Carolina Cerda

May 17, 2022

Foundations of Programming: Python

Assignment 05

https://github.com/carolinacerda/IntroToProg-Python

Creating a Script Using Text Files and a List of Dictionaries That Manages a To-Do List

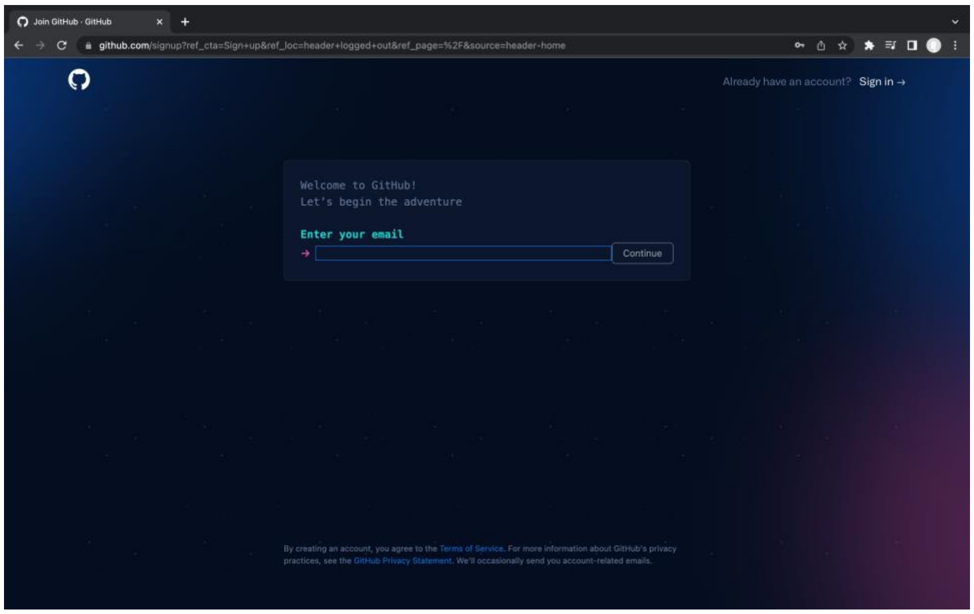
Introduction

In this assignment, text files and a list of dictionaries will be used in order to create a script to manage a To-Do list, similar to the previous week's Home Inventory assignment, in which the user is provided a menu of options to select from that will allow them to view the current data, add or remove items, save the data, and exit the program. GitHub is also introduced to create familiarity with this source control software as a means to store backups of code and other files as well as be able to share these files more easily with others.

GitHub

Traditionally, and also still common practice, code files by developers are shared with others using a network share on an organization's server which makes their files accessible by their peers and stores backups of these. With the growing internet age, it is becoming more and more common to instead use source control software to upload and share these files over the internet. One such software is GitHub which allows code and other files to be saved and shared over the Internet on the domain GitHub.com. GitHub allows backups of code files to be uploaded and makes availability to these files more readily accessible by others (R. Root, "_Mod5PythonProgrammingNotes"). GitHub also helps in saving the revision history of these files efficiently while also maintaining their integrity so it is also possible to make edits on these files directly (R. Root, "Intro to Python Mod05").

For the remaining assignments of this course, students' files will be uploaded to GitHub and shared with peers to not only provide each other feedback on our work but also become familiar with using such software in preparation for potential work with the software in the future. Firstly, an account can be made by following the prompts for signing up for an account with GitHub (Figure 1).


Join GitHub · GitHub
github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home
Already have an account? Sign in →
Continue

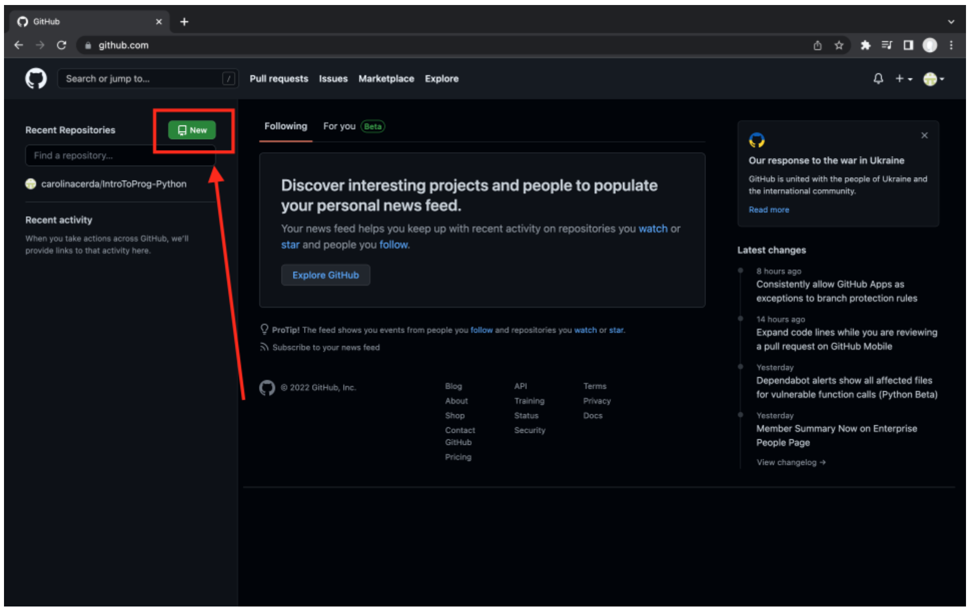
Welcome to GitHub!
Let's begin the adventure

Enter your email

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Figure 1. GitHub Sign Up Page.

After an account has been created, a repository, or a central location where the files and data will be stored in, can be created (Figure 2).



When you take actions across GitHub, we'll provide links to that activity here.

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

ProTip! The feed shows you events from people you follow and repositories you watch or star.

Subscribe to your news feed

Our response to the war in Ukraine

GitHub is united with the people of Ukraine and the international community.

8 hours ago
Consistently allow GitHub Apps as exceptions to branch protection rules

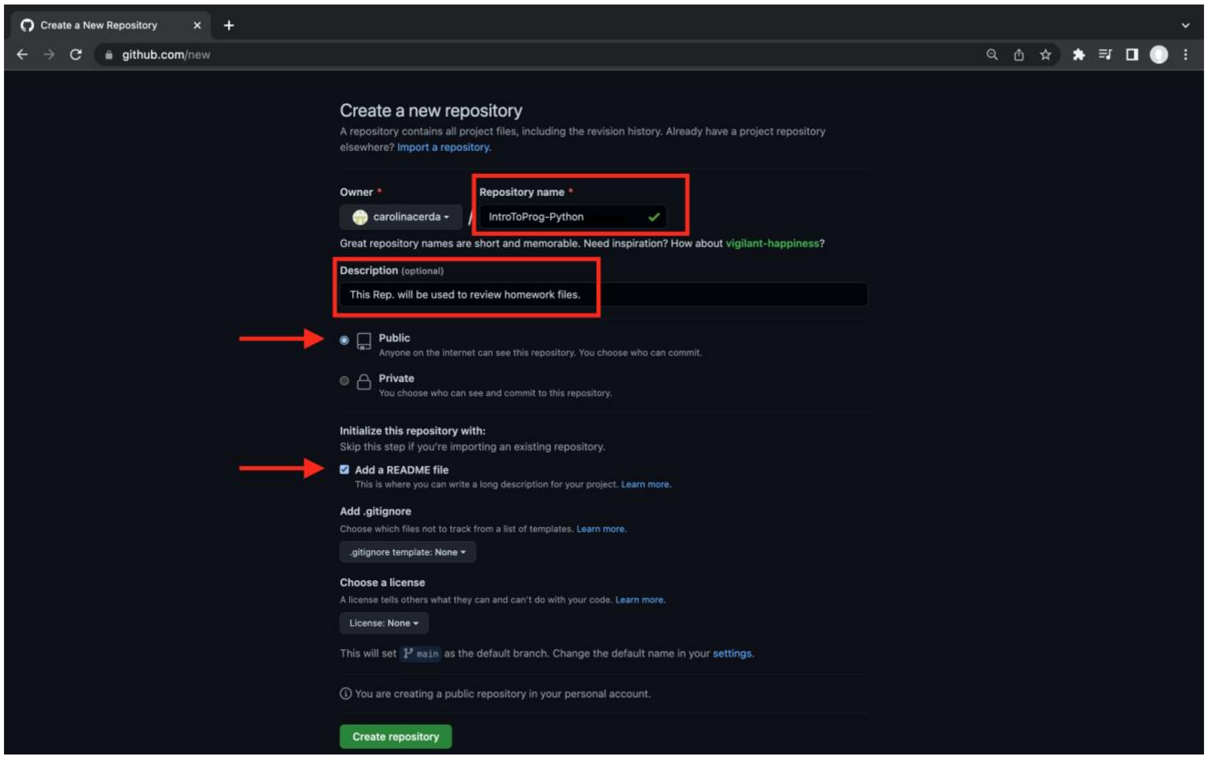
14 hours ago
Expand code lines while you are reviewing a pull request on GitHub Mobile

Yesterday
Dependabot alerts show all affected files for vulnerable function calls (Python Beta)

Yesterday
Member Summary Now on Enterprise People Page

Figure 2. GitHub Home.

A unique repository name must be created and an optional description can also be added. For this course, the repository will be named “IntroToProg-Python.” To ensure others can see the repository, the public option will be selected and a “README” file will also be selected to initialize the repository (Figure 3).


Create a New Repository

github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * carolinacerda / Repository name * IntroToProg-Python

Great repository names are short and memorable. Need inspiration? How about vigilant-happiness?

Description (optional)

This Rep. will be used to review homework files.

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☑ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

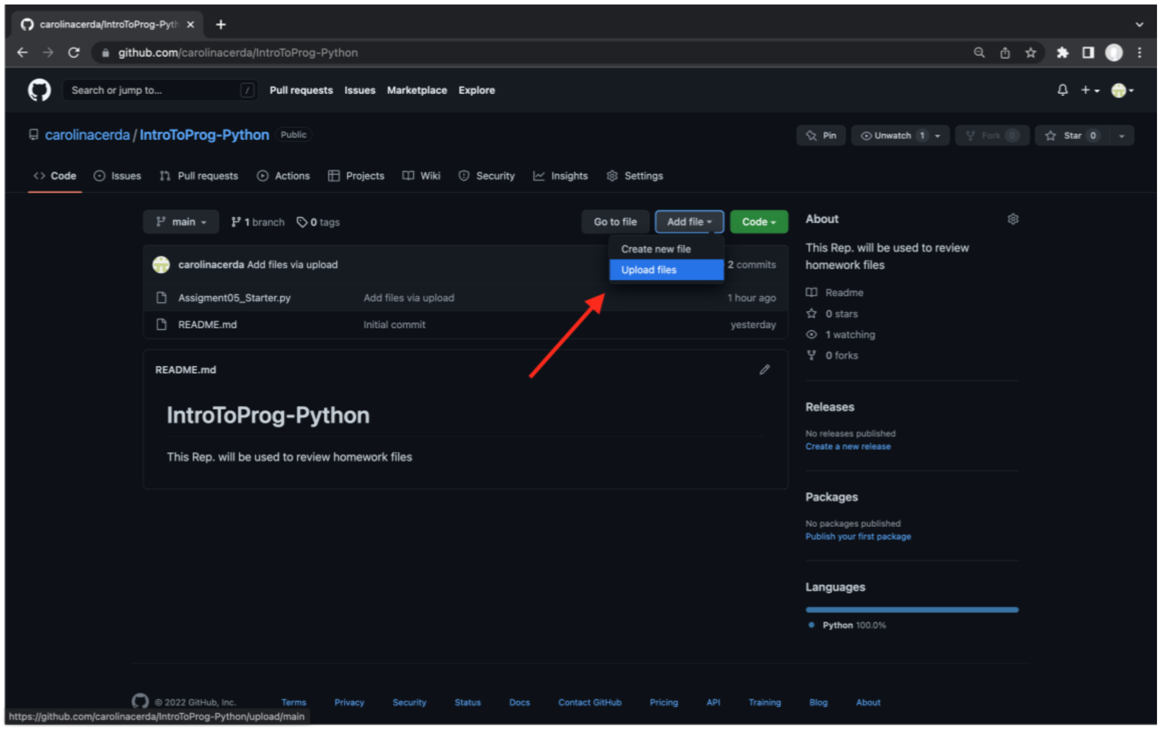
This will set main as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

Create repository

Figure 3. Creating a new repository for Foundations of Programming: Python.

Finally, files can be uploaded or created to the repository and shared with the class and others (Figure 4). This paper as well as the script for this assignment's program will both be uploaded to the repository created above. (Hi, classmates!)


carolinacerda/IntroToProg-Python

github.com/carolinacerda/IntroToProg-Python

Search or jump to... Pull requests Issues Marketplace Explore

carolinacerda / IntroToProg-Python Public

Pin · Unwatch 1 · Fork 0 · Star 0

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

main · 1 branch · 0 tags · Go to file · Add file · Code

Create new file
Upload files

carolinacerda Add files via upload — 2 commits

Assigment05_Starter.py	Add files via upload	1 hour ago
README.md	Initial commit	yesterday

README.md

IntroToProg-Python

This Rep. will be used to review homework files

About
This Rep. will be used to review homework files
Readme
0 stars
1 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages
Python 100.0%

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

https://github.com/carolinacerda/IntroToProg-Python/upload/main

Figure 4. Where to upload and create new files to the repository.

Creating the Script

This script was created by building upon a starter script provided by Professor Randal Root. The following discussion will focus on the changes and additions made to the provided script.

DATA: Declaring the Variables and Constants

This section of the script focuses on declaring any variables and constants that will be used to be more easily understood by anyone who decided to view the code. The only addition to the provided script was the constant “strFile” which would be used to locate the text file “ToDoList.txt” which is where any data will be loaded from or stored to for this program (Line 4).

```
 1  # -- Data -- #
 2  # Declare variables and constants
 3  objFile = None  # An object that represents a file
 4  strFile = "ToDoList.txt"  # A file
 5  strData = ""  # A row of text data from the file
 6  dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
 7  lstTable = []  # A list that acts as a 'table' of rows
 8  strMenu = ""  # A menu of user options
 9  strChoice = ""  # Capture the user option selection
```

PROCESSING: Loading Data from Existing Text File into a Python List of Dictionary Rows

This section distinguishes processing code from presentation code so that it is easier to go back to when editing the script. As functions are not yet being used this section is focused only on loading any data from the text file “ToDoList.txt” into a python list of dictionary rows that the script can use. Line 15 opens the file for reading. Then the data in the file returns a row of text data that can be processed into a row of data separated into elements of a dictionary by assigning the indexed strData values to keys (Line 16-18). The dicRow can then be appended to the list that acts as a ‘table’ of rows, or lstTable (Line 19). Then the object that represents the file is closed (Line 20).

```
12  # -- Processing -- #
13  # Step 1 - When the program starts, load any data you have
14  # in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
15  objFile = open(strFile, "r")
16  for row in objFile:
17      strData = row.split(",")
18      dicRow = {"Task": strData[0].strip(), "Priority": strData[1].strip()}
19      lstTable.append(dicRow)
20  objFile.close()
```

INPUT/OUTPUT:

Step 2. Displaying a Menu of Choices

The provided starter script included the code to display a menu of choices to the user. The only change to the starter script is that the menu will only be displayed once at the beginning so that the rest of the program does not appear cluttered or overly repetitive. This was achieved by moving the while loop directly after the menu of options instead of at the beginning (Line 24 → Line 31). Furthermore, a print statement is added as a stylistic choice directly below the while loop (and above the user input statement) to distinguish each time the loop comes back to the starting point by dividing it with a line created by printing the character "-" forty times that creates a divider (Line 32).

```
22  # -- Input/Output -- #
23  # Step 2 - Display a menu of choices to the user
24  print("""
25      Menu of Options
26      1) Show current data
27      2) Add a new item.
28      3) Remove an existing item.
29      4) Save Data to File
30      5) Exit Program""")
31      while True:
32          print("-" * 40)
33          strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
34          print()  # adding a new line for looks
```

Step 3. Displaying the Current To-Do List

A print statement is used to open this selected option by printing the message, "Your Current To-Do List Is:" which will be followed by the current data (Line 38). Then, to display the current items in the table, a for loop is used to pull each of the rows in the dictionary within the list table (Line 39). To print the items, since a dictionary is being used, the data is separated using the key for the values and "|" is used as a separation (Line 40). A new line is then added for aesthetics (Line 41).

```
36          # Step 3 - Show the current items in the table
37          if strChoice.strip() == '1':
38              print("Your Current To-Do List Is:")
39              for dicRow in lstTable:
40                  print("  ", dicRow["Task"], "|", dicRow["Priority"])
41          print()  # adding a new line for looks
42          continue
```

Step 4. Adding New Items to the To-Do List

The second option from the menu allows the user to input new tasks and their corresponding priorities to the To-Do List. To add new items to the To-Do List, firstly, an instruction is printed out that states, “Type in your task and its priority." (Line 46). To make the task and priority that the user will input useful to add to the list of dictionary rows, they are made into string variables as seen in Lines 47 and 48 and also stripped of any extra spaces the user may have inputted along with it. To ensure that the input is consistent in terms of presentation the .title() method is used to ensure each of the words are capitalized (Line 49). Using the input function embedded into the string function, whatever the user inputs will become the objects for creating the values for the keys “Task” and “Priority” as elements of the dictionary “dicRow”, which will then be appended to the list table, “lstTable”, as more data is inputted into the program (Lines 49 - 50).

```
44    # Step 4 - Add a new item to the list/Table
45    elif strChoice.strip() == '2':
46        print("Type in your task and its priority.")
47        strTask = str(input("  Enter a task: ")).strip()
48        strPriority = str(input("  Enter a priority [High|Medium|Low]: ")).strip()
49        dicRow = {"Task": strTask.title(), "Priority": strPriority.title()}
50        lstTable.append(dicRow)
51        print()  # adding a new line for looks
52        continue
```

Step 5. Removing Items from the To-Do List

The third option from the menu allows the user to removes previously inputted tasks from to the To-Do List. To remove items from the To-Do List, firstly, a variable, “strRemove”, is created which will use the user input obtained from the instruction, “Which task do you want to remove?: " to locate a row from the list table that matches the input (Line 56). The .lower() method is used to ensure that the two objects are consistent and will match if they are equal to each other (Line 58). Thus, a for loop is created in order to match the input against every row in the lstTable and if the input matches value for the key, “Task”, within the row then it should remove that row from the lstTable and print a statement to confirm to the user that the task has been removed will be presented (Lines 57 – 60). Once the program finds a match, it should break the loop through the break statement under the print statement (Line 61). If there is no match for the row, then it will continue to loop for each row through the pass statement under the else statement for that if loop to prevent the program from printing the confirmation or rejection statements multiple times (Lines 62 – 63). The for loop created is then broken through another else statement that should only print after it has matched the input to every row available and if there is no such task then it will print a rejection statement that will state, “Sorry, the task, “strRemove”, doesn't exist in the current To-Do List. Please try again." (Lines 64 – 67). Then, the while loop from Step 2 should loop again through the continue statement (Line 31, Line 68, respectively).

```
54  # Step 5 - Remove a new item from the list/Table
55  elif strChoice.strip() == '3':
56      strRemove = input("Which task do you want to remove?: ")
57      for row in lstTable:
58          if row["Task"].lower() == strRemove.lower():
59              lstTable.remove(row)
60              print("\n  The task, ", "'", strRemove.title(), "'", ", has been removed.", sep="")
61              break
62          else:
63              pass
64      else:
65          print("\n  Sorry, the task, ", "'", strRemove.title(), "'", ", doesn't exist in the current To-
66                                                      “Do List. Please try again.", sep="")
67      print()  # adding a new line for looks
68      continue
```

Step 6. Save Tasks to the Text File

This option of the menu allows the user to be given the choice of whether or not they want to save the inputted data into the text file. This is done by firstly printing the message “Would you like to save your data?" (Line 72). Then, a new variable is created, “strAnswer”, to record the data inputted by the user that would determine if they want the data to be saved to a text file. If the user inputs “y” to the prompt in Line 74, then the condition is met for the if function to open the text file, “ToDoList.txt”, where the data previously inputted by the user will be written into and then closed which will prompt the message “Data saved to file!” to be printed and confirm to the user that their data was saved (Lines 75 – 79). So that the data will be saved in a similar fashion to the way that the data in the file previously stored, objFile is written with concatenation so that it saves as a “table” with the format “task, priority” and continue on a new line for each item (Line 77). If instead the user inputs “n” into the prompt, then only the message “Data not saved.” will be displayed back to the user (Line 81).

```
70  # Step 6 - Save tasks to the ToDoList.txt file
71  elif strChoice.strip() == '4':
72      print("Would you like to save your data?")
73      strAnswer = str(input("  Enter 'y' or 'n': "))
74      if strAnswer.lower() == "y":
75          objFile = open('ToDoList.txt', "w")
76          for dicRow in lstTable:
77              objFile.write(str(dicRow["Task"]) + ", " + str(dicRow["Priority"]) + "\n")
78          objFile.close()
79          print("\n  Data saved to file!")
80      else:
81          print("\n  Data not saved.")
82      print()  # adding a new line for looks
83      continue
```

Step 7. Exit the Program

The final option from the menu is to exit the program which is achieved through the “break” command which breaks the while loop created previously in Step 2 (Line 88, Line 31, respectively). Before the break a simple statement is printed to ensure the user that the exit has been completed (Line 87).

```
85      # Step 7 - Exit program
86      elif strChoice.strip() == '5':
87          print("  Exit complete.")
88          break  # and Exit the program
```

Step 8. Optional Safety Net

If the user were to input anything other than a number from 1 to 5 into the prompted user input statement (Line 33) then the following “safety net” is written using an else statement so that the user is urged to provide input once again through the message, "Input not recognized! Please try again." (Lines 91 - 92). By continuing the loop, it returns the program to the start of the while loop (Line 93).

```
90      # Optional - Safety net if no recognized option is inputted by the user.
91      else:
92          print("  Input not recognized! Please try again.\n")
93          continue
```

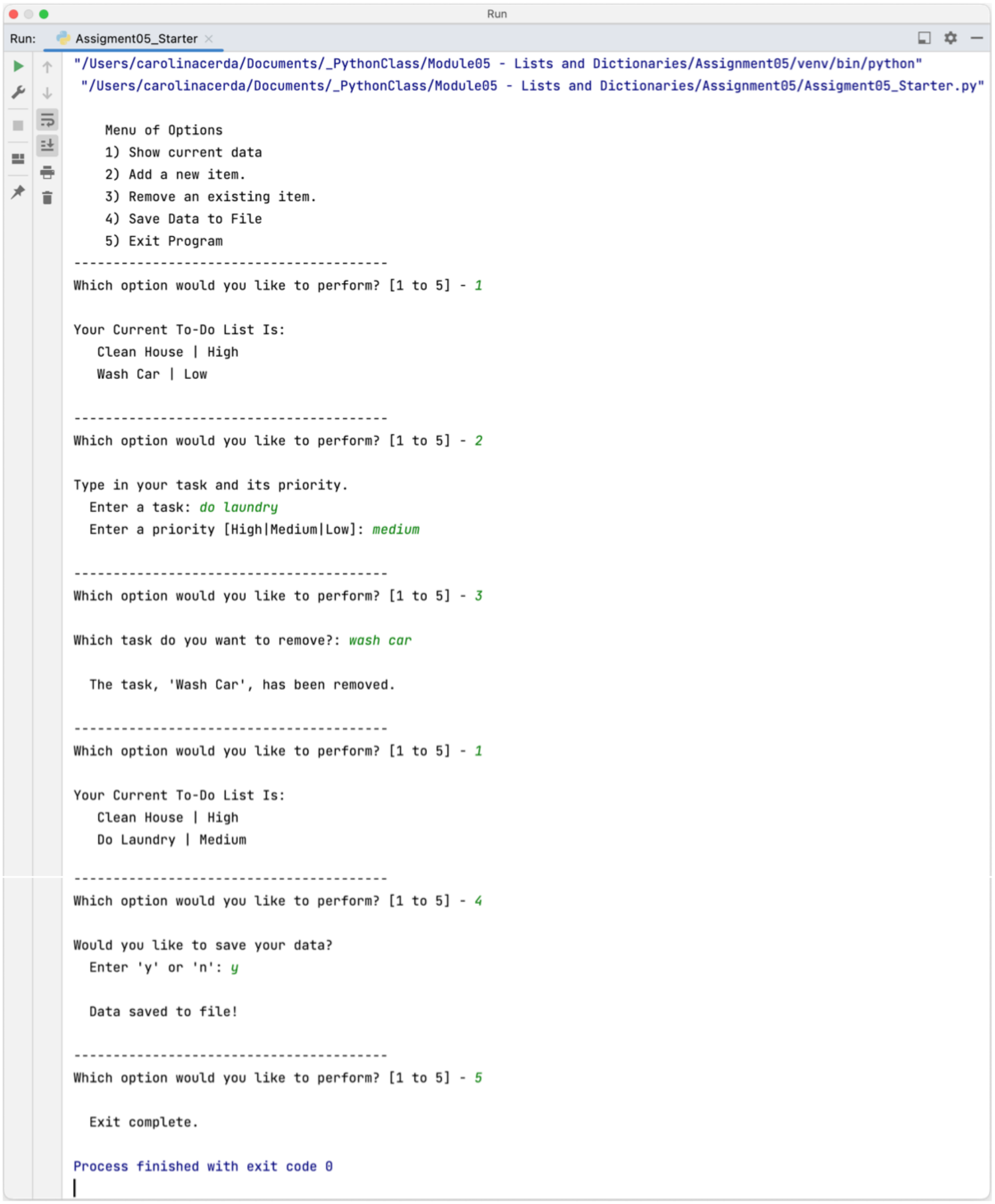
Running the Script

The previously discussed sections are put together to create the final script to manage a To-Do List as shown below.

```
 1  # -- Data -- #
 2  # Declare variables and constants
 3  objFile = None  # An object that represents a file
 4  strFile = "ToDoList.txt"  # A file
 5  strData = ""  # A row of text data from the file
 6  dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
 7  lstTable = []  # A list that acts as a 'table' of rows
 8  strMenu = ""  # A menu of user options
 9  strChoice = ""  # Capture the user option selection
10
11
12  # -- Processing -- #
13  # Step 1 - When the program starts, load any data you have
14  # in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
```

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


# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
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""")
    while True:
        print("-" * 40)
        strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
        print()  # adding a new line for looks

        # Step 3 - Show the current items in the table
        if strChoice.strip() == '1':
            print("Your Current To-Do List Is:")
            for dicRow in lstTable:
                print("  ", dicRow["Task"], "|", dicRow["Priority"])
        print()  # adding a new line for looks
        continue

    # Step 4 - Add a new item to the list/Table
    elif strChoice.strip() == '2':
        print("Type in your task and its priority.")
        strTask = str(input("  Enter a task: ")).strip()
        strPriority = str(input("  Enter a priority [High|Medium|Low]: ")).strip()
        dicRow = {"Task": strTask.title(), "Priority": strPriority.title()}
        lstTable.append(dicRow)
        print()  # adding a new line for looks
        continue

    # Step 5 - Remove a new item from the list/Table
    elif strChoice.strip() == '3':
        strRemove = input("Which task do you want to remove?: ")
        for row in lstTable:
            if row["Task"].lower() == strRemove.lower():
                lstTable.remove(row)
                print("\n  The task, ", "'", strRemove.title(), "'", ", has been removed.", sep="")
                break
            else:
                pass
        else:
            print("\n  Sorry, the task, ", "'", strRemove.title(), "'", ", doesn't exist in the current To-
                                                                                           “Do List. Please try again.", sep="")
        print()  # adding a new line for looks
        continue

    # Step 6 - Save tasks to the ToDoList.txt file
    elif strChoice.strip() == '4':
        print("Would you like to save your data?")
        strAnswer = str(input("  Enter 'y' or 'n': "))
        if strAnswer.lower() == "y":
            objFile = open('ToDoList.txt', "w")
            for dicRow in lstTable:
                objFile.write(str(dicRow["Task"]) + ", " + str(dicRow["Priority"]) + "\n")
            objFile.close()
            print("\n  Data saved to file!")
        else:
            print("\n  Data not saved.")
        print()  # adding a new line for looks
        continue

    # Step 7 - Exit program
    elif strChoice.strip() == '5':
        print("  Exit complete.")
        break  # and Exit the program

    # Optional - Safety net if no recognized option is inputted by the user.
    else:
        print("  Input not recognized! Please try again.\n")
        continue
```

The final result of the script in both PyCharm and the OS Command are shown below in Figure 5 and Figure 6, respectively.



```
"/Users/carolinacerda/Documents/_PythonClass/Module05 - Lists and Dictionaries/Assignment05/venv/bin/python"
 "/Users/carolinacerda/Documents/_PythonClass/Module05 - Lists and Dictionaries/Assignment05/Assigment05_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

Your Current To-Do List Is:
   Clean House | High
   Wash Car | Low

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

Type in your task and its priority.
  Enter a task: do laundry
  Enter a priority [High|Medium|Low]: medium

----------------------------------------
Which option would you like to perform? [1 to 5] - 3

Which task do you want to remove?: wash car

  The task, 'Wash Car', has been removed.

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

Your Current To-Do List Is:
   Clean House | High
   Do Laundry | Medium

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

Would you like to save your data?
  Enter 'y' or 'n': y

  Data saved to file!

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

  Exit complete.

Process finished with exit code 0
```

Figure 5. Final result of script in PyCharm.

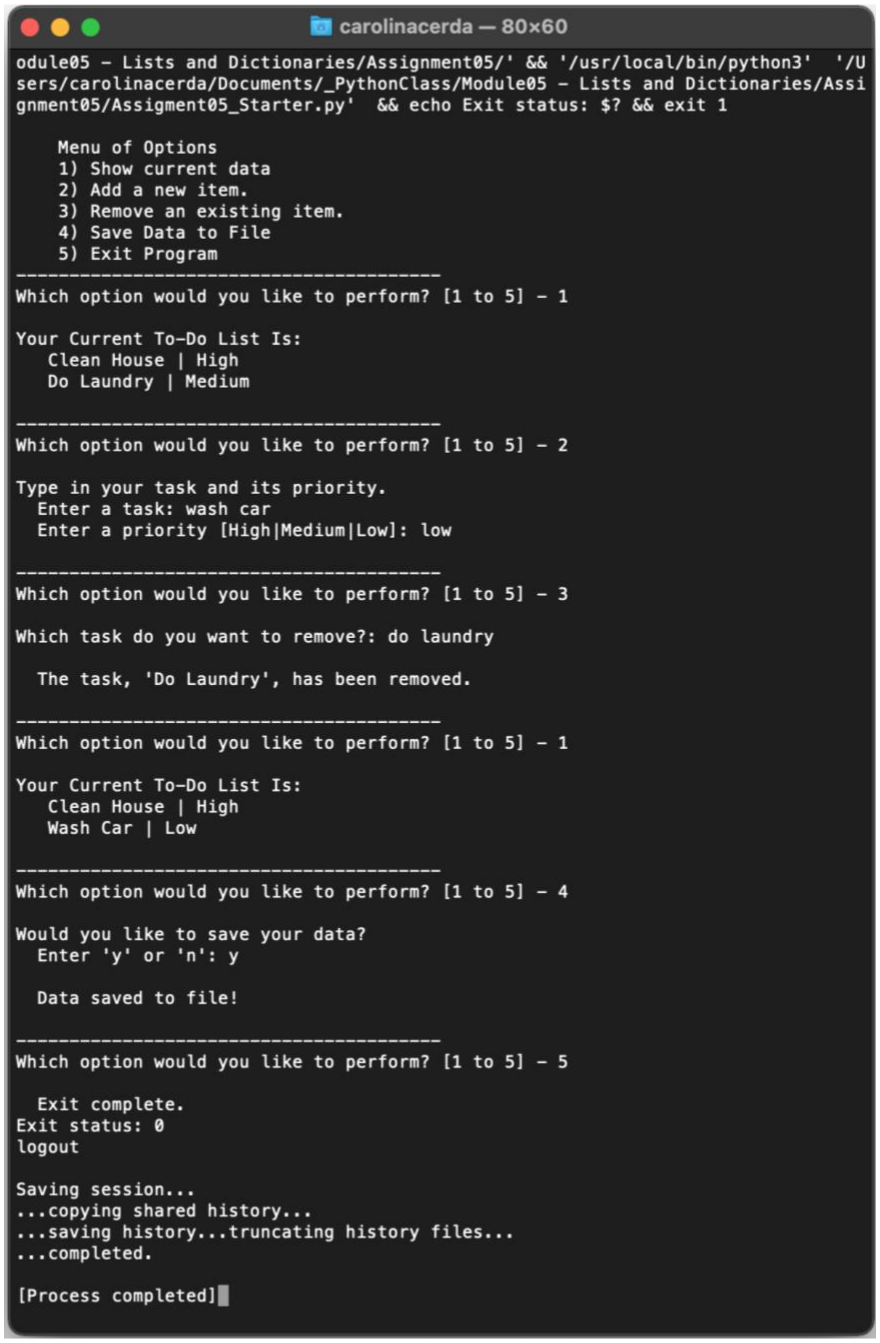

```
odule05 - Lists and Dictionaries/Assignment05/' && '/usr/local/bin/python3'  '/U
sers/carolinacerda/Documents/_PythonClass/Module05 - Lists and Dictionaries/Assi
gnment05/Assigment05_Starter.py'  && echo Exit status: $? && exit 1

    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] - 1

Your Current To-Do List Is:
   Clean House | High
   Do Laundry | Medium

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

Type in your task and its priority.
  Enter a task: wash car
  Enter a priority [High|Medium|Low]: low

----------------------------------------
Which option would you like to perform? [1 to 5] - 3

Which task do you want to remove?: do laundry

  The task, 'Do Laundry', has been removed.

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

Your Current To-Do List Is:
   Clean House | High
   Wash Car | Low

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

Would you like to save your data?
  Enter 'y' or 'n': y

  Data saved to file!

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

  Exit complete.
Exit status: 0
logout

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Figure 6. Final result of script in OS Command.

Finally, a verification of the success of the script is done by confirming that the data inputted by the user has successfully been saved within the text file, “ToDoList.txt” (Figure 7).

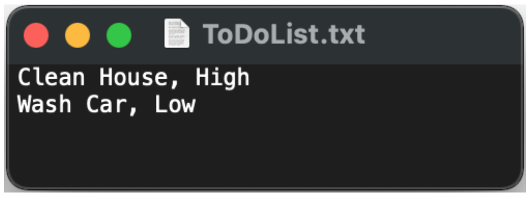

```
Clean House, High
Wash Car, Low
```

Figure 7. File used in script to save user input data.

Summary

This exercise consisted of using text files, dictionaries and lists in order to create a script that manages an interactive To-Do List. Furthermore, GitHub was introduced in order to provide familiarity to the software for potential future use as well as be able to provide feedback to our peers. The purpose of this investigation was to use lists and dictionaries in order to store collections of data in memory and text files as well as to find ways to further improve our scripts and learn how to use GitHub as a tool for file sharing and storage.