Working with Functions

# Introduction

This document will cover what functions are and how functions can be used in Python. Functions are an important tool for being able to break code up by purpose per Separation of Concerns. Additionally, this document also covers creating a GitHub web page and how to use the debugger function in PyCharm.

# Functions

To be able to separate concerns into data, processing, and presentation, functions are needed. A function is simply a grouping of commands. A function can be as simple as a single print statement or as complicated as hundreds of lines of code.

When a function, or group of statements, runs the output of those statements can be returned to to be used elsewhere in the code. As with outputs, it is also possible to feed in inputs to the functions for use. The ability of functions allows for flexibility and a single function to be used several times in the same program.

To create a function, one needs to define the function with “def” and the function name. If there are any inputs that need to be passed into the function, those will also need to be included in the name. For example, the script below shows how to define a function called print name, with the input to the function being the name.

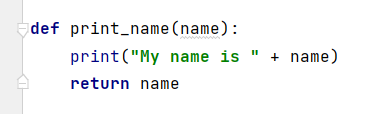


Figure 1. Example script that shows how to create a simple function that uses the input name, prints a statement with the name input, and returns the value in name as well.

## Variable Naming

While being able to input and return values in and out of functions, it’s also important to understand how to name those variables. There are two types of variables that can be used, global and local. A global variable can be used anywhere in the code, however the value the global variable has, it is not easy to follow if it’s used inside functions. One application where a global value is used, is when the global variable is a constant. No matter what is done with the global variable it never changes.

A local variable is a variable just used in the function but won’t be used anywhere else in the code. Local variables make it much easier to trace the values of the variables since they are used only in the function.

## Document Headers

Another benefit of using functions is the ability to use a document header or docstring in Python. The document header allows you to add a definition of what the function does, including what variables are used in the function. This is helpful when splitting up code. In the main part of the script, if you call the function but can’t remember what the inputs or outputs of the function are, Python will provide that information to you when hovering over the function.

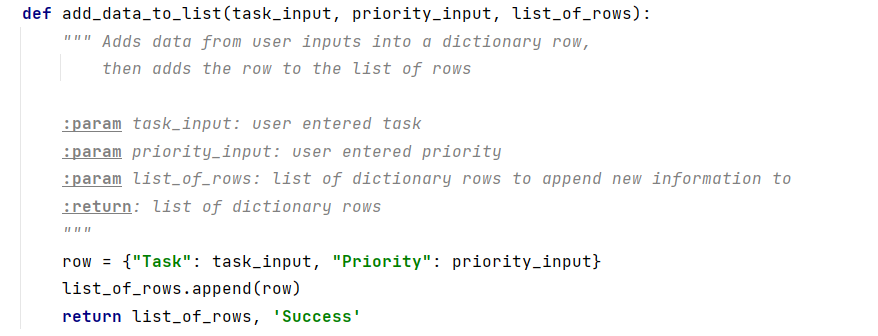


Figure 2. Example of what the script looks like to add a document header in Python.

This is helpful when splitting up code. In the main part of the script, if you call the function but can’t remember what the inputs or outputs of the function are, Python will provide that information to you when hovering over the function. See below for an example of what the document header looks like when called the function in Figure 2.

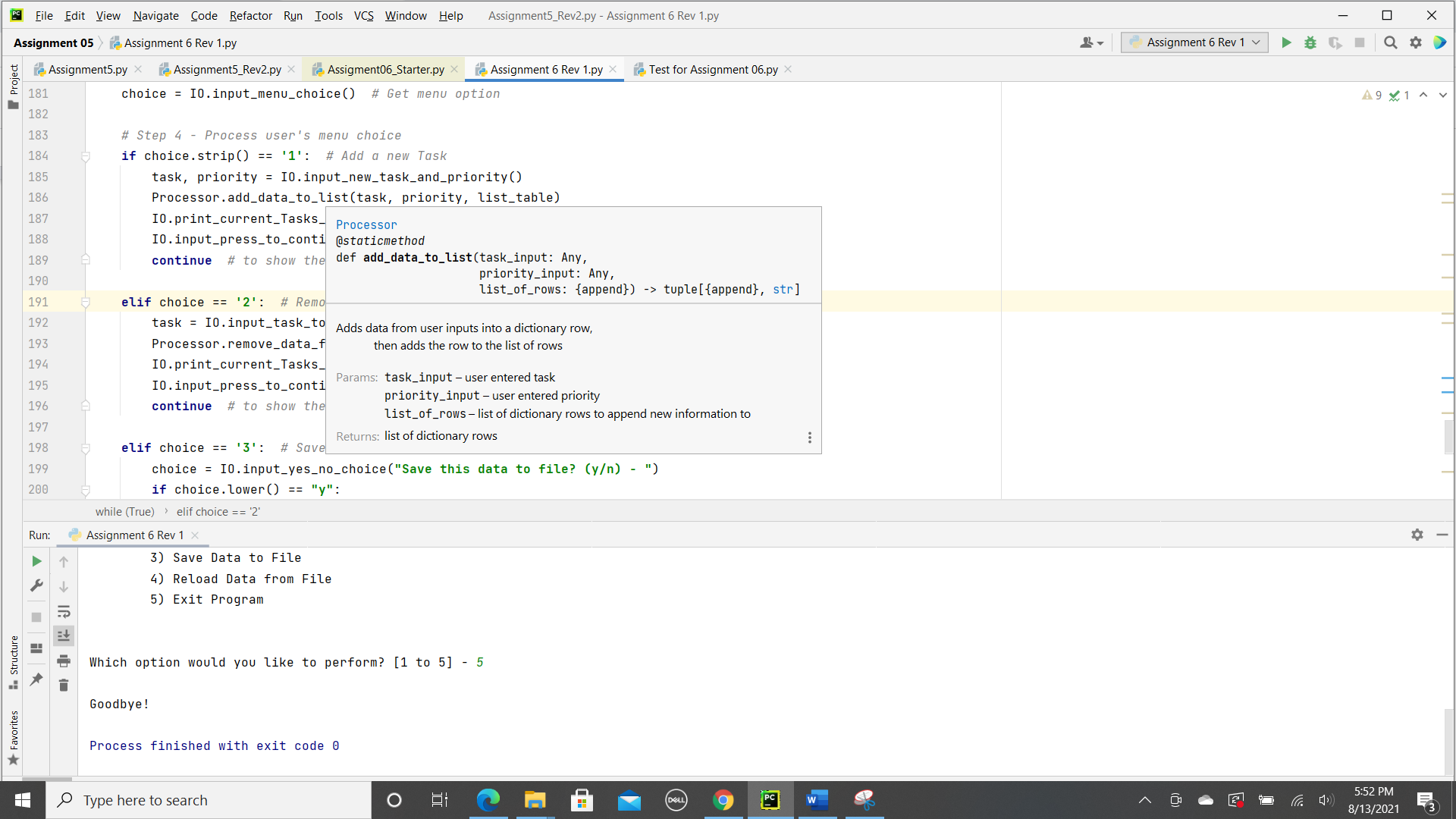


Figure 3. This figure above shows how the document header added to the function shows up later in the script when one calls the function.

## Classes

Just as functions are groups of statements, classes are groups of functions. Classes can also group variables and constants. Classes will be reviewed even more in later modules. Since I am going to use classes in the assignment, I just wanted to review what they were before talking about them.

# Working on the Assignment

As we talked about in class, I wanted to start the assignment, by breaking up the code into different sections. I’m splitting up the code into three sections as with the assignment, processing, presentation, and main script. I am breaking up all the sections to make it easier to debug the script and know if it is working or not, before working on all the code together.

## Testing the Processing Functions

Since the homework assignment was based on last week's assignment, I had the majority of the processing script completed. With the script done last week, I needed to organize into the different parts of the processing script.

Based on the starting code, I knew what functions were needed in the processing section. I would need a function to open and read data from the file, add data to the list of rows, remove data from the list of rows, and finally write the data to a file.

I started with opening the file and reading the data from the file into the list of rows. The inputs for this were the filename and the name for the list of rows. Since I wanted to be able to see the list and use it later, I also needed to return the list of rows back.

I used the starting information given to us and added the needed script to finish up the function. Now that it was ready to test, I needed to provide the variables that would be passed in and out of the function. To see if it was working, I called the function and then printed out the list of rows. With the list printed I could determine if it matched the txt file I had created.

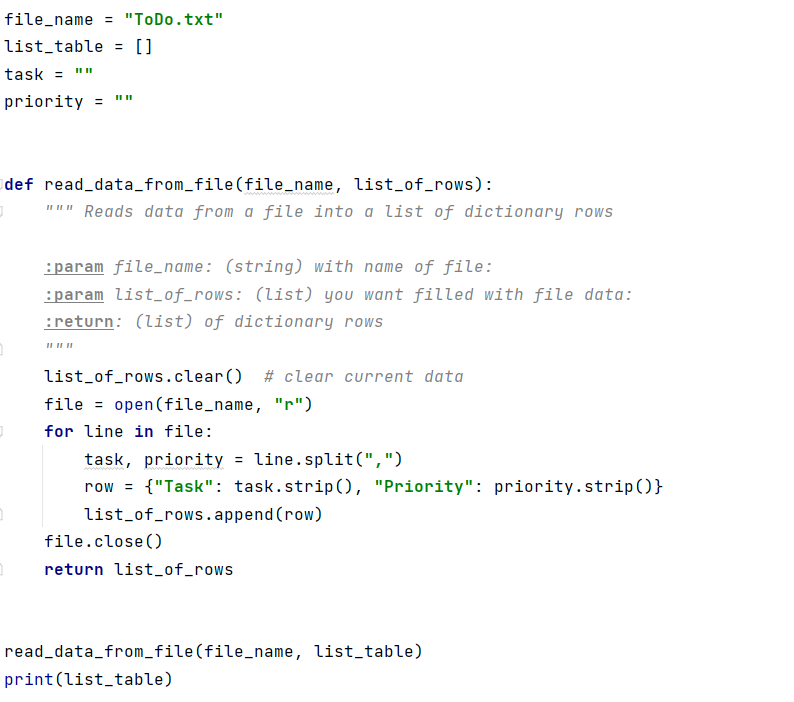


Figure 4. Example of setting up the functions for testing outside of the main assignment file.

To confirm that the function was working correctly, I printed the list of rows and confirmed that it matched my text file.

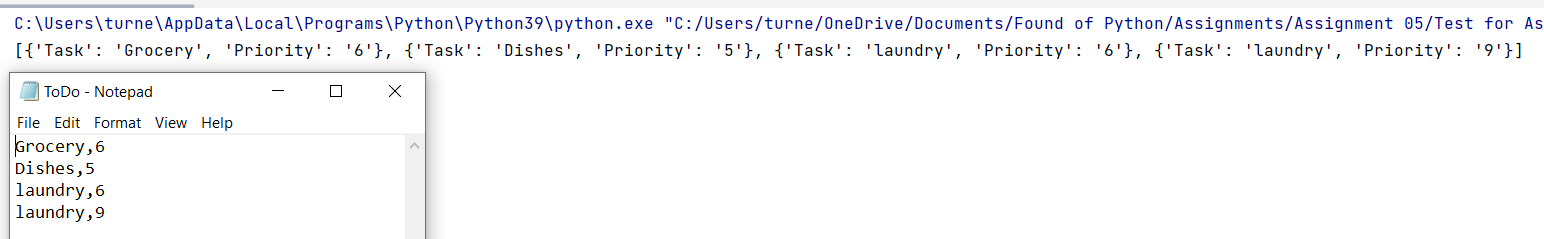


Figure 5. Output from the list of rows print script matches what I had entered into my text file.

I completed the same process with all the other functions I needed in the processing section until they were all working as I wanted them to. The last thing I did once I added them back to the main code was to update the variables where necessary to make them local instead of global so I wouldn’t mix up what values were being held where.

## Presentation Functions

The next section I worked on was creating the presentation functions necessary for getting the user input for the task and priority to be added and the input for the task to be removed. All the other presentation functions called out in the assignment were already completed.

From the past assignments, I had the script needed, it was just a matter of adding it to a function so that the correct information from the function would be returned from the function. As with the processing functions, I also added document headers so later I would know what was needed for that function.

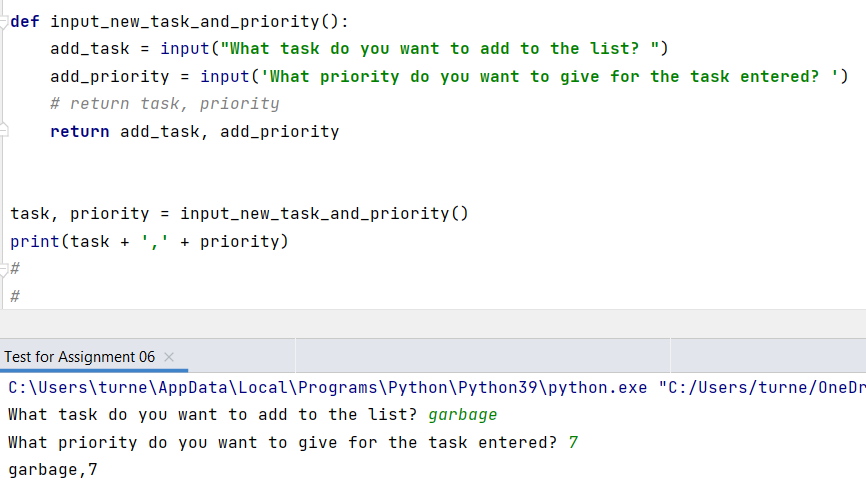


Figure 6. Example of the script used to test the function to ask for the user’s input for task and priority, and then return task and priority to be used elsewhere. To test that the inputs were returned, I used a print statement to check.

## Putting the Code Together

With all the functions built, I was ready to start calling the functions and using them in the main body of the script. I decided to start from the top and read data from the file into a list of dictionary rows. As with the other assignment I found it helpful to add a try and except block. The reason the try and except blocks help is to limit the errors the user faces if there is no file to open.

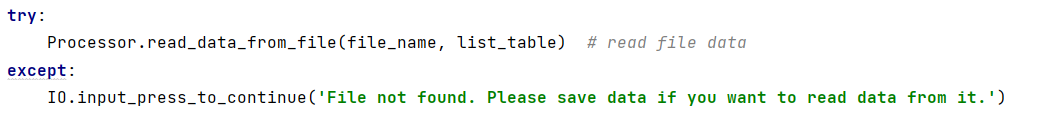


Figure 7. Example of script to open a file, write it to a list of dictionary rows. If the file does not exist, then it tells the user to save if they want to read from it later.

One thing to note is how the function is called. In Figure 7, the function “read\_data\_from\_file” is in the class of Processor. In order to call the function, first we must let python know what class the function is in and then the function name.

As I went through each of the steps outlined in the script, I added the necessary function calls to complete each of the step:

1. Print the menu
2. Adding a task and priority to the list of rows
3. Removing a task from list of rows
4. Saving the list of rows to a text file
5. Reloading the file back into a list of rows
6. Ending the program

With all the steps completed with the right function class, passing and returning the right variable, it was now time to test the script all together.

## Testing the Code

My goal with testing is always to try and break the code. My first test was just a simple walk through of viewing the current list, adding a task, removing a task, saving rows to a file, and closing the program.

With a few minor glitches, I was able to get a simple loop working. I knew there was going to be an issue with the reload function if there was no file. I stepped through the code, and sure enough it threw an error when Python couldn’t find the file to load the data from. As with the start of the code, I added a try/except block to stop the error from causing the script to exit.

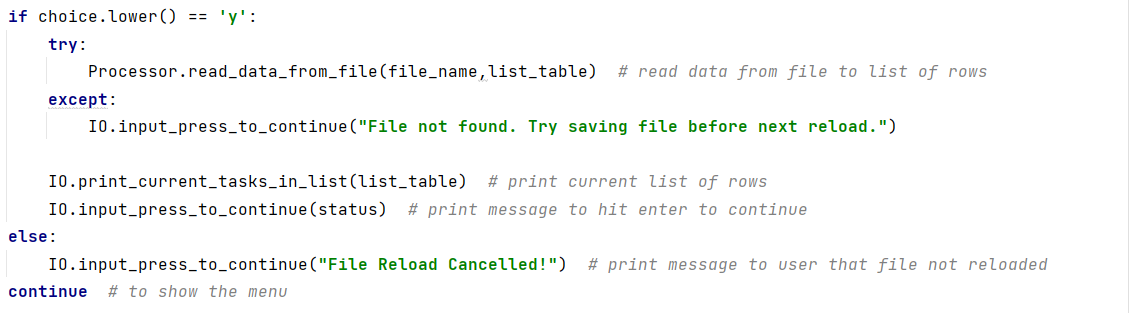


Figure 8. Example of try/except blocks added to the reload script to prevent errors from causing issues with the program stopping early.

I also ran into some issues getting the code to look as I wanted it to. Adding extra lines to make it easier and clearer for the user to read the output. But after lots of testing of the code, I eventually got the output looking the way I wanted.

## Use of the Debugger

One tool that can make it easier to fix bugs in code, is using the debugger tool built into PyCharm. The debugger tool allows a coder to step into code to see the steps Python is taking when running through different sections of the code.

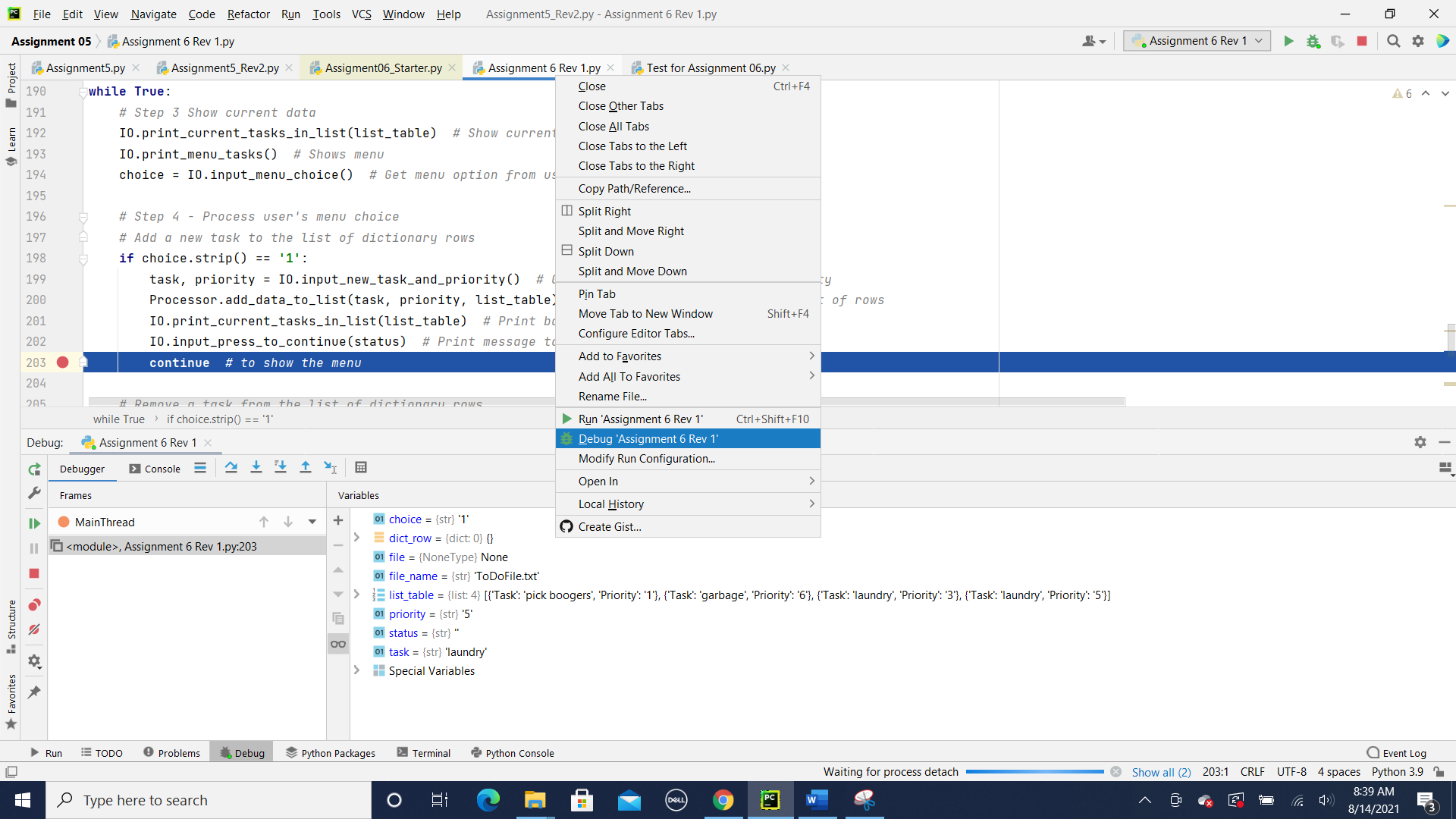


Figure 9. Picture of where to find the debugging tool (dark blue highlighted drop down menu option), and one of the outputs from the debugging tool. Stops or breakpoints can also be added to pause the code, see red dot to the left of the script.

As one can see in Figure 9, if you right click on the script you want to debug, there is a drop-down menu that allows you to run the script or debug the script. In the debug mode, you can add stops to allow a coder to check a small amount of the script at a time.

To add a breakpoint, you can right click to the left of the script and a red dot will appear. Click on the red dot to remove it when it’s not needed anymore. Click to add it back or in another part of the script.

Also in Figure 9, one can see what one of the uses of the debugger is, to see what is stored in all the variables. This information can be found in the bottom section of Figure 9. Especially when trying to troubleshoot Python, it is helpful to know what type of data Python decided to use for each part of the code.

I found it helpful to use the debugger code when testing the functions, especially once trying all the code together. Knowing what values were being passed in and out of the functions, was helpful.

## GitHub Web Page

One of the last parts of the assignment was to create a GitHub webpage. As we did last week in creating our first folder/repository, we were asked for this assignment to add another repository and create a web page.

From the last assignment it was easy to add a new repository for module 6. Once I had the repository started, this week we were supposed to add an extra folder to our repository. To do this we added a file to the module 06 repository but stored it under a secondary file called docs.



Figure 10. Example of folder structure in my GitHub. Storing my new webpage inside of the repository ITFnd100-Mod06 inside of the folder docs.

To make a webpage from our index.md file, we had to go to settings and let GitHub know where we wanted the files for the web page to come from. Once in settings, you have to scroll down to “pages” to be able to set-up the web page information.

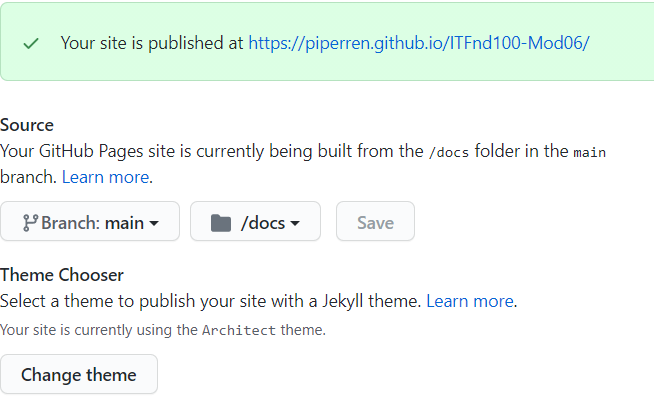


Figure 11. Example of how-to set-up a GitHub web page.

This menu can be found under setting and then under pages. The menu shows you where your web page will be published, where the documents for the web page are pulled from, and finally GitHub gives you the ability to change the theme of your web page. My web page changed themes many times trying to get one I really liked. I had lots of fun putting together my first web page.

Once I had the web page set-up, I was ready to add some content. The original file, index.md, was updated to add some links to various pages per the homework assignment. I copied the listing from the homework and added a link of my own for fun, see Figure 12.

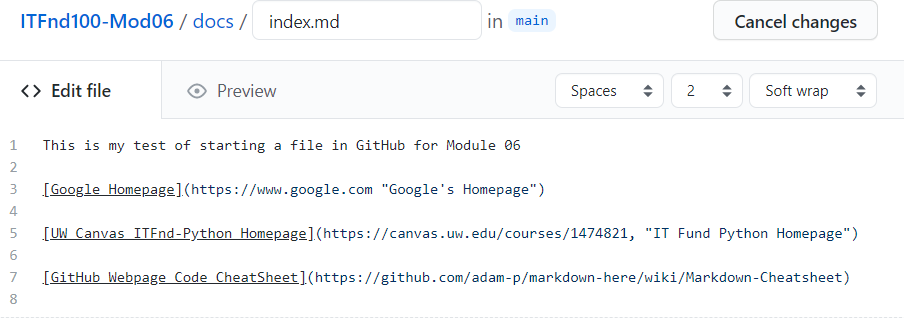


Figure 12. Example of code added to the “index.md” file to be able to display links to different web pages.

Once I had my links added to my index.md file, it was time to see what was published to the web page. At first, I had a space between the name and address, which didn’t make the links work. I updated the script to remove the space and then my web page worked as I wanted.

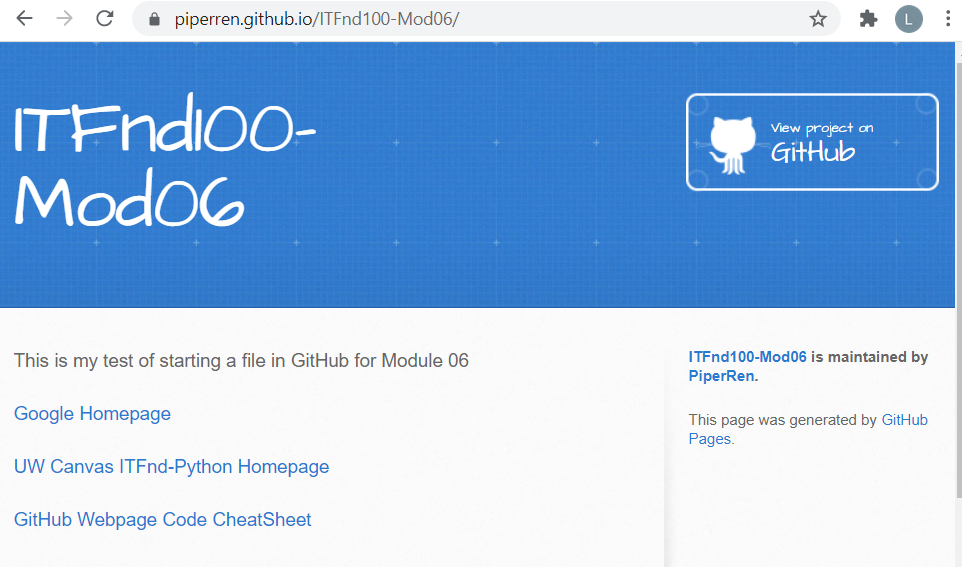


Figure 13. An example of what a web page from GitHub looks like. This web page is the one for assignment 6 with links to various other web pages, such as Google.

# 

# Summary

In module 06 we learned all about functions and how to use them. This document covered how to use functions, how to use classes, and how to use the debugger tool. The assignment reviewed taking the code from assignment 5 and using functions to get a similar output. Functions allowed the code to be broken up into different sections: data, processing, presentation, and the main body of the script. As well as learning about functions, we also touched on how to use the debugger and setting up web pages on GitHub.