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 do 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 it's 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:

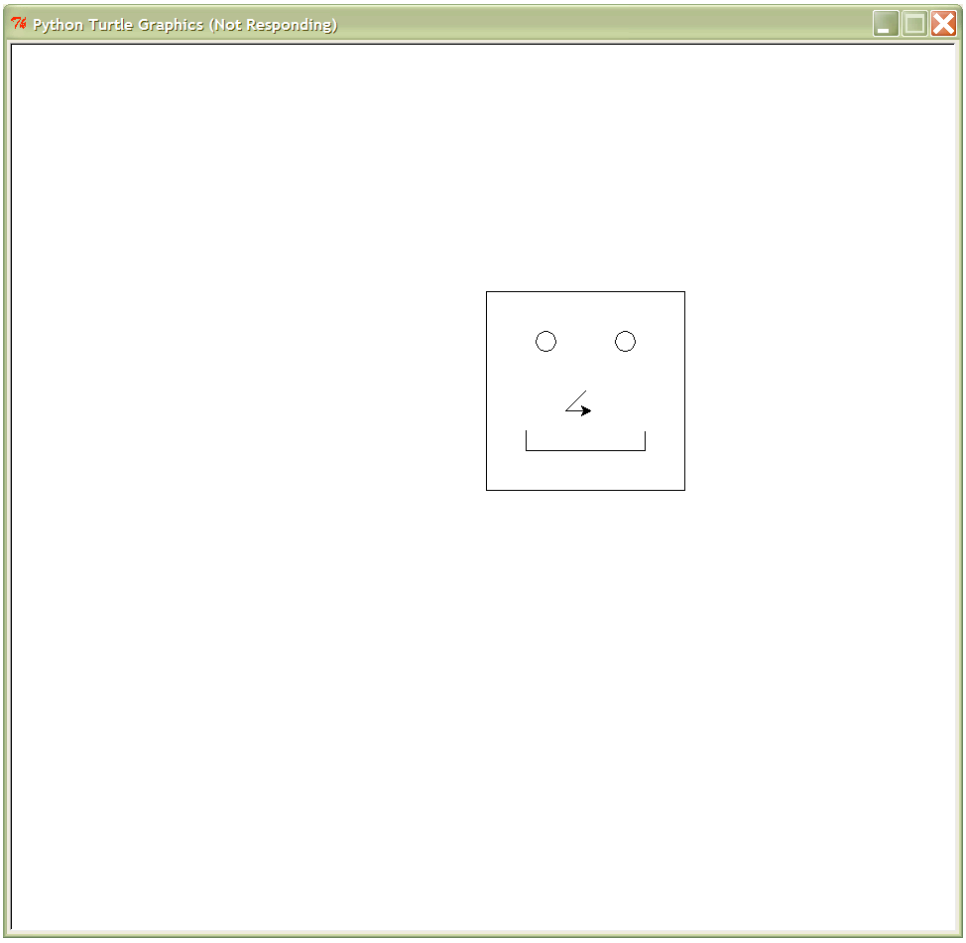
The upper left part of the mouth begins at (40,60). The other points are up to you. Hint, try a y value smaller than 60 for the bottom left of the mouth. For the right side, try x values that are larger.

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.


Python Turtle Graphics (Not Responding)

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.

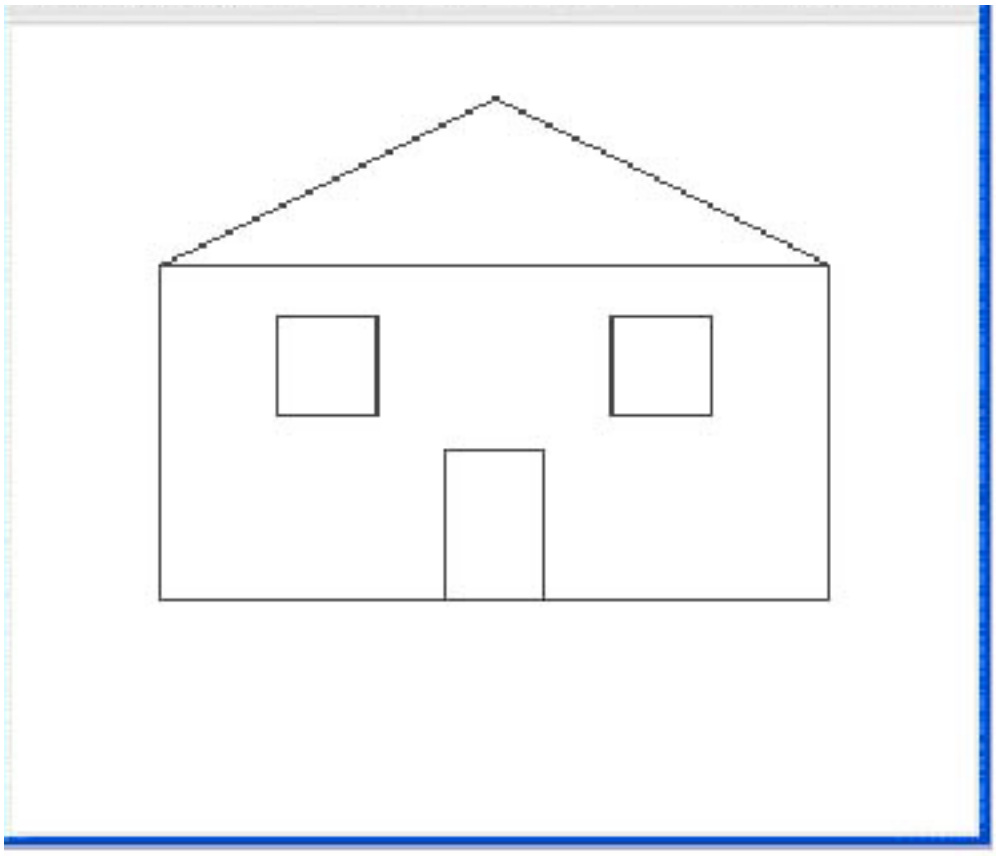
Resources:

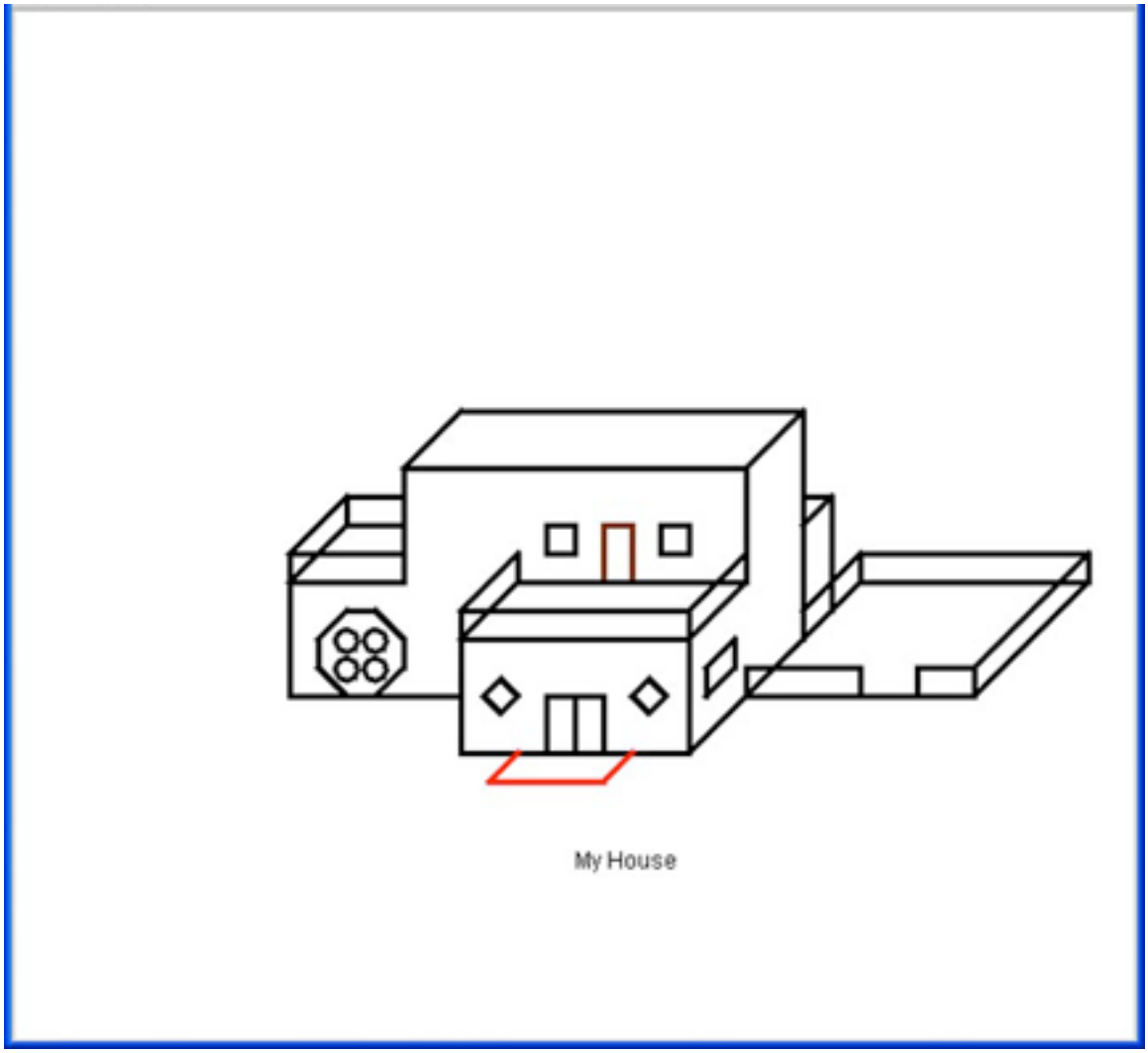
- Dream House/Car Project
- Dream House/Car Sample Rubric
- Drawing Class Reference
- Peer Grading

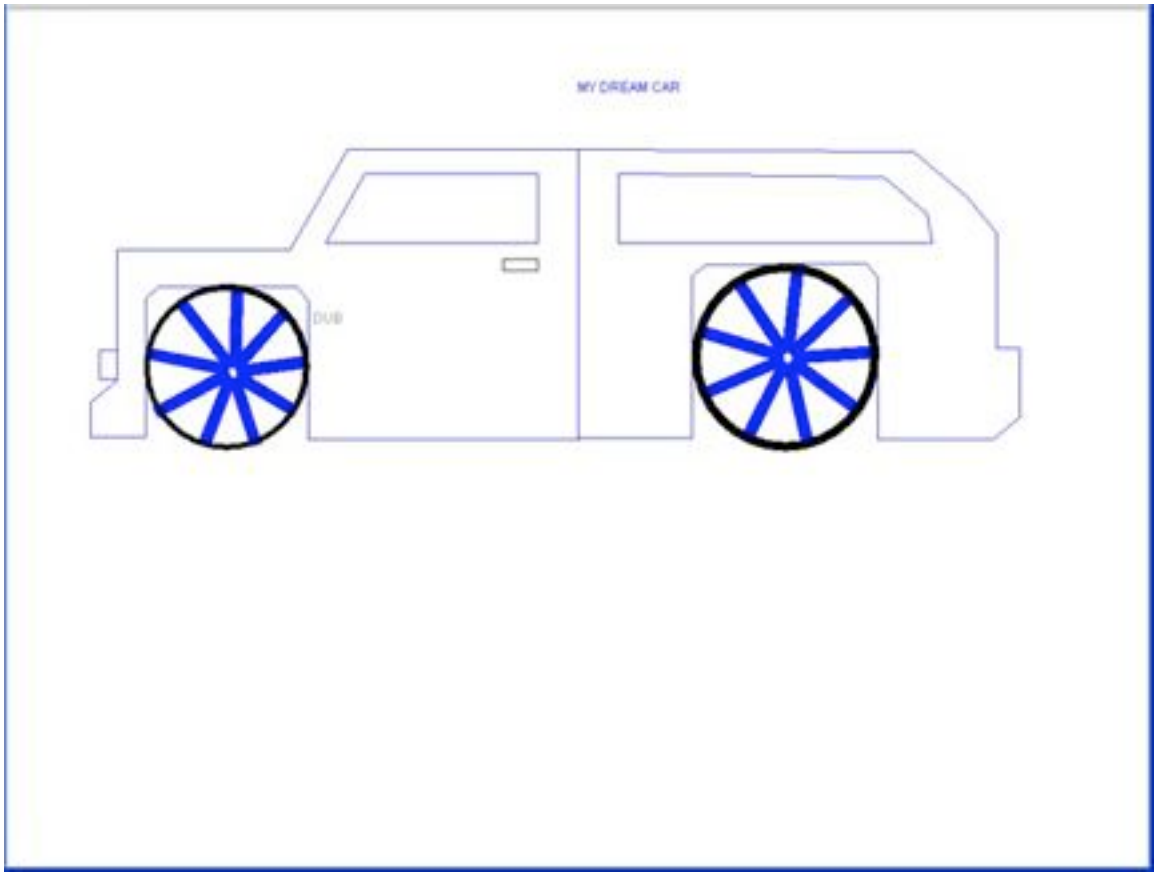
Dream House/Car Project

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:




My House


MY DREAM CAR


Welcome


My Ok House
3693

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 #____

VOTING

From **ALL** the projects, choose

1ST 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 onditional 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 hasan 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 moreliststsolution.py (10 minutes)

Student Activities:

- Complete lists.py.
- Participate in lists discussion.
- Complete morelists.py.
- Participate in review moreliststsolution.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.

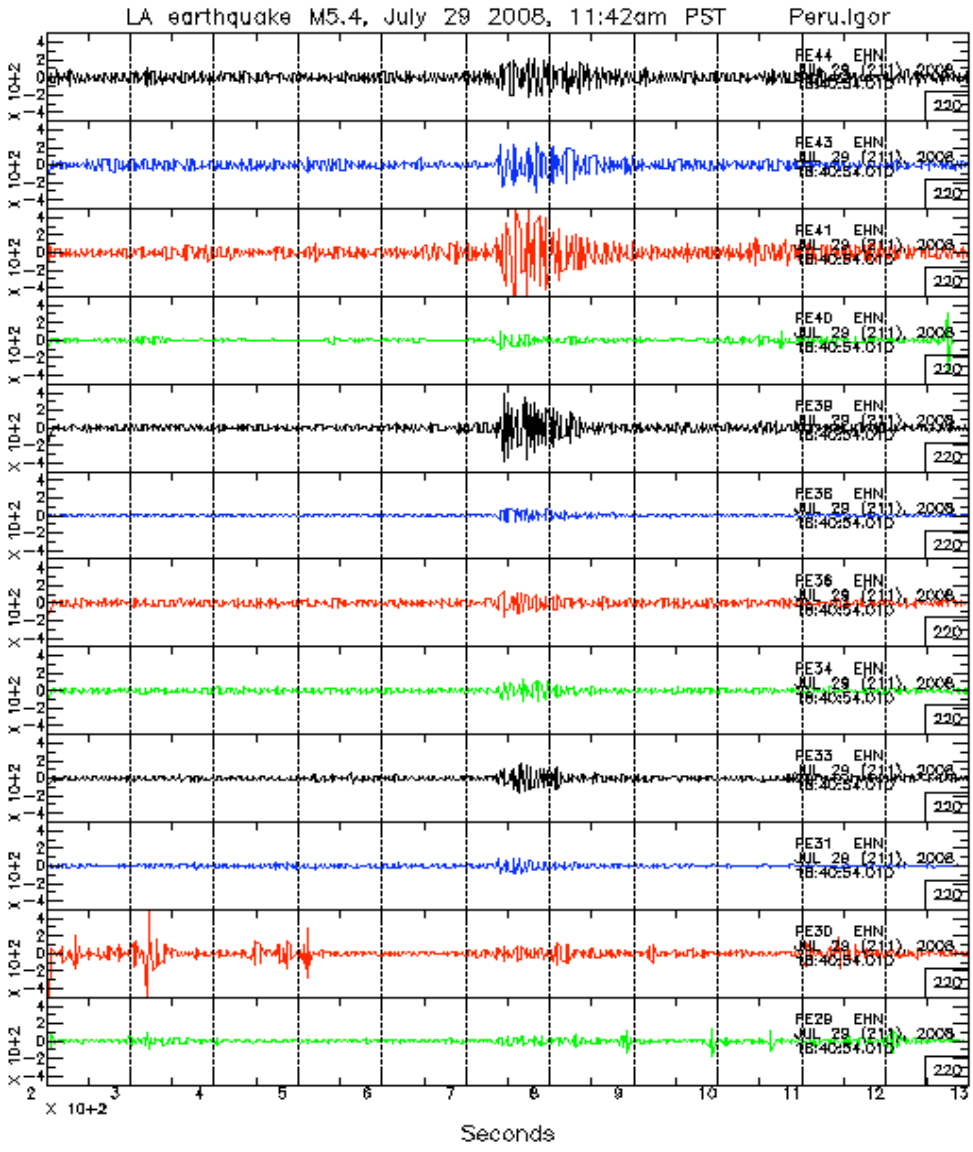
The data was collected by UCLA's CENS (Center for Embedded Networked Sensing). The sensors were setup all the way in Peru! They are numbered on the map below.

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
- 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 60th 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			