Python Basics Week 1

July 13, 2018

1 Python Basics- Day 1

We are going to be learning the basics of python.
Today we will cover basic types, user interaction, branching, and the random library

```
In [1]: print("Hello World")
```

```
Hello World
```

1.1 Basic Types

- int (integers)
- float (decimals)
- char (characters)
- boolean (logical: True, False)
- Strings (collections of characters)

```
In [1]: # Basic Types - foundational types that make up python.

        3 # int (integer)
        4.5 # float (decimals)
        .45
        45.0

        print(4 - 3)

        # String - collection of characters that is not interpreted
        # Char - single character

        print("4 + 11 =", 4 + 11)
        print("My name is Mitch.")

        # Boolean - Either True or False
        print(False)
        print(True)
```

```
        ## Variables

        num = 27

        color = "Green"

        isBlonde = True

        age = input("Enter your age: ")
        print(age)
        print(color)

        # Casting - turning a type into a different type
        # if age bigger than 10: print You are older than 10.
        # int(v)
        # float(v)
        # str(v)
        # bool(v)
        # char(v)
        numAge = int(age)
```

```
1
4 + 11 = 15
My name is Mitch.
False
True
Enter your age: 15
15
Green
```

1.1.1 Application: Temperature Conversion

Next we will create a simple application that will request a temperature in Fahrenheit and convert it to Celcius. Then it ill prompt the user for a temperature in Celcius and convert it to Fahrenheit. I have outlined the first half with comments.

```
In [1]: from sense_hat import SenseHat ## Delete this line before running
        hat = SenseHat() ## Delete this line before running
        # Ask the user for a temperature, store to variable
        Tempf = input("What is the temperature(f): ")
        # Change that temp into a float, (from a string)
        Tempf = float(Tempf)
        # Convert that number into Celcius from Fahrenheit
        # Tc = (Tf - 32) x (5/9)
        TempC = (Tempf - 32) * (5/9)
        # Print out that number. With some info
        print("Celcius:", TempC)
```

```
        # Go the other direction
        # Tf = ((9/5) x Tc) + 32
        Tc = input("What is the temperature(c): ")
        Tc = float(Tc)
        Tf = (9/5 * Tc) + 32
        print("Fahrenheit:",Tf)

        currentTemp = hat.get_temperature() ## Delete this line before running
        print("Current Temperature: (C)", currentTemp) ## Delete this line before running
        hat.show_message(str(currentTemp)) ## Delete this line before running
```

```
What is the temperature(f): 50
Celcius: 10.0
What is the temperature(c): 50
Fahrenheit: 122.0
Current Temperature: (C) 31.32342529296875
```

1.2 Branching and the Random Library

We will be using a popular library called Random to learn about branching, conditions, and generating random numbers.

Branching allows us to run certain blocks of code only when it suits our needs. Generally this will be done with the `if` statement.

```
if (condition):
____ #Do Something
____ #Do Something else
# Do something outside of the if statement
```

To use Random library we need to use the `import` keyword. This gives us access to all of the classes and functions in the random library. We will be using the `randint(a,b)` function, which returns a random integer between `a` and `b` inclusive.

```
In [2]: import random

        num = random.randint(-5,5)

        if num == 0 or num < 0:
            print("You have no health.")
        if num > 0:
            print("You have {} health".format(num))
```

```
You have 3 health
```

1.2.1 Challenge Program

Write a program that simulates ordering a meal at In-N-Out. First ask the user if they want a hamburger or a cheeseburger. A hamberger costs 3.25, and a cheeseburger costs 4.10. Then ask if

they want fries. Fries cost 2.45 Ask if they want a strawberry or vanilla shake. Strawberrys cost 3.75. Vanilla costs 3.00. Give them their total cost.

```
In [3]: from sense_hat import SenseHat # Don't do this
        s = SenseHat() # Or this
        print("Welcome to In-N-Out!")

        hb = 3.25
        cb = 4.10
        ff = 2.45
        vs = 3.00
        ss = 3.75

        price = 0.0
        burger = input("Would you like a Hamburger or a Cheeseburger? ")

        if burger.lower() == "hamburger":
            print("Got it, one hamburger.")
            price = price + hb
        elif burger.lower() == "cheeseburger":
            print("Comin up, one cheeseburger.")
            price = price + cb

        fries = input("Would you like fries? ")

        if fries.lower() == "yes":
            print("Great")
            price = price + ff

        shake = input("Would you like a Strawberry Shake or a Vanilla Shake? ")

        if shake.lower() == "strawberry":
            print("My favorite, you're all set.")
            price = price + ss
        elif shake.lower() == "vanilla":
            print("Great, you are set.")
            price = price + vs
        print("You're total price is: $"+ str(price))
        s.show_message("$"+str(price)) # Don't write this
```

```
Welcome to In-N-Out!
Would you like a Hamburger or a Cheeseburger? Hamburger
Got it, one hamburger.
Would you like fries? Yes
Great
Would you like a Strawberry Shake or a Vanilla Shake? Vanilla
Great, you are set.
You're total price is: $8.7
```

2 Python Basics - Day 2

Today we will be covering Iteration and Lists and Dictionaries.

2.1 Lists

Yesterday we learned that we can create variables to store our data. But what if we have a large set of data all related to some single topic? In this case we will often turn to a list. A list is a structure in Python that allows us to store data in an ordered way.

```
In [28]: myList = ["Green", "Blue", "Red", "Gray"]
         myNumbers = [6.5, 5.7, 5.8, 4, 5]

         print(myList)
         print(myNumbers)

         # Accessing the elements (zero based)
         print(myNumbers[0]) # Print the first element
         print(myNumbers[3]) # Prints the fourth element
         print(myNumbers[-1]) # Always prints the last element

         # Adding and removing from the list
         # Append always adds to the end of the list
         myNumbers.append(14)
         # Adds 15 to the list as the first element
         myNumbers.insert(0,15)
         # Removes the first element and gives it back to us.
         # Pop default to the last element.
         a = myNumbers.pop(0)
         print(myNumbers)
         print(a)

         # Checking for a value in the list, is "Green" in my list?
         # Counts the number of times "Green" is in the list
         hasGreen = myList.count("Green")
         # Finds where it is in the list, throws error if not found
         greenIndex = myList.index("Green")
         # Returns the number of elements in the list
         length = len(myList)
```

```
['Green', 'Blue', 'Red', 'Gray']
[6.5, 5.7, 5.8, 4, 5]
6.5
4
5
[6.5, 5.7, 5.8, 4, 5, 14]
15
```

2.2 Dictionaries

A dictionary is similar to a list in that it stores multiple pieces of data all at once. The difference is that a dictionary stores its data as key-value pairs. There is no order to a dictionary, the keys are what we use to look up our data.

```
In [50]: # Creates a dictionary
         favoriteColors = {"Mitch": "Green", "Angel": "Red", \
                           "Thomas": "Green", "Harrison": "Green"}

         # Adds a key-value pair to the dictionary
         favoriteColors["Bob"] = "Blue"

         # Remove a key-value pair
         del(favoriteColors["Mitch"])

         # Check if a key is in the dictionary
         hasBob = "Bob" in favoriteColors
         print(hasBob)

         lookUp = "Bob"
         print("{}'s favorite color is {}".format(lookUp, favoriteColors[lookUp]))
```

```
True
Bob's favorite color is Blue
```

2.2.1 Program: Queriable Dictionary

Create a program that looks up a word in a premade dictionary and prints out the definition, if the word they entered is not in the dictionary we will tell them that it is not in the dictionary. Edible

```
In [4]: myWords = {"python": "A general purpose programming language",
                   "computer": "An electrical device that makes \
                                large computations very quickly",
                   "raspberry pi": "A single board computer that is \
                                    great for many small applications"}

        query = input("What word would you like to look up? ").lower()
        if query in myWords:
            print(myWords[query])
        else:
            print("Sorry, Invalid Word.")
```

```
What word would you like to look up? Raspberry Pi
A single board computer that is great for many small applications
```

2.3 Day 3 - Looping and Functions

Often we want to repeat a process over and over again but we don't want to type our code over and over again if we can help it. There are two main types of loops that we use to avoid rewriting code.

`While` loop: runs a block of code over and over again until a condition is false.

```
In [39]: # While loops

         sum_ = 0
         x = 2
         while x <= 1000:
             sum_ += x
             x += 2

         sum2 = 0
         for x in range(0,1001,2):
             sum2 += x

         print(sum_)
         print(sum2)
```

```
250500
250500
```

2.3.1 While Loop Assigment

Find the product of the first 13 prime numbers

```
In [21]: def isPrime(num):
             x = 2
             while x <= num / 2:
                 if num % x == 0:
                     return False
                 x += 1
             return True

         ans = 1
         x = 2
         primes = 0
         while primes < 15:
             if isPrime(x):
                 ans *= x
                 primes += 1
                 print("Prime Found: {}. Ans: {}. Primes: {}"\
                       .format(x, ans, primes))
             x += 1
         print(ans)
```

```
Prime Found: 2. Ans: 2. Primes: 1
Prime Found: 3. Ans: 6. Primes: 2
Prime Found: 5. Ans: 30. Primes: 3
Prime Found: 7. Ans: 210. Primes: 4
Prime Found: 11. Ans: 2310. Primes: 5
Prime Found: 13. Ans: 30030. Primes: 6
Prime Found: 17. Ans: 510510. Primes: 7
Prime Found: 19. Ans: 9699690. Primes: 8
Prime Found: 23. Ans: 223092870. Primes: 9
Prime Found: 29. Ans: 6469693230. Primes: 10
Prime Found: 31. Ans: 200560490130. Primes: 11
Prime Found: 37. Ans: 7420738134810. Primes: 12
Prime Found: 41. Ans: 304250263527210. Primes: 13
Prime Found: 43. Ans: 13082761331670030. Primes: 14
Prime Found: 47. Ans: 614889782588491410. Primes: 15
614889782588491410
```

The second type of loop is the `for` loop. The `for` loop is similar to the `while` loop only we have more control over how many times it is run.

```
In [36]: for i in range(1,11):
             print(i)

         array = ["Mark", "Bob", "Steve"]
         for x in array:
             print(x, "says hi.")

         index = 0
         while index < len(array):
             print(array[index], "_says hi.")
             index += 1
```

```
1
2
3
4
5
6
7
8
9
10
Mark says hi.
Bob says hi.
Steve says hi.
Mark _says hi.
Bob _says hi.
```

```
Steve _says hi.
```

2.3.2 For Loop Assignment

Write a program that gets three integers from the user. Count from the first number to the second number in increments of the third number. Use a `for` loop to do it.

```
In [49]: countFrom = int(input("Count From: "))
         countTo = int(input("Count To: ")) + 1
         countBy = int(input("Count By: "))
         if (countTo - 1 - countFrom) % countBy ==0:
             for i in range(countFrom, countTo , countBy):
                 print(i)
         else:
             print("Invalid Input")
```

```
Count From: 2
Count To: 10
Count By: 2
2
4
6
8
10
```

2.3.3 Challenge Assignment: FizzBuzz

Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

```
In [ ]: num = 1
        while num <= 100:
            if num % 15 == 0:
                print("FizzBuzz")
            elif num % 5 == 0:
                print("Buzz")
            elif num % 3 == 0:
                print("Fizz")
            else:
                print(num)
            num += 1
```

2.3.4 Functions

Often we have block of code that serves one purpose and we want to be able to reuse that functionality any time we need to. We call creating a function "Defining a function". A function definition

is started with the `def` keyword followed by the name of the functions and a pair of parenthesis. Inside the parenthesis we can put any "inputs" that our function need to be able to run. We call the inputs parameters.

```
In [73]: import math

         def square(number):
             print("The square of {} is {}." \
                   .format(number, number * number))
             return number * number

         def _2xPlus5(x):
             return 2*x + 5

         def PythagTheorem(leg1, leg2):
             sumOfSquares = square(leg1) + square(leg2)
             return math.sqrt(sumOfSquares)

         for i in range(0,11):
             b = _2xPlus5(i)
             square(b)

         print(PythagTheorem(3,4))
```

```
The square of 5 is 25.
The square of 7 is 49.
The square of 9 is 81.
The square of 11 is 121.
The square of 13 is 169.
The square of 15 is 225.
The square of 17 is 289.
The square of 19 is 361.
The square of 21 is 441.
The square of 23 is 529.
The square of 25 is 625.
The square of 3 is 9.
The square of 4 is 16.


Out[73]: 5.0
```

2.3.5 Challenge Program: Area Calculator

Create a program that show the user a menu to calculate the area of a few different shapes: square, rectangle, triangle, and circle. Then find the area of the shape with the parameters given by the user.

```
In [81]: import math
```

```
        def inputSquare():
            side = input("Side Length: ")
            side = int(side)
            return side * side

        def inputRectangle():
            w = int(input("Width: "))
            l = int(input("Length: "))
            return w * l

        def inputTriangle():
            b = int(input("Base: "))
            h = int(input("Height: "))
            return (1/2)*b*h

        def inputCircle():
            r = int(input("Radius: "))
            return math.pi * r * r

        print("1. Area of a Square")
        print("2. Area of a Rectangle")
        print("3. Area of a Triangle")
        print("4. Area of a Circle")
        ans = -1
        choice = int(input("\n Type 1-4 to choose: "))
        if choice == 1:
            ans = inputSquare()
        elif choice == 2:
            ans = inputRectangle()
        elif choice == 3:
            ans = inputTriangle()
        elif choice == 4:
            ans = inputCircle()
        if ans != -1:
            print(ans)
        else:
            print("Invalid Input")
```

```
1. Area of a Square
2. Area of a Rectangle
3. Area of a Triangle
4. Area of a Circle

 Type 1-4 to choose: 5
Invalid Input
3.141592653589793
```

2.4 Classes

Classes allow us to encapulate data and functions into one group or class that lets us easily simulate behavior. For example, next we are going to create a rocket class that simulate the position and fuel levels of a rocket. The convience of wrapping our program into a class allows us to create multiple rockets with ease.

Lists - []
Dictionary - {}
Tuples - ()

In [37]:
```
import math

# Takes in two tuples and calculates the distance between them.
def distance_formula(p1, p2):
    sum_of_squares = 0
    if len(p1) == len(p2):
        for i in range(0, len(p1)):
            diff = p2[i] - p1[i]
            sum_of_squares += diff * diff
        return math.sqrt(sum_of_squares)
    else:
        return -1


class Rocket():

    ## Magic method, used to initialize our class into an object. Takes 2 parameters,
    # the starting location of the rocket and its fuel level
    def __init__(self, p0, fuel_level0):
        self._position = p0
        self._fuel_level = fuel_level0

    ## Magic method, used to create the string that is
    # printed when we call the print() function on our object.
    def __repr__(self):
        return "Position: {}; Fuel: {}".format(self._position, self._fuel_level)

    ## Assignment: edit the fly method so that:
    #         We ensure that there is enough fuel to make the trip
    #         If we do, make the trip
    #         If not, print("Not enough fuel")
    def fly(self, p2):
        distance = distance_formula(self._position, p2)
        fuel_used =  0.3 * distance
        ##### Add fuel logic here
        if fuel_used > self._fuel_level:
            print("Not enough fuel")
            return False
```

```
            else:
                self._fuel_level -= fuel_used
                self._position = p2
                return True
            #####

        def refuel(self):
            print("Filling up the fuel tank.")
            self._fuel_level = 100

        def get_position(self):
            return self._position

        def get_fuel_level(self):
            return self._fuel_level

    class Shuttle(Rocket):
        # People is an optional parameter, defaults to 0 if not included
        def __init__(self, p0, fuel_level, people = 0):
            Rocket.__init__(self, p0, fuel_level)
            self._people = people

        def __repr__(self):
            return "Position: {}; Fuel: {}; People: {}"\
                    .format(self._position, self._fuel_level, self._people)

        ## Get People function
        def get_people(self):
            return self._people

        ## Load People function
        def load_people(self, num_people):
            self._people += num_people

        ## Unload People function
        def unload_people(self, num_people):
            self._people -= num_people
            if self._people < 0:
                self._people = 0

    r = Rocket((0,0,0),100)
    print(r)
    r.fly((4,5,9))
    print(r)
    if r.get_position() == (4,5,9):
        print("You made it to the space station!")
    r.refuel()
    print(r)
```

```
        print("\n")

        ss = Shuttle((0,1,0), 100)
        print(ss)
        ss.fly((10,10,11))
        print(ss)
        ss.load_people(5)
        print(ss)
        ss.fly((0,0,0))
        print(ss)
        ss.unload_people(11)
        print(ss)
```

```
Position: (0, 0, 0); Fuel: 100
Position: (4, 5, 9); Fuel: 96.68639169484382
You made it to the space station!
Filling up the fuel tank.
Position: (4, 5, 9); Fuel: 100


Position: (0, 1, 0); Fuel: 100; People: 0
Position: (10, 10, 11); Fuel: 94.78655584090517; People: 0
Position: (10, 10, 11); Fuel: 94.78655584090517; People: 5
Position: (0, 0, 0); Fuel: 89.4116139807545; People: 5
Position: (0, 0, 0); Fuel: 89.4116139807545; People: 0
```