Hedy Khalatbari

July 27, 2021

Foundations of Programming: Python

Assignment06

GitHub URL: https://github.com/Seattle15/IntroToProg-Python-Mod06

Programming with Python: Module 6

# Introduction

# Step 1 – Watch the video for Module 6

1. **Functions:** define function and call function. Functions are typically using for processing and presenting data
2. **parameters**: as variable being used by the function; it allows you to pass values to functions. Arguments get passed into parameters and parameters receive arguments

* use snake casing for parameters (such as value\_1)

1. **using variables as arguments**
   * print(**"The Sum of %.2f and %.2f"** % (fltV1, fltV2)) the **%.2f symbols** defines a floating point variable with 2 zeroes after the decimal point
   * alternative format: print(**"The Sum of {} and {}"**.format(fltV1, fltV2))
2. **return values**

* use return key word and either evaluate function as an expression or capture the results in a variable
* allows separation of the processing and presentation codes

1. **return multiple values**

* Python returns multiple values as a tuple which can be unpacked after the function is called
* in other languages (and Python) you can use a list to return multiple values; it is more explicit but extracting the elements with index subscripts is a bit more cumbersome. Tuples read more nicely

1. **Working with arguments**
2. **positional vs named arguments**: named arguments make code easier to read and less prone to error
3. **default parameter value**: when defining the function, you may define a default value for the parameter. If the user does not put in a value, the default value will be used
4. **overloaded functions**: are multiple versions of a function. Each version uses a different number or (data) type of parameters

* the alternative is to write multiple separate functions to achieve the same results

1. **the None keyword**: used to indicate the absence of a parameter value

* often used as a default argument
* check whether a variable is NONE by using “is” operator (rather than “==”)

1. **Return data by reference**
2. value types and reference types: in Python the type is automatically chosen. In many other languages you can choose how to pass arguments into a function explicitly (reference vs value type)

* simple data types -> act as value types
* more complex data types -> act as reference types
* when in doubt check how Python is handling the data type

1. **Global vs. local variables**
2. **local variable**: declared in a container, such as a function; can not be accessed outside the container

* local variables are ‘inside the scope’ of code in the same function
* local variables are ‘outside the scope’ of code outside of the function

1. **global variable**: declared in the body of the script and can be used anywhere in the script

* when using global variables inside a container (*such as a function)* use the keyword "global" to indicate that the variable is global. Otherwise, it will be treated by default as a local variable*,* e.g., global g\_answer
  + when using global variables in a function you do not need to return the value
  + the use global variables within functions breaks the concept of "encapsulation and abstraction" and is discouraged in programming
* name global variables to indicate your intent of how you want to use them. Examples are: dic\_name, lst\_name, flt\_name, g\_variable
  + it is a common practice to name local variables without a prefix; this practice also helps in keeping your global and local variables separate

1. **shadowing a global variable**: if you use a global variable within a function (a practice to be discouraged) use the keyword "global". Otherwise, your local variable will "shadow" the global one whenever you assign a value to a variable with the same name
2. **Function document headers (doc strings)**: a header at the beginning of a function that describes what the function does. It is enclosed in triple quotes

* in IDEs you can access the doc strings; in PyCharm by highlighting the name of the function and using ‘ctrl + q’. Latter is true for built-in functions as well
* when you type triple quotes in the second line of a function, PyCharm automatically prints out the template for a docstring (Figure 1)

Graphical user interface, text, application

Description automatically generated with medium confidence

***Figure 1.*** When you type triple quotes in the second line of a function, PyCharm automatically prints out the template for a docstring

1. **Classes and functions**: functions, variables and constants can be grouped into classes

* @staticmethod
* drill into a class to get to a function: class\_name.function\_name()

1. **Using the PyCharm Debugger**
2. **debug mode and breakpoints:** breakpoints tell the debugger where to pause
3. **walking through code**: place a breakpoint in an active piece of code and use the step into, step over and step out of options to walk through your code. As you go through the code you can look at what your variables are doing (type and value)
4. **Creating and GitHub webpage**
   1. **creating a GitHub webpage**

* open a new repository, name it, and initialize it
* create new folder: Create new file -> docs -> / -> index.md
* in the ‘Edit new file’ window type a line and then ‘Commit new file’
  1. **configuring the website**
* You need to configure it before you can use it. I configured the website by following the sequential steps delineated in Figure 2
* message with link to my site: ‘Your site is published at <https://seattle15.github.io/ITFnd100-Mod06/>’
* check out the link (Figure 3)

**Graphical user interface, application

Description automatically generated**

***Figure 2.*** Configuring the website. a) Choose Settings (arrow). b) This will direct you to a new page and when you scroll down you are informed that ‘GitHub Pages’ now has its own dedicated tab (arrow). Click on ‘Check it out here’. c) In GitHub Pages under Source choose ‘Branch:main’ and ‘/docs’ then click save. The webpage will now have a blue box with the message ‘Your site is ready to be published at…’. Click on ‘Choose a Theme’. d) This opens a new webpage with several theme options. I chose ‘Cayman theme’ for now; however, my choice did not go into effect as we will see in Figure 3. I went back and selected this theme again and in Figure 4 you can see that it is now the theme of my website

Graphical user interface, text, application

Description automatically generated

***Figure 3.*** My initial website available at <https://seattle15.github.io/ITFnd100-Mod06/>

* 1. **modifying the GitHub webpage**
* go to your index.md file and use the edit (pencil-like) button to edit the file; then commit changes
* the language used in Github is a markdown language (Jekyll)
* I made changes like the example in the Module in my page (Figure 4)

Graphical user interface, application, website

Description automatically generated

***Figure 4.*** My modified website available at <https://seattle15.github.io/ITFnd100-Mod06/>

1. **Miscellaneous, formatting outputs**
2. **Formatting output using String modulo operator(%): old way of doing things! Understand it but do not use it, e.g., %4.2f**

* general syntax for a format place holder:%[flags][width][.precision]type
  + %: introduces the place holder
  + flag
  + width: number of digits in output (includes decimal point)
  + .precision: number of digits after decimal point
  + type: such as f (floating point decimal)

1. **Formatting output using the format method – the Pythonic way**

* general syntax:template.format(p0, p1, ..., k0=v0, k1=v1, ...)
* use {} to mark where a variable will be substituted and can provide detailed formatting directives
* allows positional arguments and keyword arguments.  When the positional parameters are used in the order in which they are written, you can omit the positional argument specifiers
* string method "format" works for positional and keyword arguments
  + this can be used to format your floats (especially the results of a division which may have many digits after the decimal point)
* sources [Python | Output Formatting - GeeksforGeeks](https://www.geeksforgeeks.org/python-output-formatting/#:~:text=In%20Python%2C%20there%20is%20no%20printf%20%28%29%20function,string%20modulo%20%28or%20sometimes%20even%20called%20modulus%29%20operator.) (external site), [Python Tutorial: Formatted Output (python-course.eu)](https://www.python-course.eu/python3_formatted_output.php) (external site)

# 6-1. Working with functions

My script worked as expected (Figure LAB1) and I got the corrected results.

Text

Description automatically generated

***Figure LAB1***. Function for add, subtract, multiplication and division

# LAB 6-2. Returning tuples

I declared the float variables in the data section. The function returned a tuple composed of six float numbers; I chose to enclose the tuple in () to make it more explicit. Once I called the function and unpacked it and printed out the results. The code worked as expected.

I could have named my variables more appropriately such as fltSum, fltDiff, fltProd, fltQuot – like Mr.Root did.

|  |  |
| --- | --- |
| **LAB6-2 script** | **Running** |
| *# ------------------------------------------------- # # Title: <Basic Math Functions # Description: <perform four math functions and return a tuple> # ChangeLog: (Who, When, What) # Hedy Khalatbari, 07/26/2021, Created Script # ------------------------------------------------- # # Sum, Difference, Product, and Quotient  # -- Data -- #* flt\_1 = **None** flt\_2 = **None** flt\_3 = **None** flt\_4 = **None** flt\_5 = **None** flt\_6 = **None** *# -- Processing -- #* **def** math\_functions(value1, value2) :  flt\_sum = value1 + value2  flt\_difference = abs(value1 - value2)  flt\_product = value1 \* value2  flt\_quotient = value1 / value2  **return** (value1, value2, flt\_sum, flt\_difference, flt\_product, flt\_quotient) *# a tuple   # -- Presentation (I/O) -- #* flt\_1 = float(input(**"Enter value 1: "**)) flt\_2 = float(input(**"Enter value 2: "**)) flt\_1, flt\_2, flt\_3, flt\_4, flt\_5, flt\_6 = math\_functions(flt\_1, flt\_2) print(**"The Sum of %.2f and %.2f is %.2f"** % (flt\_1, flt\_2, flt\_3)) print(**"The Difference of %.2f and %.2f is %.2f"** % (flt\_1, flt\_2, flt\_4)) print(**"The Product of %.2f and %.2f is %.2f"** % (flt\_1, flt\_2, flt\_5)) print(**"The Quotient of %.2f and %.2f is %.2f"** % (flt\_1, flt\_2, flt\_6)) | C:\Python39\python.exe C:/\_PythonClass/ModDemos/LAB6-2.py  Enter value 1: 23  Enter value 2: 34  The Sum of 23.00 and 34.00 is 57.00  The Difference of 23.00 and 34.00 is 11.00  The Product of 23.00 and 34.00 is 782.00  The Quotient of 23.00 and 34.00 is 0.68  Process finished with exit code 0 |

# LAB 6-3. Creating a class of functions

My script performed the math operations as expected. However, as I used a different code, I got one decimal place after the decimal point for the first three math functions and numerous decimal places after the decimal point for the division function; this is highlighted in the LAB6-3 script in the program listing below.

I delved deeper to learn more about formatting outputs (summarized in 9) Miscellaneous section above) and learned how I could have formatted the output with the string method. I did not rewrite the presentation section of the code but a simple work around for the quotient output is assigning the number returned from the division function to a variable and formatting the variable with the format method as follows:

division\_result = MathProcessor.DivideValues(flt\_one,flt\_two)  
print(**'{0:5.2f}'**.format(division\_result))

I reran the script and the revised result is pasted in the box below with the changes highlighted in yellow.

|  |  |
| --- | --- |
| **LAB6-3 script** | **Running** |
| *# -- define variables -- #* flt\_one = **None** flt\_two = **None** *# -- processing code -- #* **class** MathProcessor():  *""" functions for processing simple math """* @staticmethod  **def** AddValues(value1=0.0, value2=0.0):  *""" This function adds two values* **:param** *value1: (float) the first number to add* **:param** *value2: (float) the second number to add* **:return***: (float) sum of two numbers  """* **return** float(value1 + value2)   @staticmethod  **def** SubtractValues(value1=0, value2=0):  *""" This function subtracts two values* **:param** *value1: (float) the first number to subtract* **:param** *value2: (float) the second number to subtract* **:return***: (float) sum of two numbers  """* **return** float(value1 - value2)   @staticmethod  **def** MultiplyValues(value1=0, value2=0):  *""" This function multiplies two values* **:param** *value1: (float) the first number to multiply* **:param** *value2: (float) the second number to multiplpy* **:return***: (float) product of two numbers  """* **return** float(value1 \* value2)   @staticmethod  **def** DivideValues(value1=0, value2=0):  *""" This function divides two values* **:param** *value1: (float) the first number to divide* **:param** *value2: (float) the second number to divide* **:return***: (float) quotient of two numbers  """* **return** float(value1 / value2)  *# -- presentation (I/0) code -- #* flt\_one = float(input(**"Enter value 1: "**)) flt\_two = float(input(**"Enter value 2: "**)) print(**'The Sum of '** + str(flt\_one) + **' and '** + str(flt\_two) + **' is'**, end = **' '**) print(MathProcessor.AddValues(flt\_one,flt\_two)) print(**'The Difference of '** + str(flt\_one) + **' and '** + str(flt\_two) + **' is'**, end = **' '**) print(MathProcessor.SubtractValues(flt\_one,flt\_two)) print(**'The Product of '** + str(flt\_one) + **' and '** + str(flt\_two) + **' is'**, end = **' '**) print(MathProcessor.MultiplyValues(flt\_one,flt\_two)) print(**'The Quotient of '** + str(flt\_one) + **' and '** + str(flt\_two) + **' is'**, end = **' '**) print(MathProcessor.DivideValues(flt\_one,flt\_two)) | C:\Python39\python.exe C:/\_PythonClass/ModDemos/testMod6.py  Enter value 1: 23  Enter value 2: 236  The Sum of 23.0 and 236.0 is 259.0  The Difference of 23.0 and 236.0 is -213.0  The Product of 23.0 and 236.0 is 5428.0  The Quotient of 23.0 and 236.0 is 0.09745762711864407  # -------------------------------------------------------------------------------- #  # ---------------------- Running the revised script -----------------------#  C:\Python39\python.exe C:/\_PythonClass/ModDemos/LAB6-3.py  Enter value 1: 23  Enter value 2: 236  The Sum of 23.0 and 236.0 is 259.0  The Difference of 23.0 and 236.0 is -213.0  The Product of 23.0 and 236.0 is 5428.0  The Quotient of 23.0 and 236.0 is 0.10  Process finished with exit code 0 |

# Step 2 - Read a book chapter

I read the book chapter.

# Step 3 - Web pages

<http://www.learnpython.org/en/Functions>(external site)

I read the web page.

# Step 4 - External Video

Creating Functions: <https://youtu.be/qO4ZN5uZSVg>(external site)

I watched the video.

# Step 5 - Apply your knowledge

* I started with the starting template ,"Assigment06\_Starter.py", which I renamed Assigment06\_ToDoList.py"
* Created a new sub-folder called Assignment06 inside the \_PythonClass folder. Created a new project in PyCharm that uses the \_PythonClass\Assignment06 folder as its location . Added the starter file, "Assigment06\_Starter.py," to my project
* I started with starting template ,"Assigment06\_Starter.py", which I renamed Assigment06\_ToDoList.py". I updated the script header
* I read through the pseudocode to understand what the script was meant to do
* I started writing the functions for processing the data first followed by functions for the input/output and then the main script
* Once I was in the main script section, I started testing the code by running each menu choice (1, 2, 3, …)
* I learned from my failures:
  + I was mixing and matching my local and global variables and using the same names for global and local variables
  + in the main script, I listed the functions I wanted the program to perform, without correctly assigning the output of one function as the input of the next function
* I did add a menu option to show the current to do tasks – there was too much repetition for me in how in each iteration the tasks were printed out – and deleted that part from the cycle
* I also added numbering to the printout of the tasks
* I ran the code multiple times and verified that it worked both on PyCharm and the Command console – each however, saved the associated text file locally
* I also opened the text file and confirmed that the entries were being saved
* I learned that I had to first open a text file in a write or append mode and that if the text file was empty, I would get an error message
  + to circumvent the latter, I used the code ‘**if "," in** line:’ to check for the presence of a comma and to proceed with the remainder of the block only if text file was not empty. There are probably other ways to do this as well
* Ran the script in BOTH PyCharm and an OS command/shell window and capture images of it working on my computer (Figure 5).
* Verified that it worked, by locating the text file and opening it in a text editor. The file was in the same folder as my PyCharm script (Figure 5) and in a different folder when I ran script with Command console.
* The program listing as well as a sample of the running script are in the box below
* Reflections: with the Module 6 assignment I learned that I needed to think through the steps and process of what I expect the code to do before I start writing the code – if it’s too complicated then my logic is at fault. The Module 7 assignment was somewhat challenging at first and I started by writing down the names of the functions in each class and the variables on a big sheet of paper. I then tried to follow the expected flow of data to understand how the script was supposed to work. With this assignment I learned that you just have to dive in at the deep end and keep making mistakes – you will eventually learn from them.

Graphical user interface, application

Description automatically generated

***Figure 5.*** Screen captures demonstrating the script running in the Command console (a) and PyCharm (b) and the text written in the text file (c)

|  |  |
| --- | --- |
| **Assignment06 Script– To Do List with Functions** | **Running** |
| *# ---------------------------------------------------------------------------- # # Title: Assignment 06 # Description: Working with functions in a class, # When the program starts, load each "row" of data # in "ToDoList.txt" into a python Dictionary. # Add the each dictionary "row" to a python list "table" # ChangeLog (Who,When,What): # RRoot,1.1.2030,Created started script # Hedy Khalatbari,07.27.2021,Added code to complete assignment 6 # ----------------------------------------------------------------------------- #  # Data ------------------------------------------------------------------------ # # Declare variables and constants* strFileName = **"ToDoFile.txt"** *# name of the data file* objFile = **None** *# object that represents a file* dicRow = {} *# row of data; dictionary {Task,Priority}* lstTable = [] *# list of dicRow* strChoice = **""** *# Captures the user option selection* strTask = **""** *# Captures the user task data* strPriority = **""** *# Captures the user priority data* strStatus = **""** *# Captures the status of processing functions   # Processing --------------------------------------------------------------- #* **class** Processor:  *""" Performs Processing tasks """* @staticmethod  **def** read\_data\_from\_file(file\_name):  *""" Reads data from a file into a list of dictionary rows* **:param** *file\_name: (string) with name of file:* **:return***: (list) of dictionary rows  """* list\_of\_rows = [] *# local variable, list of dictionary rows* list\_of\_rows.clear() *# clear current data* file = open(file\_name, **"r"**)  **for** line **in** file:  **if "," in** line: *# added this condition as was getting an error when text file empty* task, priority = line.split(**","**)  row = {**"Task"**: task.strip(), **"Priority"**: priority.strip()}  list\_of\_rows.append(row)  file.close()  **return** list\_of\_rows   @staticmethod  **def** add\_data\_to\_list(task, priority, list\_of\_rows):  *""" Adds user input data (task, priority) to a list of dictionary rows* **:param** *task: (string) task we want to add:* **:param** *priority: (string) priority of task to add:* **:param** *list\_of\_rows: (list) of dictionary rows* **:return***:(list) of dictionary rows  """* list\_of\_rows = list(list\_of\_rows) *# declares list type* task = task.strip()  priority = priority.strip()  row = {**"Task"**:task.lower(), **"Priority"**:priority.lower()}  list\_of\_rows.append(row)  **return** list\_of\_rows   @staticmethod  **def** remove\_data\_from\_list(task, list\_of\_rows):  *""" Removes task from list of dictionaries* **:param** *task: (string) task we want to remove:* **:param** *list\_of\_rows: (list) of dictionary rows* **:return***: (list) of dictionary rows  """* task = task.strip() *# task to be removed* task = task.lower()  list\_of\_rows\_local = [] *# local list to write the new rows to* **for** row **in** list\_of\_rows:  task\_local = row[**"Task"**]  **if** task != task\_local: *# remove the user selected task* list\_of\_rows\_local.append(row)  list\_of\_rows.clear() *# clear contents* list\_of\_rows = list\_of\_rows\_local *# populate with current list* **return** list\_of\_rows   @staticmethod  **def** write\_data\_to\_file(file\_name, list\_of\_rows):  *""" Write data from a list of dictionary rows into a file* **:param** *file\_name: (string) with name of file:* **:param** *list\_of\_rows: (list) of dictionary rows:  """* file = open(file\_name, **"w"**)  **for** row **in** list\_of\_rows:  file.write(row[**"Task"**] + **", "** + row[**"Priority"**] + **"\n"**)  file.close()  *# no return necessary as no changes made to table list   # Presentation (Input/Output) -------------------------------------------- #* **class** IO:  *""" Performs Input and Output tasks """* @staticmethod  **def** print\_menu\_Tasks():  *""" Display a menu of choices to the user* **:return***: nothing  """* print(**'''  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File   4) Reload Data from File  5) Show Current To Do List  6) Exit Program  '''**)  print() *# Add an extra line for looks* @staticmethod  **def** input\_menu\_choice():  *""" Gets the menu choice from a user* **:return***: string  """* choice = str(input(**"Which option would you like to perform? [1 to 6] - "**)).strip()  print() *# Add an extra line for looks* **return** choice   @staticmethod  **def** print\_current\_Tasks\_in\_list(list\_of\_rows):  *""" Shows the current Tasks in the list of dictionaries rows* **:param** *list\_of\_rows: (list) of rows you want to display* **:return***: nothing  """* print(**"\*\* Current tasks in the To Do List are: \*\*"**)  counter = 1  **for** row **in** list\_of\_rows:  print(str(counter),**") "**,row[**"Task"**],**" ("**,row[**"Priority"**], **" )"**)  counter += 1  print(**"\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*"**)  print() *# Add an extra line for looks* @staticmethod  **def** input\_yes\_no\_choice(message):  *""" Gets a yes or no choice from the user* **:return***: string  """* choice = str(input(message))  choice = choice.strip()  choice = choice.lower()  **return** choice   @staticmethod  **def** input\_press\_to\_continue(optional\_message=**''**):  *""" Pause program and show a message before continuing* **:param** *optional\_message: An optional message you want to display* **:return***: nothing  """* print(optional\_message)  input(**'Press the [Enter] key to continue.'**)   @staticmethod  **def** input\_new\_task\_and\_priority():  *""" Asks user to input new task and priority* **:return***: (strings) task, priority  """* task = str(input(**"Enter task: "**))  priority = str(input(**"Enter priority: "**))  **return** task, priority   @staticmethod  **def** input\_task\_to\_remove():  *""" Asks user which task they would like to remove* **:return***: (string) task  """* task = str(input(**"Enter task to remove: "**))  **return** task   *# Main Body of Script ------------------------------------------------------ #* objFile = open(strFileName, **"a"**) *# I need to create a text file first if not present* objFile.close()  *# Step 1 - When the program starts, Load data from ToDoFile.txt and print To Do List* lstTable = Processor.read\_data\_from\_file(strFileName) *# read file data into list table* IO.print\_current\_Tasks\_in\_list(lstTable) *# Show current data in the list of dictionary rows  # Step 2 - Display a menu of choices to the user* **while** (**True**):  *# Step 3 Show menu and ask user to choose a menu option* IO.print\_menu\_Tasks() *# Shows menu* strChoice = IO.input\_menu\_choice() *# Get menu option   # Step 4 - Process user's menu choice* **if** strChoice == **'1'**: *# Add a new Task* strTask, strPriority = IO.input\_new\_task\_and\_priority()  lstTable = Processor.add\_data\_to\_list(strTask, strPriority, lstTable)  IO.input\_press\_to\_continue(strStatus)  **continue** *# to show the menu* **elif** strChoice == **'2'**: *# Remove an existing Task* strTask = IO.input\_task\_to\_remove()  lstTable = Processor.remove\_data\_from\_list(strTask, lstTable)  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"**:  Processor.write\_data\_to\_file(strFileName,lstTable)  IO.input\_press\_to\_continue(strStatus)  **else**:  IO.input\_press\_to\_continue(**"Save Cancelled!"**)  **continue** *# to show the menu* **elif** strChoice == **'4'**: *# Reload Data from File* print(**"Warning: Unsaved Data Will Be Lost!"**)  strChoice = IO.input\_yes\_no\_choice(**"Are you sure you want to reload data from file? (y/n) - "**)  **if** strChoice.lower() == **'y'**:  lstTable = Processor.read\_data\_from\_file(strFileName)  IO.input\_press\_to\_continue(strStatus)  **else**:  IO.input\_press\_to\_continue(**"File Reload Cancelled!"**)  **continue** *# to show the menu* **elif** strChoice == **'5'**: *# Show current data in the list of dictionary rows* IO.print\_current\_Tasks\_in\_list(lstTable) **elif** strChoice == **'6'**: *# Exit Program* print(**"Goodbye!"**)  **break** *# and Exit* **else**:  print(**"Please choose from menu options"**) | C: \\_PythonClass\Assignment06\venv\Scripts\python.exe C:/\_PythonClass/Assignment06/Assignment06\_ToDoList.py  \*\* Current tasks in the To Do List are: \*\*  \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 1  Enter task: do laundry  Enter priority: high  Press the [Enter] key to continue.  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 1  Enter task: wash dishes  Enter priority: medium  Press the [Enter] key to continue.  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 1  Enter task: mow  Enter priority: medium  Press the [Enter] key to continue.  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 5  \*\* Current tasks in the To Do List are: \*\*  1 ) do laundry ( high )  2 ) wash dishes ( medium )  3 ) mow ( medium )  \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 2  Enter task to remove: mow  Press the [Enter] key to continue.  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 5  \*\* Current tasks in the To Do List are: \*\*  1 ) do laundry ( high )  2 ) wash dishes ( medium )  \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 3  Save this data to file? (y/n) - y  Press the [Enter] key to continue.  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 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.5  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 5  \*\* Current tasks in the To Do List are: \*\*  1 ) do laundry ( high )  2 ) wash dishes ( medium )  \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*  Menu of Options:  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Show Current To Do List  6) Exit Program    Which option would you like to perform? [1 to 6] - 6  Goodbye!  Process finished with exit code 0 |

# Step 6 - Document your knowledge

I documented my learning throughout the steps in this document.

# Step 7 - Post your Files to GitHub

I created a repository called "IntroToProg-Python-Mod06" under my account.

I uploaded both of my files to the repository and committed the changes to save my work.

# Step 8 - Add a GitHub Web Page

I added a GitHub webpage to my repository and pasted supplied code to my site (Figure 6) .

My site link is: <https://seattle15.github.io/IntroToProg-Python-Mod06/>

# Step 9 - Post a Link to GitHub

I will post the link using the Canvas discussion board the week of Module 6.

<https://github.com/Seattle15/IntroToProg-Python-Mod06>

Graphical user interface, website

Description automatically generated

***Figure 6.*** Screen capture of my website for this assignment at  <https://seattle15.github.io/IntroToProg-Python-Mod06/>

# Step 10 - Submit your work

I submitted by Assignment06 word document and the Python script as a zipped file on Canvas – I included the URL to GitHub.

# Step 11 - Perform a Peer Review (Not Graded!)

# I will remember to do this the week of Module 6.

# Summary

In this module we learned about functions, local and global variables and creating GitHub webpages. We also reviewed reference and value types and classes and debugging were introduced. Module 5 was somewhat challenging for me, but the material in Module 6 was more straightforward. I look forward to Module 7.