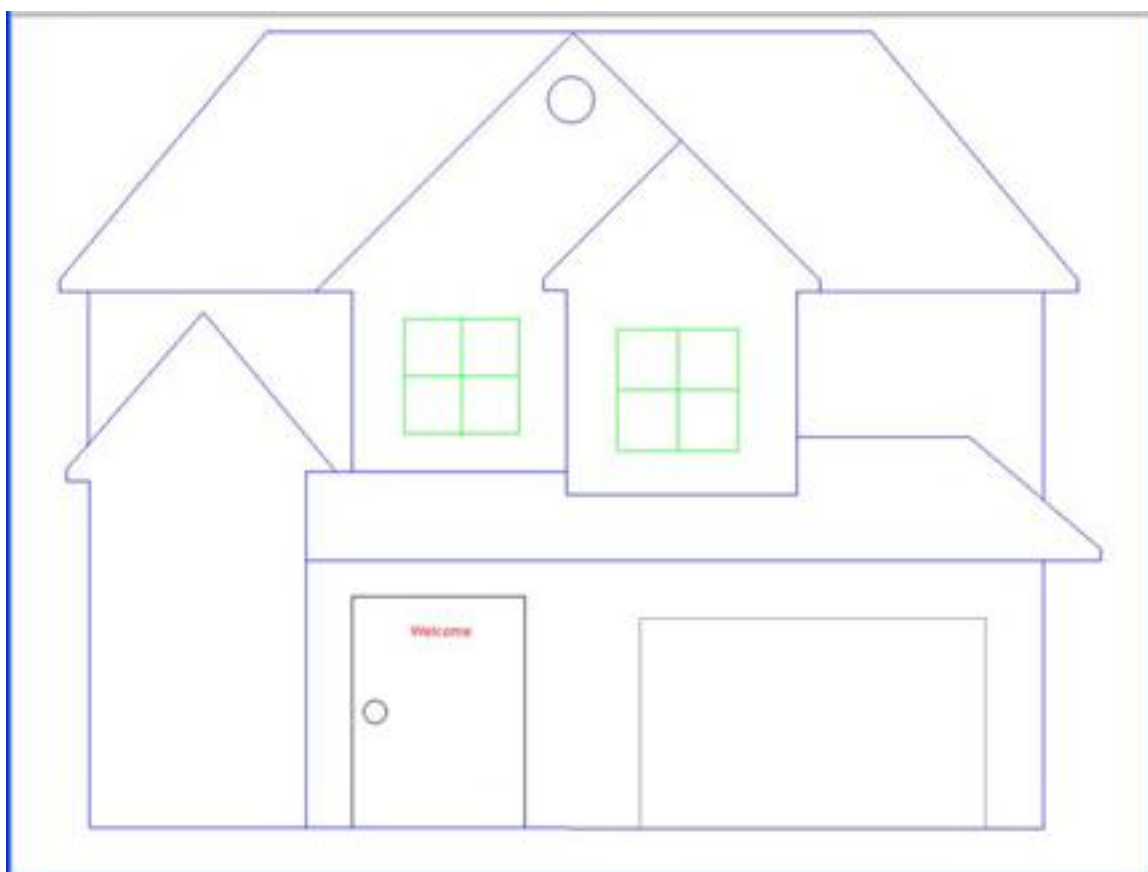
Your task is to use the Pen class to draw your dream house or car. You could actually make a drawing with both. Use your creativity to make your own original looking house or car. You may want to draw your house on paper first to figure out the points. There is extra credit for the best assignment as voted on by your peers.

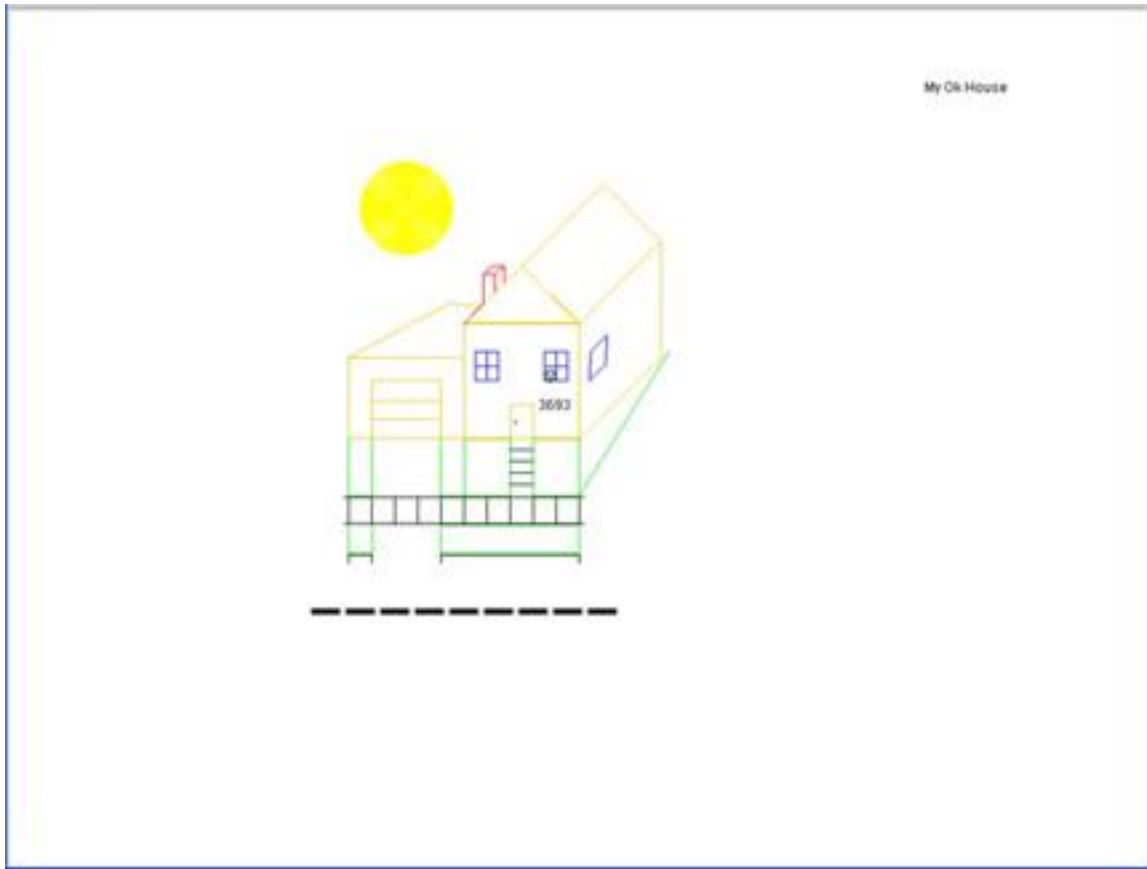
Examples:











Dream House /Car Sample Rubric

Do you have?	Points Possible	Yes	No	Points Earned
Output				
Does your project contain at least one rectangle?	20			
Do you have at least one angle that is not a right angle?	10			
Do you have a circle in your drawing?	10			
Do you use write() to write at least one label on your drawing?	10			
Do you use up() and down() to move the pen without drawing anything?	10			
Do you use the goto() method?	10			
Do you use more than 1 color?	10			
Peer Grading	20			
Extra Credit:				
Your project is voted best by your peers.	10			
TOTAL:	100			

Name_____Computer #_____

VOTINGFrom **ALL** the projects, choose1ST Place_____2nd Place_____**PEER GRADING**For **EACH** of the following give the student a score from 1 to 4.

Use the rubric online to decide the score.

4 – Student has everything on the rubric: A**3** – Student has most things on the rubric: B**2** – Student has some things on the rubric: C**1** – Student turned in project, but is missing many items: D

Student Name	Score (1-4)		Student Name	Score (1-4)

Instructional Day: 6

Topic Description: This lesson provides an introduction to dialogs in Python.

Objectives:

The students will be able to:

- Use dialogs for input and output.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Practice with dialogs (30 minutes)
- Review of answers (20 minutes)

Student Activities:

- Complete journal entry.
- Practice with dialogs.
- Review answers.

Teaching/Learning Strategies:

- Journal Entry: What are some examples of when a program asks the user to input information?
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- Practice with dialogs
 - Individually, students answer questions and follow directions to discover how to use Dialogs for input and output. (See `dialoguepractice.py`)
 - Encourage students to experiment and discover how to use the dialogs.
- Review of answers
 - Review answers with students (see `dialoguepracticesolution.py`).
 - Emphasize:
 - You can just create a variable by typing its name and using the = to save a value into it.
 - Later on, when you refer to the variable, it still has the value that you saved in it.
 - Variables in IDLE are black.
 - Words inside the quotes (string literals) are printed exactly as seen. In IDLE, they appear in green.
 - You can append a variable to a string with a + .
 - \n inside the quotes will create a newline at that point.
 - To put a space in between two variables, you need to place a + " " + in between them.
 - To place a space between a string literal and a variable, you can just put the space inside the quotes.

Resources:

- `dialoguepractice.py`
- `dialoguepracticesolution.py`

Instructional Days: 7-10

Topic Description: In this lesson, students practice using dialogs in Python. They are introduced to software development activities, models and design teams. Students work in teams using a development process to design an order form program.

Objectives:

The students will be able to:

- Write a simple Python dialog.
- Explain software development activities, models, and design teams.
- Use dialogs to create an order form program working in teams.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Review of how to use dialogs (10 minutes)
- Research on software development process, models, and design teams (30 minutes)
- Presentations of research (25 minutes)
- orderform.py (20 minutes)
- Introduction of project (10 minutes)
- Order form project (100 minutes)
- Gallery walk (20 minutes)

Student Activities:

- Complete journal entry.
- Review how to use dialogs.
- Complete research on software development process, models, and design teams.
- Complete presentations of research.
- Work on order form project.
- Participate in gallery walk.

Teaching/Learning Strategies:

- Journal Entry: What are some of the things that you remember from using dialogs yesterday?
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- Review of how to use dialogs
 - Guide students in sharing their journal responses. Make sure all of the features of dialogs are reviewed.
- Research on software development process, models, and design teams
 - Divide students into teams of 3-5, depending on the size of the class.

- Assign 1 or more teams (depending on the total number of teams) to research each of the following topics: software development activities, software development models, and software design teams.
- Presentations of research
 - Have one team for each of the topics present their findings. If there is more than one team per topic, have the other teams add anything that the presenting team left out.
- Order Form
 - Have students run `order form.py` as a prelude to writing an order form program.
- Introduce Project
 - Explain that they will be working in teams to develop an order form.
 - Student teams should develop a project plan, assign team roles, and complete the project.
- Order form project
 - Circulate room and help students get organized as a team and create their design.
- Gallery Walk
 - Have students circulate the room trying everyone else's order form program (everyone go one computer clockwise, etc). Have them complete the Peer Grading form and vote for the best project.

Resources:

- `orderform.py`
- `orderformsolution.py`
- Order Form SampleRubric
- Peer Grading

Order Form Rubric

Name: _____

Do you have?	Points Possible	Yes	No	Points Earned
The Order Form				
Does your order form ask the user for the parts of the address separately?	10			
Does your order ask the user for 3 separate items?	10			
Do you show your order summary at the end?	10			
Do you correctly input all the user's information into the order?	10			
Are there spaces between all the words in your order?	10			
Do you use \n to place newlines in your order?	10			
Coding Style				
Does your file run without errors	10			
Do you have meaningful variable names (ex. name)	10			
Peer Grading	20			
Extra Credit				
Include an introduction to your online store	10			
TOTAL:	100			

Instructional Day: 11

Topic Description: This lesson introduces students to numerical types and math in Python.

Objectives:

The students will be able to:

- Use dialogs to receive numerical input from the user.
- Write a tip calculator program.

Outline of the Lesson:

- readnum.py (10 minutes)
- Number discussion (15 minutes)
- calculatetip.py (30 minutes)

Student Activities:

- Complete readnum.py.
- Participate in number discussion.
- Complete calculatetip.py.

Teaching/Learning Strategies:

- readnum.py
 - Have students follow directions in readnum.py answering questions on paper.
- Number discussion
 - Review answers to readnum.py (see readnumsolutions.py)
 - Emphasize:
 - Integers are whole numbers only (including the negatives).
 - Float (short for Floating Point number) are decimals.
 - str() needs to be used to output integers and floats.
 - The whole reason for the different types is so we can do math on variables that hold integers or floats.
 - Addition is +
 - Subtraction is –
 - Multiplication is *
 - Division is /
 - To save an answer, we need a variable followed by an equals.
 - Example: average = 15 / 4.0
 - This saves 3.75 into the variable average
- calculatetip.py
 - Circulate and help students complete calculatetip.py. (See calculatetipsolution.py.)

- Don't be concerned if dollar amounts in output do not have 2 decimal places.

Resources:

- readnum.py
- readnumsolutions.py
- calculatetip.py
- calculatetipsolutions.py

Instructional Day: 12

Topic Description: This lesson introduces functions in Python.

Objectives:

The students will be able to:

- Use functions in calculations.
- Write a function.

Outline of the Lesson:

- gpa.py (15 minutes)
- Review of gpa.py (5 minutes)
- functions.py (20 minutes)
- Functions discussion (15 minutes)

Student Activities:

- Complete gpa.py.
- Participate in review of gpa.py.
- Complete functions.py.
- Participate in functions discussion.

Teaching/Learning Strategies:

- gpa.py
 - Give students gpa.py. Have them follow the directions in the file and answer the questions.
- Review of gpa.py
 - Review answers in gpasolution.py.
 - In the answer for number 6, make some connections to where integer division is useful:
 - If you have \$18 and pizzas cost exactly \$5 each. If you want to know how many pizzas you can buy, it would be $18/5 = 3$. You wouldn't care about the remainder since they won't sell you fractions of a pizza.
- functions.py
- Functions discussion
 - Review answers to functions.py. (See functionssolution.py.)
 - Give a formal explanation of functions.
 - Functions are defined using the def statement followed by the function name, argument list in parentheses, and a :
 - def functionname(arguments):
 - A function may have any number of arguments (also called parameters). It can even have none.
 - Example of a function with no arguments:

- Tell a student to run the function `raisehand()`.
 - The student should be able to raise their hand without requiring more information (an argument).
- Example of a function that requires an argument:
 - Tell a student to run the function `call()` to call someone on a phone.
 - The student should ask you whom to call. They should not be able to call without knowing the name or number of the person to call (they need an argument for that).
 - The function call should be more like `call("John")`.
- The body of the method must be indented using a tab. The method can be several lines long.
- The function may or may not have a return statement to send information back to the user.
 - Example of a function without a return:
 - Again, tell a student to run the function `raisehand()`.
 - They raise their hand, but don't verbally respond. In other words, they act but do not return any information back.
 - Example of a function with a return:
 - Tell a student to run the function `getage()`.
 - The student should respond by saying their age. In other words, the function `getage()` returns a number that is the age.
- When python sees the `def` statement, it defines the function but does not run it. It is not run until it gets to the bottom of the file where the functions are called.
- The whole idea behind functions is to reuse code and also to not have to repeat the same code over and over.
- You can call methods within methods. In other words, you can use methods as building blocks to build more sophisticated methods.

Resources:

- `gpa.py`
- `gpasolution.py`
- `functions.py`
- `functionssolutions.py`

Instructional Day: 13

Topic Description: In this lesson students will write programs that include functions.

Objectives:

The students will be able to:

- Write a program to exchange currencies.
- Write a program to calculate measurements.

Outline of the Lesson:

- Journal Entry (5 minutes)
- exchange.py (25 minutes)
- measurements.py (25 minutes)

Student Activities:

- Complete journal entry.
- Complete exchange.py.
- Complete measurements.py.

Teaching/Learning Strategies:

- Journal Entry: What do you remember about functions from yesterday?
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- exchange.py
 - Have students help you write the method `pesostodollars()` on the board.
 - Have students follow directions in exchange.py (for answer, see `exchangesolution.py`). Grouping can be individual or in groups of two.
 - After a few minutes, write the answer to `dollarstopesos()` on the board so students can check that they are on track.
- measurements.py
 - Have students follow directions in measurements.py (for answer, see `measurementssolution.py`). Grouping can be individual or in groups of two.

Resources:

- exchange.py
- exchangesolution.py
- measurements.py
- measurementssolutions.py

Instructional Day: 14

Topic Description: This lesson explores the use of conditionals in Python.

Objectives:

The students will be able to:

- Use conditionals to complete Python versions of the Age and Rock, Paper, Scissors programs from Unit 4.

Outline of the Lesson:

- Journal Entry (5 minutes)
- Conditional Discussion (10 minutes)
- age.py (15 minutes)
- Rock, Paper, Scissors (rps.py) (25 minutes)

Student Activities:

- Complete journal entry.
- Participate in Conditional Discussion.
- Complete age.py.
- Complete Rock, Paper, Scissors (rps.py).

Teaching/Learning Strategies:

- Journal Entry: What do you remember about ifs and elses from Scratch?
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.

Conditional Discussion

- Ask students to share their journal entries and write down how conditionals work in Scratch. Then use that to help introduce them in Python. In other words, In Scratch it looks like this while in Python it looks like this.
- The basic structure of an if in python is:
 - `if somecondition:`
- The conditions could be comparisons like in Scratch
 - Equal to: `==`
 - Less than: `<`
 - Greater than: `>`
 - Less than or equal to: `<=`
 - Greater than or equal to: `>=`
 - Not equal to: `!=`

- In Scratch, the body of the if is determined by placing the other puzzle pieces within the if. In Python, the body is determined by indentation (just like the functions).
- In Scratch, the if–else is a separate puzzle piece. In Python, you can place an else after any if.
 - ```
if somecondition:
 body
else:
 body
```
- You may still nest ifs inside of ifs or elses like in Scratch.
- Placing an if inside an else is so common, that python has an elif (else if in other languages). This is known as an if-else chain. It means that the program will execute only one of the choices. It will also stop checking the other elifs once it finds a condition that is true.
  - ```
if somecondition:
    body
elif somecondition:
    body
elif somecondition:
    body
else:
    body
```
- age.py
 - Individually, students follow the instructions in age.py. (See also agesolution.py.)
- Rock, Paper, Scissors (rps.py)
 - Individually, students follow the instructions in rps.py. (See also rpssolution.py.)

Resources:

- age.py
- agesolution.py
- rps.py
- rpssolution.py

Instructional Day: 15-17

Topic Description: This lesson requires students to apply their knowledge of conditionals and functions to develop a Choose Your Own Adventure program in Python.

Objectives:

The students will be able to:

- Use conditionals and functions to write a Choose Your Own Adventure Program.

Outline of the Lesson:

- Presentation of assignment (10 minutes)
- Choose Your Own Adventure Program (100 minutes)
- Presentations/ peer grading (55 minutes)

Student Activities:

- Develop Choose Your Own Adventure program.
- Participate in presentations and peer grading.

Teaching/Learning Strategies:

- Presentation of assignment
 - Show students chooseexample.py.
 - Show students Decision Tree Sample
 - Trace through the tree as you run the example so that the students can see the correlation.
 - Show students Choose Your Own Adventure Sample Rubric.
 - Give students choose.py as a starting place.
 - Recommend writing the decision tree first, and then writing the code.
- Choose Your Own Adventure Program
 - Individually, students develop their adventures.
- Presentations/ Peer Grading
 - Students take turns presenting their adventures in front of the class. The student will read the story as the program is executing.
 - First, allowing the class to make all the choices
 - Next, running the story again making his or her choices
 - Finally, presenting their decision tree
 - The other students complete Peer Grading form making sure to score each student and vote for the best at the end.

Resources:

- Choose Your Own Adventure SampleRubric
- Decision Tree Sample
- choose.py
- chooseexample.py
- Peer Grading

Choose Your Own Adventure Sample Rubric

Name _____

A Choose Your Own Adventure Story is one where the reader is the main character and gets to make choices that affect how the story goes. Depending on the choices that the reader makes, the story can have a good, ok or bad ending.

You will have to come up with your own story and give the user choices. A good place to start is with a decision tree. That is a diagram that shows all the possible choices and endings. It is a required part of the project.

Do you have?	Points Possible	Yes	No	Points Earned
The Story				
Does your story give the user 1 or more sets of choices?	5			
Does your story give the user 2 or more sets of choices?	10			
Does your story give the user 3 or more sets of choices?	10			
Does your story give the user 4 or more sets of choices?	5			
Does your story have an ending for all possible combinations of choices?	10			
Do you have varying endings, some good, some ok and some bad?	5			
Coding				
Do you use functions to separate out the different parts of the story	10			
Does your story handle errors (the person inputs the wrong number)	5			
Decision Tree diagram mapping out all the choices and endings for your story	20			
Peer Grading	20			
Extra Credit				
Have the best project as voted on by peers	Up to 10			
TOTAL:	100			

Instructional Day: 18

Topic Description: This lesson introduces students to the while loop in Python.

Objectives:

The students will be able to:

- Explain uses for the while loop.

Outline of the Lesson:

- Journal Entry (5 minutes)
- busy.py and count.py (15 minutes)
- Iteration Discussion (15 minutes)
- menu.py (20 minutes)

Student Activities:

- Complete journal entry.
- Complete busy.py and count.py.
- Participate in iteration discussion.
- Complete menu.py.

Teaching/Learning Strategies:

- Journal Entry: What do you remember about the forever and repeat blocks from Scratch?
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- busy.py and count.py
 - Individually, students follow directions and answer questions in busy.py and count.py. See busysolution.py and countsolution.py.
 - Students might have to close the Python Shell window to stop busy.py.
- Iteration Discussion
 - Have students start by sharing some of what they remember about the forever and repeat blocks in Scratch.
 - Review answers in busysolution.py and countsolution.py.
 - Have students share their answers to #8 in count.py..
 - Some key points to share:
 - The structure of a while is just like an if. There is a condition section followed by a :.
 - The body of the while is also determined by indenting.
 - The while is like an if that keeps repeating as long as the condition is true.
 - The forever block in Scratch is like the while true in busy.py

- The repeat _ block in Scratch can be made with a while and a counting variable like in count.py.
 - The main use of the while loop is to repeat code without having to rewrite it. This is especially useful for things like menu.py since you can keep taking orders without having to know how many things the user is going to order before you start.
- menu.py
 - Individually, students follow directions in menu.py. See menusolution.py.

Resources:

- busy.py
- busysolution.py
- count.py
- countsolution.py
- menu.py
- menusolution.py

Instructional Day: 19

Topic Description: This lesson introduces students to the for loop in Python.

Objectives:

The students will be able to:

- Use the for loop to make a sunburst and a bottles of root beer program.

Outline of the Lesson:

- Journal Entry (5 minutes)
- for Loop Discussion (10 minutes)
- sunburst.py (20 minutes)
- rootbeer.py (20 minutes)

Student Activities:

- Complete journal entry.
- Participate in for Loop discussion.
- Complete sunburst.py.
- Complete rootbeer.py.

Teaching/Learning Strategies:

- Journal Entry: Describe the way that the while loop was used in count.py.
 - Monitor students responding in journal.
 - Instruct students to share responses with their elbow partners.
- for Loop Discussion
 - In Python, the for loop has a few different uses. Here we are going to use it to iterate a variable through various numbers.
 - The for loop is like a more organized while loop.
 - The basic structure of an if in python is:
 - `for somevariable in range(start, end, increment):`
 - Just like the if and while, the body of the loop is defined by indenting.
 - The loop will start at *start*, but will end before *end*.
 - If you leave off the increment part, it will automatically increment the variable by 1.
 - For loops can be made infinite. See fourth example below.
 - Test the students with some examples:
 - `for x in range (0, 10):`
`print x`
 - output: 0 1 2 3 4 5 6 7 8 9
 - `for x in range (0, 10, 2):`

- print x
 - output: 0 2 4 6 8
 - for x in range (5, 0, -1):
 - print x
 - output: 5 4 3 2 1
 - for x in range (0, 10, -1):
 - print x
 - infinite loop
- sunburst.py
 - Show the students the output of a completed sunburst.py so they know what they are trying to accomplish. (See sunburstsolution.py.)
 - Individually, students follow directions in sunburst.py.
- rootbeer.py
 - Individually, students follow directions in rootbeer.py. See rootbeersolution.py.
 - Make sure to check that their program stops when the number of bottles gets to 0. A common mistake is to have the song continue and end with -1 bottles of soda on the wall (To this, I would ask the student to show me what -1 bottles of soda looks like).

Resources:

- sunburst.py
- sunburstsolution.py
- rootbeer.py
- rootbeersolution.py

Instructional Day: 20

Topic Description: In this lesson students are introduced to the concept of storing data in lists.

Objectives:

The students will be able to:

- Create lists.
- Write some simple list algorithms.

Outline of the Lesson:

- lists.py (15 minutes)
- Lists discussion (15 minutes)
- morelists.py (15 minutes)
- Review of morelistssolution.py (10 minutes)

Student Activities:

- Complete lists.py.
- Participate in lists discussion.
- Complete morelists.py.
- Participate in review morelistssolution.py.

Teaching/Learning Strategies:

- lists.py
 - Distribute lists.py and have students complete the questions and instructions individually or in groups of 2.
- Lists Discussion
 - Review listssolution.py
 - Lists can be created with initial contents (i.e. `nums = [6, 8, 86]`) or empty (i.e. `nums = []`).
 - Lists can be accessed like arrays in most languages using `[]`'s and the index number.
 - The index numbers start with 0 and go to one less than the length of the List.
 - When the List is being used on the left side of an `=` it means save the value on the right into the List. When the List is on the right side of the equals, it means lookup the value in the List.
 - Example:


```
nums = [6, 8, 4]
nums[2] = 86
#nums is now [6, 8, 86]
temp = nums[1]
#temp now equals 8
```
 - Items in the list can be incremented.

- Example:


```
nums = [6, 8, 4]
nums[2] += 1
#nums is now [6, 8, 5]
```
- To get the length of the List (how many items does it hold), use the `len()` function. For example, `len(numbers)` returns the length of the List, `numbers`.
- There are two major ways to iterate through a List:
 - A for loop
 - Example:


```
for n in numbers:
    print n
```
 - This traverses the List item by item in order.
 - During each iteration, `n` is the actual data in the List.
 - This is like using a for each loop in Java.
 - A for loop using range
 - Example:


```
for i in range(0, len(numbers), 1):
    print numbers[i]
```
 - You have to use the `[]`'s to extract the data out of the List.
 - This gives you flexibility to pick the exact range you want to traverse as well as the direction.
 - You may even skip items by changing the incrementing number in the for loop (change 1 to 2 for every other item).
 - This is more like traditional use of the for loop with arrays in many languages.
- Much like ArrayLists in Java, the size of the list can change.
- There are many functions of lists that we are not covering in this unit (appending, slicing, etc.).
- You may want to give students additional examples to check for understanding.
- `morelists.py`
 - Students follow directions in `morelists.py`.
 - If students are stuck, have them first choose the appropriate for loop to accomplish the required task.
- Review `morelistssolution.py`
 - Have students contribute their answers.

Resources:

- `lists.py`
- `listssolution.py`
- `morelists.py`
- `morelistssolution.py`

Instructional Day: 21-24

Topic Description: This lesson requires students to use their knowledge of loops, conditionals, and lists to develop an opinion poll program.

Objectives:

The students will be able to:

- Use Lists, loops, and conditionals to create an opinion poll.

Outline of the Lesson:

- Introduction of project (20 minutes)
- Opinion Poll project (170 minutes)
- Gallery walk of opinion polls (30 minutes)

Student Activities:

- Watch presentation of sample.
- Develop opinion poll programs.
- Participate in a gallery walk of opinion polls.

Teaching/Learning Strategies:

- Introduction of project
 - The Opinion Poll will ask a question and allow the user to pick from a predefined set of answers. When the user votes, the program keeps a tally and displays the results in a bargraph.
 - Run sample poll.py for students so they can see the finished project.
 - Show students Opinion Poll Sample Rubric.
 - Students will need bargraph.py and poll.py in the same folder for the program to work. They only need to edit poll.py. bargraph.py does all the graphing using the Pen class. They might want to peruse it if interested.
 - When you try to run poll.py, it will create a bargraph.pyc file. This is a compiled python file.
 - If students run their program before they do #4, the program will run infinitely.
 - Warning: Using three Lists is not the best way to design this. It would have been better to have one List of some type of object that held the votes, label and color. Three Lists were used in this case to get students to practice using them. Depending on your students, you may want to comment on this as a kind of look ahead to object oriented programming.
- Opinion Poll project
 - Divide students into teams. These teams should be different from the previous team project. Use this opportunity to point out the importance of diverse ideas and styles in a successful project.
 - Circulate room and help students get organized as a team and decide on the topic and questions for their opinion polls.

- Explain to students that they should each ask friends and family to provide their answers to the question posed; they will enter the responses once they have completed their program. (This can be done for homework.)
- Ensure that all teams have data prior to beginning to start programming.
- Circulate room and help students.
- Step 5 in poll.py can be done as


```
votes[vote] += 1
```

 or


```
if vote == 0:
    votes[0] += 1
elif vote == 1:
    votes[1] += 1
...
```
- If students are doing the extra credit and decide to edit the bargraph.py, they will need to run the file before running poll.py in order to see the changes (This should create a new bargraph.py.).
- Gallery walk of opinion polls
 - Facilitate students in circulating the room voting on answers to each other's polls.

Resources:

- Opinion Poll Sample Rubric
- poll.py
- bargraph.py
- sample poll.py

Opinion Poll Sample Rubric

The Opinion Poll will ask a question and allow the user to pick from a predefined set of answers. When the user votes, the program keeps tally and display the results in a bar graph.

Do you have?	Points Possible	Yes	No	Points Earned
Do you ask an appropriate poll question?	15			
Do you provide numbered answers for the user to choose from?	10			
Do you correctly add the user's vote to the poll results?	10			
Do you create a Bar Graph?	10			
Does your Bar Graph have the correct labels for each column?	15			
Does your poll allow the user to quit easily?	10			
Does your Bar Graph use different colors?	10			
Make the answer(s) with the most votes appear in a different color in the bar graph. (For example, the answer with the most votes shows up in blue and everything else shows up in yellow.)	10			
Did you answer the question below?	10			
Extra Credit				
Have your graph display the percentages of each vote under the labels. (For example, 14% blue, 86% silver, and 0% red)	up to 10			
TOTAL:	100			

1. Write down three ways that you used Lists (our friends with the []'s) to help you create your opinion poll.
2. Why should your first choice that the user can vote on be 0?
3. What was the best part of this project?

Instructional Days: 25-30

Topic Description: Complete the final project on analyzing earthquake data.

Objectives:

The students will be able to:

- Incorporate all objectives in the unit into the final project.

Outline of the Lesson:

- Presentation of final project (20 minutes)
- Earthquake Analysis program (255 minutes)
- Presentations (55 minutes)

Student Activities:

- Watch presentation of the project.
- Develop Earthquake Analysis Program.
- Complete presentations.

Teaching/Learning Strategies:

- Presentation of final project
 - Distribute and explain Final Project description.
 - Emphasize that this is real data collected from the Chino Hills quake from July of 2008.
 - Run quakesolution.py so students can see a working version.
 - Distribute and explain Final Project Sample Rubric.
- Develop Earthquake Analysis Program
 - Pair students and give each group a quake.py and a data file. (See seismic data files folder.)
 - Give students some guidance on what they should consider as a timeline for completion of each part in quake.py so that they can plan. (These times obviously will vary by pair and do not include time for the extra credit.)
 - Question # 1: 40 minutes
 - Question # 2: 40 minutes
 - Question # 3: 40 minutes
 - Question # 4: 40 minutes
 - Question # 5: 40 minutes
 - Presentation: 55 minutes
 - The window for the graph may seem too wide depending on the resolution of the students' screens. This was done in order to have the data fit onto one graph.
 - When students experiment with the sampling rate and there is more data than what fits on one graph, it is possible to clear the screen and reset the x to start at the left side of the screen. This goes inside your graphing loop:

```

if x == 490:
    pen.clear()
    pen.up()
    x = -490
    pen.goto(x, 0)
    pen.down()

```

- If students are stuck, encourage them to go step by step.
 - It might help for them to look back on their previous work (especially lists.py and morelists.py).
- When students finish their program, have them start their presentations.
 - Aside from an oral explanation, students should prepare their presentations on powerpoint or a poster; encourage them to be creative.
- Presentations
 - Facilitate pairs in presenting their projects.
 - Have other students grade each group on Peer Grading form.

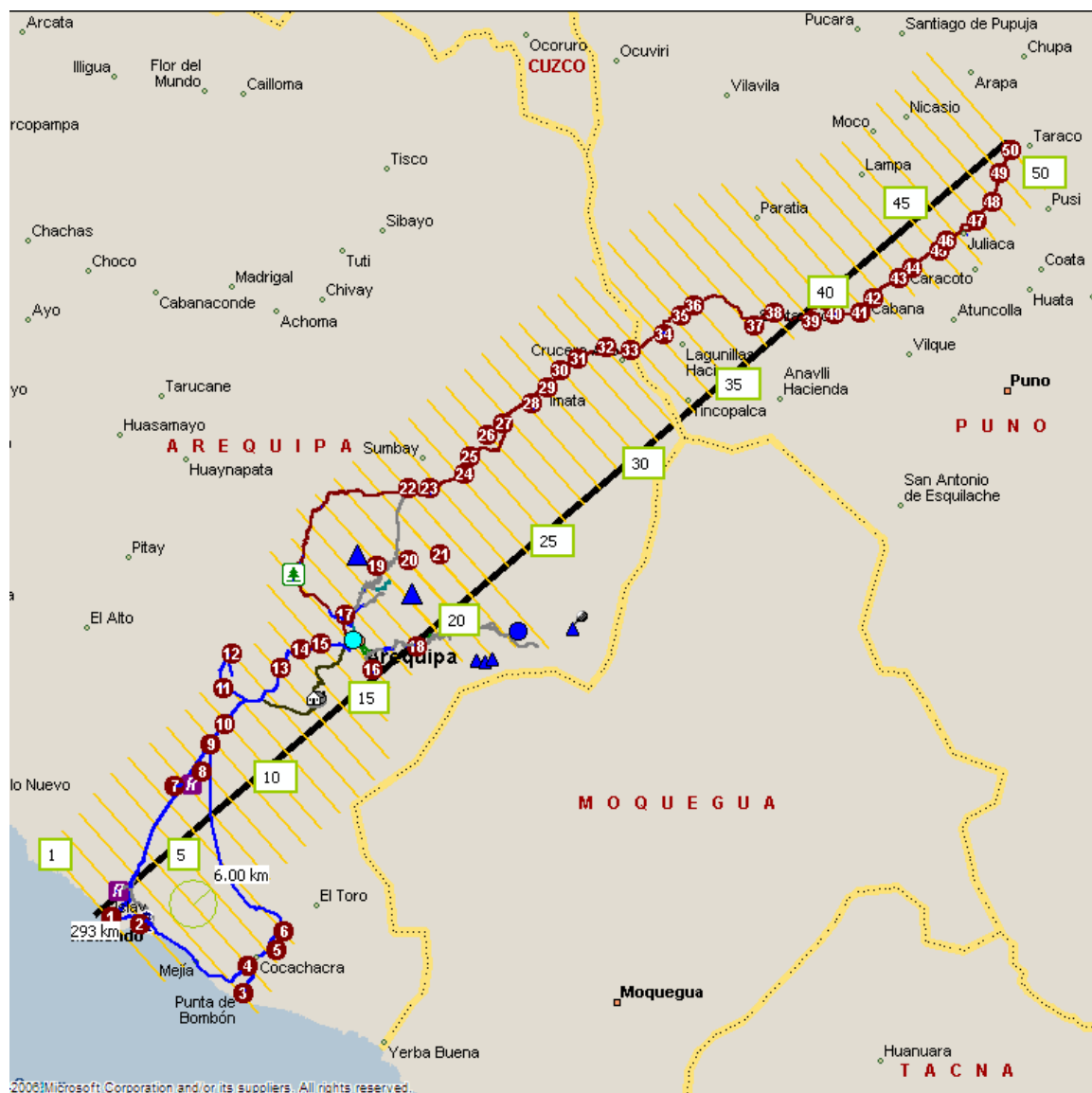
Resources:

- Final Project
- Final Project Sample Rubric
- quake.py
- quakesolution.py
- seismic data files folder(UCLA CENS)
- Peer Grading

Final Project

Where were you at 11:42 AM on July 29th, 2008? If you were in the Los Angeles area you felt a magnitude 5.4 earthquake centered in Chino Hills. Your task is going to be to analyze and graph some of the data collected from that earthquake.

The data was collected by UCLA's CENS (Center for Embedded Networked Sensing). The sensors were setup all the way in Peru! They are numbered on the map below.

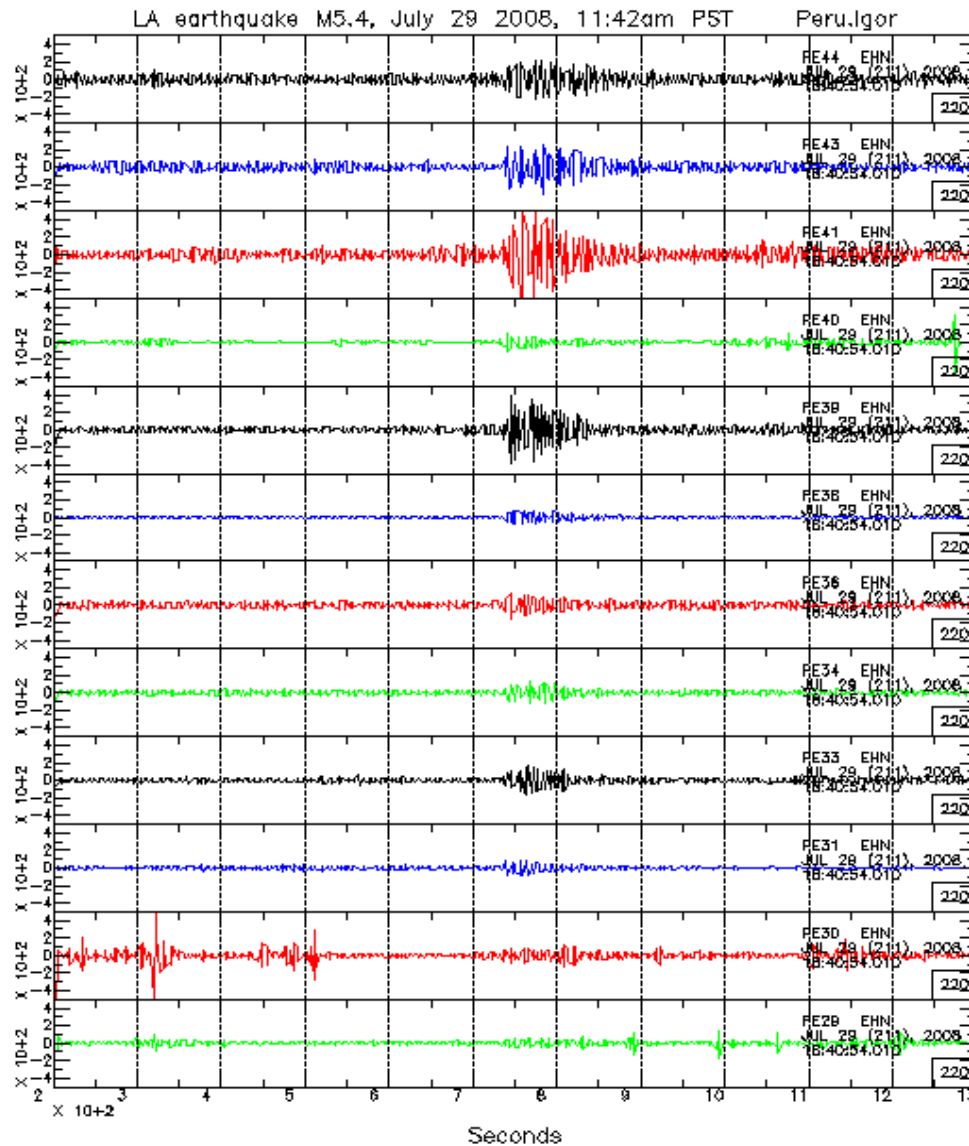


The data files have the sensor names in them. For example, PE03.EHE.ascii is from sensor 3. Each sensor has three files, E, N, and Z to record movement in each of the three dimensions. You will only have to use one of the data files. The data files contain two columns of information; time and the seismic signal. The time is relative. The data is seismic signal is recorded 100 times a second.

Your tasks include the following:

- Find the average of the seismic signals. This information can be used to calibrate the sensors.
- Find the minimum and maximum values.
- Graph the data.

Example graphs (you only need to make one graph):



Links to more information:

- Chino Hills Earthquake Wikipedia Entry - [http://en.wikipedia.org/wiki/Chino_Hills_\(Los_Angeles\)_earthquake_\(CA,_USA\),_2008](http://en.wikipedia.org/wiki/Chino_Hills_(Los_Angeles)_earthquake_(CA,_USA),_2008)
- CENS seismic research - <http://research.cens.ucla.edu/areas/2007/Seismic/>

Final Project Sample Rubric

Do you have?	Points Possible	Yes	No	Points Earned
Program				
Find average of data values	10			
Output the max value in the data	10			
Output the min value in the data	10			
Output possible earthquake if the max exceeds a certain amount	10			
Create graphs of earthquake data	10			
Use data sampling by only looking at every 60 th item in data List	10			
Have graph change color when values are above the threshold for a possible earthquake	10			
Use different sampling of every 30 and every 100 items	5			
Presentation				
State your name(s) and which data set you have	5			
Run your program	5			
Explain the graphs, how the sampling size changes the graph, and the pros and cons of different sampling sizes.	5			
Answer the question: How did you use Computer Science to analyze the earthquake data. In other words, how did you take a file with a bunch of numbers in it and produce a graph that is easier for people to understand. Make sure to be descriptive.	10			
Extra Credit:				
Have your program graph data from multiple files in the same window. See the example graphs section of the project description.	Up to 10			
TOTAL:	100			