Working with Dictionaries

## Introduction

In this document, I will cover some of the different ways one can work with dictionaries. In previous modules we have reviewed strings, tuples, and lists. Dictionaries allow even more ways to manipulate and store data within Python. A file can be written to a dictionary. Data can be added and removed from dictionaries. This document will cover dictionaries in more depth and will also demonstrate how they can be used and manipulated to complete Assignment 05.

## Dictionaries

Dictionaries are another way to store data. The advantage of a dictionary is it has even more functions and abilities than a list. As with a list, a dictionary is just rows of data. But like with a dictionary there is a key and a value.

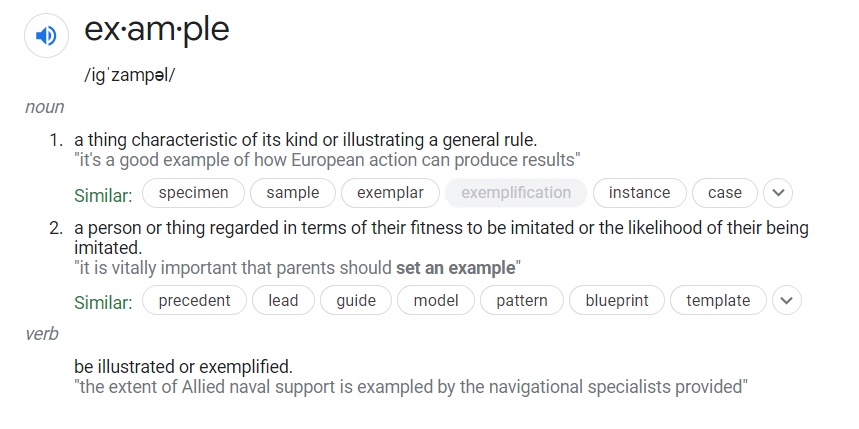


Figure 1. Example of a dictionary definition of what the word example means. Example is the key and the definitions are the values for the key. (external link: [Google Search](https://en.wikipedia.org/wiki/Google_Search))

Tuples are defined using () brackets, lists are defined using [] and dictionaries defined using {} brackets. When writing a dictionary row, you have the opportunity to name the components of the row which can make it easier to recall the data later. Below are some examples of how you can write inputs to a dictionary. A single dictionary row with no titles for the key and value is the first option. The second option is to name the key and value.

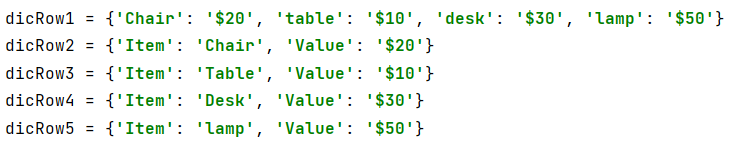


Figure 2. Examples of different ways to enter data into a dictionary. One can use a single row or multiple rows when creating a dictionary. The second dictionary row method allows for the keys and values to be named.

For a list you have to use indexes to call a specific piece of data in a row. If one names the dictionary key and values, then it allows the coder to call the specific pieces of data in that dictionary row by name instead of index. When working with a large amount of data, it can be helpful to call the data by name instead of index.

Once a dictionary has been created, there are many already built in functions in Python to allow viewing and manipulating the dictionary. See below for some examples from [W3Schools](https://www.w3schools.com/python/python_ref_dictionary.asp) (external link).

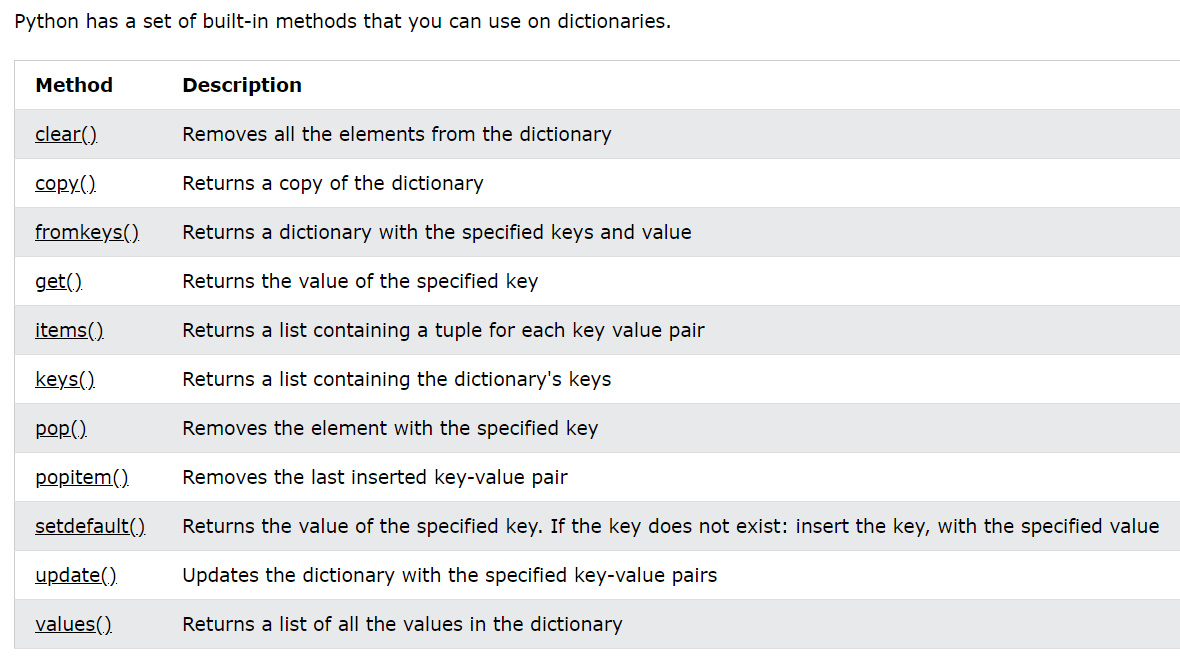


Figure 3. Examples of what built in functions are available for dictionaries in Python from W3Schools. (External Link: [W3Schools](https://www.w3schools.com/python/python_ref_dictionary.asp))

## Starting the Assignment

For Assignment 05, the goal was to use a dictionary to do various tasks. The first step of the assignment is to transpose the information in a text file into a dictionary. Once the data is in the dictionary the next part of the program is to display the information in the dictionary to the user. After the user knows what is in the dictionary, allow the user to enter or remove items/rows from the dictionary. When the user is done changing the dictionary, write the information to a text file. Finally, once the user is done, allow them to exit the program.

In the following sections, I will go in depth about how I was able to work with dictionaries for Assignment 05.

### Writing a File to a Dictionary

To do anything with a file in Python, we first need to open the file. Since we are only trying to get data out of the file and add it to our dictionary, we only need to be able to read the file. In this example, the text file is called “ToDoList.txt”. I’m going to ask Python to open the file in read mode.



Figure 4. How to open a text file in read mode in Python. Text file is called “ToDoList.txt”.

I’m assuming that the text file is going to have more than one item in the list. Since I need Python to continue to keep looking for new rows in the text file, we need to use a for loop. For all the rows in the text file perform these actions.

I’m also going to assume, since it is common practice, that there is a comma that separates the task and priority in each row of the text file. Since there is a comma separating the task and priority, I can use the comma as a signal to split the line/row of data.



Figure 5. Starting example of setting up a for loop to split all the rows of data in the file.

Finally, we need to write that row of information into the dictionary. Each row of data is going to have two parts: a task and a priority. In the dictionary, we want to make sure that we can easily see what a task is and what is a priority. To do this we are going to name the first index of the row of data: task and the second index priority. Then we are going to write that to a dictionary row.

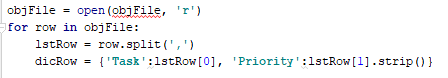


Figure 6. Example of writing a row of data in a text file to a dictionary row.

Note that in the above example, the second index (priority), has a strip at the end of the input. I added this strip to remove any extra spaces after the priority in the file. This strip just cleans up the data to make it easier to manipulate.

The final step to write the row of data from the text file to the dictionary, is to append the dictionary to a “table” of dictionary rows.

To check to make sure that the data is stored how we want it to, I can print out the lines of my dictionary to see what the output will be. There are several ways to print out the dictionary values. I can print the task and priority, or I can print the whole dictionary list or a mixture depending on what I want to see.

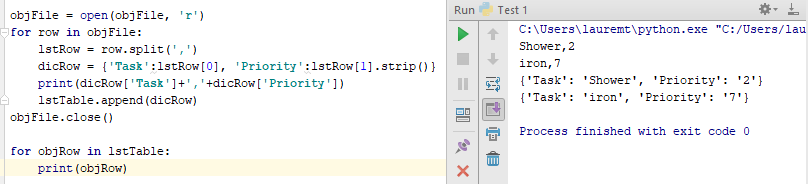


Figure 7. Examples of how to print dictionary rows with different information depending on what a user wants to see.

### Displaying Dictionary Information

As a part of the last section, I touched on printing the dictionary rows so we could test to make sure the code was working. In this section, I’ll go into more detail about how to print different parts of the dictionary list.

Since I’m assuming that the list is going to have more than one row of data, I know I’m going to need a for loop to go through all the rows or lists of data. I’m going to go over two options. The first option is to print the entire dictionary list/row. To do this as I’m looping through all the rows, I’m just going to simply print the row.

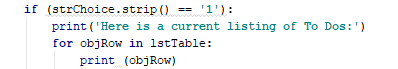


Figure 8. This shows how to print the entire dictionary row with all the brackets and names for each of the dictionary items.

The second option is to print just the task and priority without the additional notes and brackets. To accomplish that, I’m going to have to call out the task in the dictionary row and priority in the dictionary row. See the below figure for the code to print a simplified dictionary row.



Figure 9. Example of the code to print the simplified dictionary row with just the task and priority split with a comma.

To see the difference in outcomes from the simple to the full dictionary row see the below example.

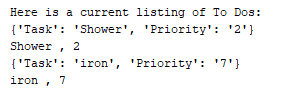


Figure 10. This is an example of both print outputs. The top line is the full dictionary line, and then the second line is the simplified print line that just shows the task and priority with no brackets or quotes.

### Adding information to a Dictionary

Now that I can see what is in the dictionary, the next step will be adding new rows of data to my dictionary. I know that I’m going to need to ask the user for inputs, a task and a priority for that task. From previous documentation we have already gone over how to do this. The twist for this section is how to add it to the dictionary.

As with a list I’m going to add a row of data to my dictionary. To define a dictionary, I know I’m going to need to use curly brackets ({}) to let Python store it as a row of dictionary data and not a list data. Once I have the data in the dictionary row, I now need to add the row of data to the larger dictionary. To add the row, I use the append function to add the row of data to the existing dictionary.

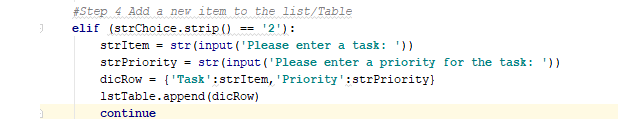


Figure 11. Example of script used to ask the user for inputs for task and priority. Then the script creates a dictionary row (dicRow) and then appends it to the main dictionary called lstTable.

With a row added to the dictionary, how do I check that the row was added to the list? PRINT it of course. In the previous section we learned how to print the dictionary row. To test the script, I’m going to add laundry with a high priority.

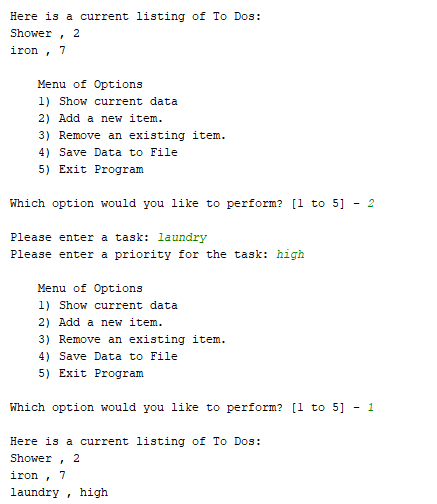


Figure 12. Final output from the script to add a dictionary row to the main dictionary. This output proves the script works to add a row. Print when in doubt.

### Deleting Information from a Dictionary

With data in the dictionary, how do we remove a row of data? First, I needed to ask the user for what task or priority they wanted me to remove from the list. Then I need to loop through all the rows of data to check that the input isn’t in one of the rows. To accomplish this, I’m going to need an if statement to check to see if the input is in a row.

One part that was tricky was trying to figure out how to call each of the dictionary terms. When we wrote the dictionary, we wrote it as a list of dictionary rows. Each line was a dictionary, and we had a table of dictionary rows.

After some research, I found a site to help figure out the script needed [geegsforgeeks.org](https://www.geeksforgeeks.org/python-removing-dictionary-from-list-of-dictionaries/) (external link). The website went over how to call a dictionary in a list of dictionaries and break apart each dictionary row to search it for the user input. I copied the code below and adapted it for the variables I had in my script.

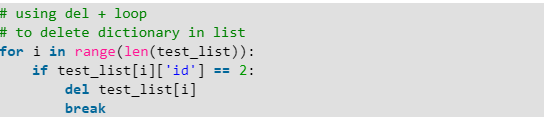


Figure13. Example of the script used to check if an input is in a dictionary list.

I was able to get my script to work with the script from geekstogeeks.org, but thought there must be another way to make it work. The script from geektogeeks.org didn’t make sense to me, but I knew it worked after trying it. After watching the course zoom video, I figured out there was another way to write the script to remove rows.

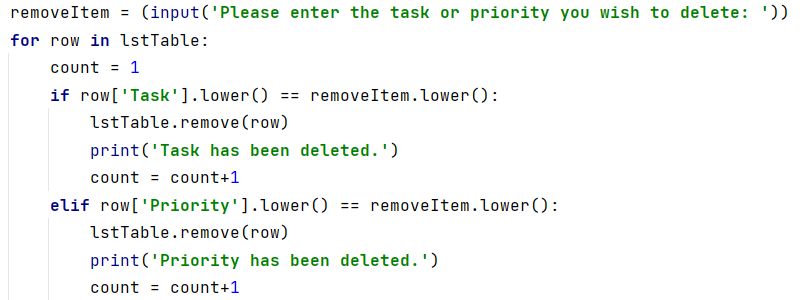


Figure 14. Example of code to remove a row based on if the input from the user matched the task or priority.

Once I find a matching task or priority, I need to remove it and let the user know what happened. Did a row get deleted or not? If a task or priority is found and deleted, I can write a print statement to let the user know. But what happens when the input isn’t found in the dictionary?

If I add it to the end of my if loop looking for the input, it is always going to print if it isn’t in that row of data. To prevent many lines of unnecessary notifications, I devised a plan to use a counter. If the input was found in the row, the count would go up by 1. Then once the loops were completed and data may be deleted, I could add a second if statement to determine the value of the count variable. If the count was greater than the initial 1 it started with, then I knew lines of code had been deleted.

If the count was greater than 1, I could print a line of text to let the user know the input wasn’t in the list. Now the user could try again to remove a different task or priority.

### Writing New List to Text File

Now the user can see what is on the To Do List, add a To Do item to the list, and remove a To Do item from the list. The next step will be to write the dictionary list back to a text file so the user can exit the program but not lose any of the updates they did to the list.

To start with I will need to open the file that I want to write the data to. Since we imported the list from the text file, I don’t want to append the list, I want to write the new list back to the file. For this reason I open up the text file in write mode not append mode.

As with the other steps in the script, I know I’m going to need a loop to write each of the rows to the file. Also to make the output text file easy to read, I know I just want to write the task and priority separated by a comma. Then I’m going to have to write each row of the dictionary to the file. Once all the rows of data have been written to the file, I need to close the file. As feedback to the user, I added a final print line to let them know the file was saved.

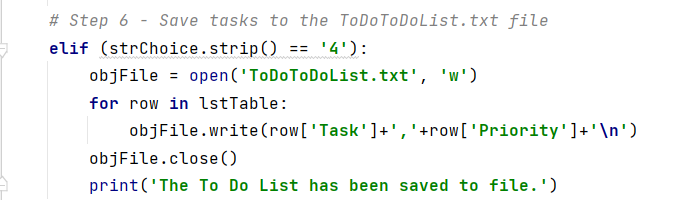


Figure 15. Script to write the dictionary rows back to a text file.

### Exiting The Script

With the data updated and saved to a file, the final step would be to exit the script. I wanted to add a final check for the user to verify they actually want to exit and didn’t type in the wrong input. Once the user types “y” to exit, the loop breaks and the script ends. I added a final line of print to let the user know the program ended.

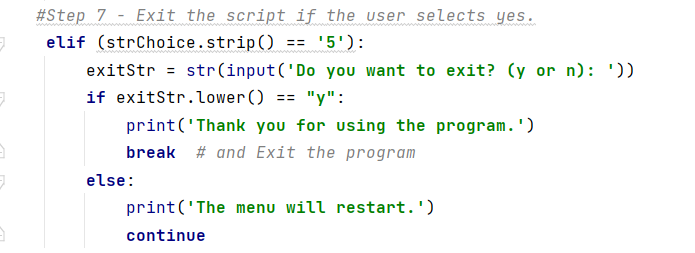


Figure 16. This figure is an example of how to write a script to ask the user if they really want to exit by asking for a user input again. If the user types “Y” then the program will end. If the user doesn’t want to exit, the menu will repeat and they can choose a different menu option.

### 

### Testing the Script

The final step to writing a code is testing it to make sure it works and different user inputs don’t cause errors. To make the code easier to write, I broke the code down by step. Then I could focus on one step and get it working before moving on to the next step.

With all the different parts of the script working individually, it was time to put it all together. When I tested the code, I tried different strings for all the input to verify the code acted the way it should. This meant it would be hard for the user’s input to throw an error in the script.

The final test was looping through all the different menu options and going back and forth between each of them to make sure the program acted as it should. One place I struggled was in the removal step. A few times I ran it, Python threw some errors out. The errors were mostly due to issues in the if statements. After I stepped through the code, I was able to find the issues and restructure my if statements to work.

Working with the count variable to determine if rows were deleted also took some testing to make sure it was written correctly. I started with the count variable starting at 1 inside the if statement, which caused issues, but then I remembered to remove it out of the loop.

Once the removal step was working, I looped the program through all the menu options in order and then out of order, repeated steps, and tried lots of different removal sequences to double check all the code was working. I created a second file, “ToDoToDoList.txt”, to write the final list from the program to, so that I could double check how the program was working.

As a last addition to my script, I added a try/except loop for reading the file. I added this because if a file doesn’t exist it throws an error since it can’t find the file to read from. Instead of using a try/except loop, it means I can bring a line of code to the user letting them know the file doesn’t exist, and to create a text file use menu option 4. Without the try/except loop, there is an error and the program stops.

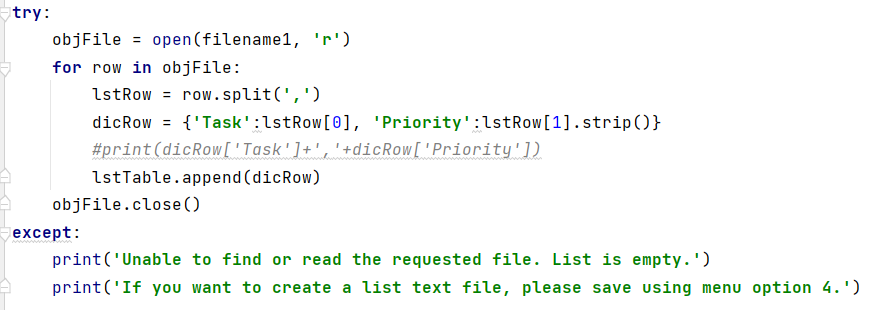


Figure 17. Example of a try/except loop for reading a text file and writing it to a list of dictionaries.

## Organizing Thoughts

One of the last topics of the module was reviewing code structure. The important thing about code structure is making it easy for coders to read and also debug later. An organization scheme so others can easily pick up where you left off. The starting point is a description of the program and a log of how it has changed. To make it easier to input the starting description script, it can be saved as a template in PyCharm. Instead of typing it out each time, you can call the template and update it as needed.

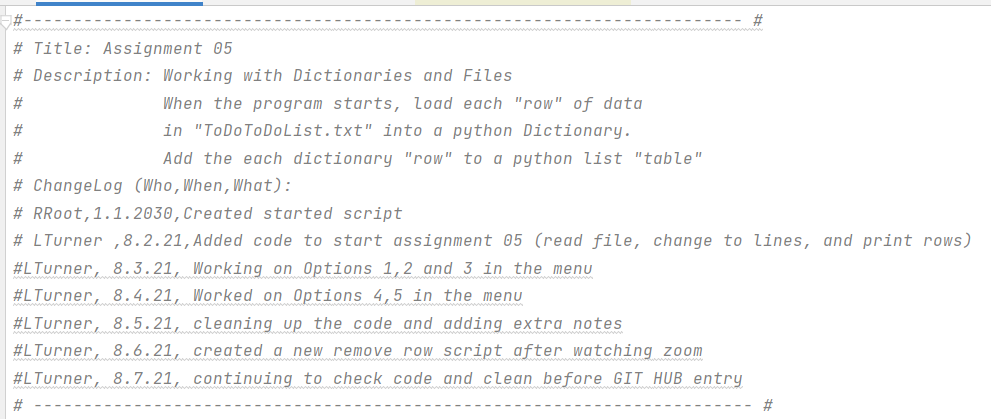


Figure 18. Example of script description with a change log. This will help others who come to look at your code see what is changed and where the last coder left off.

Next, it’s important to declare all the variables being used in the script. Not only does this help others following your work, but it also can help improve script. I learned in the last module, that declaring the variables outside of the while loop improved my script by allowing me to call variables at different parts of the script.

In this module we also went over the concept Separation of Concerns. This concept means that the script should be broken up into different sections based on what each piece of script is doing. It is common to have 3 sections, data, processing, and presentation. The data section is where all the variables or data used is defined. Then all parts of the script to process the data can be grouped together. This is normally accomplished by using functions. Functions allow the processing script to be written together, but be called by name into the processing section.

The final section is the presentation or input/output section. This section holds all the print and user input script, or rather information presented to the user. Breaking up the sections of script not only make it easier to read and understand, but also for debugging.

Without the use of functions, it was hard to break up the code for Assignment 05. I made sure to put all the data/variables at the top of the script in one section. Then I read from the file and wrote to dictionary lists in the processing section. The last and largest section was the presenting section, this was all the print statements and input statements to make the program work.

## GitHub

The last part of Module 5 and Assignment 5 was to post our work to GitHub so that others in the class could review our code for things we did well and things we could check. GitHub is a platform that allows you to upload scripts and post them publicly so others can review your code, use your code, or work together on code.

GitHub is easy to access and simple to upload files to. Below are the basic steps I took to access GitHub.

1. Go to the GitHub Website. <https://github.com/>
2. Sign-Up for a GitHub account
3. Enter the information required to create an account
   1. Name and Email
   2. Create username and password
4. Wait for the confirmation code to appear in you email to confirm you account
5. YOU ARE NOW PART OF GitHub! Congrats!

After creating an account the next part was to upload your code to GitHub and post a link so others could check your work. To upload a script into GitHub, below are a few simple steps.

1. In the top left corner of GitHub there is an icon to create a repository.
   1. A repository is a file folder to store your work in
2. After naming the repository, you’ll need to add a description and decide how you want the files you upload to be shared
   1. You can choose private or public for who can view the files
   2. There is also an option for a readme file, .gitignore, or create a license that limits how people can use your code.
3. Once you have selected your options you can hit the create repository button at the end to complete the repository set-up.

It was great to be able to review other’s code and see how they accomplish certain tasks. I was impressed with how the script differed from person to person depending on how they read the assignment. A few folks really went above and beyond the assignment.

Here is a link to my GitHub repository for Assignment 5.

<https://github.com/PiperRen/IntroToProg-Python>

## Summary

This module’s main lesson revolved around dictionaries and how they can be used. Dictionaries have even more abilities than lists. In this document, I also reviewed organizing thoughts and how to use GitHub. Separation of concerns is organizing code by its purpose; data, processing, and input/output. The last part of the assignment was to upload our scripts to GitHub so others in the course could review our code and provide feedback.