David Brady

February 15th, 2021

Foundations of Programming: Python

Assignment 06

https://github.com/D500844/GitAssignments/tree/main/Assignment06

**Classes, Functions and Reusability**

**Overview**

In the assignment for Module 06 we work extensively with Classes and Functions to further create identity within our Separation of Concerns. A large assignment with multiple areas in need of additional code, we are working with large pre-written files now, but fortunately for us- Classes and Functions will largely be made of the code we have already used in previous assignments. While largely we will be continuing to write the program we have written before, we are now using a combination of multiple advanced coding techniques to do so.

**Functions**

In many languages including Python a Function is a way to combine many actions into a single process, ideal for a situation when the process may need to be used multiple times. When declaring a function, first you use the “def” keyword (short for define) and then write the name of the function afterward.

def doesNothing():

pass

doesNothing()

Functions have parameters which allow you to pass values into a function for processing. Values passed into parameters are officially called arguments by taking values and using them as arguments in parameters. We can access those values multiple times very easily.

Functions can return one or more values. You can also use them to capture returning values of a function in variables. Values can be received or saved by functions, and when a return value is used a function can act as an expression. As a result of this- functions results can be used immediately without being stored in a variable.

Capturing the results in a variable allows you to use the variable of a result multiple times without calling the function again, but using it as an expression does not.

**Variables**

Functions can be totally sealed off and independent from each other and the main part of your program. The only way to get information into them is through their parameters, and the only way to get information out of them is their return values, unless you use global variables.

Variables inside a function are called a local variable, local to the function, and global variables are accessible across the program.

Inside(local)

Def myFunction()

v1= 15 #local variable (to myFunction)

Print(v1) #works

myFunction()

print(v1) #does not work, v1 is Local within myFunction above

v1 = 10 #This would have been Global, outside of myFunction

Organizing code using functions is good for a few reasons.

Functions allow for a large amount of code to be compressed and reused, allowing for more modular programming.

**Re-Writing Code**

Largely in our Assignment we found many functions by looking through the program starter we were given and finding groups of written code that could be saved into a function- compressing its space taken up in the separation of concerns. This gave a really strong sense of identity to each concern, and made some of their more complicated and technical combinations of tools come alive to achieve their local goals.

**Summary**

The assignments have gotten to a point now where the code has become substantial and our techniques are becoming broader. We are focusing on learning things that will really benefit us in the long run- and may be useful as soon as our Resume. We are learning how to read large batches of code we did not write, and work within it by recognizing features within the code- and the inter-relationships, regardless of the differences in the way people write code. This is extremely tricky, and personally I am going to have to open up more time to work on the assignments and review this chapters concepts again, though I think studying the core and pillars/fundamentals is something that will likely help me for many years after the course I’m sure.