**Monday**

**Morning class: Introduction** (180 minutes)

Objective: Coders will get comfortable with each other, the instructor, and the format of the course. Coders will understand the basics of Python.

* (~15 minutes) Encourage the coders to get to know each other with a few ice breakers
* (~5 minutes) Discuss why they are here and what they would like to know by the end of the week
* (~5 minutes) Give the students an idea of the schedule for the week
* (~5 minutes) Classroom expectations: Let the students know what you expect from them - raising hands, being respectful to one another, collaborative not competitive, etc. Expectations on the computer: staying on the site the teacher tells you to be on. Encourage students to ask questions when they hear words they don’t know or get lost during a presentation.
* (~50 minutes) Have coders work through the Lightbot Hour of Code (<https://lightbot.com/flash.html>). This will get them into the problem-solving mindsets they need for the class.
* (~5 minutes) Start with a discussion of computers and computer science. Ask students about all the different things they can do on a computer. How does the computer know how to do those things? Prompt for: someone programmed it, someone coded it. Did they program all those things in the same language? No. Different languages are used for different things. Ask for any programming languages students are familiar with.
* (~30 minutes) Offline activity: Graph Paper Programming. Description and files to print are in the Monday folder, in the “Graph Paper Programming” folder.
* (~15 minutes) Introduce Python. Have the coders set up their programming environments: a window to edit projects in Sublime Text 2, and a window of the terminal in which to run projects.
* (~10 minutes) Show the command line: ls, cd, mkdir, rmdir, rm, nano, clear, javac, java.
* (~10 minutes) Command line Practice activity: in the Day 1 Folder, give students the file called “Command Line Practice” and have them replicate that structure on their desktops using only the command line.
* (~10 minutes) HelloWorld Program: go over HelloWorld.py in the In Class Examples folder. Explain briefly that the quotation marks make the “Hello, World” a String. Obviously, the work “print” makes what follows print onto the screen. Run the program from the terminal. What do they think will happen if you changed “Hello, World” to “Hello, Class”? Then, have the students create and run their own Hello, World Program. Each time they finish a project, they should email it to the instructor so that it will be saved to the cloud for them.
* (~20 minutes) Explain the difference between creating projects and saving them and running them from the terminal and using interactive Python. Have the coders do Assignment 1: Interactive Python in the Monday folder.

**Afternoon class: Variables, Booleans, and User Input** (180 minutes)

Objective: Coders will be familiar with several variable types in Python. They will learn how to use math operations and shortcuts, and understand boolean logic operators for numbers and strings. Coders will learn how to capture and use input typed by users.

* (~15 minutes) Quick review of basic Python. Go over Basics.py in the In Class Examples folder. Go over comments (code that is not executed, has a # in front of it). Shortcut to comment multiple lines of code in Sublime: select them, then press control-/.
* (~10 minutes) Introduce Variables: integers, doubles, booleans and Strings. Discuss how to define a variable (name = value), how to access the variable (name), and the convention for multiword variable names (first\_second\_third).
* (~15 minutes) Go over Variables.py in the In Class Examples folder.
* (~20 minutes) Have coders do Assignment 2: Variables.
* (~20 minutes) More on booleans: Sometimes, we define these explicitly as True or False. Sometimes, we use statements about other variables to define them. Explain how to use logic operators on numbers: ==, !=, >, <, >=, <=. Explain compound operators as well: not (not), &(and), and | (or).
* (~30 minutes) Have coders do Assignment 3: Booleans
* (~10 minutes) Go over how to capture user input in Python. The example file is called KeyboardInput.py.
* (~60 minutes) Have coders do Assignment 4: User Input.

**Tuesday**

**Morning class: Turtle Graphics and Conditionals** (180 minutes)

Objective: Coders will learn how to graph points in the plane and how angles and degrees work to give them the background they need for graphics. Coders will learn how to use Turtle, Python’s graphics module. They will also learn how to do conditionals in Python.

* (~15 minutes) Introduce Turtle. Go through the example program Turtle.py, showing how to create a turtle, how to change its shape using the shape method, how to move a turtle, and how to make the window disapear when the user clicks.
* (~20 minutes each, ~40 minutes total) There are 2 math topics that need to be covered for Turtle. Worksheets for each topic are in the Tuesday folder.
  + Graphing in the cartesian plane. Teach the coders how to graph an (x,y) point, identify the x-axis and the y-axis, and figure out the coordinates from a given point. Have the coders do Assignment 5: Graphing Puzzle in the Tuesday folder.
  + Explain how to use this in turtle using the methods goto(x,y),
  + Angles: Introduce the 360-degree circle, right angles, 180-degree straight lines, and acute and obtuse angles. Show how to measure angles using a protractor. Have the coders do Assignment 6: Angle Types and Measuring in the Tuesday folder.
  + Explain how to use this in turtle using the methods left(degrees) and right(degrees).
* (~20 minutes) Have coders do Assignment 7: Turtle Practice.
* (~15 minutes) Introduce the color, pensize, penup, and pendown turtle methods.
* (~20 minutes) Have coders do Assignment 8: Shapes.
* (~10 minutes) Do the Conditionals activity in the Tuesday folder.
* (~10 minutes) Go through Conditionals.py in the In Class Examples folder. Change the height a few times and have the coders guess what will happen. Then, uncomment the else and try again. Then, uncomment the age section, have the coders guess what will happen, and try a few different ages on that part.
* (~40 minutes) Have coders do Assignment 9: Conditionals.
* (~10 minutes) Have coders do Assignment 10: Shape Shifter.

**Afternoon class: While Loops, Random Numbers, and Events** (180 minutes)

Objective: Coders will be exposed to the concept of while loops and they will practice them in several assignments. They will learn how to use random number generators and use them for typed and graphics assignments.

* (~20 minutes) Do the Loops activity.
* (~15 minutes) Explain While Loops in Python. Explain that they are often used to keep doing a process until something happens - until a user enters the magic password, rolls doubles, etc. Sometimes they are used to do something a set number of times using a counter. Walk through the example file WhileLoops.py in the in Class Examples folder.
* (~30 minutes) Have the coders do Assignment 11: While Loops.
* (~20 minutes) Have the coders do Assignment 12: Shape Shifter Revisited
* (~15 minutes) Explain random number generation using random.randint(min,max). Go over file NumberGame.py in the In Class Examples folder.
* (~40 minutes) Have coders do Assignment 13: Doubles to practice random number generation and while loops.
* (~40 minutes) Have coders do Assignment 14: Random Shapes.

**Wednesday**

**Morning class: Lists and For Loops** (180 minutes)

Objective: Coders will learn how to use lists in Python. They will then learn how to use For Loops to iterate through a list.

* (~60 minutes) Have coders do Assignment 15: High or Low to continue with random numbers, conditionals, and while loops.
* (~20 minutes) Introduce lists! A list is a collection of elements that may have different types. Explain how to create a new list, how to put values in it, how to use the length, check for elements, access by index, and remove elements by going through the Lists.py file in In Class Examples.
* (~40 minutes) Have coders do Assignment 16: Lists.
* (~15 minutes) Introduce for loops by going through the In Class Example ForLoops.py.
* (~45 minutes) Have coders do Assignment 17: For Loops.

**Afternoon class: Projects and Functions** (180 minutes)

Objective: Coders will apply everything they learned to make a game. They will get an introduction to functions and a chance to use functions. Then, they will pick their final projects and begin working on them.

* (~60 minutes) Have coders do Assignment 18: Choose Your Own Adventure.
* (~20 minutes) Introduce functions: defining functions, inputs and outputs, and using functions. Go through the In Class Example Functions.py
* (~60 minutes) Have coders do Assignment 19: Functions.
* (~40 minutes) Introduce the final projects. They will have all day tomorrow to work on them. Encourage them today to just pick a project and try to plan out how to do it. A list of choices for final projects is in the Wednesday folder. Student-chosen projects are allowed provided they fulfil the following guidelines:
  + Take the whole time to create
  + Include at least 1 function, 1 list, 1 loop, and 1 user interaction
  + Have at least 20 lines of code
  + If it is a game, it should be possible to win the game

**Thursday**

**Morning class: Final Python Project** (180 minutes)

Objective: Coders will use all they’ve learned about programming to create a final Python program.

* (~180 minutes) Coders work on their final projects.

**Afternoon class: Final Python Project 2** (180 minutes)

Objective: Coders will continue to work on their final projects and aim to finish them in this class.

* (~180 minutes) Coders work on their final projects.

**Friday**

**Morning class: Final Presentation Prep** (180 minutes)

Objective: Coders will wrap up their Python projects and create and practice their presentations.

* (~5 minutes) Today, we will be making and practicing our final presentations for families which will happen later today.You must run your Python project and then walk the audience through each line of code and what it does. If the project is interactive, either demonstrate it or call a volunteer to use/play it.
* (~60 minutes) Coders work on their presentations.
* (~45 minutes) Coders break into small teams and practice presenting each other and giving constructive feedback.
* (~70 minutes) Coders can continue to work on their presentations, put finishing touches and additional features on their projects, or work on a challenge assignment for Snap (in Friday folder).

**Electives**

For any elective, if the students should be working on their projects instead, then the elective period can be extra work time, essentially a lab. Otherwise, they can do the following:

* Elective 1: How Computers Work activity in Electives folder
* Elective 2: Graph Paper Maze Solving in Electives folder
* Elective 3: Requires computers! Code.org is an introduction to programming for non-coders. Students can do any hour of code tutorial from this list: <https://code.org/learn>

**Sources**

1. <https://code.org/>
2. <http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/Exercises.html>
3. <https://github.com/zhiwehu/Python-programming-exercises/blob/master/100%2B%20Python%20challenging%20programming%20exercises.txt>
4. <http://www.practicepython.org/solution/2014/07/25/13-fibonacci-solutions.html>