Kerry Bosworth  
Nov 5, 2019  
Assignment05

# Working with Lists and Dictionaries

**Introduction**

This week we worked with lists and dictionaries and learned how to load data into and reference them. The key difference between a list and a dictionary is that a dictionary is like a hash. Instead of subscript or index as a number, dictionaries use a key which is text. They also use curly braces { } instead of [ ].

When working with data it is best practice to load it into memory and manipulate it. Then store it back in the file. A file with multiple rows turns into a table and is read into a list. It can store rows of values and dictionaries for more complex data. This can be done by opening the file and feedings each row into a table. If it holds key/value pairs, you can load the rows as dictionaries into the list.

**Dictionaries**

Here is how you read data from a file into a Dictionary that has entries like ‘item,priority’ per line and prints it on the screen.  
  
# File to List  
objFile = open(strFile, "r")  
for row in objFile:  
 lstRow = row.split(",") # Returns a list!  
 dicRow = {"item": lstRow[0], "amount": lstRow[1]}  
 print(dicRow)  
objFile.close()

If you need to store it into memory, you can put it in a table:  
  
# File to List  
objFile = open(strFile, "r")  
for row in objFile:  
 lstRow = row.split(",") # Returns a list!  
 dicRow = {"item": lstRow[0], "amount": lstRow[1]}  
 lstTable.append(dicRow)  
objFile.close()  
  
**Separates of Concerns and Functions**  
The concept of the programming pattern called “Separations of Concerns” is to compartmentalized your code for easy reuse in other places. This helps keep complex code in order.

For Example:  
  
#-- Data --# # Example: Declare variables and constants

#-- Processing --# # Example: Perform tasks on data

#-- Presentation (Input/Output) --# # Example: Get user input

Sometimes sections contain more than one type shown above. This is where functions can help. Functions can help control the flow of the program and makes code more reusable. This allows processing to stay in processing and can be called elsewhere. This keeps processing code in the expected place and easily found and reviewed.

Functions are not executed or run when they are first encountered. Instead the program calls it later. You can put them in separate script and link the scripts (that’s a module). Or you can keep in same script and call them throughout the program as needed. This should reduce the number of lines of code overall.

**Templates**

It is best practice to create templates. This allows for a clean starting point each time a new program is written. When used across a team, it keeps everyone’s work consistent. Later as authors leave the environment it can be easily consumed by new parties. Templates start with a formatted header. For editable parts use <> to indicate these sections should be filled in later. PyCharm and other IDE’s allow you to save templates (Tools -> Save File as Template). Then you can right click -> New -> Template anytime you are starting a new file. Voila! All files will be consistent.

For Example:  
# ------------------------------------------------------------------------ #  
# Title: <Type the name of the script here>  
# Description: <Type a description of the script>  
# ChangeLog (Who,When,What):  
# <Example Dev,01/01/2030,Created Script>  
# ------------------------------------------------------------------------ #

# -- Data -- #  
# -- Processing -- #  
# -- Input/Output -- #

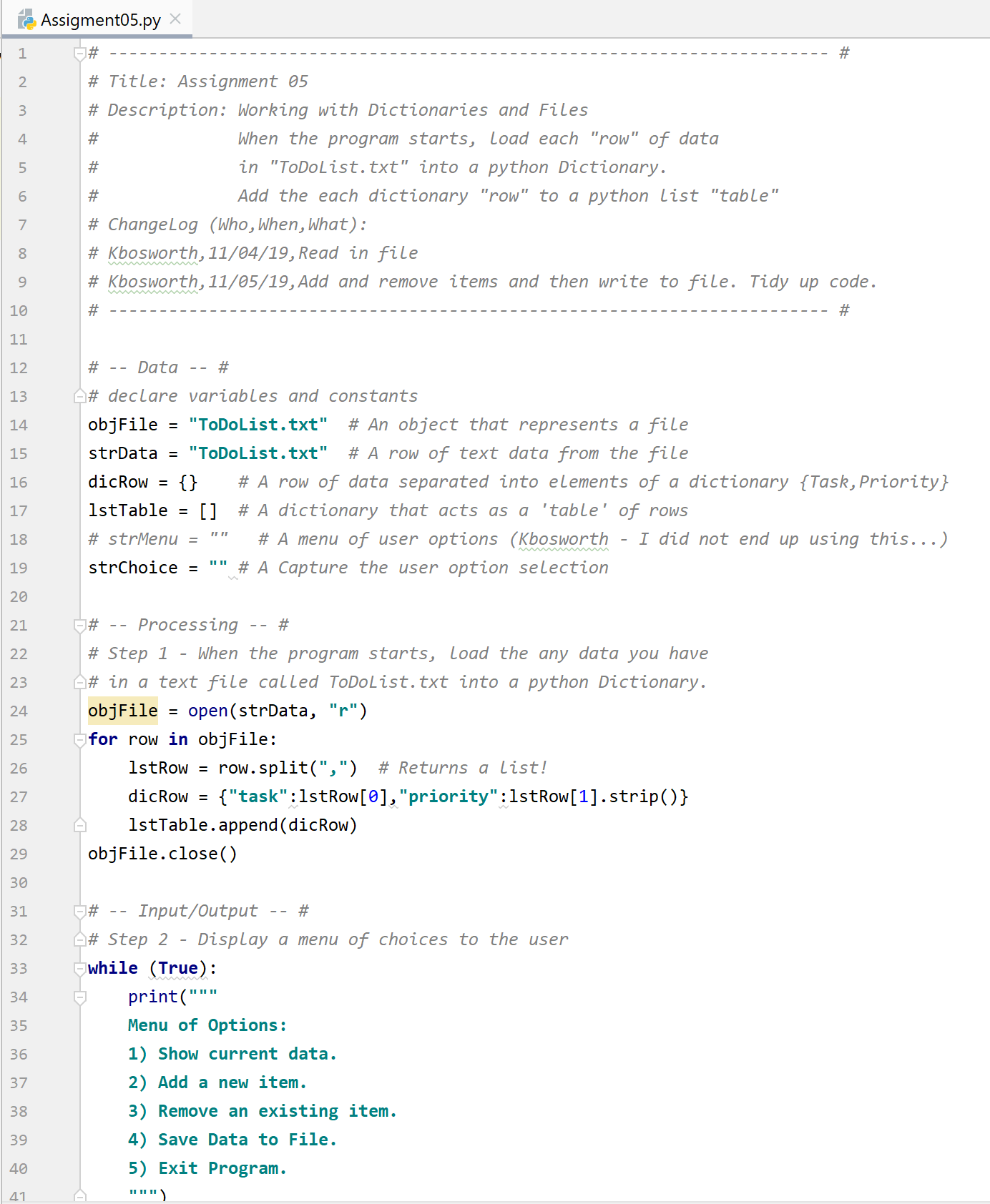
**Error Exception Handling**

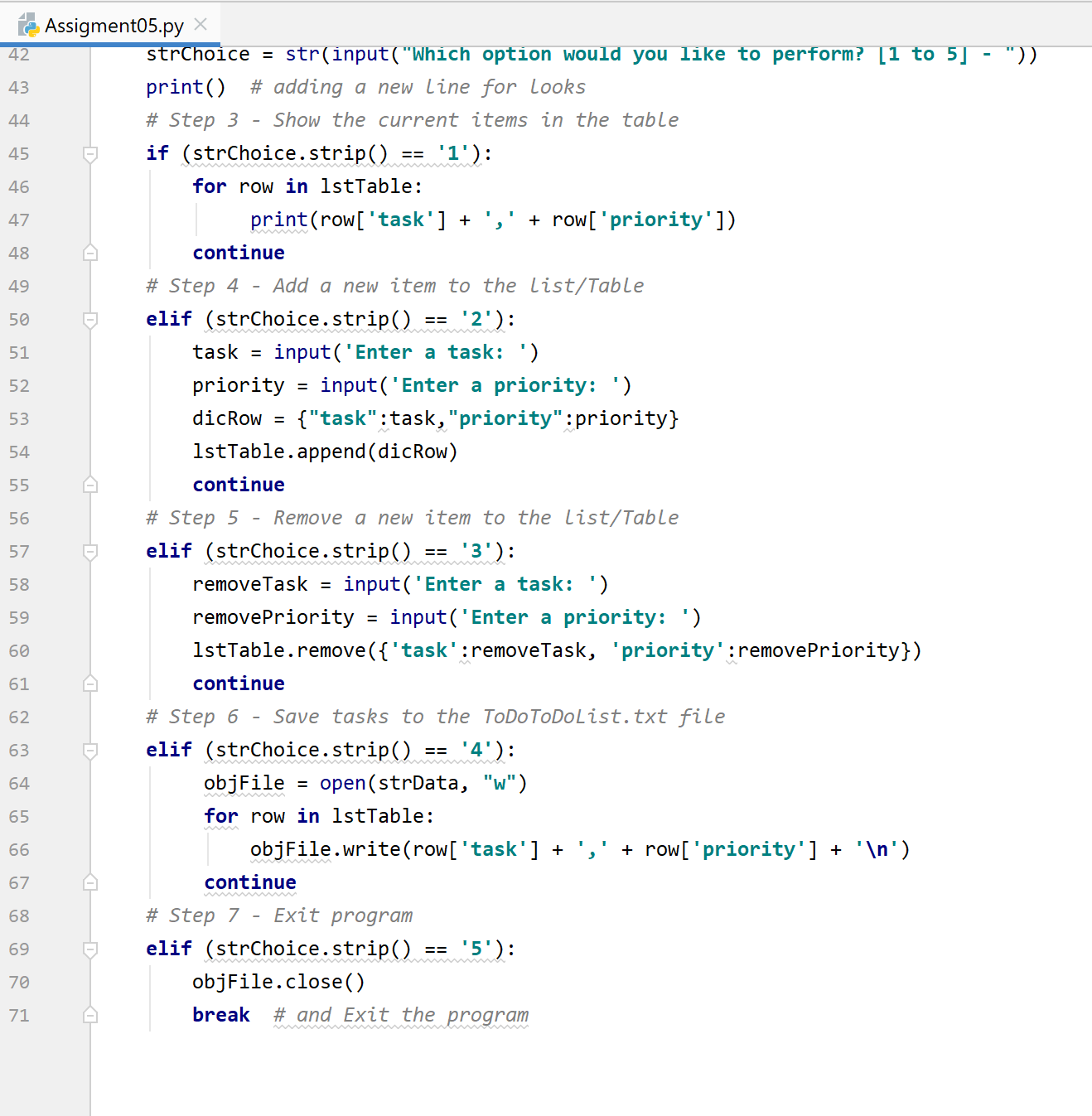
Error handling is critical but unfortunately very variable. It is often left up to the program author and it’s quality depends on how thoughtful he or she was when they wrote it. When a program encounters an error, it will display the system errors by default. If you capture the processing code within a ‘try’ and ‘except’ statement, you can print a customize message if it does not execute successfully.  
  
For example:  
  
try:  
 num1 = 44  
 num2=0  
 divideNum=num1/num2  
except:  
 print(“An error occurred when trying to divide”, num1, “by”, num2)

**GitHub**  
  
Get organized by putting code in a central place that can be accessed by multiple people. This helps with code review and sharing and is also a good backup strategy. Previously, hard drives, a network share, or Team Foundation Servers were where it was saved, but now it is common to put it out on the internet. There is can be used by a greater audience and the source code control repositories track changes. This introduced version control so that you could go back and look at previous copies if needed.

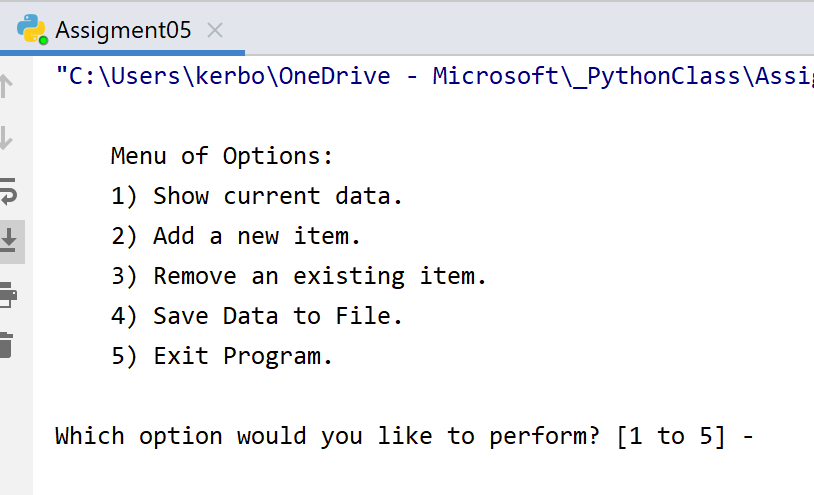
**Programming Assignment**

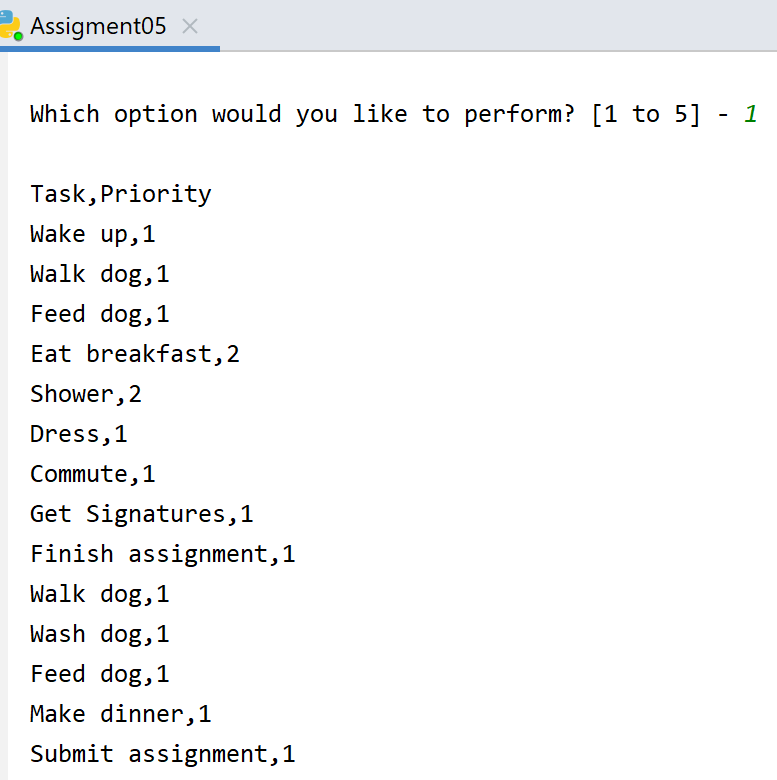
In this programming assignment the objective was to read a file into a dictionary and then add and remove entries in memory before writing permanently to a file. Using lists and dictionaries made it more complex. It was easy to read into a dictionary, but then later it was more challenging determining how to interact with the data, list or as a dictionary. In my code, when I needed to add, remove or write I manipulated the table stored in a list (See Figure 1 below).

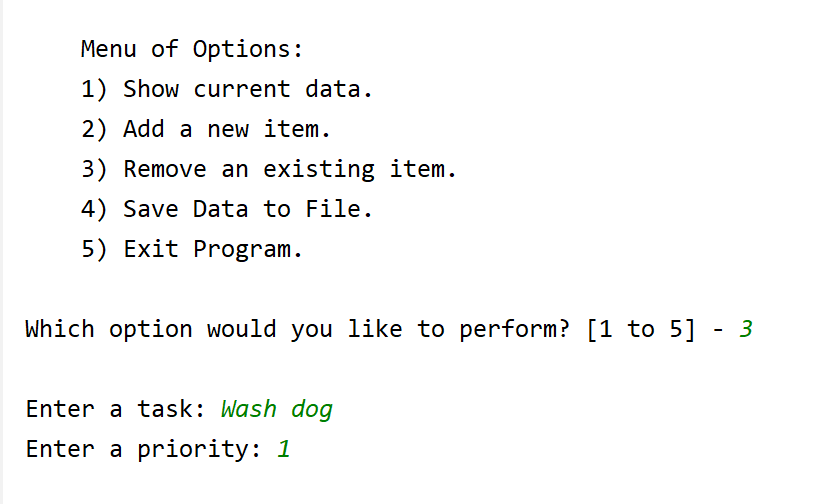


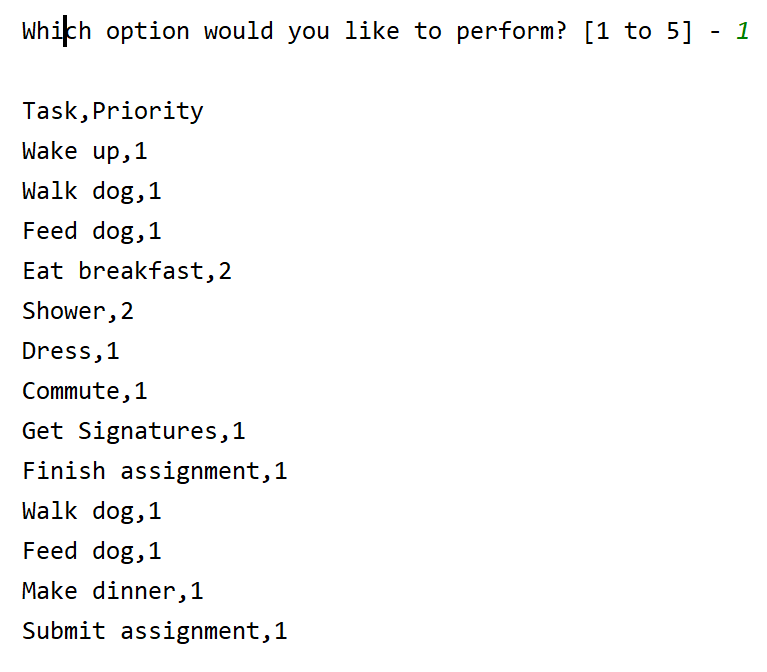


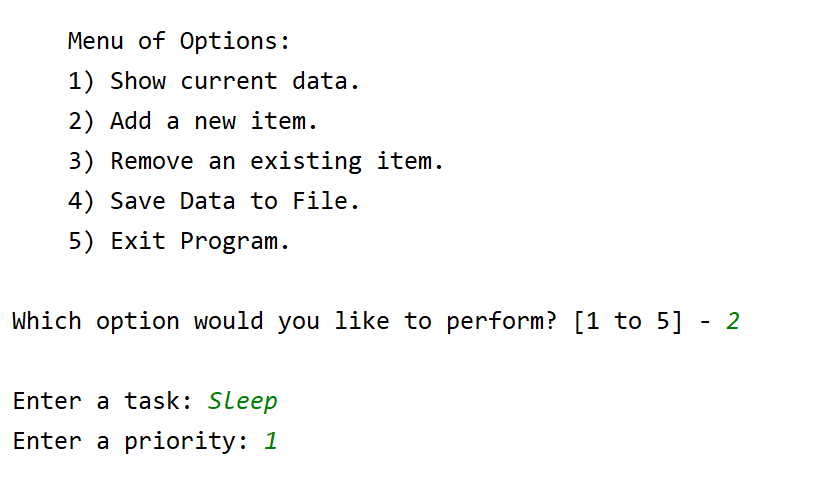
**Figure 1: To Do List program**  
Here is a clean run of the program (see Figure 2). The file has a list of tasks, then allows the user to add and remove tasks from the list.

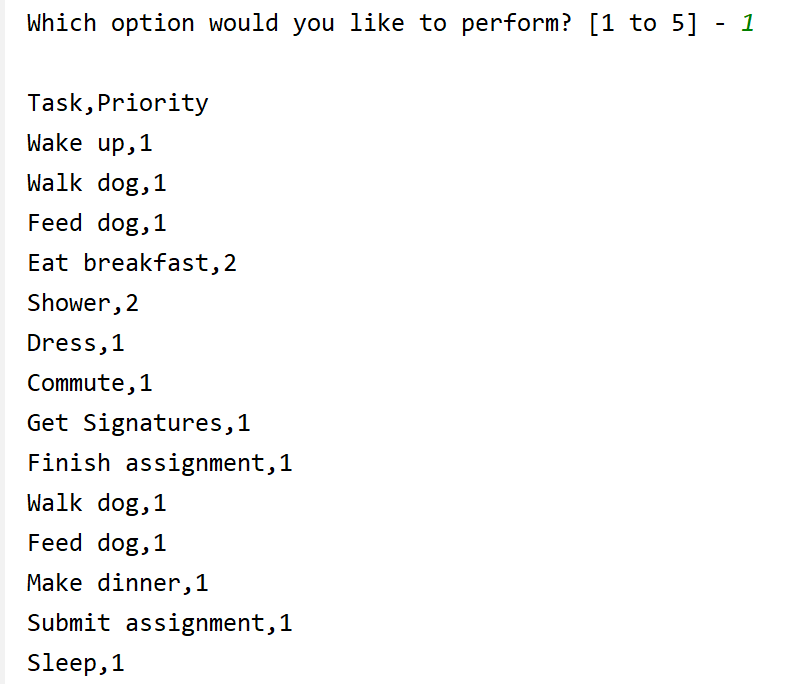
****

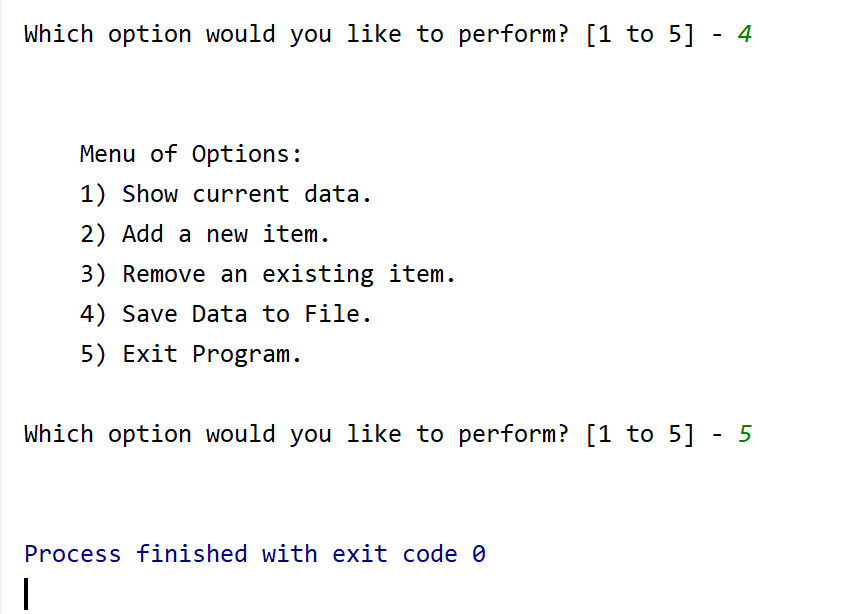
****

****

****

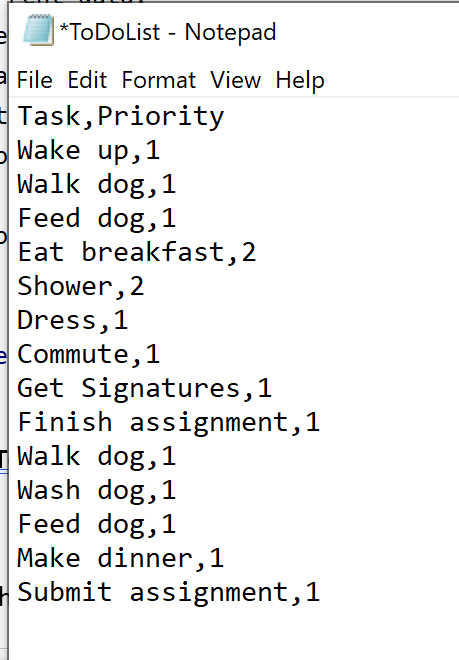
****

****

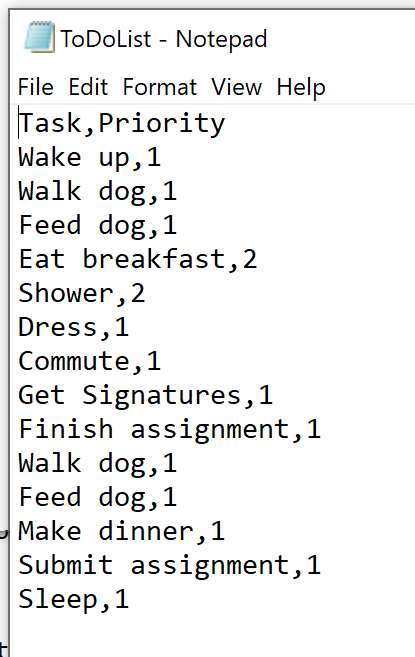
****

**Figure 2: Running To Do List program in PyCharm**

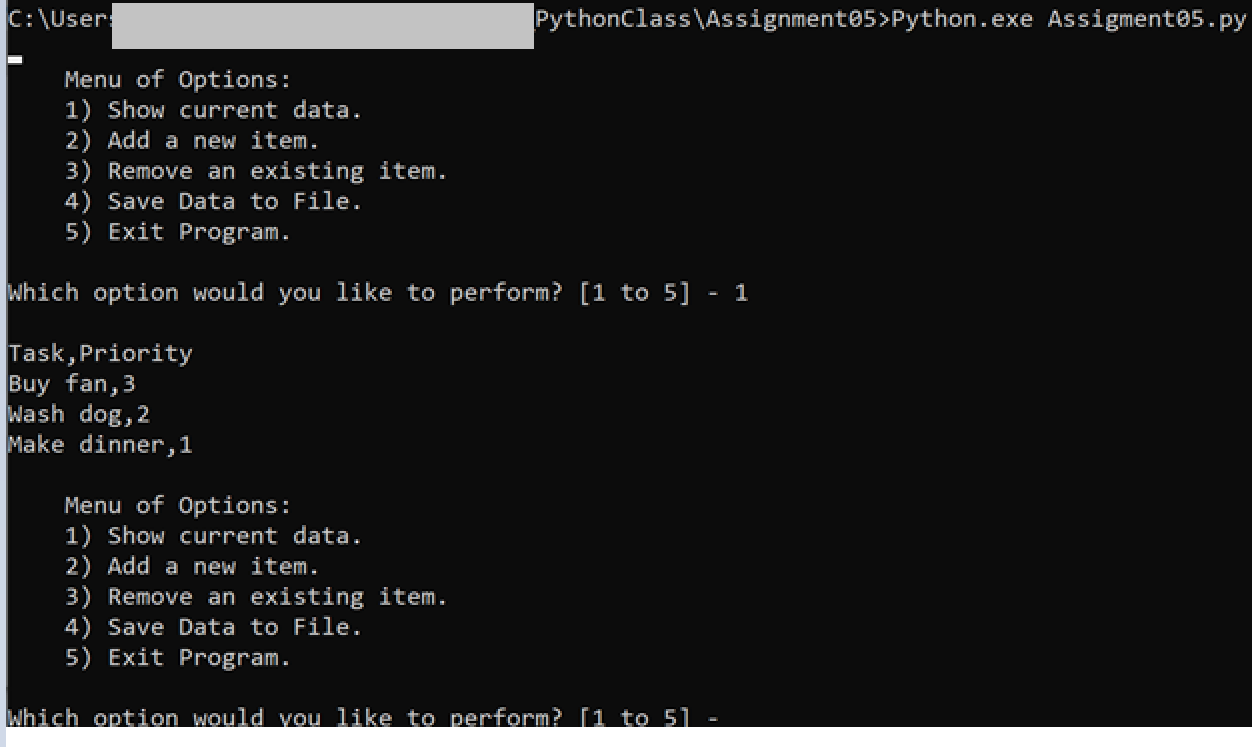
Here is the file before (see Figure 3):

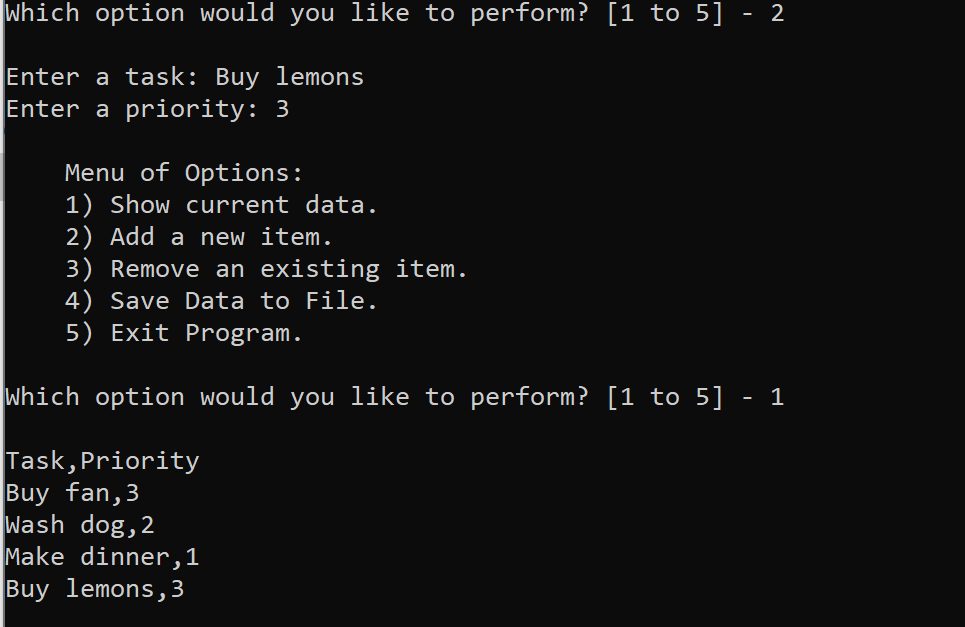
  
  
**Figure 3: Original list**

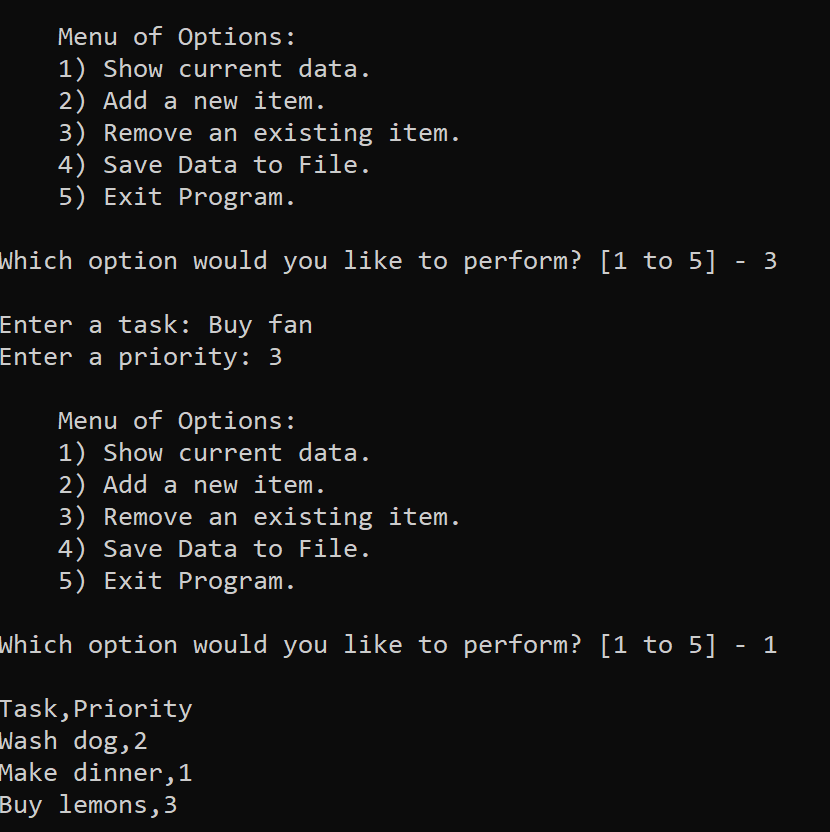
Here is the file after (see Figure 4):

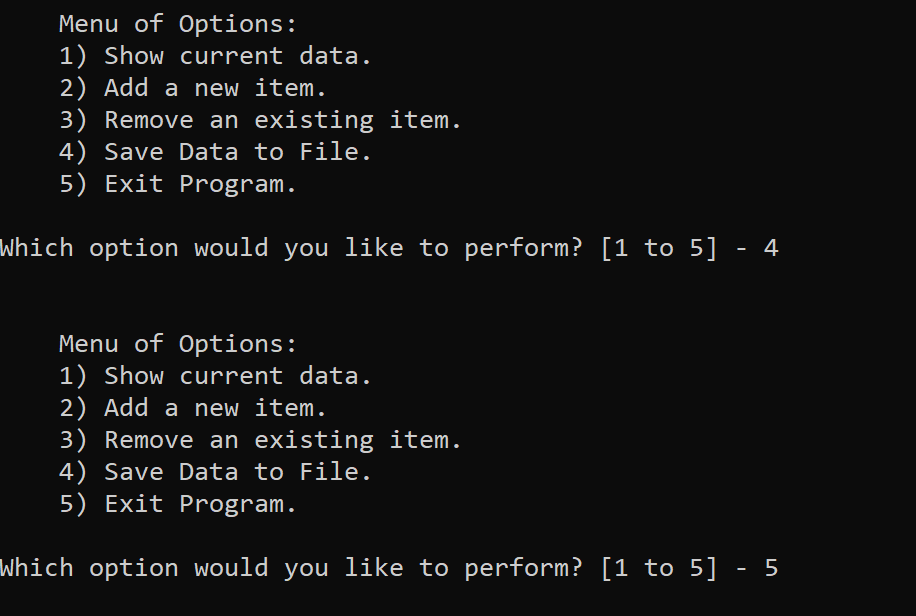


**Figure 4 Updated list**

This run (see Figure 5) demonstrates the same thing as above.  


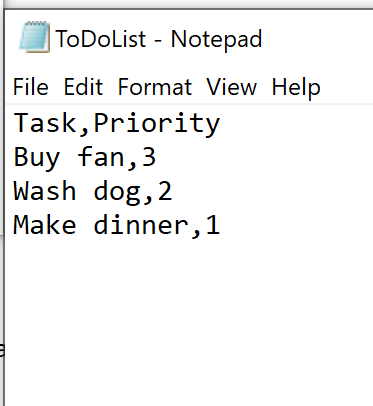




****

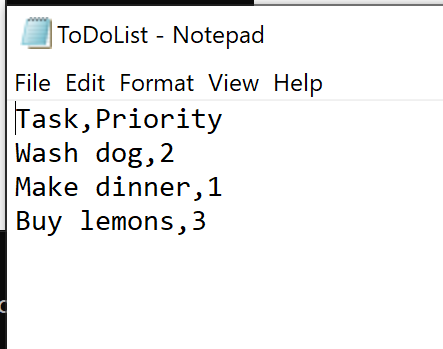
**Figure 5: Running To Do List program in Command line**

Here is the file before (see Figure 6):



**Figure 6: Original list**

Here is the file after (see Figure 7):

****

**Figure 7: Original list**

**Summary**

We learned how to work with data both horizontally (dictionaries) and vertically (table list). This made me confront one of the things that has challenged me in the past with programming. To trust the code and not overthink or manipulate. I spent many hours thinking into it too much when in the end the solutions were very few lines of code.