Using Functions and Organization

**Introduction and Script**

The assignment is mainly about using functions to reorganize script from a previous assignment. It’s organized into three areas being: Processing, Presentation, and Main Script. Having the code grouped like that makes it easier to find what is needed, along with reuse functions when applicable.

**Part 1**

When the script opens it automatically opens up the file to see what is already inside, captures the data in memory to use in the script, then opens up the greeting along with a menu for the user to start using. Both processes are done through functions to keep the main script organized.

# -- Main Script -- #

Processor.read\_data\_from\_file("C:\\_PythonClass\ToDoFile.txt", lstTable)

IO.greetings()

while True:

# Selection Menu

Processor.print\_current\_tasks\_in\_list(lstTable)

IO.print\_menu\_tasks()

strChoice = IO.input\_menu\_choice("self")

**Part 2**

Once that’s up the user can add or take away data selecting options in the menu. The processes to do both are also kept at functions in the processing area. They use arguments made by the user to operate.

@staticmethod

def add\_data\_to\_list(task, priority, lstTable):

dicRow = {"Task": task, "Priority": priority}

lstTable.append(dicRow)

print("\n=====", task, "has been added with priority", priority, "=====")

@staticmethod

def remove\_data\_from\_list(removal):

lstTable.pop(removal)

print("\n===== Row", removal, "has been taken out of the table =====")

**Part 3**

There is an option to save the current list of data to the file. It opens up the file and writes inside the text file. In the main script it is kept short due to being a function.

@staticmethod

def save\_data\_to\_file():

objFile = open("C:\\_PythonClass\ToDoFile.txt", "w")

for row in lstTable:

objFile.write(row["Task"] + "," + row["Priority"] + "\n")

objFile.close()

print("===== The current table has been saved =====\n")

# Save Data

elif strChoice.strip() == '3':

strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ")

if strChoice.lower() == 'y':

Processor.save\_data\_to\_file()

else:

IO.input\_press\_to\_continue("Save Cancelled!")

**Part 4**

The last function the script performs is reloading the data currently in the file, not in memory. It basically ignores what is currently in the script and writes again what is inside the file.

@staticmethod

def reload\_file(file\_name, lstTable):

objFile = open(file\_name)

for row in objFile:

txtdata = row.split(",")

dicRow = {"Task": txtdata[0], "Priority": txtdata[1].strip()}

lstTable.append(dicRow)

objFile.close()

return lstTable

# Reload Data

elif strChoice.strip() == '4':

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':

Processor.reload\_file("C:\\_PythonClass\ToDoFile.txt",lstTable)

else:

IO.input\_press\_to\_continue("File Reload Canceled!")

**Conclusion**

The exercise shows how functions may be used multiple times, like in the case of the “yes or no” functions, along with using their properties of being away from where they are getting used to organize the script. By keeping them in sections debugging is made easier and reading it is a bit simpler.

Code in Process:











