Basics/Setup

- Two Basic ways of running code, in an interpreted mode or in script mode.
- To run in interpreted mode, open a terminal and Type “Python” and then hit return.
    - In interpreted mode, code is processed as it is entered.
- To run in script mode. create a file with a “.py” extension and place all your code in there.
    - Once you have your file and are ready to run it, navigate to the directory in the terminal and type the command “python” followed by the name of your program. Once you hit return, your program will process and you should see your output.
    - You can create a python script in almost any text editor.

Coding Conventions

- In python commands are based on lines. So each line features one command and no semi colons are needed.
- In order to use multi line commands end the line with a colon(:) and indent one tab, each command that is under that code block should be indented one tab.
- Comments are designated using the pound symbol(#)
- Python scripts should have the line “#!/usr/bin/env python” at the top
- To extend your code onto another line use the \

Variables

- Variables are handled differently in python than other languages. When you define a variable you do not specify a type, it is inferred.
- Python accepts many types of variables, but we are mainly concerned with Int, float, boolean, and string.
- To declare a string: testString =”This is a string”
- To declare an int: testInt = 12
- To declare a float: testFloat = 12.213
- To declare a boolean: testBoolean = true
- To check what type of variable something is, you can use the function “type()”
    - Just enter the command “type(testInt)” to see what kind of variable it is.
- Something that is used quite often is the global variable. If you declare a variable outside of a function and want to access it as a global variable you can use the global keyword infront of it, to say “Hey I want to use the global version and not a new local one.”
    - global testInt = 2
- To print out the values of a variable you can either just type the variable and hit return or you can type “print testInt”

Functions

- Functions are useful for defining computations/or “functions” that you want to call over and over again. So you can create a function to convert from celsius to Fahrenheit, or reverse a string, … etc. They help you write cleaner and condensed code.
- To create a function use the command “def nameOfFunction():” after this line, indent once and every command that you want to happen in that function should be indented under it as well.
- Functions can also take in variables so that you can manipulate data. To take in data inside the parenthesis just type the name of your new variable that you want to work with.
- To make your program exit out of a function use a return statement. If you return a value then a value is sent back to the calling function, if you just put return it just returns back to where you were.
  Example:

```
def Celsius(Fare):
    Temp = Fare
    Temp = Temp - 32
    Temp = Temp* .5555555
    return Temp
```

- Python comes with some built in functions that can be found by importing libraries

Libraries

- Python has a bunch of libraries that are quite useful and can be used as long as you follow some basic rules
- To use a library all you have to do is use the import command. To use the import command just type “import nameOfLibrary” This allows you to use any function defined in that library inside of your program.
  - To access a function in the library, just use “nameOfLibrary.nameOfFunction”
    - Good example is the Math library
    - import math
    - math.pi
    - math.sqrt()
- Another way to use a library is the import from command.
  - from math import pi -- This allows you to use pi in your program
  - from math import * -- this imports everything so you can use any function without the dot operator.

Data Structures

- Python features quite a few handy data structures, of note is the list data structure.
- A list is a sequence of values and is most similar to an array in other languages.
- To create a list all you have to do is type: nameOfList = [“hello”, “world”, “Cool”]
    - This creates a list of the data you entered on the right hand side. If you want you can also change the values at any point. Just type “nameOfList[0] = “cool”
        - This will change the first item in the list. Each list item has an index and is numbered from 0 to end of list. So the first item is 0 the second is 1 the third is 1 etc, etc ,etc.
- List have several built in functions. One handy one is the append function which takes an object and adds it to the end of a list. nameOfList.append(object)
    - hello = [1,2,3,4,5]
    - hello.append(6)
    - print hello
        - [1,2,3,4,5,6]
- Sort is useful to sort from low to high: list.sort()
- Combine two lists a+b combines list a and b into one big list
- List slice, allows you to take part of a list.
    - a[1:3] takes items 1 through 3.
    - Allow you to change multiple items in a list
        - b[1:3] = [x,y,z]

Control Structures

- While loop
    - used to perform a task while a certain condition holds, i.e x>0 or x<y
    - loop checks is condition true? if so perform loop, if not exit loop
    - syntax: while(x>0):
      x=x-1
    - Very useful for iterating
- For Loop
    - used to iterate over something, unlike in other languages, python iterates over an a sequence in the order that they appear.
    - Standard syntax:
        - for w in words: -- where words is a list
    - If you want to iterate over something by index then you have to combine a few functions, but it does work
        - for i in range(len(words)):
- If Statements
    - Useful for evaluating conditions
    - general syntax: if x>y:
    - can combine with elif and else to create more control

Use Input

- Taking in user input is very simple all you have to do is use the input function for numbers and the raw_input function for strings
- a = input(“Prompt message goes here”)
- b = raw_input(“Prompt here”)
- Both are very handy for dealing with user input