 Zefeng Shen

August 15, 2020

IT FDN 110A

Assignment06

<https://github.com/zzsocool/ITFnd100-Mode06/tree/master/docs>

How to Use Classes and Functions in Python

**Introduction:**

This week I learn a lot of information about classes and functions. There is more detail about how to use classes and functions to organize the script and make my code reusable, how to pack and unpack the tuple return from the function, and how to use global variable and local variable to pass the value into the function.

**Classes:**

A class is a code template for creating objects. Objects have member variables and have behavior associated with them. In python a class is created by the keyword class.

An object is created using the constructor of the class. This object will then be called the instance of the class. In Python we create instances in the following manner

A class by itself is of no use unless there is some functionality associated with it. Functionalities are defined by setting attributes, which act as containers for data and functions related to those attributes. Those functions are called methods.

You can assign the class to a variable. This is called object instantiation. You will then be able to access the attributes that are present inside the class using the dot . operator.

Once there are attributes that “belong” to the class, you can define functions that will access the class attribute. These functions are called methods. When you define methods, you will need to always provide the first argument to the method with a self keyword.

(Hackerearth, <https://www.hackerearth.com/practice/python/object-oriented-programming/classes-and-objects-i/tutorial/>) (External website)

**Functions:**

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

* Function blocks begin with the keyword **def** followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.   
    
  (tutorialspoint, <https://www.tutorialspoint.com/python/python_functions.htm>) (External website)

**Global and Local Variables:**

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

(tutorialspoint, <https://www.tutorialspoint.com/global-vs-local-variables-in-python>) (External website)

**Filling up script:**

In this assignment, there were two classes were partial filled with functions. One was processor class, the other one was IO (input and output) class Also, the body of the script need to be added with the function from these two classes.

I started with IO class. I used input() to crab user’s input data. At first, I did not understand how to use the return function. I thought the user’ input would just save in the memory; I can call it out anytime. However, later on when I tested the code, I did not get any input data. Then I used the return function and uppack the data as shown in figure 1.

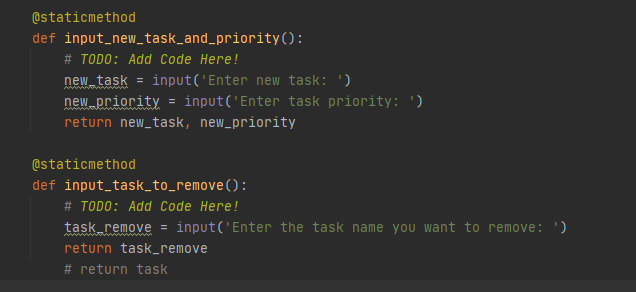


 Figure 1: Return variables.

Then I moved on to processor function. Most of the code I already did in assignment 6 so I just need to copy and paste the code and edit it to make them work in the function. At first, I made same mistake as I did for IO class that I did not use return function to get the variable. Second mistake I made was I did not keep the variable consistently, so the new row of the data did not append on the original data as shown in figure 2.

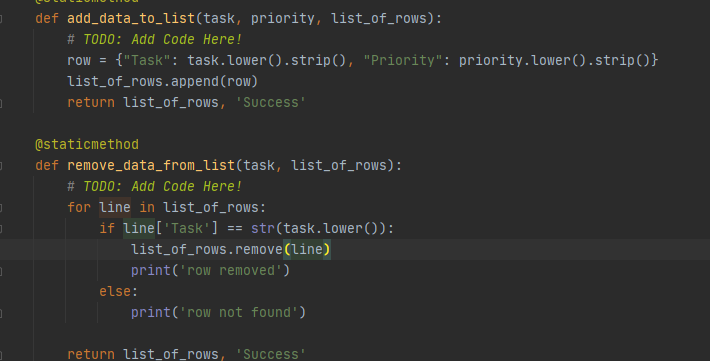


Figure 2: inconsistent variable: row and line.

When I combined the functions from these two classes, I unpacked function and called out the function. However, when I run the code, the same function got processed twice which was unnecessary as shown in figure 3 and figure 4.



Figure 3: Process the function twice.

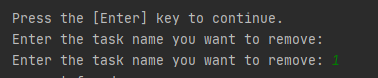


Figure 4: Same result shows twice.

So, I just need to unpack the function would assign the value to variable and run the entire function. Here are my three parts of the script: processor class, IO class, and main script as shown in figure 5, 6, and 7.

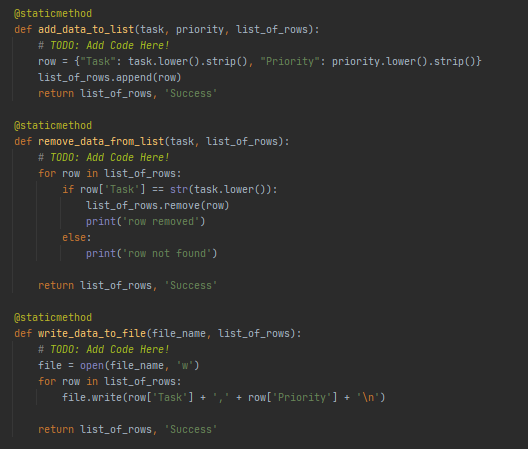


Figure 5: Process code.

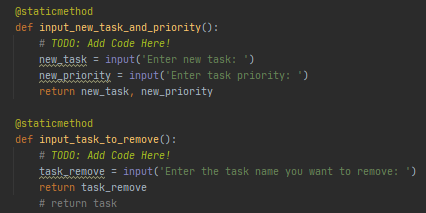


Figure 6: IO code.



Figure 7: Main body of code.

**Summary:**

Functions and classes are useful tool to organize the code. They can be used over and over in the same or different scripts without retyping the code. Function will be processed when unpack the function. While using variable, do not name the same variables in both global and local.