Peter McGiffin

May 22, 2022

IT FDN 110A

Assignment 06

Introduction

Assignment 06 required code to be completed to perform several functions and complete a required task. The starter code declared variables and constants and provided several processing functions, (defined as processor class), a presentation input/output functions, and finally the main script to run the “To Do List” function.

Classes and Functions

This assignment used many different functions to perform tasks. The processor section used the class method to define processor functions. This was called in the presentation (input/output) section by using the code *class IO:* (shown in figure 1).

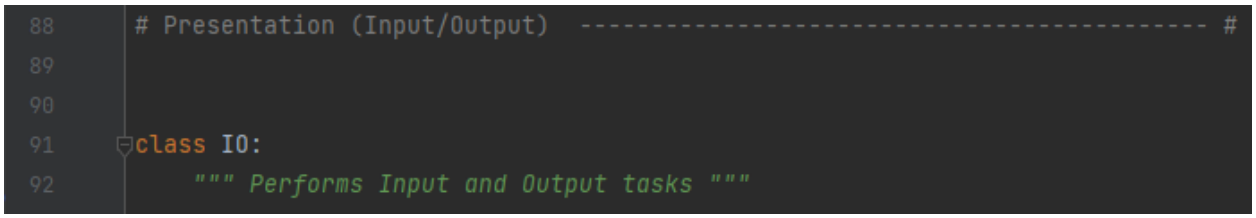

Figure 1. Showing the class IO function

Defining a class allows the programmer to then assign functions under the class. Functions that follow the class assignment are assigned using the *def function()* code. Functions are very useful as they allow a script to be assigned and then called multiple times at a later date. Functions allow the programmer to avoid repeating themselves. Values can be placed inside functions by using *def function(value):* code. This is seen multiple times in Assignment06, (one is shown in figure 2 below).

```python
@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
```

Figure 2. Assignment of output_current_tasks_in_list prints the list with “list of rows” as the value.

Lists and Dictionaries

Lists and Dictionaries are both used in assignment 6 to store data and recall it at a later point. The list assignment is done using square brackets “[ ]”. In assignment 06, the list is declared and called as ‘table_lst’, which is a list that acts as a table of rows. The rows are made up of python dictionary values, (assigned as row_dic). The dictionary rows are defined using “{ }”. Dictionaries assign a value and a key to that value. They are coded in python using {“value”: key}. In assignment 6, the value is the task while the key is the priority. Each dictionary value is stored on a row and then the subsequent value is stored on the next row down. This is achieved by the function *write_data_to_file* shown in figure 3 (below).

```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
    """
    # TODO: Add Code Here!
    file_obj = ("ToDoList.txt")
    open(file_obj, 'w')
    for row in table_lst:
        file_obj.write(dicRow["Task"] + ' ', + dicRow["Priority"] + '\n')
    file_obj.close()
    return list_of_rows
```

Figure 3. Takes input data and writes it to the file “ToDoList.txt”.

As shown in figure 3, each new dictionary input is assigned to a row. The ‘\n’ code creates a new line for the code.

Removing code from the list

In assignment 6, a function was created to remove data from the list. This function (*def input_task_to_remove():)* (shown in figure 4 below), uses a Tuple flag and the “del” method to delete data from the table_lst.

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

    :return: (string) with task
    """
    strKeyToRemove = input("Which task would you like to remove? ")  # TODO: Add Code Here!
    blnItemRemoved = False
    intRowNumber = 0
    for row in table_lst:
        task, priority = dict(row).values()
        if task == strKeyToRemove:
            del table_lst[intRowNumber]
            blnItemRemoved = True
        intRowNumber += 1
    if(blnItemRemoved == True):
        print("The task was removed. ")
    else:
        print("The task could not be found. ")
```

Figure 4. Function to remove code from the list.

Conclusion

Assignment 6 built on the skills we have learned previously by introducing functions and function methods to make code more detailed, and reduce repetitions. Functions are very useful and one of the most important concepts in understanding Python.