Week 5

This week's assignment:

1. Go to the link below and watch sections 7, 8.1, 8.4, and 8.5

`https://learning.oreilly.com/videos/python-for-data/9780135687253/9780135687253-pfds_01_05_02_00`

6 Functions

6.1 If Functions

You can use Python to create if functions. The concept is simple. If ___ happens, then Python will do ___.

Consider a student much like yourself, working on some programming. How much time will that student spend on one line of code before they give up? You can set up an if statement to state the answer.

```
#time spent on one line of code
time= 12
If (time > 15):
    print("I give up")
Else:
    print("This isn't too hard")

Out: This isn't too hard
```

You can customize your If Statements even more by using elif (short for else if). Using elif, you could create an entire scale of difficulty for the student, ranging from "This is easy" to "Death would be preferable to this."

```
If (time < 5):
    print("Wow this is easy")
elif (time > 5 and time < 15):
    print("This isn't too hard")
elif (time > 15 and time < 30):
    print("I don't think this is possible")
else (time > 30):
    print("Death would be preferable to this")
```

6.2 Lambda Function- Another Best Friend

Throughout the semester, you will get homework with multiple problems that require you to use the same equation. Lambda functions are a way of storing those equations so you don't have to type them over and over again. They are relatively easy to use, but you may have to refer back to this guide often to make sure you get all the steps right.

We're going to use the same growth function that we've been using throughout the guide. First, name your function and then call the lambda command. Next, write down all the variables in your equation. You only have to write the variable once if they appear multiple times in the equation. Place a colon, and write down your equation. All together it should look like this:

```
growth= lambda A, r, t: A * (1 + r) ** t
```

When you want to use the function, all you have to do is this:

```
# if you have already defined the variables
print(growth(A, r, t))

# if you haven't defined the variables.
print(growth(100, .03, 60))
```

You can place an array within the lambda function the same way you put an array in a regular equation. When it is time to solve the equation, simply reference the array that you want to use. For example:

```
initial= np.array([0, 8, 20, 100])
print(growth(initial, r, t))

#Or

print(growth(initial, .03, 60))
```

6.3 Block Function

The block function is essentially a way to create your own custom command. This is useful when you can't find a command in Python to do exactly what you want. The block function programs a new command into Python, complete with a name, input parameters, and what you want Python to do when that command is called.

Below is an example of using the block function to create your own command.

```
# Function definition is here
def printinfo( name, age ):
   #This prints a passed info into this function
    print "Name: ", name
    print "Age: ", age
    return;

# Now you can call printinfo function
printinfo("miki", 50 )

Out: Name: miki
Out: Age:  50
```

When using the block function for mathematical purposes, you essentially add equations in as arguments for the new command. Below is an example of using a block function to create a Profit Calculator.

```
def calculateTradeProfit(buyQ, buyP, sellQ, sellP):
    buyCost = sellQ * buyP
    saleRevenue = sellQ * sellP
    print("You made $" + str(saleRevenue - buyCost)
```

You can use the function like this:

```
buyQ = 100
buyP = 120
sellQ = 50
sellP = 200

# with predefined variables
calculateTradeProfit(buyQ, buyP, sellQ, sellP)

# without predefined variables
calculateTradeProfit(100, 120, 50, 200)
```

7 Loops

Loops are another method for functions that need to be calculated multiple times. They are essential in upper level programming. I encourage you to

use loops in your homework assignments from here on out. Loops can be complicated, but fortunately there are many explanations and resources in this section to help you. Take time to review and understand these examples.

7.1 "For" Loops

There are two basic types of loops: 1) "For" loops and 2) "While" loops. "For" loops are slightly easier to work with than "while" loops, so make sure you understand this section completely before you move on.

`https://www.w3schools.com/python/python_for_loops.asp`

For Loops in Math
What if you were given a system of equations that look like this?

$$x_0 = 10 \tag{3}$$

$$\Delta x = x_i * \frac{1}{2} - x_i * \frac{1}{3} \tag{4}$$

$$x_{i+1} = x_i + \Delta x \tag{5}$$

This system of equations states that the next value of x depends on the current value of x and the change in x (a.k.a. Δx). If you were to calculate x_1, you would have to plug x_0 into equation 4, like so:

$$\Delta x = 10 * \frac{1}{2} - 10 * \frac{1}{3} \tag{6}$$

This would get you a Δx of approximately 1.67. You would then have to plug Δx and x_0 into equation 5 like this:

$$x_1 = 10 + 1.67 \tag{7}$$

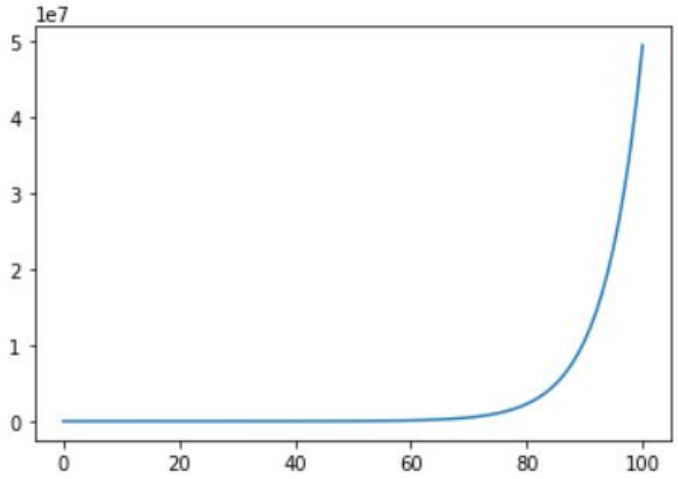
This would make $x_1 = 11.67$. If you wanted to find the value of x_2, you would have to do the whole process over again, this time plugging in the value of x_1 into equation 4. If you wanted to find the value for x_{100}, you would have to calculate all of the values in equations 4 and 5 one hundred times. That would take a ridiculous amount of time, but with Python, you can do it in seconds.

```
1 x = np.zeros(101) #one more than range in loop
2
3 x[0] = 10 #setting 1st number in x to 10
4
```

```
#end range at 1 less than number of zeros
for i in range (0, 100):
    #x_change uses the value in the x array
    #with the same index as the range
    #if i = 0, x_change will use x[0]
    x_change = (x[i] * 1/2) - (x[i] * (1/3))
    #fills in the next value of np.zeros array
    x[i + 1] = x[i] + x_change

print(x[-1]) #prints last value in x
plt.plot(x) #plots new x array

Out: 49508408.19899595
```



In order to see how the loop works, copy the code below into **this** website and press visualize. The code below is the same as the code above, but it doesn't use *np.zeros()* because the visualization website doesn't have NumPy installed.

```
x = ([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

x[0] = 10

for i in range (0, 9):
    x_change = (x[i] * 1/2) - (x[i] * (1/3))
    x[i + 1] = x[i] + x_change
```

7.2 While Loops

While loops continue to run until a condition is met. For example, while $x < 10$, add 1 to x. The website below will tell you everything you need to know about while loops.

`https://www.w3schools.com/python/python_while_loops.asp`