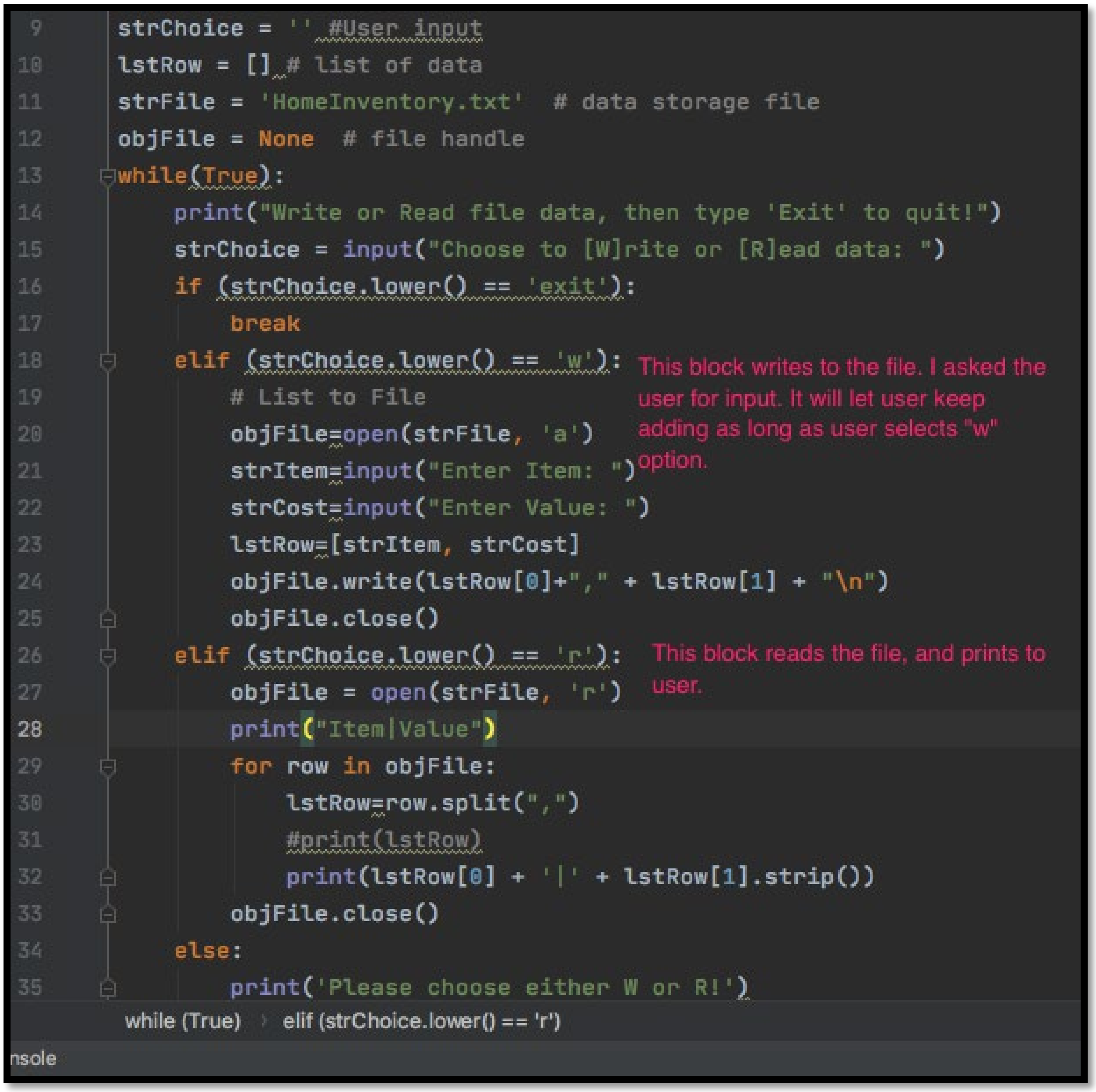
Roslyn Melookaran
5/9/2020
Foundations of Programming, Python
Assignment 5
https://github.com/roslynm/IntroToProg-Python

Introduction

In this module we learn the differences between lists and dictionaries. We also learn about how to use dictionaries, and how to access information from them via keys. Through the labs and in the homework assignment, we begin to read files into lists and dictionaries. In addition, this module gives us practice on how to organize our scripts using the "Separation of Concerns" method, as well experience modifying and adding to existing scripts. We also gain exposure to the website, GitHub, which allows programmers a space to share and modify scripts.

Loading Data

In previous modules, we have learned how to write data into a file. In this module, we learn how to read data from a file. It is important to load the data from your hardrive into memory first, then use it from memory. In the example shown in **Figure 1**, below, you can see the script asks user for input, and writes to the file. If the user would like to see the inventory they have entered, they can choose to read the file. Note, the .strip() function can be used to strip any extra character off of the beginning and end of a string. In this case, we are stripping off spaces AND the invisible carriage returns.



```
strChoice = '' #User input
lstRow = [] # list of data
strFile = 'HomeInventory.txt'  # data storage file
objFile = None  # file handle
while(True):
    print("Write or Read file data, then type 'Exit' to quit!")
    strChoice = input("Choose to [W]rite or [R]ead data: ")
    if (strChoice.lower() == 'exit'):
        break
    elif (strChoice.lower() == 'w'):
        # List to File
        objFile=open(strFile, 'a')
        strItem=input("Enter Item: ")
        strCost=input("Enter Value: ")
        lstRow=[strItem, strCost]
        objFile.write(lstRow[0]+"," + lstRow[1] + "\n")
        objFile.close()
    elif (strChoice.lower() == 'r'):
        objFile = open(strFile, 'r')
        print("Item|Value")
        for row in objFile:
            lstRow=row.split(",")
            #print(lstRow)
            print(lstRow[0] + '|' + lstRow[1].strip())
        objFile.close()
    else:
        print('Please choose either W or R!')
```

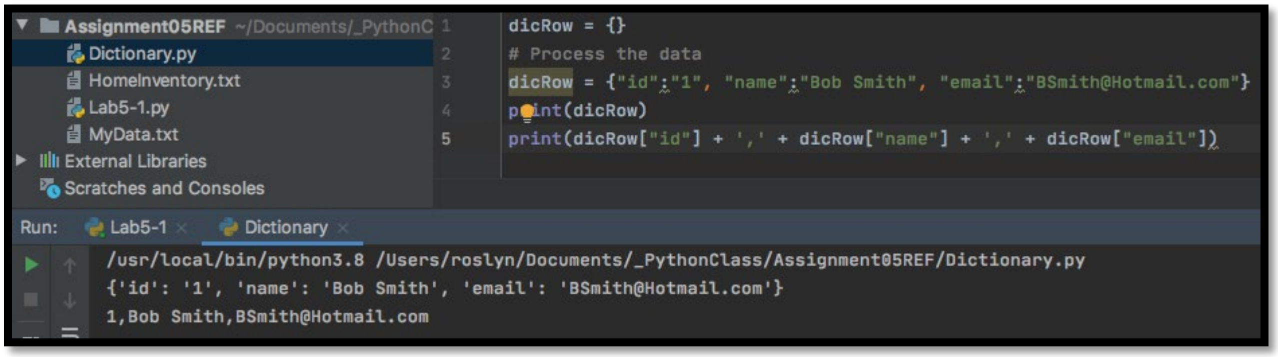
This block writes to the file. I asked the user for input. It will let user keep adding as long as user selects "w" option.

This block reads the file, and prints to user.

Figure 1. Example of reading from a file.

Dictionaries

When working with lists, touples and strings, we can identify components of these collections by using indexes (numeric subscript). You can access components of a dictionary similarly, except we use a key in the form of a character subscript. You use {} brackets to identify a dictionary. See **Figure 2** for an example of using dictionaries. Here, “ID”, “Name” and “Email” are considered the keys.

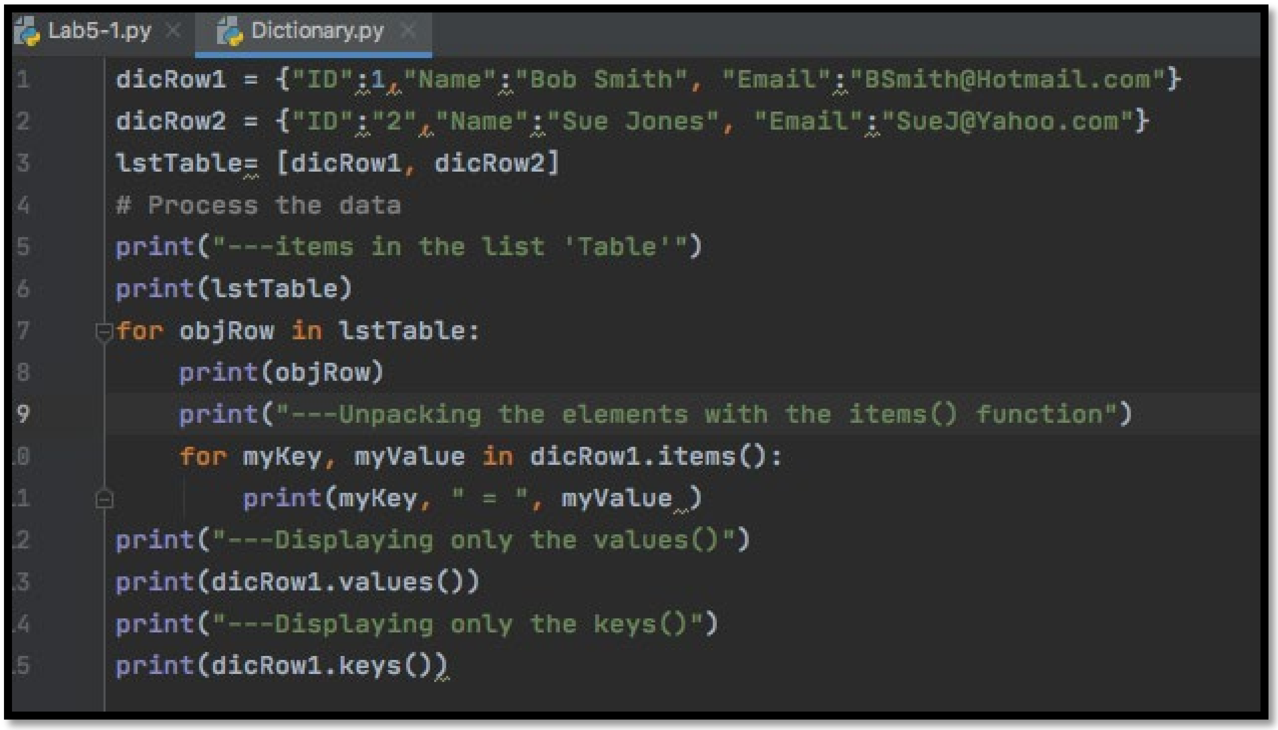


```
dicRow = {}
# Process the data
dicRow = {"id":"1", "name":"Bob Smith", "email":"BSmith@Hotmail.com"}
print(dicRow)
print(dicRow["id"] + ',' + dicRow["name"] + ',' + dicRow["email"])
```

```
/usr/local/bin/python3.8 /Users/roslyn/Documents/_PythonClass/Assignment05REF/Dictionary.py
{'id': '1', 'name': 'Bob Smith', 'email': 'BSmith@Hotmail.com'}
1,Bob Smith,BSmith@Hotmail.com
```

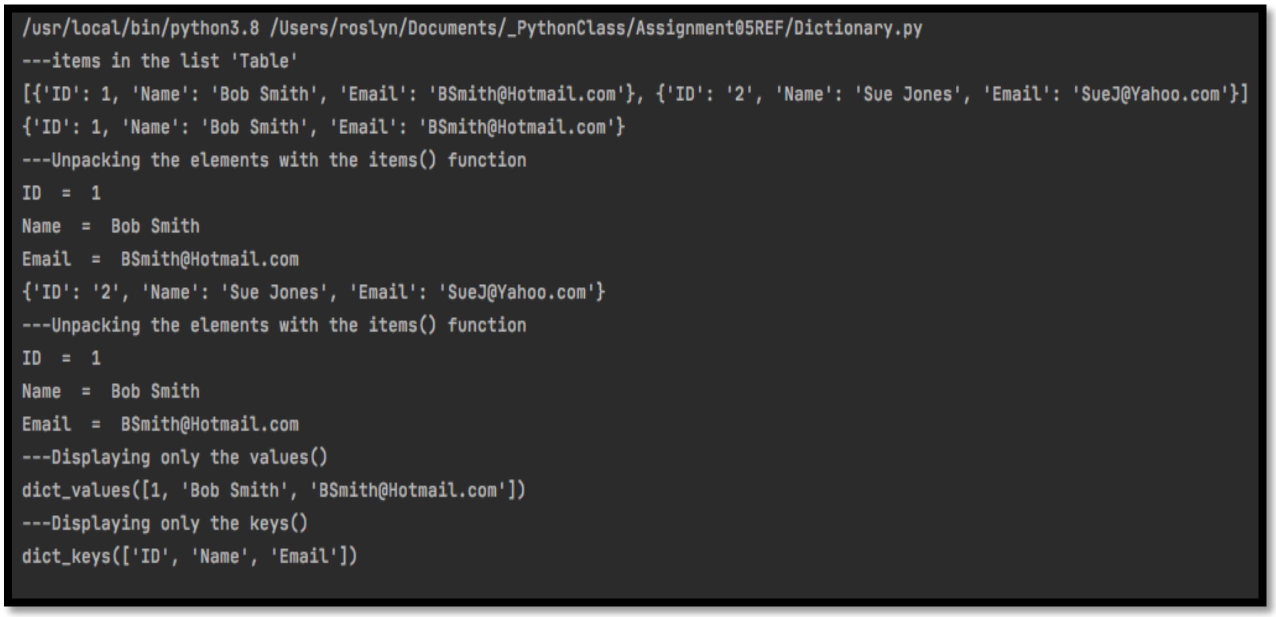
Figure 2. Example of using a dictionary.

In the **Figure 3a/b** example, below, we take dictionaries one step further and make a list of dictionaries. Here you can see that the way we print dictionaries is slightly different than what we were using for lists and touples. The .items(), .key(), and .value() functions can help unpack dictionaries for printing and other uses.



```
dicRow1 = {"ID":1,"Name":"Bob Smith", "Email":"BSmith@Hotmail.com"}
dicRow2 = {"ID":"2","Name":"Sue Jones", "Email":"SueJ@Yahoo.com"}
lstTable= [dicRow1, dicRow2]
# Process the data
print("---items in the list 'Table'")
print(lstTable)
for objRow in lstTable:
    print(objRow)
    print("---Unpacking the elements with the items() function")
    for myKey, myValue in dicRow1.items():
        print(myKey, " = ", myValue )
print("---Displaying only the values()")
print(dicRow1.values())
print("---Displaying only the keys()")
print(dicRow1.keys())
```

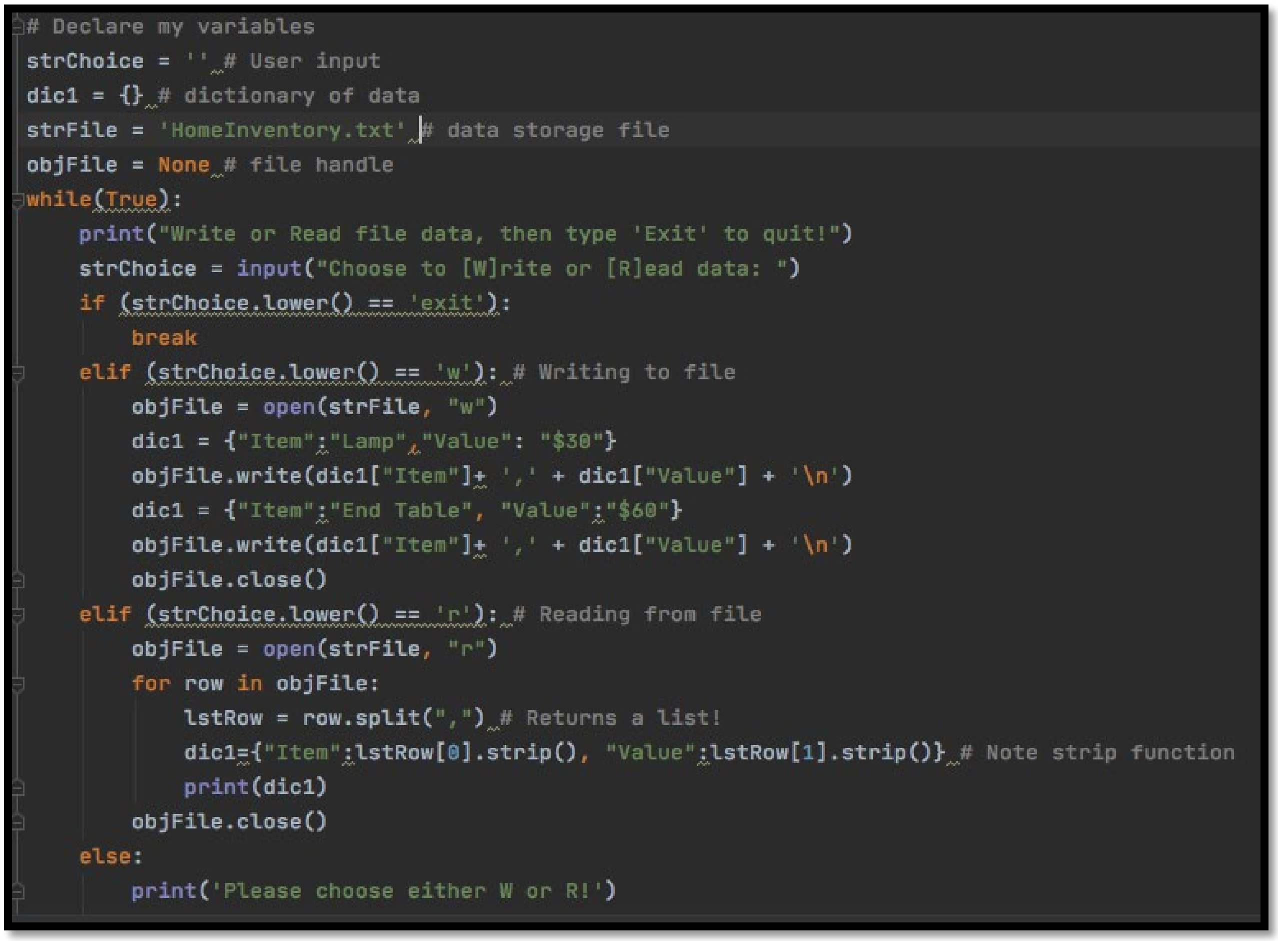
Figure 3a. Example script showing list of dictionaries



```
/usr/local/bin/python3.8 /Users/roslyn/Documents/_PythonClass/Assignment05REF/Dictionary.py
---items in the list 'Table'
[{'ID': 1, 'Name': 'Bob Smith', 'Email': 'BSmith@Hotmail.com'}, {'ID': '2', 'Name': 'Sue Jones', 'Email': 'SueJ@Yahoo.com'}]
{'ID': 1, 'Name': 'Bob Smith', 'Email': 'BSmith@Hotmail.com'}
---Unpacking the elements with the items() function
ID  =  1
Name  =  Bob Smith
Email  =  BSmith@Hotmail.com
{'ID': '2', 'Name': 'Sue Jones', 'Email': 'SueJ@Yahoo.com'}
---Unpacking the elements with the items() function
ID  =  1
Name  =  Bob Smith
Email  =  BSmith@Hotmail.com
---Displaying only the values()
dict_values([1, 'Bob Smith', 'BSmith@Hotmail.com'])
---Displaying only the keys()
dict_keys(['ID', 'Name', 'Email'])
```

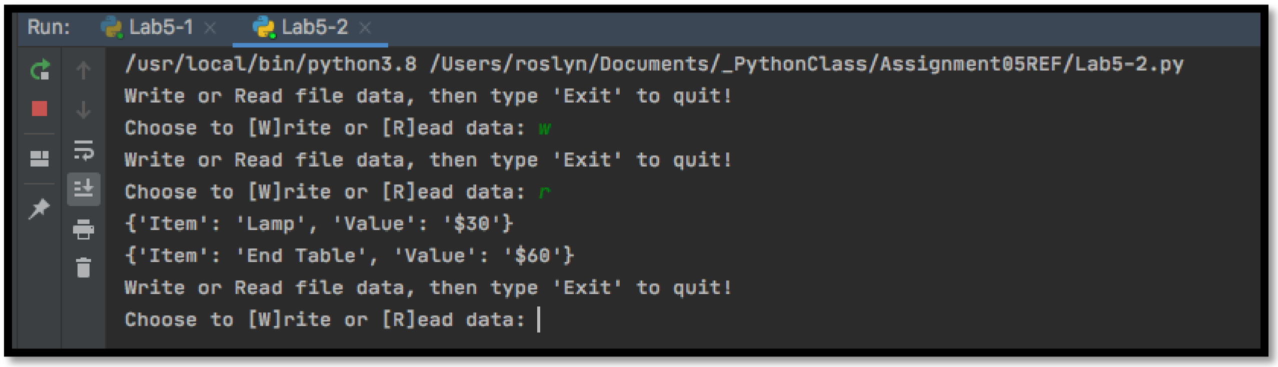
Figure 3b. Script running in PyCharm

Similar to what we practiced in lab 5-1, we also practiced writing a dictionary to file, as well as reading from a file to a dictionary. You can write a dictionary to file, one key at a time. In order to read from a file, you must first read the data in the file into a list, then convert the list to a dictionary. Note the use of the strip function when reading the file. See **Figure 4a/b** for Lab 5-2.



```
# Declare my variables
strChoice = '' # User input
dic1 = {} # dictionary of data
strFile = 'HomeInventory.txt' # data storage file
objFile = None # file handle
while(True):
    print("Write or Read file data, then type 'Exit' to quit!")
    strChoice = input("Choose to [W]rite or [R]ead data: ")
    if (strChoice.lower() == 'exit'):
        break
    elif (strChoice.lower() == 'w'): # Writing to file
        objFile = open(strFile, "w")
        dic1 = {"Item":"Lamp","Value": "$30"}
        objFile.write(dic1["Item"]+ ',' + dic1["Value"] + '\n')
        dic1 = {"Item":"End Table", "Value":"$60"}
        objFile.write(dic1["Item"]+ ',' + dic1["Value"] + '\n')
        objFile.close()
    elif (strChoice.lower() == 'r'): # Reading from file
        objFile = open(strFile, "r")
        for row in objFile:
            lstRow = row.split(",") # Returns a list!
            dic1={"Item":lstRow[0].strip(), "Value":lstRow[1].strip()} # Note strip function
            print(dic1)
        objFile.close()
    else:
        print('Please choose either W or R!')
```

Figure 4a. Example script reading data from a text file



```
Run: Lab5-1 × Lab5-2 ×
/usr/local/bin/python3.8 /Users/roslyn/Documents/_PythonClass/Assignment05REF/Lab5-2.py
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: w
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: r
{'Item': 'Lamp', 'Value': '$30'}
{'Item': 'End Table', 'Value': '$60'}
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: 
```

Figure 4b. Example of script running in PyCharm

Separation of Concerns

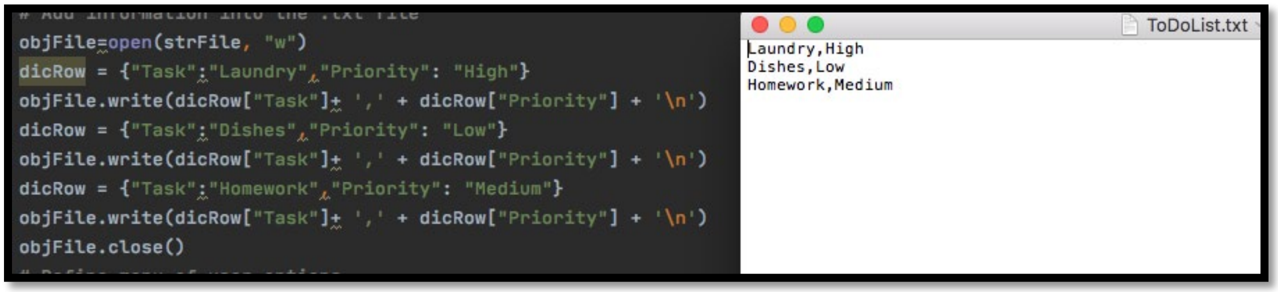
As scripts become longer, and more complex, it is important to organize the script into a concise way. One way of doing this is separating the script into distinct sections. Each of which, addressing a certain concern. This can help in separating out the code to different programmers, and handling errors. A common way to do this is to seperate the code into a “Data”, “Process”, and “Present” sections.

Functions

Functions are a set of statements, grouped together under a given name. These functions are defined, and then called upon in the main program. Using functions in your program can make your program easier to read, and enables you to use the same function, in multiple scripts. We will dive more into functions in Module 6.

Assignment

To begin the assignment script, I first loaded some tasks to populate the text file. You can see this in Figure 5 below.



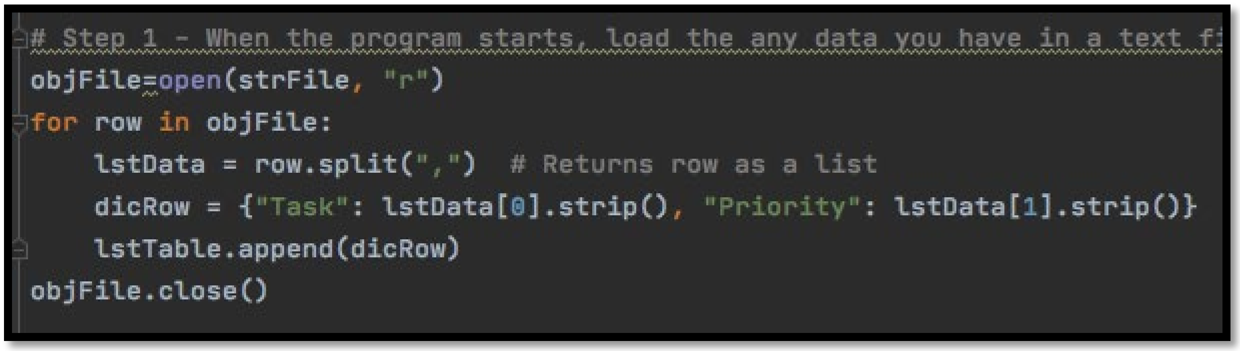
```
# Add information into the .txt file
objFile=open(strFile, "w")
dicRow = {"Task":"Laundry","Priority": "High"}
objFile.write(dicRow["Task"]+ ',' + dicRow["Priority"] + '\n')
dicRow = {"Task":"Dishes","Priority": "Low"}
objFile.write(dicRow["Task"]+ ',' + dicRow["Priority"] + '\n')
dicRow = {"Task":"Homework","Priority": "Medium"}
objFile.write(dicRow["Task"]+ ',' + dicRow["Priority"] + '\n')
objFile.close()
```

ToDoList.txt

```
Laundry,High
Dishes,Low
Homework,Medium
```

Figure 5. Initializing the task list

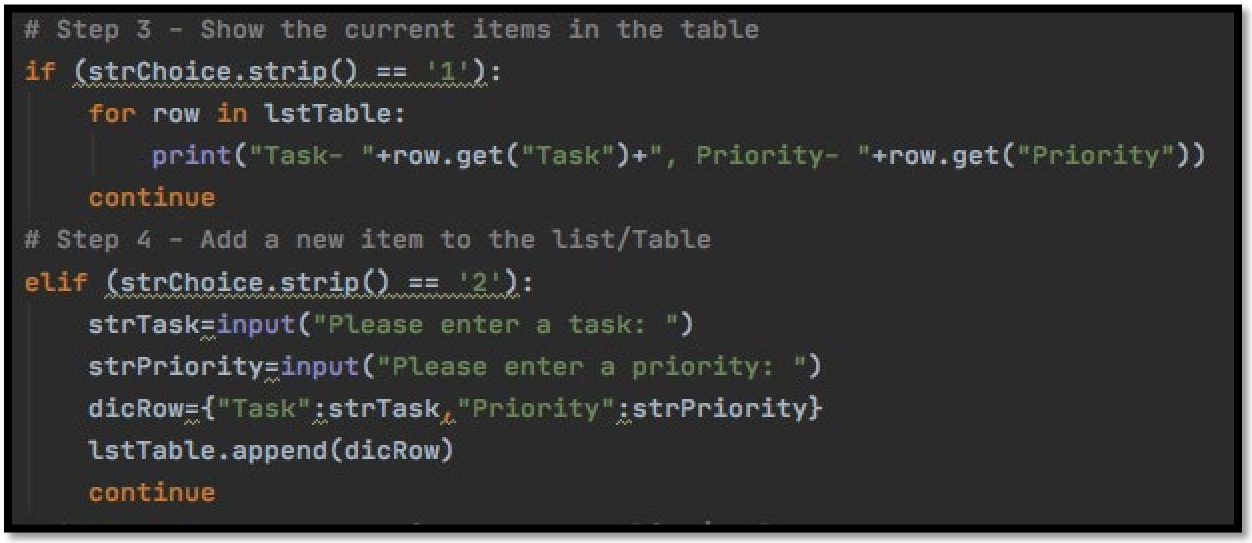
After I loaded initial values into the text file, I then load them from the text file into my LstTable variable using a For loop. In Figure 6, you can see that I first loaded each row of the text file into a list. I then created a dictionary from that list, then created a list out of those dictionary rows. The use of the strip() function takes away any extra spaces or carriage returns at the beginning and end of each row.



```
# Step 1 - When the program starts, load the any data you have in a text fi
objFile=open(strFile, "r")
for row in objFile:
    lstData = row.split(",")  # Returns row as a list
    dicRow = {"Task": lstData[0].strip(), "Priority": lstData[1].strip()}
    lstTable.append(dicRow)
objFile.close()
```

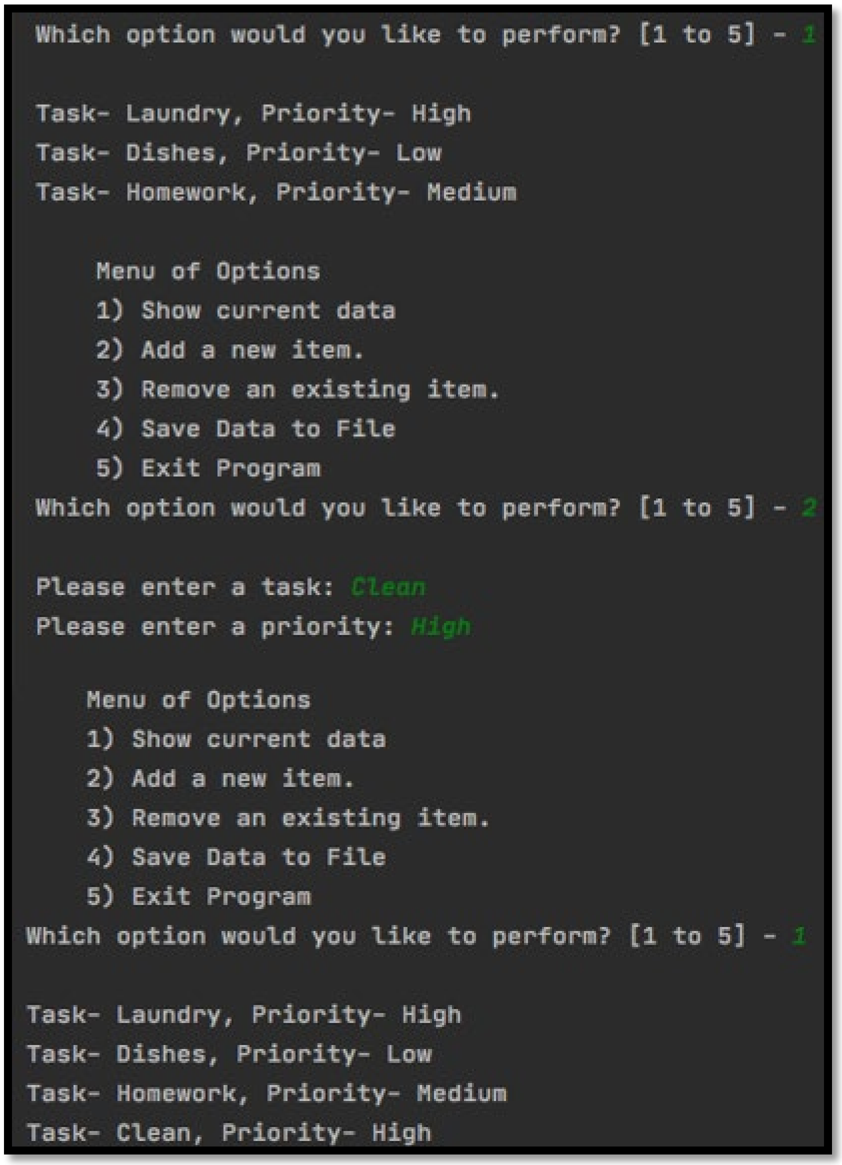
Figure 6. Reading data from file

Now we begin the Input/Output portion of the script. If the user selects 1, the data from the text file is displayed. If the user would like to add to the file, they select 2. In **Figure 7a/b**, you can see an example of adding to the list of task.



```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    for row in lstTable:
        print("Task- "+row.get("Task")+", Priority- "+row.get("Priority"))
    continue
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    strTask=input("Please enter a task: ")
    strPriority=input("Please enter a priority: ")
    dicRow={"Task":strTask,"Priority":strPriority}
    lstTable.append(dicRow)
    continue
```

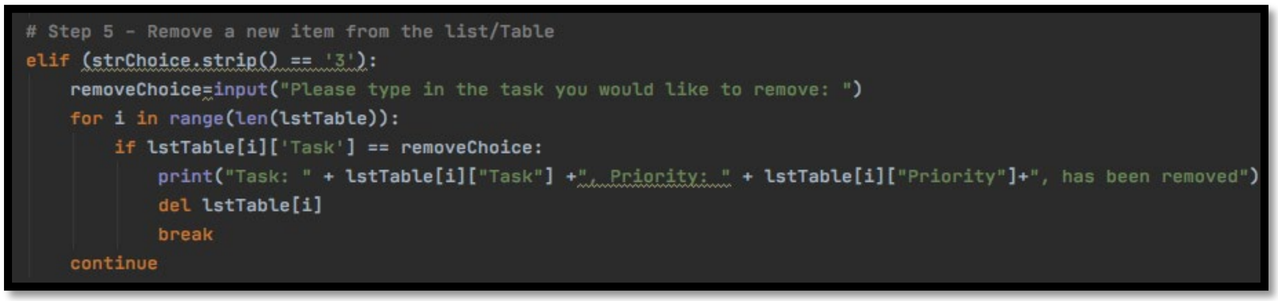
Figure 7a. Script of option 1 or 2 selection. Showing list or adding to a list.



```
Which option would you like to perform? [1 to 5] - 1

Task- Laundry, Priority- High
Task- Dishes, Priority- Low
Task- Homework, Priority- Medium

     Menu of Options
     1) Show current data
     2) Add a new item.
     3) Remove an existing item.
     4) Save Data to File
     5) Exit Program
Which option would you like to perform? [1 to 5] - 2

Please enter a task: Clean
Please enter a priority: High

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 1

Task- Laundry, Priority- High
Task- Dishes, Priority- Low
Task- Homework, Priority- Medium
Task- Clean, Priority- High
```

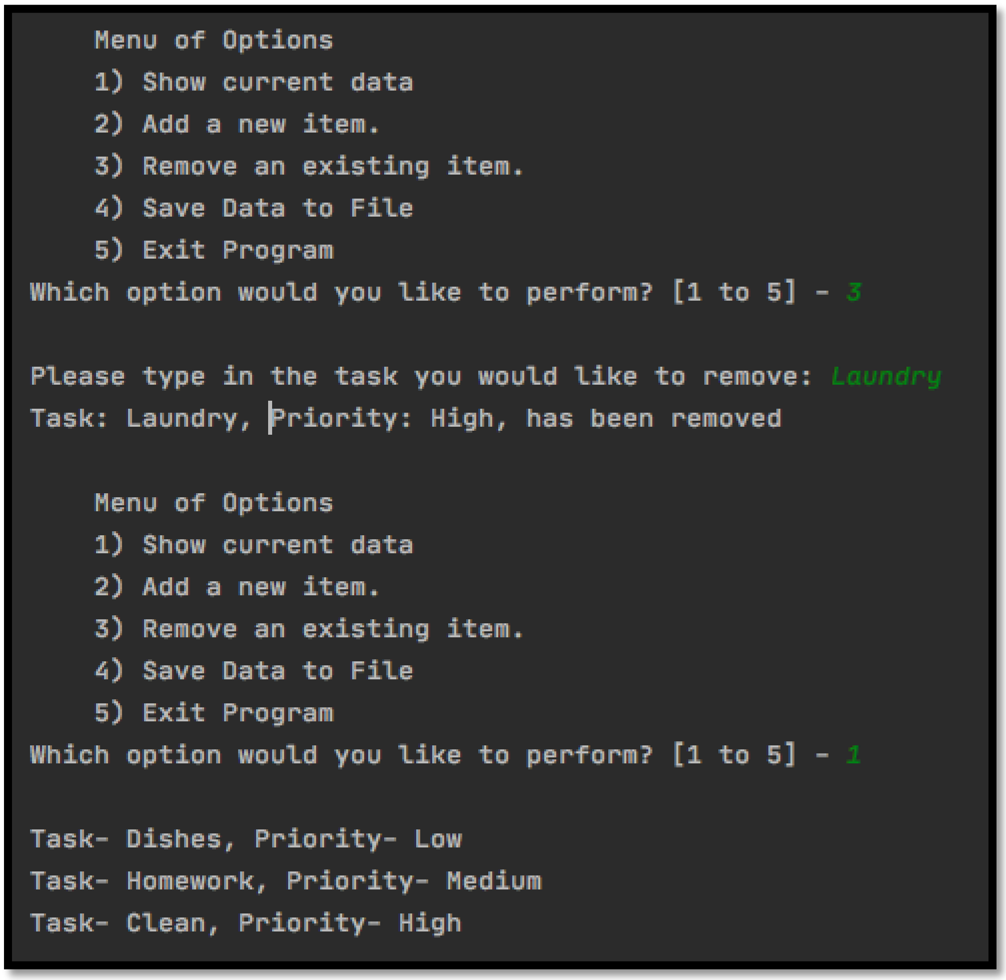
Figure 7b. Script of option 1 or 2 selection running in PyCharm

If the user chooses option 3, they can remove an item from the list. In **Figure 8a/b**, you can see that we are removing "Laundry" from the task list.



```python
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    removeChoice=input("Please type in the task you would like to remove: ")
    for i in range(len(lstTable)):
        if lstTable[i]['Task'] == removeChoice:
            print("Task: " + lstTable[i]["Task"] +", Priority: " + lstTable[i]["Priority"]+", has been removed")
            del lstTable[i]
            break
    continue
```

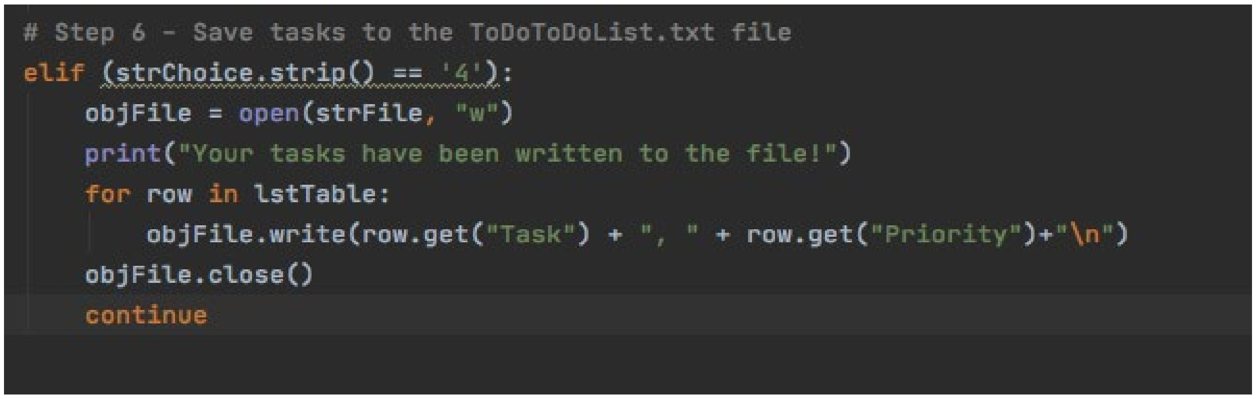
Figure 8a. Script of option 3 selection, removing task from list.



```
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 3

Please type in the task you would like to remove: Laundry
Task: Laundry, Priority: High, has been removed

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 1

Task- Dishes, Priority- Low
Task- Homework, Priority- Medium
Task- Clean, Priority- High
```

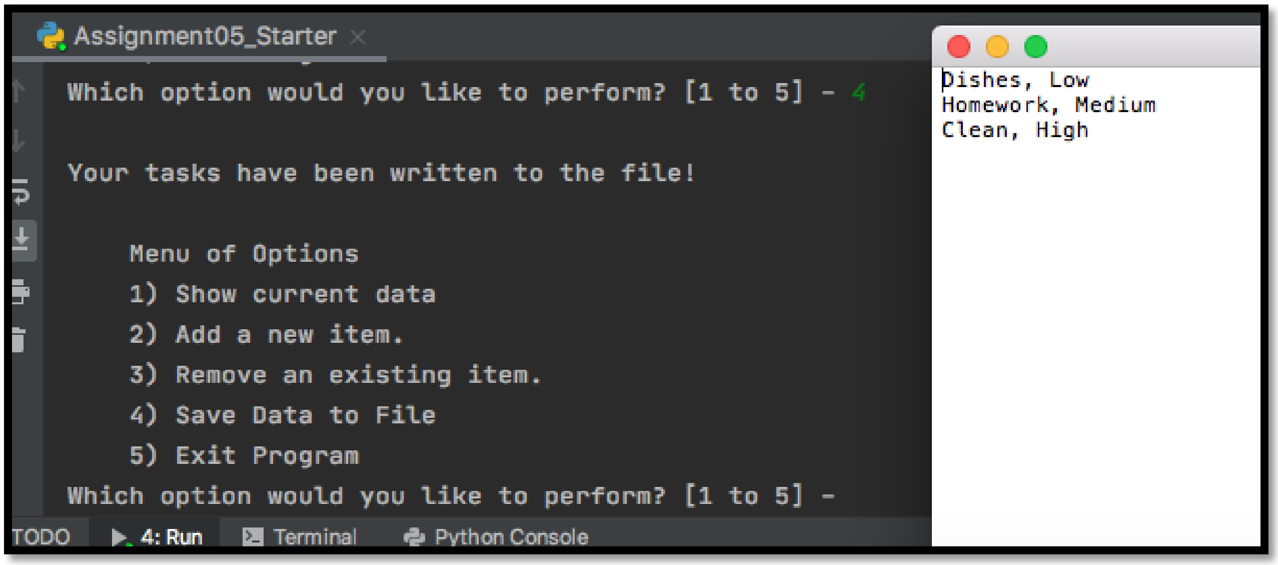
Figure 8b. Script of option 3 selection running in PyCharm.

Once we have added and removed the tasks from our To Do list, we want to write this finalized list back to the text file. When the user selects option 4, the list of tasks is written back to the file. You can see the task list written back to the file in **Figure 9a/b**.



```
# Step 6 - Save tasks to the ToDoToDoList.txt file
elif (strChoice.strip() == '4'):
    objFile = open(strFile, "w")
    print("Your tasks have been written to the file!")
    for row in lstTable:
        objFile.write(row.get("Task") + ", " + row.get("Priority")+"\n")
    objFile.close()
    continue
```

Figure 9a. Script of option 4 selection, writing task list back to file.

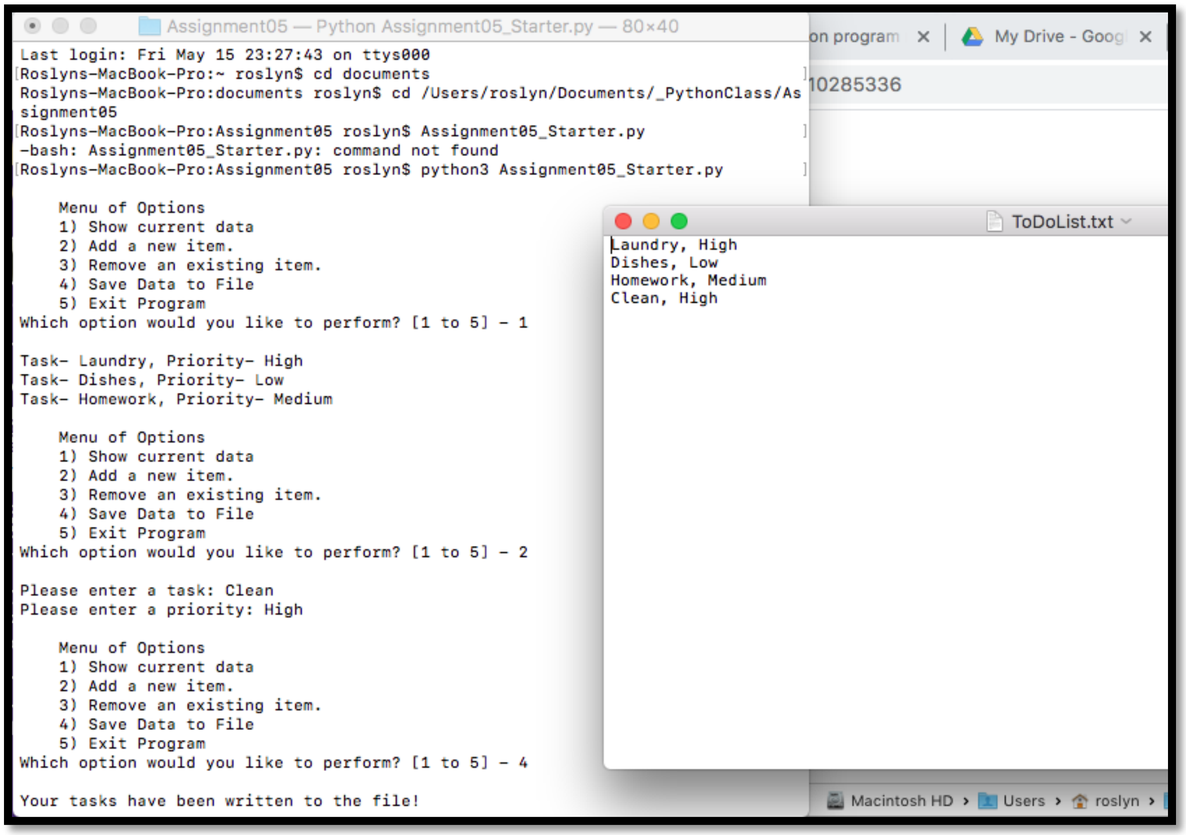


```
Which option would you like to perform? [1 to 5] - 4

Your tasks have been written to the file!

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] -
```

```
Dishes, Low
Homework, Medium
Clean, High
```

Figure 9b. Script of option 4 selection running in PyCharm.

It is important to check and make sure that scripts are functional in your PyCharm window AND through your terminal. In **Figure 10**, you can see an example of the script running from the terminal.



```
Last login: Fri May 15 23:27:43 on ttys000
Roslyns-MacBook-Pro:~ roslyn$ cd documents
Roslyns-MacBook-Pro:documents roslyn$ cd /Users/roslyn/Documents/_PythonClass/As
signment05
Roslyns-MacBook-Pro:Assignment05 roslyn$ Assignment05_Starter.py
-bash: Assignment05_Starter.py: command not found
Roslyns-MacBook-Pro:Assignment05 roslyn$ python3 Assignment05_Starter.py

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 1

Task- Laundry, Priority- High
Task- Dishes, Priority- Low
Task- Homework, Priority- Medium

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 2

Please enter a task: Clean
Please enter a priority: High

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
Which option would you like to perform? [1 to 5] - 4

Your tasks have been written to the file!
```

```
Laundry, High
Dishes, Low
Homework, Medium
Clean, High
```

Figure 10. Script running in terminal

Conclusion

In this module we furthered our base of python skills learning how to use dictionaries and how to read data from files. Through the labs and assignments, we were able to get great practice writing and reading from files, and using while and if loops to complete certain tasks based on user input. Exposure to adding to and modifying existing scripts will also help us to be more adaptable programmers moving forward.