Trace Kershaw

Assignment 6

IT FDN 100

**Working with Functions**

The purpose of this assignment was to learn how work with functions in python that can make tasks simpler. The main focus of this assignment was to redo the ToDoList code we did in Assignment 5 and replace much of the codes with functions. There were 3 main aspects of the code we replaced with functions: Data processing tasks, processing tasks, and Input/Output tasks. The purpose is to understand how functions provides immense flexibility in coding, where it allows you to use overall functional placeholders and pipe in variables to execute the functions. This allows one a lot of flexibility in coding and also allows you to separate different types of tasks such as processing and presentation tasks.

**Using Dr. Root’s Template**

* For this assignment I used Dr. Root’s Assignment06\_start template which listed out the code from Assignment 5.
* In various places it suggested we replace the current code with functions, and then tested whether the coding still worked.
* The first thing I did was modify the header for the 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  
  # RRoot,1.1.2030,Fixed bug by clearing the list before it was refilled  
  # Tkershaw,11/10/18,Modified code to complete assignment 6  
  # ------------------------------------------------------------------------ #
* This code creates a To Do List for users, and uses a menu of options that allow users to add, show, delete, and save to do items. The core menu is below:

Menu of Options

1) Show current data

2) Add a new item.

3) Remove an existing item.

4) Save Data to File

5) Reload Data from File

6) Exit Program

* There were three main classes of processing that I added functions to: FileProcessing (reading or writing data to a file, IO (or Input/Output, and processing (manipulating data and variables in the actual program. The startup file already had the FileProcessing and IO classes of functions (for which I added functions. I created the new Processing Class to add Processing functions.
* Looking through the file there were 5 functions to add to the program
  + WriteListDataToFile to save data to a text file, which I created in the DataProcessing Class
  + AskforNewTask to ask users to add a new task and priority to their To Do List, which I created in the IO Class
  + AppendNewTask which appended new tasks and priorities to the dictionary list and data from the text file, which I created in the Processing Class
  + SearchTables which searched tables for a task that the user entered in order to delete it from the To Do List, which I created in the Processing Class
* So first I created the function WriteListDataToFile, which would save any entered data into the text file.
* I defined the function and listed out the purpose of the function and appropriate parameters.
* @staticmethod  
  def WriteListDataToFile(file\_name**,** list\_of\_rows):  
   *"""  
   Desc - Writes current data to a file into a list of dictionary rows* ***:param*** *file\_name: (string) with name of file:* ***:param*** *list\_of\_rows: (list) you want saved to file data:* ***:return****: (list) of dictionary rows  
   """* objFile = open(file\_name**,** "w")  
   for list\_of\_rows in lstTable: # Write each row of data to the file  
   objFile.write(list\_of\_rows["Task"] + "," + list\_of\_rows["Priority"] + "\n")  
   objFile.close()
* Then in the body of the code based on the condition of individuals asking to save the data, I called up the function, substituting the file name (strFileName) and list of rows (dicRow).

# Step 3.4 - Save tasks to the ToDoFile.txt file  
elif(strChoice == '4'):  
  
 #Step 3.4.a - Show the current items in the table  
 IO.ShowCurrentItemsInList(lstTable) # Show current data in the list/table  
  
 #Step 3.4.b - Ask if user if they want save that data  
 if("y" == str(input("Save this data to file? (y/n) - ")).strip().lower()): # Double-check with user  
  
 FileProcessor.WriteListDataToFile(strFileName**,** dicRow) # write file data  
  
 input("Data saved to file! Press the [Enter] key to return to menu.")  
 else: # Let the user know the data was not saved  
 input("New data was NOT Saved, but previous data still exists! Press the [Enter] key to return to menu.")  
 continue # to show the menu

* Next, I created a function AskforNewTask that allows people who choose to add a new task, to add a task and priority.
* @staticmethod  
  def AskforNewTask(task**,**priority):  
   *""" Asks user for new task* ***:param*** *task: user asked for task* ***:param*** *priority: user asked for priority* ***:return****: nothing  
   """* task = str(input("What is the task? - ")).strip() # Get task from user  
   priority = str(input("What is the priority? [high|low] - ")).strip() # Get priority from user  
   print() # Add an extra line for looks  
   return task**,** priority
* For this function, I am having users enter in a task and a priority and then returning a tuple of task and priority
* Next, in the body of the code I call up that function when people responded 2 (that the wanted to add a new item).
* # Step 3.2 - Add a new item to the list/Table  
  elif(strChoice.strip() == '2'):  
   vTask**,** vPriority= IO.AskforNewTask(strTask**,**strPriority)  
   print ("Your new task is %s with a priority of %s" %(vTask**,** vPriority))
* I had to unpack the tuple here so I could then append the data to the data table.
* So next, I created a function that first creates a new dictionary row, and then appends that row to the existing data table.

@staticmethod  
def AppendNewtask(Task**,** Priority):  
 *"""  
 Desc - creates a new dictionary row and appends task and priority* ***:param*** *Task: task entered from user* ***:param*** *Priority: priority entered from user* ***:return****: dictionary row appended to table  
 """* row = {"Task": Task**,** "Priority": Priority} # Create a new dictionary row  
 lstTable.append(row) # Add the new row to the list/table

* Then I call that function in the body of the text as part of the process that asks users for a new task.

# Step 3.2.b Add item to the List/Table  
Processing.AppendNewtask(vTask**,** vPriority)  
IO.ShowCurrentItemsInList(lstTable) # Show current data in the list/table  
continue # to show the menu

* So this step appends the new task and then uses the function to show the current items in the newly appended table.
* Finally, I created a function SearchTables for those who entered 3 (to remove an item, that searches the table for the value that users enter, and then removes the item if it finds the value in the table.

@staticmethod  
def SearchTables(removeitem**,** flag):  
 *"""  
 Desc - searches table for item that users wants to remove and removes the item* ***:param*** *removeitem: item entered from user that they want to remove* ***:param*** *flag: flag variable created to stop iteration loop* ***:return****: none  
 """* intRowNumber = **0** # Create a counter to identify the current dictionary row in the loop  
 while (intRowNumber < len(lstTable)):  
 if (removeitem == str(list(dict(lstTable[intRowNumber]).values())[**0**])): # Search current row column 0  
 del lstTable[intRowNumber] # Delete the row if a match is found  
 flag = True # Set the flag so the loop stops  
 intRowNumber += **1** # Increase counter to get next row

* So in the body of the code, I then call up that function after the user enters in which Task they like to remove (strKeyToRemove)

# Step 3.3 - Remove a new item to the list/Table  
elif(strChoice == '3'):  
  
 # Step 3.3.a - Ask user for item and prepare searching while loop  
 strKeyToRemove = input("Which TASK would you like removed? - ") # get task user wants deleted  
 blnItemRemoved = False # Create a boolean Flag for loop  
  
 # Step 3.3.b - Search though the table or rows for a match to the user's input  
 Processing.SearchTables(strKeyToRemove**,**blnItemRemoved)  
  
 # Step 3.3.c - Update user on the status of the search  
 if(blnItemRemoved == True):  
 print("The task was removed.")  
 else:  
 print("I'm sorry, but I could not find that task.")  
 print() # Add an extra line for looks

* Finally, I posted my code and assignment on GitHub.