Miguel Quintana
May 17, 2023
IT FDN 100 A
Assignment 05

Introduction:

Lists and dictionaries are very similar ways to store data; the biggest difference is how we can find the collection of objects that this data holds. On the list, each object has an index that represents its place on the list. Instead, dictionaries don't use indexes but keys and values; each value has a key, and that is how we can find and organize data in a dictionary.

Most of the data is processed in computer memory, but it's backed up on files. The way we can read the data from a list file is by declaring first our variables; in this case, it will be an empty list, which we can call lstRow = [ ] because it will represent a row in a table. The second variable needed is the one with the file that we will use; let's call it strData = "MyData.txt". Finally, we will create a variable that will process the data, let's call it objData = none.

Once we declare the variables, we can start to process the data by opening the file and specifying that we want to read the data. After that, we can loop through the list and output the data in the desired format. And would look like this:

```
objFile = open(strFile, "r")
for row in objFile:
        lstRow = row.split(",")
        print(lstRow[0] + '|' + lstRow[1] + '|' + lstRow[2].strip())
 objFile.close()
```

Listing 1

If we want to do the same thing with data but put it in a dictionary, the process is similar but not quite the same. If we declare the same variables as on the list but add an empty dictionary variable, the syntax will look like this: dicRow = { }. In dictionaries, instead of square brackets, use braces.

If we are extracting data from a list file, we need to iterate through the list and append the data to the dictionary with the desired format. The syntax would be like this:

```
objFile = open(strFile, "r")
for row in objFile:
  lstRow = row.split(",")
  dicRow = {"id":lstRow[0],"name":lstRow[1],"email":lstRow[2].strip()}
  lstTable.append(dicRow)
objFile.close()
```

Listing 2

It is not explicitly mentioned that the split() function will extract the data that comes after each "," on the list. The strip() function will eliminate any empty space or unwanted things that can interfere with the extraction of the data.

Another big topic this week is how to improve our scripts. From here came the concept of separation of concerns or "SoC". We can say that separation of concerns is the standard way to structure our programs. We can assume that most of the programs will be separated into three big sections: data, processing, and presentation. Different languages can call this process differently, but the principle is the same.

Another way to organize your program is with functions. Functions can hold a set of programming statements; they have to be declared before they are called, and they are a very nice way to reuse codes in different programs.

Another tool that can make our scripts easier to use is script templates. Script templates are so common in the industry that many IDLEs have a built-in function to save templates so you can reuse them or send them to coworkers or yourself in the future.

A big part of making a program is also making mistakes, and sometimes finding what is not working in our code can be a bit tricky. The error messages are not very user friendly and can be confusing at times. One way to fix this problem is with Try-Except. When an error occurs, it is very possible that it will occur again in the future, Try-Except is a way that we can document these mistakes by leaving an error message that is more user-friendly and can make much more sense than the old school error message.

The best way to save time on the creation of a program is through the reusability of the codes; none of this would be useful if we couldn’t save it or access it anywhere we wanted. One of the most famous and well known tools in the industry is github, On this website, we can upload our code, and make it public, private, or share it with our company. In the same way, we can download code from ourselves, or other people, creating a big sharable community. The way Github works is with a software control called Git and the website Github to store the codes.

Video Review:

Reading from a text file: https://youtu.be/m0o0CkYsDzI: on this video we learned how to manipulate data with dictionaries. A big part of what we learned in this video is described in depth in this week's lecture and paper.

Assignment 5:

In this assignment, we are going to create a program with a menu where we can show current data, add new data items to the file, remove existing items, save data to a file, and exit the program.

The first step is to import a pre-made script with the instructions into the program. Once we do that, we will start to declare the variables that we will use in this program:

```
objFile = None  # An object that represents a file
strData = ""  # A row of text data from the file
dicRow = {}  # A row of data separated into elements of a dictionary
{Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
strMenu = ""  # A menu of user options
strChoice = ""  # A Capture the user option selection
strFile = "ToDo.txt"
```
Figure 1.

After that, we will extract the data from the existing file to process it in memory. We will open the file to read, and after that, we will iterate through the list and put the data in a dictionary. After that, we will put the dictionary into a list:

```
objFile = open(strFile, "r")
for row in objFile:
   lstRow = row.split(",")
   dicRow = {"Item": lstRow[0], "Priority": lstRow[1].strip()}
   lstTable.append(dicRow)
objFile.close()
```

Figure 2.

After we complete this task, we will display a menu of choices to the user, so while this is true, we will keep the program open. This part of the program will receive input from the user with the desired option:

```
while (True):
   print("""
   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
   """)
   strChoice = str(input("Which option would you like to perform? [1 to 5] -
"))
   print()  # adding a new line for looks
```

Figure 3.

After that, we will start to complete each option with the proper code. The first option will show the current data in the table. An If statement will be in place, and it will iterate through the file, showing the current data in the desired format:

```
if (strChoice.strip() == '1'):
   for dicRow in lstTable:
       print(dicRow["Item"] + ", " + dicRow["Priority"])
   continue
```

Figure 4.

Option 2 is about adding a new item to the list table, To do that, we will receive input from the user with the item and the priority, Those inputs will be added to a dictionary representing a row of data, and that row will be added to the list. After iterating through the dictionary, the result with the newly added data will show on the screen.

```
elif (strChoice.strip() == '2'):
   strItem = str(input("What is the Item?: ")).strip()
   strPriority = str(input("What is the Priority?: ")).strip()
   dicRow = {"Item": strItem, "Priority": strPriority}
   lstTable.append(dicRow)
   print("Current Data in the Table: ")
   for dicRow in lstTable:
       print(dicRow)

   print("Current items in ToDo file: ")
   for row in lstTable:
       print(row["Item"] + "(" + row["Priority"] + ")")

   continue
```

Figure 5.

The next step is the most complex part of the code. Option 3 will allow you to remove an item from the list table. The first part of the code will receive input from the user with the item that the user wants to delete, and if the item is found, we will delete it. If the item from the user is not found, it will print a message letting the user know that this item is not found and will show the current data.

```
 elif(strChoice.strip() == ‘3’):
   strKeyToRemove = input("Which Item would you like to remove?: ")
   blnItemRemoved = False
   intRowNumber = 0
   while (intRowNumber < len(lstTable)):
       if(strKeyToRemove ==
str(list(dict(lstTable[intRowNumber]).values())[0])):
           del lstTable[intRowNumber]
           blnItemRemoved = True
       intRowNumber += 1
   if (blnItemRemoved == True):
       print("the Item was removed ")
   else:
       print("I'm sorry i couldn't find that Item")

   print("Current items in ToDo file: ")
   for row in lstTable:
       print(row["Item"] + "(" + row["Priority"] + ")")
   continue
```

Figure 6.

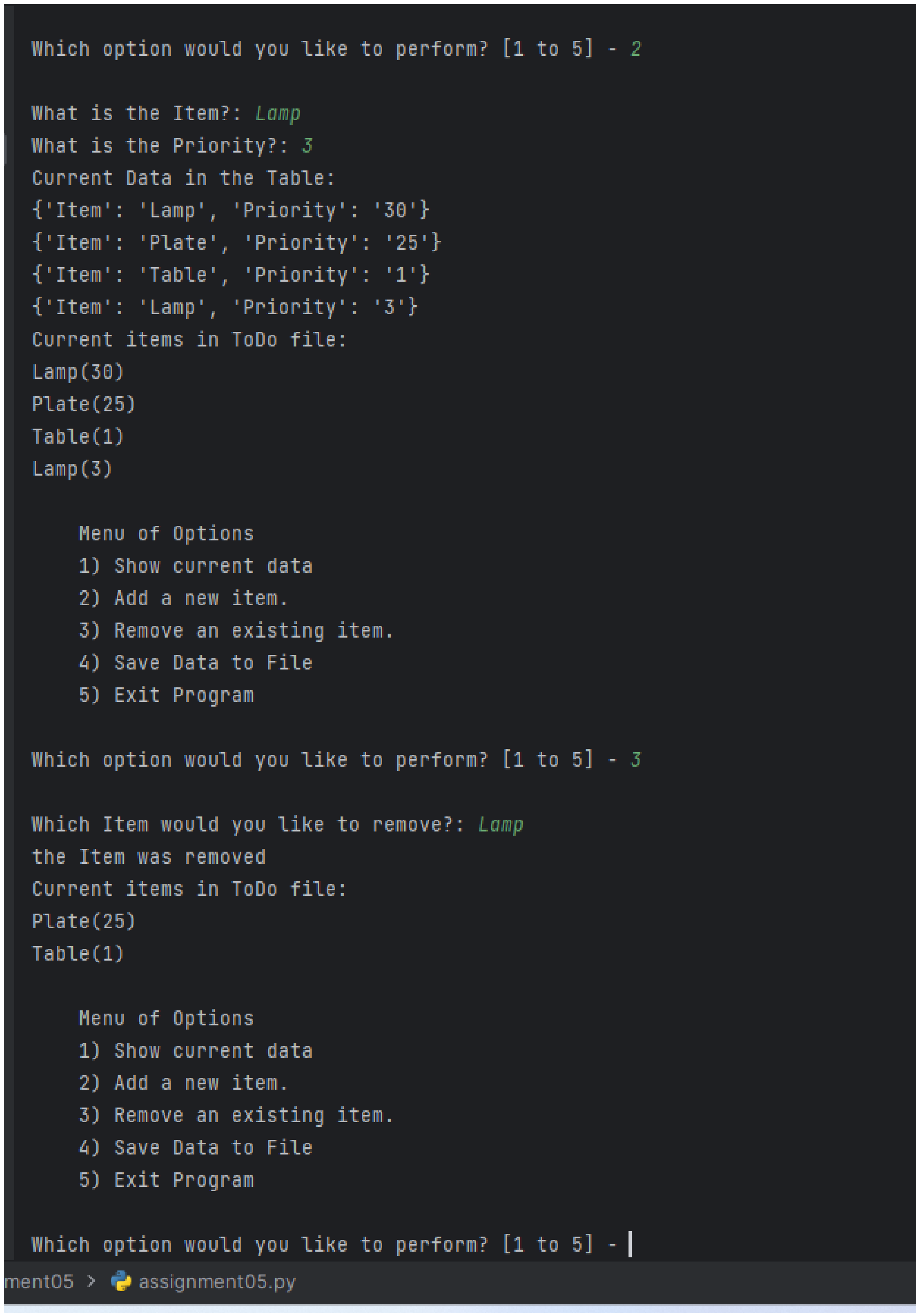
The next step is option 4, which will save the tasks in the file. It will let the user choose if they want or don't want to save the data, and once they choose the desired option, it will let them return to the menu:

```
elif (strChoice.strip() == '4'):
   print("Current items in ToDo file: ")
   for row in lstTable:
       print(row["Item"] + "(" + row["Priority"] + ")")

   if("y" == str(input("save this data to a file y/n: ")).strip().lower()):
       objFile = open(strFile, 'w')
       for dicRow in lstTable:
           objFile.write(dicRow["Item"] + "," + dicRow["Priority"] + "\n")
       objFile.close()
       input("Data saved, press the [Enter] key to return to the menu")
   else:
       print("data was not saved, but previous data still existing, press
[Enter] to return to the menu")

   continue
```
Figure 7.

Finally the last option, if the user choose option 5 it will exit the program.

```
elif (strChoice.strip() == '5'):

   break  # and Exit the program
```
Figure 8.

Here is an example of our program running, we added the Item Lamp twice on our file, and then we deleted the item Lamp, and did that successfully.



```
Which option would you like to perform? [1 to 5] - 2

What is the Item?: Lamp
What is the Priority?: 3
Current Data in the Table:
{'Item': 'Lamp', 'Priority': '30'}
{'Item': 'Plate', 'Priority': '25'}
{'Item': 'Table', 'Priority': '1'}
{'Item': 'Lamp', 'Priority': '3'}
Current items in ToDo file:
Lamp(30)
Plate(25)
Table(1)
Lamp(3)

    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

Which Item would you like to remove?: Lamp
the Item was removed
Current items in ToDo file:
Plate(25)
Table(1)

    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] - 
```

Figure 9.