Elena McDonald

May 20, 2021

Foundations of Programming: Python

Assignment 06 – Functions

# Module 6: Functions

## Introduction

Learning objective of module 06 is to learn how to use functions in Python, organize functions in classes and use arguments to pass values to function parameters. We also looked at debugging functionality in PyCharm and learned how to make a webpage in GitHub.

## Functions

Functions group code into logical sections that allow to manage large code bases. You can review and make changes to code without causing downstream effects to everything in the script. In Python functions are defined before the main body of the script. Then functions can be used (called) as needed. Functions can contain one or more statements. There are some built-in functions in Python (like print(), input(), but anyone can write a function by following Python syntax.

## Parameters

Functions can have parameters. Parameters indicate that values should be passed to function for processing. These values that are passed to the function are called ‘arguments. Parameters are optional.

Example of a function with parameters:

read\_data\_from\_file(file\_name, list)

custom function ‘read\_data\_from\_file’ expects values to be passed into parameters file\_name and list

Parameters can be set in the function as default values. For example:

def AddValues(value 1=0,value 2=1):

## Arguments

Arguments are values that are passed to functions’ parameters for processing. These are often defined as variables in Python scrips, as it makes it possible to use them multiple times in the code. In Figure 1 variables are defined at the start of the script, then function is defined. Then variables are passed into functions parameters:

strTask = "" # Variable captures the user task data strPriority = "" # Variable captures the user priority data

@staticmethod  
def add\_data\_to\_list(task, priority, list\_of\_rows): # functions with parameters task, list\_of\_rowsrow = {"Task": task.strip(), "Priority": priority.strip()}  
 list\_of\_rows.append(row)  
 return list\_of\_rows

strTask, strPriority = IO.input\_new\_task\_and\_priority(). # variables strTask, strPriority get their values  
lstTable = add\_data\_to\_list(strTask, strPriority, lstTable) #variables are passed to function  
print(lstTable)

Figure 1. Example of variables being passed to function as arguments.

Variables can be global and local. Local variables are used insed a function and are not available to the rest of the script. Variable in the main body of the crips are ‘global’ and can be used anywhere in the script. Global variable can be used inside a function, but in this case if should be explicitly declared a global variable. Example:

global strFileName

Even so, using global variable is not preferred since it can cause confusion and unexpected results.

## Return

Return statement allows to put result of function processing into a variable. This allows to use function’s output several times. This is not a required step, function can also be used ‘as is’ expression immediately after processing. For example:

print(AnyFunction()) will print the output of AnyFunction, but the result can’t be used again without calling this AnyFunction every time.

## Classes

Classes are used to group functions into collections. This makes it easier to organize and ready code. Example of function input\_new\_data belonging to class Process:

Process.input\_new\_data(param1,param2)

Class is defined before functions it contains. For example, in Figure 2 class Processor is defined before its child function read\_data\_from\_file(param1, param2)

class Processor:  
 *""" Performs Processing tasks """* @staticmethod  
 def read\_data\_from\_file(param1, param2):

Figure 2. Class and its function are declared in Python

## PyCharm Debugger

PyCharm has built in debugger functionality that helps find problems with code. Placing dot (breakpoint) next to a line of code (as shown in Figure 3 below) will pause code in this spot so can debugging information can be analyzed.

Graphical user interface, application, website

Description automatically generated

Figure 3. Breakpoint in PyCharm

## GitHub Webpage

Webpages can be added to GitHub by using markdown scripting. To add a webpage, open a folder in a repository with a file inside by clicking “New File”.

## Assignment 06

Objective: To modify a script that manages a "ToDo list." by adding more functions to perform tasks listed in the menu. The "ToDo" file will contain two columns of data, "Task" and "Priority."

### Step 0

I downloaded starting script "Assigment06\_Starter.py" to Assignment06 project in PyCharm. I also created a starting text file ToDoList.txt and loaded 2 rows of data into it, representing a task and its priority (1-5]:

make bed,1  
floss,2

### Step 1

I added code lines for the functions in class Processor that:

1. Add new task and priority to the list
2. Remove requested task from the list
3. Write updated list to the file

Here is my code:

@staticmethod  
def add\_data\_to\_list(task, priority, list\_of\_rows):  
 """This function added new task and priority to the list"""  
 # ***TODO: Add Code Here! -- EM: added next 2 lines*** row = {**"Task"**: task.strip(), **"Priority"**: priority.strip()}  
 list\_of\_rows.append(row)  
 print(**"Added!"**)  
 return list\_of\_rows  
  
@staticmethod  
def remove\_data\_from\_list(task, list\_of\_rows):  
 """This function removes task and its priority from the list"""  
 # ***TODO: Add Code Here! --EM added 7 lines below ---*** success\_status = False  
 for row in list\_of\_rows:  
 if row[**"Task"**].lower().strip() == task.lower().strip():  
 lstTable.remove(row)  
 success\_status = True  
 print(list\_of\_rows)  
 return list\_of\_rows, success\_status  
  
@staticmethod  
def write\_data\_to\_file(file\_name, list\_of\_rows):  
 """This function writes updated list to the file"""  
 # ***TODO: Add Code Here! --EM: added 7 lines of code below*** success\_status = False  
 objFile = open(file\_name, **"w"**)  
 for row in list\_of\_rows:  
 objFile.write(str(row[**'Task'**]) + **','** + str(row[**'Priority'**]) + **'**\n**'**)  
 objFile.close()  
 success\_status = True  
 return success\_status

### Step 2

I added code lines to functions in the class IO that handles presentation (input and output).

@staticmethod  
def input\_new\_task\_and\_priority():  
 "This functions asks user to input new task and priority"  
 #pass # ***TODO: Add Code Here! --EM: added 4 lines below*** task = str(input(**"Please enter task: "**)).strip() # EM: collect input for new task  
 priority = str(input(**"Please enter priority 1 to 5: "**)).strip()  
 print(**"You entered: "**, task,**','**,priority)  
 return task, priority # return task, priority  
  
@staticmethod  
def input\_task\_to\_remove():  
 "This function asks user to input task to be removed"  
 #pass # ***TODO: Add Code Here! -- EM: added 2 lines below*** task = str(input(**"Task to remove: "**)).strip()  
 return task

### Step 3

Next I added code lines to call functions according to menu choices selected by the user:

# Step 4 - Process user's menu choice  
if strChoice.strip() == **'1'**: # Add a new Task  
 # ***TODO: Add Code Here --EM:Added 4 lines below*** strTask, strPriority = IO.input\_new\_task\_and\_priority()  
 lstTable = Processor.add\_data\_to\_list(strTask, strPriority, lstTable)  
 print(lstTable)  
 IO.input\_press\_to\_continue(**"Added!"**)  
 continue # to show the menu  
  
elif strChoice == **'2'**: # Remove an existing Task  
 # ***TODO: Add Code Here --EM: Added 8 lines below*** strTask = IO.input\_task\_to\_remove()  
 lstTable, Status = Processor.remove\_data\_from\_list(strTask, lstTable)  
 if Status is True:  
 strStatus = **"Task Removed!"** else:  
 strStatus = **"Task not found. Please try again!"** IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu  
  
elif strChoice == **'3'**: # Save Data to File  
 strChoice = IO.input\_yes\_no\_choice(**"Save this data to file? (y/n) - "**)  
 if strChoice.lower() == **"y"**:  
 # ***TODO: Add Code Here! EM: added 4 lines below*** Processor.write\_data\_to\_file(strFileName, lstTable)  
 IO.input\_press\_to\_continue(**"File updated!"**)  
 else:  
 IO.input\_press\_to\_continue(**"Save Cancelled!"**)  
 continue # to show the menu

### Step 4

### GitHub

I created new repository ***IntroToProg-Python-Mod06*** *to which I uploaded my files. I also created starter webpage ‘Index.md’. Here is a link to repository:*

[*https://github.com/GromitLipton/IntroToProg-Python-Mod06*](https://github.com/GromitLipton/IntroToProg-Python-Mod06)

### Testing in PyCharm

Select menu option 1 for adding data, add task: code, priority: 3. Confirmation received:

Graphical user interface, text, application, email

Description automatically generated

Next, remove task ‘floss’ from the list. Confirmation received:

Text

Description automatically generated with medium confidence

Write updated list to a file:

Graphical user interface, text, application

Description automatically generated

This is what end result looks like in the file:

Graphical user interface

Description automatically generated

### Testing in Command Line Mac OS:

In Mac OS my code only worked when I provided absolute path for the text file in my code.

First, I added new task wash dishes, priority 4 – see output below:

Text

Description automatically generated

Next, I removed task code:

Text

Description automatically generated

And saved updated list:

Text

Description automatically generated

File appears updated:

