Pablo Montijo

February 22nd, 2021

Foundations of Programming: Python

Assignment6

<https://github.com/pablomarcel/IntroToProg-Python-Mod06>

**Introduction**

In this document I go over functions, arguments, parameters return values and classes. I briefly discuss how to use a debugger and the meaning of GitHub Web Pages. At the end I present the results of the python script.

**What is a function?**

A construct that might receive parameters performs a series of tasks with the parameters (if any) and returns a value if applicable.

**What are parameters?**

Values or variables received by a function when it’s being invoked.

**What are arguments?**

Values or variables passed to a function when it’s called out

**What is the difference between parameters and arguments?**

The argument is passed to a function whenever it is called out. The parameters are received by a function when it’s executed.

**What are return values?**

These are variables returned by a function. The return value might be the product of a series of operations on variables inside the function or can be a function’s own parameter.

**What is the difference between a global and a local variable?**

The local variable is encapsulated within a function. Its state only changes inside the function. A global variable scope is wider. It can change state inside a function or outside a function. If used inside a function, it must be declared as global.

**How do you use functions to organize your code?**

Functions can be used to avoid writing the same code over and over again and therefore keep the code clean. Functions help with code reuse.

**What is the difference between a function and a class?**

A class represents a category of objects on which different functions can be called on. The class has attributes and can have multiple functions or methods. A function, on the other hand is more limited in scope. Functions are limited at doing tasks and returning values.

**How do functions help you program using the “Separations of Concerns" pattern?**

Functions can be used to handle the data, processing and presentation. A program can be divided in sections. Each section can be handled by its related function. For example, one function can be used to handle the operations, that is the processing layer, and another function can be used to handle the ‘input-output’, that is, the presentation layer.

**How are the debugging tools use in PyCharm?**

It’s pretty simple. First breakpoints must be set. On Mac, is sufficient to click on the gutter to set a breakpoint for a line of code. Then, click on ‘debug “*name\_of\_file*”’ next to ‘run “*name\_of\_file*”’. The debugging tool will run the code up to the breakpoint and it is very helpful because it tells you what type of variables are being returned. It follows that is easy to fix the code once potential mistakes in relation to variable types are identified.

**What is a GitHub webpage?**

A welcome page that tells you what the repository is all about. It can have generic information, a menu, and/or a list of hyperlinks to relevant websites or other GitHub webpages.

**Python Script Example**

Choice 1. Calls the *‘input\_new\_task\_and\_priority’* method of class ‘IO’ to capture a new task and a new priority. Then, calls the *‘add\_data\_to\_list’* method of class ‘Processor’ to add the new task to the list table (figure 1, lines 164-168).

Choice 2. Calls the *‘input\_task\_to\_remove’* method of class ‘IO’ to capture a new task to remove. Then, calls *‘remove\_data\_from\_disk’* method of class ‘Processor’ to remove the task from the list table (figure 5, lines 174-177).

Choice 3. Calls the *‘write\_data\_to\_file’* method of class ‘Processor’ to write data to file. The file name and list table are passed as arguments (figure 5, lines 184-185).

Choice 4. Calls the *‘read\_data\_from\_file’* method of class ‘Processor’ to overwrite data to the list table. The file name and list table are passed as arguments. Of course, the contents of the list are cleared, and the list overwritten (figure 7, lines 195-196).

Choice 5. Exit

Text

Description automatically generated

Figure 1. Shows choice 1

Text

Description automatically generated

Figure 2 Shows Class IO, method ‘input\_new\_task\_and\_priority’

Text

Description automatically generated

Figure 3 Shows Class Processor, method 'add\_data\_to\_list'

Text

Description automatically generated

Figure 4 Shows Class Processor and method ‘write\_data\_to\_file’

Text

Description automatically generated

Figure 5 Shows choice 2 and 3

Text

Description automatically generated

Figure 6 Shows Class Processor and method ‘remove\_data\_from\_list’

Text

Description automatically generated

Figure 7. Shows choice 4 and 5

The output from the terminal shows the program is behaving as expected.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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] - 1

enter task: speak

enter priority: 87

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

speak (87)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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] - 2

enter task to remove: walk

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

run (1)

write (8)

publish (88)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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] - 1

enter task: breathe

enter priority: 10

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

breathe (10)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

breathe (10)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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] - 1

enter task: play

enter priority: 1

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

breathe (10)

play (1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Warning: Unsaved Data Will Be Lost!

Are you sure you want to reload data from file? (y/n) - y

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

walk (1)

run (1)

write (8)

publish (88)

breathe (10)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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] - 5

Goodbye!

**Summary**

In this assignment took me less time to complete as classes and functions really make programming easier and less confusing. Everything is encapsulated and it is easy to identify mistakes with the help of the debugger. I run into an issue with return values, but my case, it was easy to zero in and delete the return value ‘success’ to keep it simple and finish faster. I learned about classes, methods, arguments and parameters. Also, reinforced concepts such as dictionaries, lists and iterators.