

# Programming with Python: Module 5

---

## Introduction

In this module we learn about using collections of data in lists and dictionaries, improving our scripts, and the Github repository. The tables and lab solutions are located at the end of the Step 1 section.

## Step 1 – Watch the video for Module0x

### 1) Lists

- a. **Lists: methods and functions** (Tables 1 and 2)
  - source [Python - Lists - Tutorialspoint](#) (external site) and [Python List extend\(\) Method \(w3schools.com\)](#) (external site)
- b. **Storing list data in a file**
  - open text files to write to (in “w” mode); remember to close files
  - use an intermediary object file
- c. **Loading list data from a file**
  - open text files to read from (in “r” mode); remember to close files
  - use an intermediary object file
  - load data to memory and then use the data from there, i.e., you do not use the data directly from memory
  - `split(“,”)` method: used to split the content of each line at the “,”
    - `strip()` method: used to strip spaces (e.g., the `\n` that was added in the module)

### 2) Dictionaries:

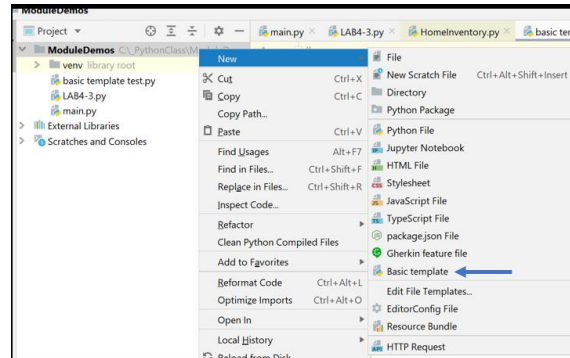
- a. Definition: “A dictionary is a collection which is ordered\*, changeable and does not allow duplicates. Dictionaries are used to store data values in key: value pairs.”
  - arrays (collections) in Python: list, tuple, set, and dictionary

- sets are enclosed in curly brackets and are used to store multiple items in a single variable. The items in a set are unordered, unchangeable, and may not have duplicate values.
- source [Python Dictionaries \(w3schools.com\)](https://www.w3schools.com/python/python_dictionaries.asp) (external site) and [Python Sets \(w3schools.com\)](https://www.w3schools.com/python/python_sets.asp) (external site)
- a. Subscripts: in some languages you may use either the key or the index number; in Python you can only use the key
- b. Working with dictionaries
  - **a table is a list of dictionaries:** a dictionary would correspond to each row of the table (the key-value pair corresponds to the column heading and the value for each entry) and the collection of these dictionary rows constitutes a list (that represents the data in the entire table)
  - **dictionary views:** `.keys()`, `.values()`, `.items()`
    - class of dictionary views are: `dict_keys`, `dict_values`, and `dict_items` respectively
  - **methods:** `.update({key_x: new_value})`, `.pop('key_x')`, `.popitem()`, `.clear()`, `.copy()`, `.get()`, `.setdefault()`, `.fromkeys()`
  - **functions:** `del dict_name['key_x']`, `del dict_name`
  - **iterate:** through items with for loop
  - **unpacking:** key-value pairs can be unpacked with `.items()` method using a for loop
- source [Python Dictionaries \(w3schools.com\)](https://www.w3schools.com/python/python_dictionaries.asp) (external site)

### 3) Improving your script

- a. **Separation of concerns**
  - “In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, so that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program. Separation of concerns results in more degrees of freedom for some aspect of the program's design, deployment, or usage. Common among these is increased freedom for simplification and maintenance of code. When concerns are well-separated, there are more opportunities for module upgrade, reuse, and independent development.” [Separation of concerns - Wikipedia](https://en.wikipedia.org/wiki/Separation_of_concerns) (external site)
  - three sections of code: data, processing, and presentation (input/output). Exact separation is not always possible
  - in some programming languages you are obliged to declare and initialize the value; Python is more forgiving
- b. **Functions:** define and call a function. Functions help you to keep concerns separate (specifically presentation and processing) and to reuse your code.

- c. **Script templates:** I saved a script template (script header and the three sections of data, processing, and presentation) in PyCharm. I named it 'Basic template' and it can be accessed in the Project Explorer tree as New > 'Basic template' (Figure 1) .



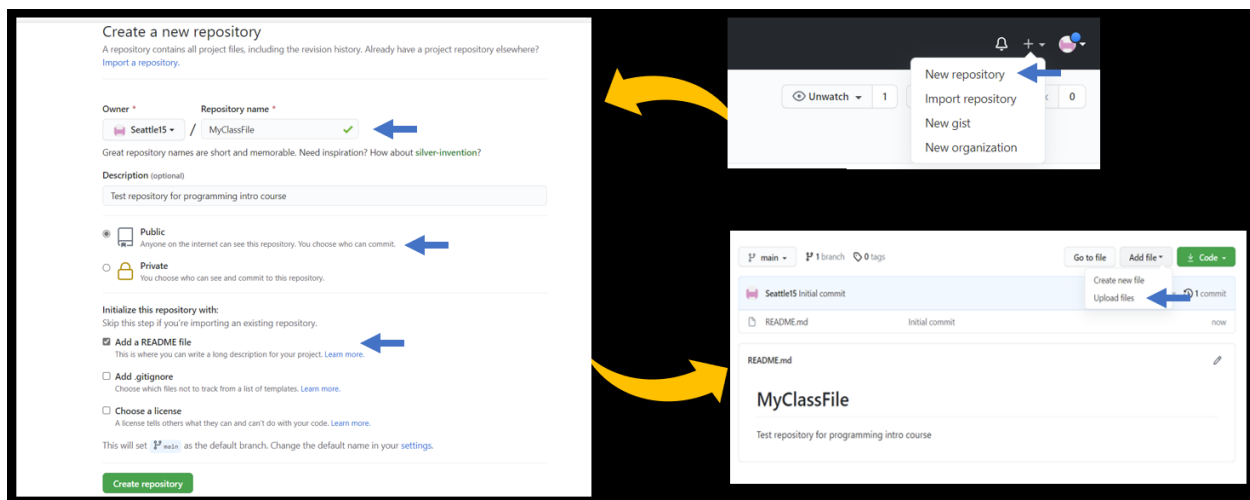
**Figure 1.** My script template can be accessed in the Project Explorer tree as New > Basic template

- d. **Error handling (try-except):** error handling improves scripts by managing errors. In programming languages, the error messages that pop up are meant for the developer and are too technical for the end user. A try-except construct can be used to trap errors and customize the error messages (i.e., structured error handling).
- when an error (or exception) occurs, Python will stop and generate an error message. The programmer can test a block of code with the *try block* and if an exception is detected it can be handled with the except block. A try block may be followed by one or more except blocks. Optional additional blocks are
    - *else block*: used to define a block of code that is executed if no errors are raised
    - *finally block*: used to define a block of code that is executed whether or not the try block raises an error
- source [Python Try Except \(w3schools.com\)](https://www.w3schools.com/python/python_try_except.asp) (external site)

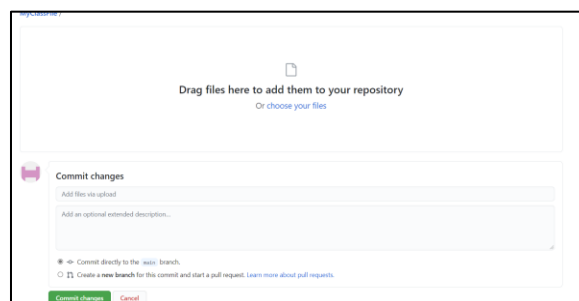
#### 4) Github and Git: Git is a source control software that is used to store code on the internet.

- a. Source control applications manage (and store) versions of your files; originally these were stored in the organization's internal servers.
- b. "GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project." [GitHub - Wikipedia](https://en.wikipedia.org/wiki/GitHub) (external site)
- c. Github tutorial: reviews how to "create and use a repository, start and manage a new branch, make changes to a file and push them to GitHub as commits, and open and merge a pull request." [Hello World · GitHub Guides](https://guides.github.com/) (external site)

- repository: typically contains folders and files, images, videos, spreadsheets, and data sets and is used to organize a single project. Use a *README* or a file containing information about the project. Steps in creating a repository and uploading files to it are demonstrated in Figures 2 and 3
  - branching: used to work on different versions of a repository at the same time
  - commit: saved changes are called commits
  - pull requests: used to propose changes to collaborators and to request review and merge of changes into their branch
- source [Hello World · GitHub Guides](#) (external site)



**Figure 2.** Demonstrates the sequential steps in creating a repository in Github and uploading files on it.



**Figure 3.** Interface for uploading files on Github. Pdfs and Python files can be viewed on Github. Python files may be edited on Github, and all versions will be saved on Github.

**Table 1. Lists: methods and functions**

Methods	
.append()	Adds an item to the end of the list
.remove()	Counts number of occurrences of an element
.insert(i, x)	Inserts an element(x) into a specific location(i)
.clear()	Removes all items
.pop([i])	Removes item at position i; removes last item if no index is specified
.count()	Counts number of occurrences of an element
.index()	Returns the index of the first time the element appears in list
.sort()	Sorts the items in the list; no return value (print list to see changes)
.reverse()	Reverses the order of the items in the list; no return value (print list to see changes)
.copy()	Returns a copy of the list
.extend()	Adds the elements of a list to the end of the current list
Functions	
len()	Returns number of elements in list
sorted()	Sorts the list and generates a new list
list()	Converts specific objects into lists
range()	Creates a range object that is converted to a list using the list() function; print(list(range(9)))
zip()	Takes 2 or more lists as inputs and returns an object that is converted to a list using the list() function. The elements in the new list are tuples with one element from each of the input lists. zipped_list = zip(list_1, list_2); converted_list = list(zipped_list)
Slicing syntax	Extracts a portion of the list: list_name[start, end, jump]

Table 1 sources are [Python - Lists - Tutorialspoint](#) (external site) and [Python List extend\(\) Method \(w3schools.com\)](#) (external site)

**Table 2. Script to test the methods and functions for lists**

List method and functions: script	Running (spacing adjusted in print out to match the script line)
<pre> lst_bones = ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius'] print(lst_bones) print('append') lst_bones.append('metacarpal') print(lst_bones) print('insert') lst_bones.insert(0, 'metatarsal') print(lst_bones) print('remove') lst_bones.remove('femur') print(lst_bones) print('pop') lst_bones.pop(1) print(lst_bones) print('insert') lst_bones.insert(2, 'metacarpal') print(lst_bones) print('count') count_metacarpal = lst_bones.count('metacarpal') print(count_metacarpal) print('sort') lst_bones.sort() print(lst_bones) print('reverse') </pre>	<pre> ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius']  append ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius', 'metacarpal']  insert ['metatarsal', 'femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius', 'metacarpal']  remove ['metatarsal', 'tibia', 'fibula', 'ulna', 'humerus', 'radius', 'metacarpal']  pop ['metatarsal', 'fibula', 'ulna', 'humerus', 'radius', 'metacarpal'] insert ['metatarsal', 'fibula', 'metacarpal', 'ulna', 'humerus', 'radius', 'metacarpal']  count 2  sort ['metatarsal', 'fibula', 'metacarpal', 'ulna', 'humerus', 'radius', 'metacarpal']  reverse </pre>

<pre> lst_bones.reverse print(lst_bones) print('copy') lst_bones_copy = lst_bones.copy() print(lst_bones_copy) print('slice') print(lst_bones[0:7:2]) lst_bones = ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius'] lst_extremity = ['lower extremity'] * 3 + ['upper extremity'] * 3 print('lst_bones') print(lst_bones) print('lst_bones_copy') print(lst_bones_copy) print('lst_extremity') print(lst_extremity) print('zipped lst_extremity and lst_bones') bones_zipped = list(zip(lst_bones, lst_extremity)) print(bones_zipped)  print('extend') lst_bones.extend(lst_extremity) print(lst_bones)  print('index') print(lst_bones.index('tibia'))  print('clear') print(lst_bones_copy.clear()) </pre>	<pre> ['metatarsal', 'fibula', 'metacarpal', 'ulna', 'humerus', 'radius', 'metacarpal']  copy ['metatarsal', 'fibula', 'metacarpal', 'ulna', 'humerus', 'radius', 'metacarpal']  slice ['metatarsal', 'metacarpal', 'humerus', 'metacarpal']  lst_bones ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius'] lst_bones_copy ['metatarsal', 'fibula', 'metacarpal', 'ulna', 'humerus', 'radius', 'metacarpal'] lst_extremity ['lower extremity', 'lower extremity', 'lower extremity', 'upper extremity', 'upper extremity', 'upper extremity'] zipped lst_extremity and lst_bones [('femur', 'lower extremity'), ('tibia', 'lower extremity'), ('fibula', 'lower extremity'), ('ulna', 'upper extremity'), ('humerus', 'upper extremity'), ('radius', 'upper extremity')]  extend ['femur', 'tibia', 'fibula', 'ulna', 'humerus', 'radius', 'lower extremity', 'lower extremity', 'lower extremity', 'upper extremity', 'upper extremity', 'upper extremity'] index 1  clear None </pre>
---	---

## LAB 5-1. Working with files and lists

I wrote the code for each section and iteratively checked that it was working as expected. I added a new line in between each entry. In the version that I wrote the user put in the entries (lamp and end table). I used a string (item\_value) to write the item and its value to the file as I got an error message when I tried to write the lstRow to the file.

Looking at Mr.Root's solution, I realized that we could write the item and value to the file if we indexed each as an element of lstRow. And that the expectation was to add the two items in the code – not as user input. Another way, my script differed was that I wrote it to the file as a string with the "|" in between the item and its value; once I used a for loop to read and print each row no additional code was required. Hence, I wrote version 2 of the script (the modified part is pasted here); I still chose to have the user input the data.

LAB 5-1. Working with files and lists	
Script – version 1	Running the script
<pre> # ----- # # Title: Lab 5-1 # Description: Writing and Reading Data from a file # ChangeLog (Who,When,What): # RRoot,1.1.2030,Created started script # &lt;Hedy Khalatbari&gt;,&lt;07.25.2021&gt;, Added read/write to file code # ----- #  # Declare my variables strChoice = "" # User input lstRow = [] # list of data </pre>	<pre> Enter the item and their values to create or add to your home inventory.  Enter a Name: Lamp  Enter a Value: 30  Write or Read file data, then type 'Exit' to quit! </pre>

<pre> strFile = 'HomeInventory.txt' # data storage file objFile = None # file handle  # Get user Input while(True):     print("Write or Read file data, then type 'Exit' to quit!")     strChoice = input("Choose to [W]rite or [R]ead data: ")      # Process the data     if (strChoice.lower() == 'exit'): break     elif (strChoice.lower() == 'w'):         # List to File         print("\nEnter the item and their values to create or add to your home inventory.")         item = input("Enter a Name: ")         est_value = input("Enter a Value: ")         item_value = item + "   " + est_value + "\n"         objFile = open(strFile, 'a')         objFile.write(item_value) # appends the items and values to text file         objFile.close()      elif (strChoice.lower() == 'r'):         # File to List         print("\nItem   Value")         objFile = open(strFile, 'r')         for row in objFile:             print(row)         objFile.close()     else:         print('Please choose either W or R!') </pre>	<p>Choose to [W]rite or [R]ead data: w</p> <p>Enter the item and their values to create or add to your home inventory.</p> <p>Enter a Name: End Table</p> <p>Enter a Value: 60</p> <p>Write or Read file data, then type 'Exit' to quit!</p> <p>Choose to [W]rite or [R]ead data: r</p> <p>Item   Value</p> <p>Lamp   30</p> <p>End Table   60</p>
<p>Script – version 2 (did not paste the script header)</p>	
<pre> # Declare my variables strChoice = "" # User input lstRow = [] # list of data strFile = 'HomeInventory2.txt' # data storage file objFile = None # file handle  # Get user Input while(True):     print("Write or Read file data, then type 'Exit' to quit!")     strChoice = input("Choose to [W]rite or [R]ead data: ")      # Process the data     if (strChoice.lower() == 'exit'): break     elif (strChoice.lower() == 'w'):         # List to File         print("\nEnter the item and their values to create or add to your home inventory.")         item = input("Enter a Name: ")         est_value = input("Enter a Value: ")         lstRow = [item, est_value]         objFile = open(strFile, 'a')         objFile.write(lstRow[0] + " , " + lstRow[1] + "\n") # appends the items and values to text file         objFile.close()      elif (strChoice.lower() == 'r'):         # File to List         print("Item   Value")         objFile = open(strFile, 'r')         for row in objFile:             if row == "": continue </pre>	<p>Enter the item and their values to create or add to your home inventory.</p> <p>Enter a Name: lamp</p> <p>Enter a Value: 30</p> <p>Write or Read file data, then type 'Exit' to quit!</p> <p>Choose to [W]rite or [R]ead data: w</p> <p>Enter the item and their values to create or add to your home inventory.</p> <p>Enter a Name: end table</p> <p>Enter a Value: 60</p> <p>Write or Read file data, then type 'Exit' to quit!</p> <p>Choose to [W]rite or [R]ead data: r</p> <p>Item   Value</p> <p>lamp 30</p>

<pre> lstRow = row.split(",") print(lstRow[0] + " " + lstRow[1].strip()) objFile.close() else: print('Please choose either W or R!') </pre>	<pre> end table 60 </pre>
---	---------------------------

## LAB 5-2. Working with a table of dictionaries

I had a hard time going back and forth between the data formatted as a dictionary and then written to the file as a string. I made it unnecessarily complicated for myself – I was initially trying to write both the key and value of the dictionary items to the file, and in doing so changing tuples to strings ... I was successful in writing it to the file but could not figure out how to read it and put it back into a dictionary.

This was the first lab exercise, that I had to watch Mr.Root's solution to figure it out as I realized I was making in too complicated when it should not be so convoluted! My script (after getting guidelines from the instructor's video) is pasted below.

What I liked: The script prints out the results as a dictionary, which is a clean format. Once I watched Mr. Root's explanation of the script, I appreciated the simplicity of going between and list and a dictionary and the logic of only saving the values (and not the keys) in the file.

What could be improved: What could be improved is the presentation of the printed output, for example it could read as,

'The first household item is a lamp with an estimated value of \$30.

The second household item is an end table with an estimated value of \$60.'

Or the printed output could align the household items as one column and the values as a second column and print out:

Household item		Estimated value
Lamp		\$30
End table		\$60'

LAB 5-2. Working with a table of dictionaries	
Script	Running
<pre> # ----- # # Title: Listing 9 # Description: Writing and Reading Data from a file # ChangeLog (Who,When,What): # RRoot,1.1.2030,Created started script # &lt;Hedy Khalatbari&gt;,&lt;07.25.2021&gt;,Changed rows from lists to dictionaries # ----- #  # Declare my variables from typing import List, Any  strChoice = "" # User input dictRow = {} # dictionary of data for writing and reading lstRow = [] # list of data for reading </pre>	<pre> 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'} </pre>



```

strFile = 'HomeInventory.txt' # data storage file
objFile = None # file handle

# Get user Input
while(True):
    print("Write or Read file data, then type 'Exit' to quit!")
    strChoice = input("Choose to [W]rite or [R]ead data: ")

    # Process the data
    if (strChoice.lower() == 'exit'): break
    elif (strChoice.lower() == 'w'):
        # List to File
        item_value = ""
        objFile = open(strFile, "w")
        dictRow = {"item": "Lamp", "value": "$30"}
        objFile.write(dictRow["item"] + ", " + dictRow["value"] + "\n")
        dictRow = {"item": "End Table", "value": "$60"}
        objFile.write(dictRow["item"] + ", " + dictRow["value"] + "\n")
        objFile.close()
    elif (strChoice.lower() == 'r'):
        # File to List
        objFile = open(strFile, "r")
        for row in objFile:
            lstRow = row.split(",")
            dictRow = {"item": lstRow[0], "value": lstRow[1].strip()}
            print(dictRow)
        objFile.close()
    else:
        print('Please choose either W or R!')

```

## Step 2 - Read a book chapter

I read chapter five and the assigned webpage and watched the assigned webpage before watching the module 5 video.

- Lists work with sequences of information.
- Dictionaries work with pairs of data (the key-value pair); you can use one value (the key) to look up another value
- Working with lists: create, index, slice, add, delete, append, sort, and nested sequences
  - methods for adding and removing elements: .append(value), .remove(value), insert(i, value), pop([i])
  - methods for sorting, ordering, and counting: .sort(), sort(reverse=True), reverse(), count(value), index(value)
  - examples of functions: del list\_name[index], del list\_name[index\_1:index\_2]
  - nested sequences: sequences inside other sequences, e.g., lists containing lists or lists containing tuples
    - use multiple indexing to get directly to a nested element, e.g., list\_name[inde\_1][index\_2]
    - .sort() method sorts based on the first element in the nested tuple/list. For example, when the nested tuple is a pair of name/scores, save the score as the first element in the nested tuple so that you can use the sort method on the scores

- Unpacking a sequence: assign variable names directly to elements in a sequence with one line of code (requires knowing number of elements in sequence), e.g., `element_1 , element_2 = list_name`
- Lists are different from tuples as they are mutable:
  - assign a new list element by index: `list_name[index] = ['string_one']`
  - assign a new list slice: `list_name[index_1:index_2] = ['string_one', ...]`
  - delete an element: `del list_name[index]`
- When to use tuples
  - tuples are faster: speed difference is relevant in more complex applications
  - tuples are immutable: perfect for creating constants; adds safety and clarity to your code
  - tuples are required by Python: such as in creating some kinds of dictionaries
- Beware of shared references when using mutable values
- Working with dictionaries: create (enclosed by curly brackets), add and delete pairs of data
  - assign a new item or reassign an item: `dictionary_name[key_1] = "value"`
  - `del dictionary_name[key_1]`
  - keys are unique and immutable; the same is not true for values
- Retrieving values from a dictionary:
  - a) use key: `dictionary_name[key_1]`
  - b) test with the *in* operator -> use key: if `key_1 in dictionary_name:`  
`print(dictionary_name[key_1])`
  - c) `get()` method: `print(dictionary_name.get(key_1, [default_value]))`
- Retrieving all keys, values, and items (key-value) pairs from a dictionary with methods:
  - dictionary views: `keys()`, `values()`, and `items()` ; can use a for loop to iterate
- random module, `random.choice()` function

## Step 3 - Web pages

<https://www.afterhoursprogramming.com/tutorial/>

I read the "List", "Dictionary", and "Reading Files" pages from the tutorial; I scrolled through the tutorial to find these pages as I could not locate the menu option. It reinforced material that had already been covered.

- a dictionary has keys, which are like the indexes used in a list; the latter are numbers, whereas the keys do not necessarily have to be numbers

## Step 4 - Additional Videos

Reading from a text file: <https://youtu.be/m0o0CkYsDzl> (external site). I watched the video.

- `open()` function: used to both create a new file and open an existing file  
`new_file = open("new_file.text", "w")`
- `write()` and `read()` methods to write to and read from a file opened in write ("w") and read ("r") modes respectively
  - `readline()` method: reads one line at a time
  - `readline()[line_number]`: only reads that specific line number

- close() method to close file

## Step 5 - Apply your knowledge

Steps in writing the script:

- created a new sub-folder called Assignment05 inside the \_PythonClass folder
- created a new project in PyCharm that uses the \_PythonClass\Assignment05 folder as its location
- added the starter file, "Assignment05\_Starter.py," to my project
- updated the changelog in the script's header
- added code to the script to perform that assignment's task. I broke it into sections and iteratively ran each section to verify that it worked. And there were a lot of glitches before I got it right

My observations and mistakes and what I learnt from them:

- I had to remind myself to think of the table as a list of dictionaries consistently though out the different sections as at times it seemed simpler to handle the data as a list of strings or as nested lists
- local and global variables caused my problems. For example, I realized that I had to reset lstTable to an empty list otherwise I would get duplications of my entries in the printout (that was not reflected in the actual text file)
- I think my solution/ logic for removing a task was unnecessarily long – I look forward to a more stream-lined approach
- I did remember to use the .lower() and .strip() methods on the inputs and the latter when reading the file contents; I wonder if I could have achieved the same results with fewer lines of code
- functions would have come in handy when reading and writing from files and shortened the code
- I tried to add comments to share my thought process. I don't know if I have formatted the comments well and look forward to feedback on how I can have a cleaner code
- I did read and write the tasks to the file in each step – I wonder if Mr. Root meant for us to save inputs to a table (as the user in adding and deleting them) and only save it to the text file once the user chooses option 4 (Save Data to File)

Assignment05 script	Running in PyCharm
<pre># ----- # # Title: Assignment 05 # Description: Working with Dictionaries and Files #     When the program starts, load each "row" of data #     in "ToDoList.txt" into a python Dictionary. #     Add each dictionary "row" to a python list "table" # ChangeLog (Who,When,What): # RRoot,1.1.2030,Created started script # &lt;Hedy Khalatbari&gt;,&lt;07.26.2021&gt;,Added code to complete assignment 5 # ----- #  # -- Data -- # # declare variables and constants strFile = "ToDoList.txt" # Data storage file objFile = None # An object that represents a file</pre>	<pre>C:\Python39\python.exe C:/_PythonClass/Assignment05/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  Your current to do list in order of priority is: {'priority': 'high', 'task': 'make a dentist appointment'} {'priority': 'high', 'task': 'take puppy to vet'}</pre>

<pre> strData = "" # A row of text data from the file dicRow = {} # A row of data separated into elements of a dictionary; {Priority, Task} lstTable = [] # A list that acts as a 'table' of dictionary rows strMenu = "" # A menu of user options strChoice = "" # A Capture the user option selection  # -- Processing -- # # Step 1 - When the program starts, load any data you have # in a text file called ToDoList.txt from a Python list of dictionaries rows # Step 1A - Add all data as dicRow to lstTable dicRow = {"priority": "high", "task": "take puppy to vet"} lstTable.append(dicRow) dicRow = {"priority": "low", "task": "schedule gutter cleaning"} lstTable.append(dicRow) dicRow = {"priority": "low", "task": "find a new gardener"} lstTable.append(dicRow) dicRow = {"priority": "high", "task": "make a dentist appointment"} lstTable.append(dicRow) # Step 1B - Write the value of the dictionary elements to the text file objFile = open(strFile, "w") for row in lstTable:     objFile.write(row["priority"] + ", " + row["task"] + "\n") objFile.close()  # -- Input/Output -- # # Step 2 - Display a menu of choices to the user 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 presentation      # Step 3 - Show the current items in the table in order of priority (high     versus low)     if (strChoice.strip() == '1'):         lstTable = [] # reset to empty list to ensure we do not get duplicate         entries         objFile = open(strFile, "r")         for row in objFile:             strData = row.split(",")             dicRow = {"priority": strData[0], "task": strData[1].strip()}             # high priority tasks are added to the beginning of the table             if strData[0] == 'high':                 lstTable.insert(0, dicRow)             # low priority tasks are added to the end of the table             elif strData[0] == 'low':                 lstTable.append(dicRow)         # print each dictionary on a separate row         print("Your current to do list in order of priority is: ")         for item in lstTable:             print(item)         objFile.close()         continue      # Step 4 - Add a new item and write it to the text file     elif (strChoice.strip() == '2'):         # ask for task input and perform lower and strip methods on input         task = input("Enter task: ") </pre>	<pre> {'priority': 'low', 'task': 'schedule gutter cleaning'} {'priority': 'low', 'task': 'find a new gardener'}  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  Enter task: cleaning Enter priority (high or low): high Task added to to-do 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] - 2  Enter task: mowing Enter priority (high or low): low Task added to to-do 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] - 2  Enter task: carpet cleaning Enter priority (high or low): loo Invalid choice. Please choose from menu  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 task would you like to remove? cleaning The task was on your to-do list and is now deleted  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 task would you like to remove? take puppy to vet The task was on your to-do list and is now deleted </pre>
--	--

<pre> task = task.lower() task = task.strip() # ask for priority input and perform lower and strip methods on input priority = input("Enter priority (high or low): ") priority = priority.lower() priority = priority.strip() # check for correct user input of priority as either high or low if priority == 'high' or priority == 'low':     # add inputs to a dictionary     dicRow = {"priority": priority, "task": task}     # write inputs to text file in append mode     objFile = open(strFile, "a")     objFile.write(dicRow["priority"] + ", " + dicRow["task"] + "\n")     objFile.close()     print("Task added to to-do list") else:     print("Invalid choice. Please choose from menu") continue  # Step 5 - Remove an item from the list/Table elif (strChoice.strip() == '3'):     priorities = [] # local list variable     tasks = [] # local list variable     # ask user for input and perform lower and strip methods on input     task_remove = input("Which task would you like to remove? ")     task_remove = task_remove.lower()     task_remove = task_remove.strip()     # open file to read each row and append contents to tasks and priorities     lists     objFile = open(strFile, "r")     for row in objFile:         strData = row.split(",")         priority = strData[0]         priorities.append(priority)         task = strData[1]         task = task.strip() # strip the \n         tasks.append(task)     objFile.close()     # check if task_remove is in tasks list and if present remove it &amp;     # delete the corresponding priority from priorities list     if task_remove in tasks:         removed_index = tasks.index(task_remove) # check for the index         tasks.remove(task_remove)         del priorities[removed_index] # remove the same index from priorities     list     print("The task was on your to-do list and is now deleted")     # otherwise let user know that the task was not in the task list     else:         print("The task was not on your to-do list")     # update the text file with remaining priority-task pairs     lstTable = [] # reset to empty list to ensure we do not get duplicate     entries     counter = 0     for i in tasks:         dicRow = {"priority": priorities[counter], "task": tasks[counter]}         counter += 1         lstTable.append(dicRow)     objFile = open(strFile, "w")     for row in lstTable:         objFile.write(row["priority"] + ", " + row["task"] + "\n")     objFile.close()     continue  # Step 6 - Save tasks to ToDoList.txt file # The data is saved to file with each step, therefore no action is required here - </pre>	<p>Menu of Options</p> <ol style="list-style-type: none"> <li>1) Show current data</li> <li>2) Add a new item</li> <li>3) Remove an existing item</li> <li>4) Save Data to File</li> <li>5) Exit Program</li> </ol> <p>Which option would you like to perform? [1 to 5] - 1</p> <p>Your current to do list in order of priority is:</p> <pre>{'priority': 'high', 'task': 'make a dentist appointment'}</pre> <pre>{'priority': 'low', 'task': 'schedule gutter cleaning'}</pre> <pre>{'priority': 'low', 'task': 'find a new gardener'}</pre> <pre>{'priority': 'low', 'task': 'mowing'}</pre> <p>Menu of Options</p> <ol style="list-style-type: none"> <li>1) Show current data</li> <li>2) Add a new item</li> <li>3) Remove an existing item</li> <li>4) Save Data to File</li> <li>5) Exit Program</li> </ol> <p>Which option would you like to perform? [1 to 5] - 3</p> <p>Which task would you like to remove? mowing</p> <p>The task was on your to-do list and is now deleted</p> <p>Menu of Options</p> <ol style="list-style-type: none"> <li>1) Show current data</li> <li>2) Add a new item</li> <li>3) Remove an existing item</li> <li>4) Save Data to File</li> <li>5) Exit Program</li> </ol> <p>Which option would you like to perform? [1 to 5] - 1</p> <p>Your current to do list in order of priority is:</p> <pre>{'priority': 'high', 'task': 'make a dentist appointment'}</pre> <pre>{'priority': 'low', 'task': 'schedule gutter cleaning'}</pre> <pre>{'priority': 'low', 'task': 'find a new gardener'}</pre> <p>Menu of Options</p> <ol style="list-style-type: none"> <li>1) Show current data</li> <li>2) Add a new item</li> <li>3) Remove an existing item</li> <li>4) Save Data to File</li> <li>5) Exit Program</li> </ol> <p>Which option would you like to perform? [1 to 5] - 4</p> <p>Data saved to file</p> <p>Menu of Options</p> <ol style="list-style-type: none"> <li>1) Show current data</li> <li>2) Add a new item</li> <li>3) Remove an existing item</li> <li>4) Save Data to File</li> <li>5) Exit Program</li> </ol> <p>Which option would you like to perform? [1 to 5] - 5</p> <p>Press the ENTER key to exit</p> <p>Process finished with exit code 0</p>
---	---

```

# other than a message letting the user know the data is saved
elif (strChoice.strip() == '4'):
    print("Data saved to file")
    continue

# Step 7 - Exit program
elif (strChoice.strip() == '5'):
    print("Press the ENTER key to exit")
    input()
    break # and Exit the program

```

```

C:\Users\hedik>python.exe "C:\_PythonClass\Assignment05\Assignment05_Starter.py"
Microsoft Windows [Version 10.0.19042.1083]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hedik>python.exe "C:\_PythonClass\Assignment05\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

Your current to do list in order of priority is:
{'priority': 'high', 'task': 'make a dentist appointment'}
{'priority': 'high', 'task': 'take puppy to vet'}
{'priority': 'low', 'task': 'schedule gutter cleaning'}
{'priority': 'low', 'task': 'find a new gardener'}

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

Enter task: cleaning
Enter priority (high or low): high
Task added to to-do 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

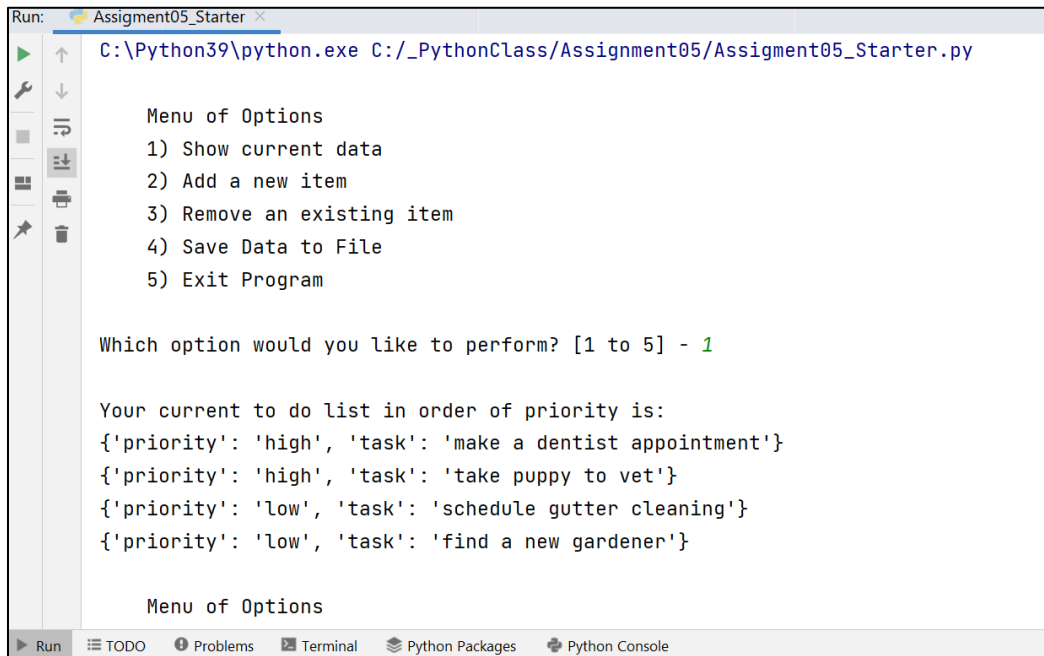
Which option would you like to perform? [1 to 5] - 3

Which task would you like to remove? find a new gardener
The task was on your to-do list and is now deleted

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

```

**Figure 4.** Screen capture of assignment script running in Command console.



```
Run: Assignment05_Starter x
C:\Python39\python.exe C:/_PythonClass/Assignment05/Assignment05_Starter.py

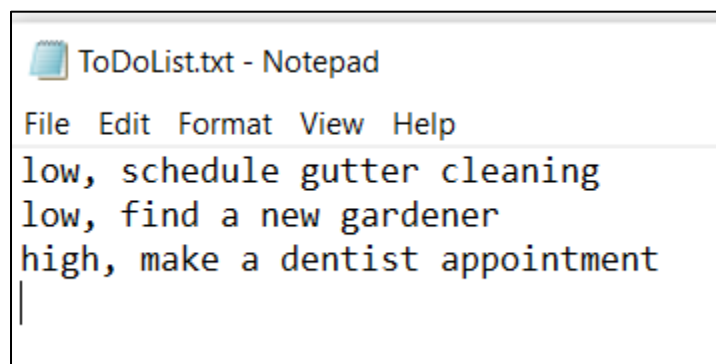
Menu of Options
1) Show current data
2) Add a new item
3) Remove an existing item
4) Save Data to File
5) Exit Program

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

Your current to do list in order of priority is:
{'priority': 'high', 'task': 'make a dentist appointment'}
{'priority': 'high', 'task': 'take puppy to vet'}
{'priority': 'low', 'task': 'schedule gutter cleaning'}
{'priority': 'low', 'task': 'find a new gardener'}

Menu of Options
```

**Figure 5.** Screen capture of assignment script running in PyCharm.



```
ToDoList.txt - Notepad
File Edit Format View Help
low, schedule gutter cleaning
low, find a new gardener
high, make a dentist appointment
```

**Figure 6.** Screen capture of text file verifying that the code performs as expected.

## Step 6 - Document your knowledge

I documented my learning process as I went through the steps in the assignment in this Word document.

## Step 7 - Post your Files to GitHub

I posted my files (assignment word document and Python file) on a public GitHub repository.  
GitHubURL: <https://github.com/Seattle15/IntroToProg-Python>

I watched the assigned videos and summarized my learning and actions as follows:

- GitHub Tutorial 2020 - Beginner's Training Guide: <https://youtu.be/iv8rSLsi1xo>
  - downloaded Github desktop
- How to use GitHub for Beginners: <https://youtu.be/E8TXME3bzNs>
  - a repository is essentially a project
  - work on your desktop version of Git and commit to the remote repository periodically
  - do not upload private/research projects on a public account in Github
  - conflicts in scripts and how to manage it
  - commands: git add, git commit, git push, git status
- Setting up a GitHub Account: [https://youtu.be/Sk1\\_DU2ky48](https://youtu.be/Sk1_DU2ky48)
  - tutorial on setting up a GitHub account
  - how to share a link to a file in repository
- I also watched the first half of [Git Tutorial for Beginners - Git & GitHub Fundamentals In Depth - YouTube](#)
  - nice overview of main and branch files, pulling and committing
  - covered downloading Git on your computer and initiating a local repository

## Step 8 - Post a Link to GitHub

I will share the link (pasted below and in the assignment header) to Github once the Canvas discussion board for this module opens. GitHubURL: <https://github.com/Seattle15/IntroToProg-Python>

## Step 9 - Submit your work

I submitted the assignment of Canvas and posted the GitHub link there as well.

## Step 10 - Perform a Peer Review (Not Graded!)

I will perform this step in week 5 of this course.

## Summary

In this module we learn about working with lists and dictionaries, improving our scripts, and creating a Github repository. I initially reviewed the material on lists and dictionaries in the Code Academy Python 3 modules. I then read the assigned book chapter and web page, watched the assigned webpage, and wrapped it up by watching the video for Module 5. I finished steps 1-9 and submitted my work on July



26<sup>th</sup>; I will have to remember to perform Step 10 in week 5 of this course. I learned a lot in this module – writing each section of code was a fulfilling journey.