# Morgan Farmer

# August 17th, 2021

# Python Course

# Assignment06

<https://morganfarmer.github.io/ITFnd100-Mod06/>

Task & Priority Program with functions

Introductory

On assignment 06 we are introducing functions in a class. In each section I will be explaining what Mr.Roots intentions were to accomplish each task in the menu like the last assignment. In this assignment we us the common header required for each assignment.

# 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.  
# Add the each dictionary "row" to a python list "table"  
# ChangeLog (Who,When,What):  
# RRoot,1.1.2030,Created started script  
# RRoot,1.1.2030,Added code to complete assignment 5  
# <Your Name>,<Date>,Modified code to complete assignment 6  
# Dev: Morgan Farmer, 08/18/21

Declare variables and constants

In this section we are simply making a cheat sheet to name the variables and constants as a reference guide through the assignment. This was provided by Mr.Root.

strFileName = "ToDoFile.txt" # The name of the data file  
objFile = None # An object that represents a file  
dicRow = {} # A row of data separated into elements of a dictionary {Task,Priority}  
lstTable = [] # A list that acts as a 'table' of rows  
strChoice = "" # Captures the user option selection  
strTask = "" # Captures the user task data  
strPriority = "" # Captures the user priority data  
strStatus = "" # Captures the status of an processing functions

Processing

In the processing section we are defining each task that is being processed below. This serves as a place holder. Each menu option must have a defined task, reading data to a file, adding a row to a list and write a list to a file. To call it as a processor you must use the variable, static method. Down below is an example.

@staticmethod  
def WriteListDataToFile(file\_name, list\_of\_rows):  
 # *TODO: Add Code Here!* for dicRow in list\_of\_rows:  
 objFile = open(file\_name, 'w')  
 objFile.write(dicRow['Task'] + ',' + dicRow['Priority'] + '\n')  
 objFile.close()  
  
 return list\_of\_rows, 'Success'

Presentation

Presentation is similar. Presentation aka IO input output is a process that performs an input output task. Mr.Root provided examples of how the menu is supposed to look along with, current tasks in a list, input yes or no choices, input press to continue. Here is an example below.

@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("\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*")  
 for row in list\_of\_rows:  
 print(row["Task"] + " (" + row["Priority"] + ")")  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 print() # Add an extra line for looks

Main body of script

Note that in all options we are using if statements.

Option 1, I declared the variable string task and priority. Next created an input statement asking if the priority is high or low. In pseudo code I called the dictionary row and appending the list because the code is like the addressed processing code above. In my code it looks like this.

if strChoice.strip() == '1': # Add a new Task  
 # *TODO: Add Code Here* strTask = str(input('Enter a Task'))  
 strPriority = str(input('Enter a Priority: [high | low] -')).strip()  
 print()  
 #dicRow = {'Task': strTask, 'Priority': strPriority}  
 #lstTable.append(dicRow)  
 FileProcessor.AddRowToList(strTask, strPriority, lstTable)  
  
 IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu

Option 2, Remove an existing task. First, I get the task that the user wants to delete. Next create a Boolean for a loop. Then last create a counter for a dictionary row in the loop. Using the counter, you can identify which task the user is trying to pick to delete. Last, I use the delete list table variable.

elif strChoice == '2': # Remove an existing Task  
 # *TODO: Add Code Here* strKeyToRemove = input('Which task to be removed?')  
 blnItemRemoved = False  
 intRowNumber = 0  
 while(intRowNumber < len(lstTable)):  
 if(strKeyToRemove == str(list(dict(lstTable[intRowNumber]).values())[0])):  
 del lstTable[intRowNumber]  
 blnItemRemoved = True  
 intRowNumber += 1  
  
 IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu

Option 3, saving the data to a file. I use the processor that was created in the processing area above and put the actual code in pseudo code. For now, though explaining the steps, first I open the text file and declare the file name variable and write the text to the file. Next, I open the dictionary and list the close the file.

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!* #objFile = open(strFileName, 'w')  
 #for dicRow in lstTable:  
 #objFile.write(dicRow['Task'] + ',' + dicRow['Priority'] + '\n')  
 #objFile.close()  
 FileProcessor.WriteListDataToFile(strFileName, lstTable)  
 IO.input\_press\_to\_continue(strStatus)  
 else:  
 IO.input\_press\_to\_continue("Save Cancelled!")  
 continue # to show the menu

Option 4, Reload the data from file. First creating the if statement. Using the file processor created from above that reads data from a file.

lstTable.clear()  
 FileProcessor.read\_data\_from\_file(strFileName,lstTable)  
  
 IO.input\_press\_to\_continue(strStatus)  
else:  
 input('File data was not reloaded.')  
 IO.input\_press\_to\_continue("File Reload Cancelled!")  
continue # to show the menu

Option 5 was completed by printing goodbye to the user and then break to breakout of the program and end it.

elif strChoice == '5': # Exit Program  
 print("Goodbye!")  
 break # and Exit

Conclusion

In conclusion, the program runs the options. Learning that I can make variables for what I want them to do is helpful and seems like it would save a lot of time for programs in the future. Only thing that isn’t making much sense is when I create the variable and call it it doesn’t always work.