Matthew Kojetin
05/25/20
IT FDN 100 A
Assignment 06
https://github.com/matthewkojetin/IntroToProg-Python-Mod06

Module 06: Functions & classes

Introduction

In this assignment I'll describe how to use functions, store them in simple classes, and apply those concepts to cleanly follow the separation of concerns design principle to organize my code.

Functions

Named set of one or more statements that you can call later in a script (Figure 1).

```
def DoThisAction(): # this block of code loads but does not run yet
  return (int1 / int2)
...
DoThisAction() # Calling the function at the appropriate time, later in the script
```

Figure 1. Basic function syntax.

Parameters

You can pass a value (an argument) into a function for processing (Figure 2).

```
def DoThisAction(string_text): # this block of code loads but does not run yet
  print(string_text)
...
DoThisAction("Argument that is being passed to the parameter 'string_text'")
```

Figure 2. Passing an argument to a function, using the functions parameters.

You can also use a variable as the argument that you're passing.

Return values

You can return a set of values back when a function is called, and those can be assigned to variables for additional processing. When you return multiple values you can do this by packing them into a collection such as tuple packing and unpacking (Figure 3). Alternatively you could use a list or dictionary to store the data.

```
def DoThisAction(string_text): # this block of code loads but does not run yet
  new_text1 = string_text + " 1"
  new_text2 = string_text + " 2"
  return new_text1, new_text2 # packing tuple
...
strNewText1, strNewText2 = DoThisAction("Argument text") # unpacking tuple
```

Figure 3. Assigning return values to variables.

It's important to differentiate that in this example `new_text1` is a local variable (specific to the function), and `strNewText1` is a global variable that can be used elsewhere. You could define a variable within a function as a global variable by saying `global new_text1`.

Classes

Classes are a way of grouping functions. They can contain multiple functions (Figure 4).

```
class Math():

  @staticmethod
  def add(value_1, value_2):
    sum = value_1 + value_2
    return sum

  @staticmethod
  def multiply(value_1, value_2):
    product = value_1 * value_2
    return product
...
Math.add(1,2) # calling the add function from the Math class. Passing arguments
Math.multiply(1,2) # calling the multiply function from the Math class. Passing ar
```

Figure 4. Syntax for a class of functions.

To-do script (v2)

To complete the assignment, I first went through the list of functions in the Processor and IO classes. I found the equivalent chunk of code for each task from last assignment, then replaced the variable names to ensure they matched the local variable names defined in the parameters.

Then I went through each section of the main body of the script and called the appropriate functions to perform the IO and Processor tasks for that selection. If a selection returned multiple values, I followed the model of packing a tuple, and unpacking it to a global variable. A good example of this is the "Add new task" step (Figure 5.)

```
strTask, strPriority = IO.input_new_task_and_priority() # Assigns returned values
Processor.add_data_to_list(strTask, strPriority, lstTable) # Processes data in var
```

Figure 5. Unpacking the return of the input_new_task_and_priority function into global variables, then passing them as arguments to the add_data_to_list function.

I then went through and made sure wherever the `list_of_rows` parameter was required, that I was passing the `lstTable` global variable as the argument.

The script runs in PyCharm (Figure 6).

```
Run:  Assignment06
Mop,Very high saved to file.
Order pizza,Pretty high saved to file.
Fix car,low saved to file.

Press the [Enter] key to continue.
******* The current tasks to-do are: *******
clean (low)
Mop (Very high)
Order pizza (Pretty high)
Fix car (low)
*****************************************


        Menu of options
        1) Add a new task
        2) Remove an existing task
        3) Save data to file
        4) Reload data from file
        5) Exit program


Which option would you like to perform? [1 to 5] -
4: Run   6: TODO   Python Console   Terminal
```

Figure 6. Script running in PyCharm.

Windows command line (Figure 7).

```
Command Prompt - Assignment06.py

Which option would you like to perform? [1 to 5] - 2

Remove an item from the list:

Enter task would you like to remove: Order pizza
Order pizza removed.

Press the [Enter] key to continue.
******* The current tasks to-do are: *******
clean (low)
Mop (Very high)
Fix car (low)
*********************************************


        Menu of options
        1) Add a new task
        2) Remove an existing task
        3) Save data to file
        4) Reload data from file
        5) Exit program


Which option would you like to perform? [1 to 5] - 3

Save this data to file? (y/n) - y
clean,low saved to file.
Mop,Very high saved to file.
Fix car,low saved to file.

Press the [Enter] key to continue.
```

Figure 7. Script running in Windows command line.

Summary

Grouping your processing and IO tasks as functions (and even further organizing as classes) is a clean way to keep your code organized and reusable. You could easily add a class to another project and call the same functions as you need them.