Name: Amruth D

Date: 25thMay2023

Course: IT FDN 110 A

Assignment 06

The ‘To DO List’ Script – Part 2 (Python)

Introduction

In this assignment, we will understand, what is a Function, Parameters, Arguments, Difference between Parameters and Arguments and Return Values. For the assignment this week we had to redo the same exercise as last week in terms of writing a program that would allow us to edit and store a to do list

Function, Parameters, Arguments, Return Values

A **function** is a named block of reusable code that performs a specific task. It is a fundamental building block of any programming language and allows you to organize your code into modular and reusable units

A function is defined using the ‘**def’** keyword followed by the function name and parentheses. It can also take parameters (inputs) inside the parentheses.

Parameters are variables declared in the function definition that represent the inputs the function expects to receive. They act as placeholders for the values that will be passed into the function when it is called. Parameters allow functions to be flexible and work with different values each time they are called.

Arguments are the actual values passed into a function when it is called. They correspond to the parameters defined in the function's parameter list. Arguments provide the specific data on which the function will operate. When calling a function, you provide arguments that match the function's parameters in terms of position and number.

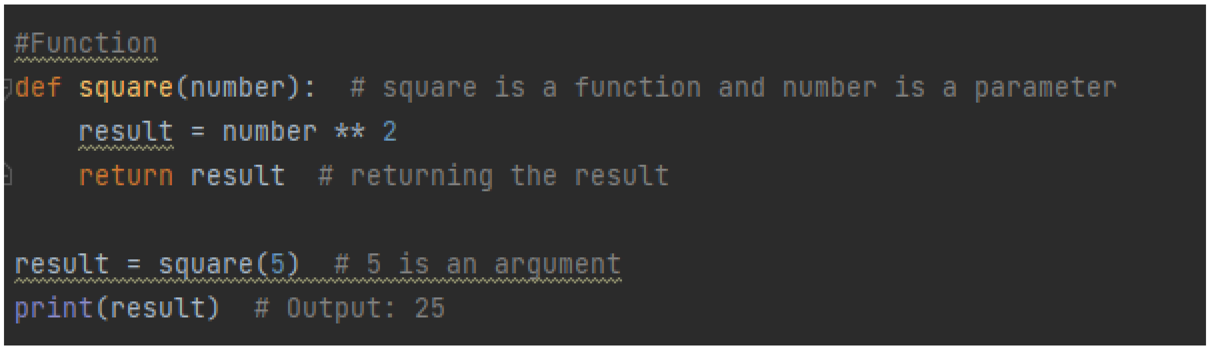
Difference between Parameters and Arguments

The terms "parameters" and "arguments" are often used interchangeably, but they have distinct meanings:

- Parameters: These are the variables declared in the function definition. They act as placeholders and specify the types and order of the values that a function expects to receive.
- Arguments: These are the actual values passed into a function when it is called. They are assigned to the corresponding parameters defined in the function. Arguments provide the specific data for the function to work with.

In simple terms, parameters are used in the function definition, while arguments are the values passed when calling the function.

Return values are the values that a function sends back as a result after performing its task. When a function encounters a **return** statement, it immediately exits the function and passes the specified value or expression back to the caller. The return value can be assigned to a variable, used in expressions, or passed as an argument to other functions. If a function does not have a **return** statement, it returns **None** by default.



```
#Function
def square(number):  # square is a function and number is a parameter
    result = number ** 2
    return result  # returning the result

result = square(5)  # 5 is an argument
print(result)  # Output: 25
```

Fig 1.0 Function Example

In the above example, the function is named **square** and takes one parameter **number**. The function calculates the square of the given number and returns the result.

To call a function, you simply write the function name followed by parentheses and any required arguments inside the parentheses.

And then, the **square** function with an argument of **5** and assigns the returned value to the **result** variable. The value **25** is then printed to the console.

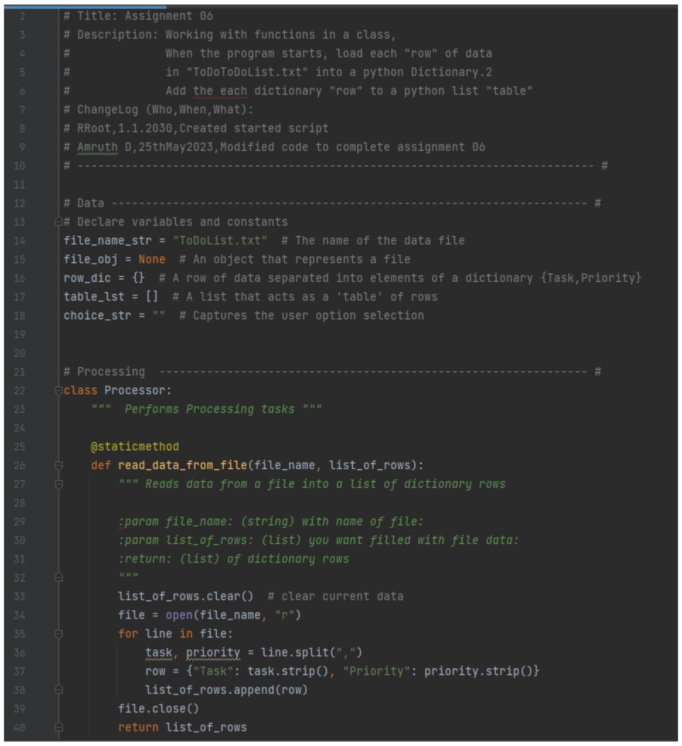
GitHub Webpage

GitHub not only serves as a code repository but also provides the capability to host simple webpages. These webpages have the extension ".md" and follow a specific style guide to achieve a desired appearance. They can serve as front pages for repositories hosted on GitHub.

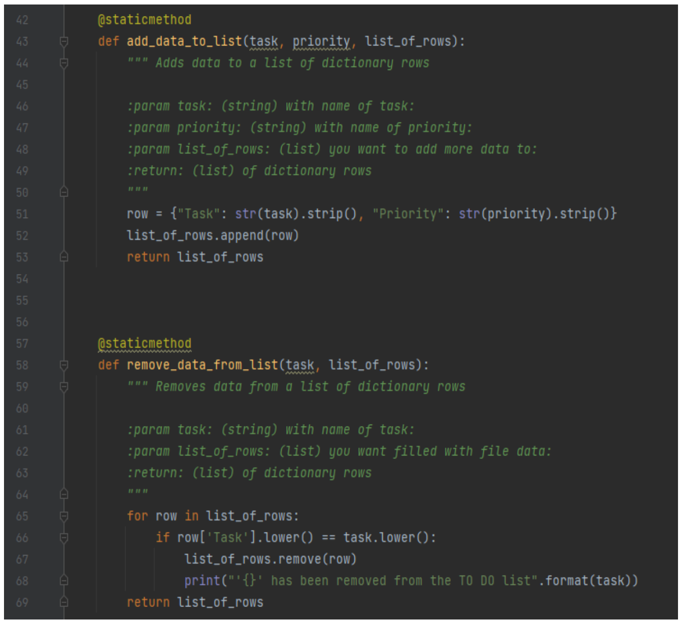
To access the GitHub webpage created for the assignment, please visit the following link: https://amruthd-92.github.io/IntroToProg-Python/

Writing The TO-DO Python Script

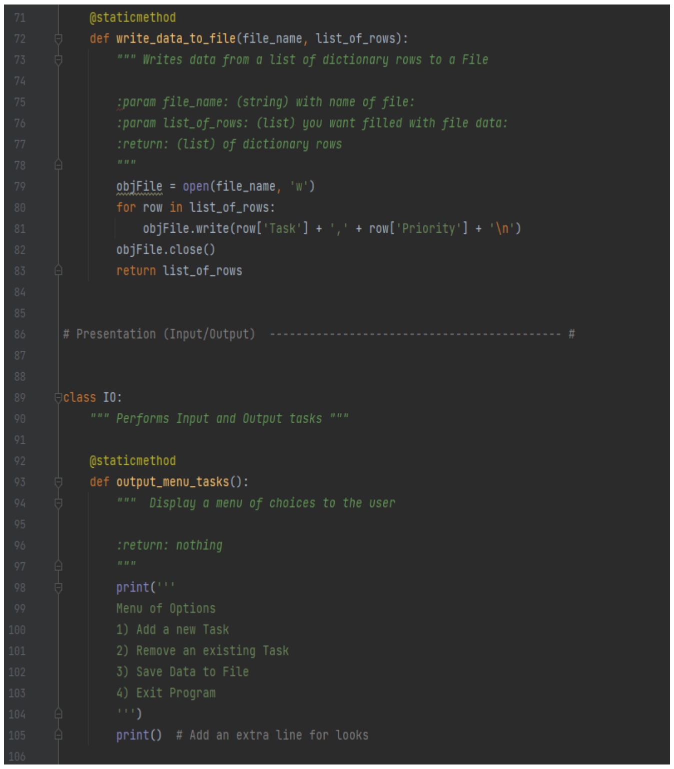
For this week's assignment, we were tasked with revisiting the same exercise as last week, which involved creating a program to manage and store a to-do list. While a significant portion of the code was provided to us, we had to make a few additions to certain functions to complete the task. Fortunately, I was able to leverage much of the code I had written last week to fill in the missing parts. I successfully executed the program both in PyCharm and an OS command, as depicted in the screenshots below.



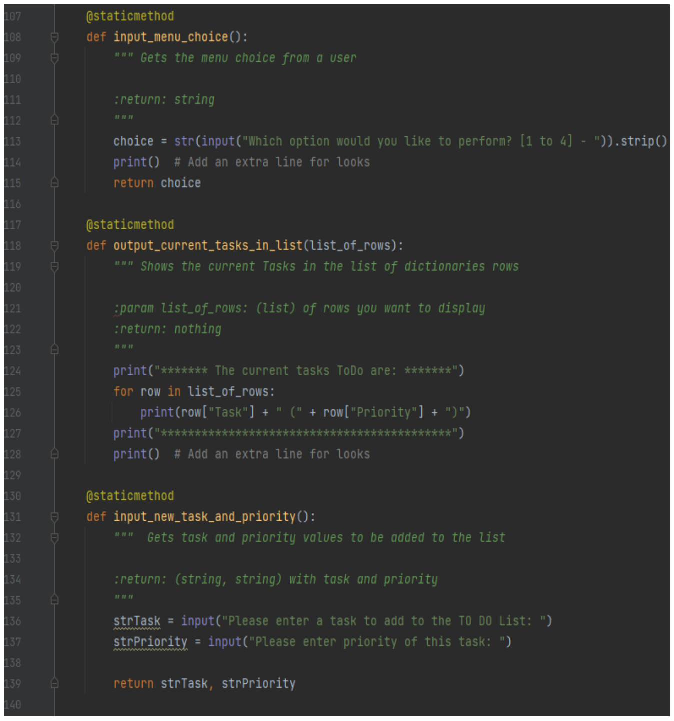
```python
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.2
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# Amruth D,25thMay2023,Modified code to complete assignment 06
# ----------------------------------------------------------------------------- #

# Data ----------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoList.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection


# Processing  ---------------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
        """ Reads data from a file into a list of dictionary rows

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        list_of_rows.clear()  # clear current data
        file = open(file_name, "r")
        for line in file:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows
```



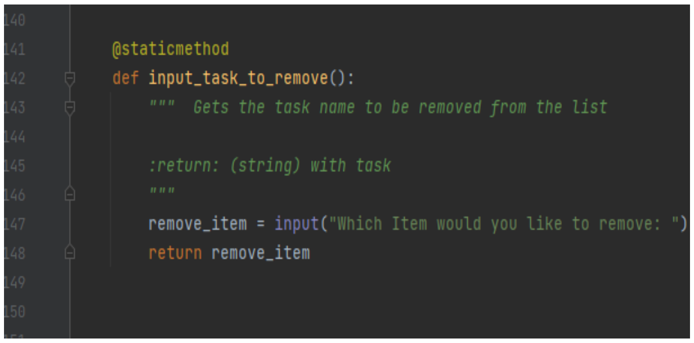
```python
    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds data to a list of dictionary rows

        :param task: (string) with name of task:
        :param priority: (string) with name of priority:
        :param list_of_rows: (list) you want to add more data to:
        :return: (list) of dictionary rows
        """
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        list_of_rows.append(row)
        return list_of_rows



    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes data from a list of dictionary rows

        :param task: (string) with name of task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        for row in list_of_rows:
            if row['Task'].lower() == task.lower():
                list_of_rows.remove(row)
                print("'{}' has been removed from the TO DO list".format(task))
        return list_of_rows
```



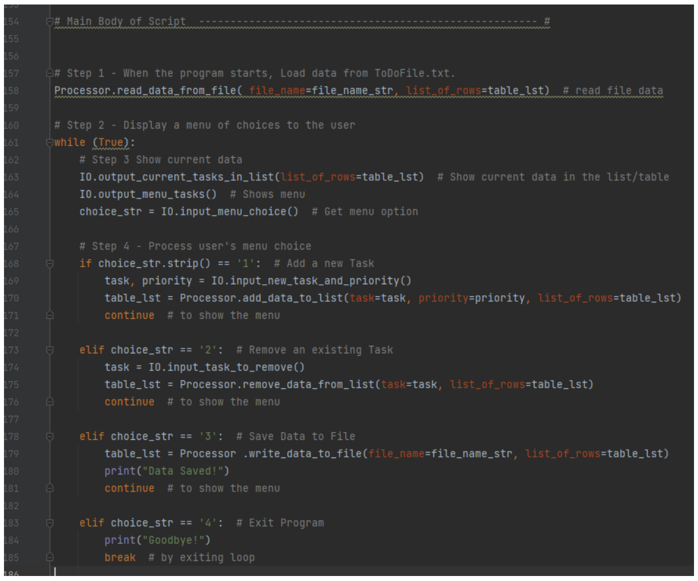
```python
    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Writes data from a list of dictionary rows to a File

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        objFile = open(file_name, 'w')
        for row in list_of_rows:
            objFile.write(row['Task'] + ',' + row['Priority'] + '\n')
        objFile.close()
        return list_of_rows


# Presentation (Input/Output)  -------------------------------------------- #


class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_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) Exit Program
        ''')
        print()  # Add an extra line for looks
```



```python
    @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 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice

    @staticmethod
    def output_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("******* The current tasks ToDo are: *******")
        for row in list_of_rows:
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("*******************************************")
        print()  # Add an extra line for looks

    @staticmethod
    def input_new_task_and_priority():
        """  Gets task and priority values to be added to the list

        :return: (string, string) with task and priority
        """
        strTask = input("Please enter a task to add to the TO DO List: ")
        strPriority = input("Please enter priority of this task: ")

        return strTask, strPriority
```



```python
    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        remove_item = input("Which Item would you like to remove: ")
        return remove_item
```



```python
# Main Body of Script  ------------------------------------------------------ #


# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file( file_name=file_name_str, list_of_rows=table_lst)  # read file data

# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved!")
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # by exiting loop
```

Fig 3.0 Completed Code

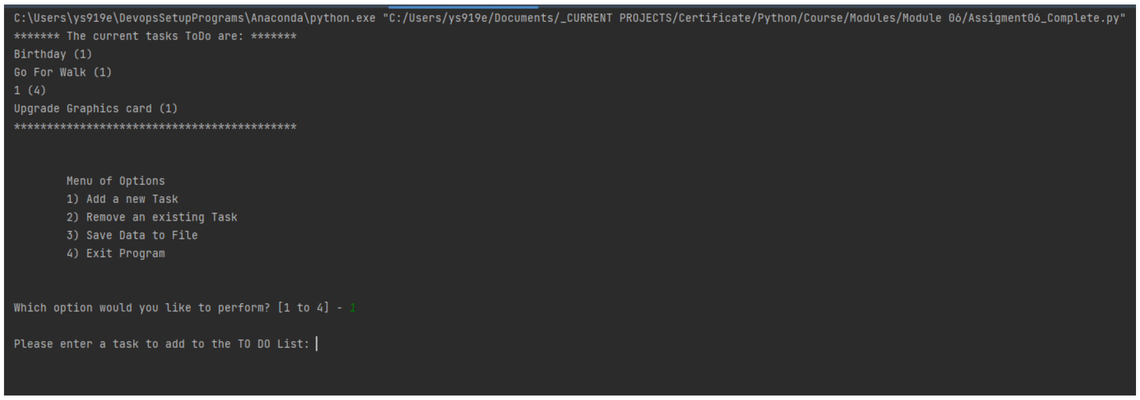
The code begins with a "Title Block" which typically refers to a section of comments at the beginning of the script that provides information about the script, such as its purpose, author, and date of creation.

- Declaring Variables - several variables are defined such as **file_name_str**, **file_obj**, **row_dic**, **table_lst**, and **choice_str** to store data and user choices.
- Next, a class named **Processor** is defined, which contains static methods for processing tasks related to reading, adding, removing, and writing data.
    - The **read_data_from_file** method reads data from a file and populates a list of dictionary rows.
    - The **add_data_to_list** method adds a new row of data to the list.
    - The **remove_data_from_list** method removes a specific row from the list.
    - The **write_data_to_file** method writes the list of dictionary rows back to a file.
- Another class named **IO** is defined to handle input/output tasks.
    - The **output_menu_tasks** method displays a menu of options to the user.
    - The **input_menu_choice** method prompts the user to enter their choice and returns it.

    - The **output_current_tasks_in_list** method displays the current tasks in the list of dictionary rows.
    - The **input_new_task_and_priority** method prompts the user to enter a new task and priority, returning both values.
    - The **input_task_to_remove** method prompts the user to enter the task they want to remove and returns it.
- Finally, the main body of the script is executed.
    - It starts by loading data from a file using the **read_data_from_file** method.
    - It then enters a loop where it displays the current tasks, shows the menu of options, and processes the user's choice accordingly.
        - If the choice is '1', it calls **input_new_task_and_priority** to get task and priority, then adds the data using **add_data_to_list**.
        - If the choice is '2', it calls **input_task_to_remove** to get the task to be removed, then removes it using **remove_data_from_list**.
        - If the choice is '3', it writes the data to the file using **write_data_to_file**.
        - If the choice is '4', it exits the loop and terminates the program.

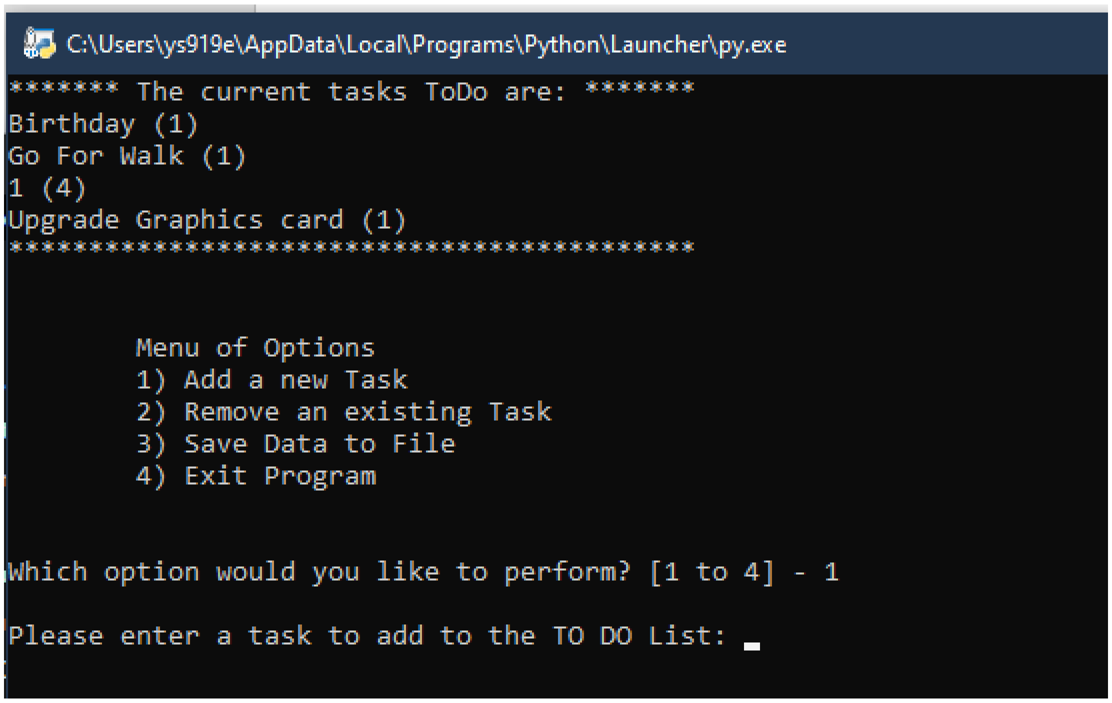
Executing the Python Script

Now that the script is written, the next step is to execute the script to check the correctness.



```
C:\Users\ys919e\DevopsSetupPrograms\Anaconda\python.exe "C:/Users/ys919e/Documents/_CURRENT PROJECTS/Certificate/Python/Course/Modules/Module 06/Assigment06_Complete.py"
******* The current tasks ToDo are: *******
Birthday (1)
Go For Walk (1)
1 (4)
Upgrade Graphics card (1)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 1

Please enter a task to add to the TO DO List: 
```

Fig 12.0 Output of script in PyCharm



```
C:\Users\ys919e\AppData\Local\Programs\Python\Launcher\py.exe
******* The current tasks ToDo are: *******
Birthday (1)
Go For Walk (1)
1 (4)
Upgrade Graphics card (1)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 1

Please enter a task to add to the TO DO List: _
```

Fig 10.0 Output of script in Command Prompt

Summary

In the sixth week, we learnt about Functions, Parameters, Arguments and Return values

Also, I used a starter script provided for the assignment and completed the code. Overall, it was little challenging to complete the assignment but it was an exciting week of learning.