

Corbin Bridge

November 21, 2022

IT FDN 110 A Au 22

Assignment06

<https://cbridge-uw.github.io/IntroToProg-Python-Mod06/>

The To Do List Script with Functions

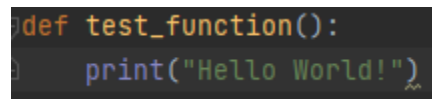
Introduction

In the following knowledge document, the reader will be introduced to the process of creating functions in Python. While Python comes with several built-in functions, it is possible for the developer to create their own. I will cover how to create a function, adding a doc string, and enhancing a script with the use of custom functions.

Creating a Function

Before diving into the script, I will briefly discuss how a developer can create their own custom function in Python.

The first step is to declare the function name. Like variables, names should be relevant to what action the function performs. All function names are preceded by the def keyword (Figure 1).



```
def test_function():  
    print("Hello World!")
```

Figure 1: Creating a custom function.

Like other blocks of code that we have worked with, while-loops and if-statements, functions make us of indentation. All code indented under the function is grouped together.

The code within a function is loaded into memory when the script starts but does not perform any action until called later in the main body section.

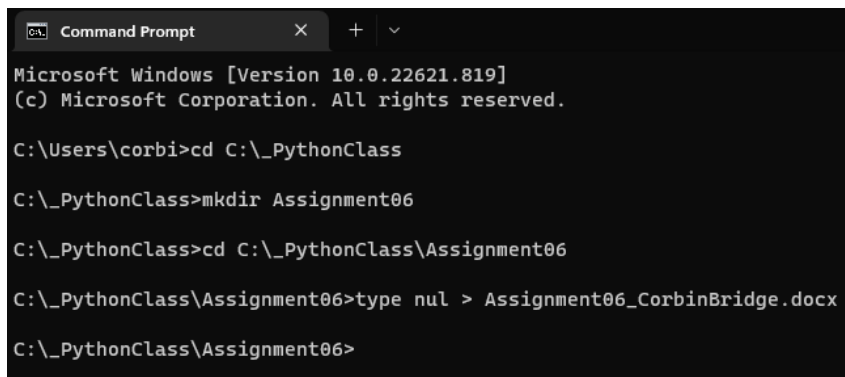
Functions are very helpful when writing longer scripts. They allow the developer to break up code into sections that perform a single task very well. This also helps for testing as functions can be isolated and tested. Finally, functions are useful in that a change can be made in one place and be used in any other section of the script that calls that function.

Creating the Script

Assignment06 requires that updates are made to a provided To Do list script so that more functions are used to organize the code.

Initial Setup

The initial setup follows the pattern of prior assignments. I created a new sub-folder in the `_PythonClass` folder and added a new knowledge word document (Figure 2).



```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\corbi>cd C:\_PythonClass

C:\_PythonClass>mkdir Assignment06

C:\_PythonClass>cd C:\_PythonClass\Assignment06

C:\_PythonClass\Assignment06>type nul > Assignment06_CorbinBridge.docx

C:\_PythonClass\Assignment06>
```

Figure 2: Initial setup.

A starter file was provided by Professor Root in the .zip folder found on Canvas. I made a copy of the file and added it to the Assignment06 folder.

Finally, I created a new project in PyCharm in the Assignment06 folder (Figure 3).

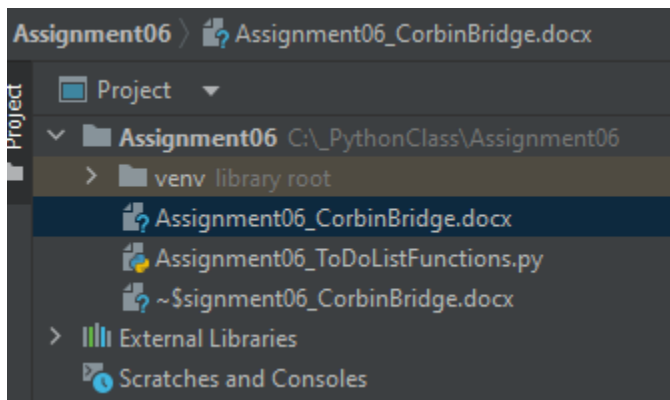


Figure 3: Assignment06 PyCharm project.

With the basic setup completed, it was time to create some functions.

add_data_to_list

The first section of code to address was the `add_data_to_list` function. I appended the dictionary row to the existing list of rows (Figure 4). Row consists of a task and priority in the typical key: value pairs in a Python dictionary. The return statement is used frequently in functions. It serves to return the functions result to the caller. In this case, return will send the updated `list_of_rows` to the section of the code that made the function call.

```
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :param task: (string) with name of task:
    :param priority: (string) with name of priority:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    # TODO: Add Code Here!
    list_of_rows.append(row)          # add new row to list_of_rows
    return list_of_rows
```

Figure 4: Appending row to list_of_rows.

remove_data_from_list

I next revised the processing function that removes an item from the list. A for loop is used to review each item in the list_of_rows. A nested-if statement checks if the task, in lower case and with whitespaces stripped, is the same as the task specified by the user. If so, the item is removed from list_of_rows and the updated list is returned (Figure 5).

```
@staticmethod
def remove_data_from_list(task, list_of_rows):
    """ Removes data from a list of dictionary rows

    :param task: (string) with name of task:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    # TODO: Add Code Here!
    for item in list_of_rows:
        if item["Task"].lower().strip() == task.lower().strip():
            list_of_rows.remove(item)
    return list_of_rows
```

Figure 5: Removing items from list_of_rows.

I used the lower and strip methods on both the item in list_of_rows and the user input to ensure that a false negative is not returned. This could happen if a user enters a task in a different case than is stored in list_of_rows or has extra whitespaces. This issue is likely to occur and would create a bad user experience.

write_data_to_file

Writing data to the file follows the process used in Assignment05. The file is opened in write mode. A for loop is then used to write each item in the list_of_rows to the file. Each item contains a comma separated task and priority. As is best practice, the file is closed before returning the list_of_rows (Figure 6).

```
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :param file_name: (string) with name of file:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    # TODO: Add Code Here!
    file = open(file_name, "w")
    for item in list_of_rows:
        file.write(item["Task"] + "," + item["Priority"] + "\n")
    file.close()
    return list_of_rows
```

Figure 6: Writing data to the file.

input_new_task_and_priority

This function is used to gather a task name and priority from the user for a new entry. Both task and priority are requested using the input statement. The input is converted to string. Note that string is the default value type for input. Explicitly making them a string, via str, is intended to help others understand my intent (Figure 7).

```
@staticmethod
def input_new_task_and_priority():
    """ Gets task and priority values to be added to the list

    :return: (string, string) with task and priority
    """
    # TODO: Add Code Here!
    task = str(input("Enter a task: ")).title() # get task from user
    priority = str(input("Enter a priority (Low, Medium, High): ")).lower() # get priority from user
    return task, priority
```

Figure 7: Requesting a new task and priority from the user.

I use the title and lower method for task and priority, respectively, so that each entry has a uniform case regardless of how the user entered the values (i.e., using all uppercase).

input_task_to_remove

The final function that required attention allows the user to specify the task to be removed. An input statement requests the task name from the user. The lower and strip methods are both applied to the input before returning the task (Figure 8).

```
@staticmethod
def input_task_to_remove():
    """ Gets the task name to be removed from the list

    :return: (string) with task
    """
    # TODO: Add Code Here!
    task = str(input("Enter the task name to be removed: ")).lower().strip()
    return task
```

Figure 8: Requesting the task to be removed.

Other Modifications

In addition to modifying the existing functions to complete the assignment, I added some additional code to make the program run smoothly.

First, I included a try-except block in Step 1 of the main body of the script. This returns a user-friendly error message if ToDoList.txt is not yet created (Figure 9). If try-except is excluded, an error message is returned and the code stops running.

```
# Step 1 - When the program starts, Load data from ToDoFile.txt.
try:
    Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst) # read file data
except:
    print("No To Do List created yet. Try adding some items!\n")
```

Figure 9: Using the try-except code block for error handling.

Next, I added a while loop along with an if-else statement to the input_menu_choice function. As the menu only has options 1-4, I wanted to return an error if the user enters any other value. I made use of the in keyword to test if the user entered a value between 1 and 4. If so, the code proceeds as expected. If not, then an error message is returned, and the user must enter a new option (Figure 10).

```

@staticmethod
def input_menu_choice():
    """ Gets the menu choice from a user

    :return: string
    """
    while True:
        choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
        if choice in ("1", "2", "3", "4"):
            print() # Add an extra line for looks
            return choice
        else:
            print("Not a valid option. Only enter 1-4.\n")

```

Figure 10: Testing for valid user entry.

Before exiting the program, I wanted to prompt the user to save before leaving. While a diligent user would save frequently, others might forget to do so.

I created a new function, `save_reminder`, in the I/O section. This function asks a user if they would like to save before existing. The return value is either a Y or a N (Figure 11).

```

@staticmethod
def save_reminder():
    """ Reminds the user to save before exiting

    :return: string
    """
    while True:
        reminder = str(input("Would you like to save before exiting the program? [Y]es or [N]o: ")).lower().strip()
        if reminder in ("y", "n"):
            return reminder
        else:
            print("Not a valid option. Enter [Y] or [N].")

```

Figure 11: Creating `save_reminder`.

In keeping with best practice, I included a brief docstring describing the function's purpose and return value type.

Next, I added `save_reminder` to the main body of the script when the user selection equals 4 to exit the program.

If the user does want to save before exiting, when `save_reminder` is "y", then the same code included when the user selection is three (save data to file) is executed. If `save_reminder` is "n", then the program ends (Figure 12).

```

elif choice_str == '4': # Exit Program
    if I0.save_reminder() == "y":
        table_lst = Processor.write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved!\n")
        print("Goodbye!")
        break # by exiting loop
    else:
        print("Goodbye!")
        break # by exiting loop

```

Figure 12: Enhanced exit program block.

Testing the Script

Before submitting the assignment, I tested the script in both PyCharm and the command window.

PyCharm

I first validated that the try-except block was working and added a new task to the list (Figure 13).

```

No To Do List created yet. Try adding some items!

***** The current tasks To Do are: *****
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program

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

Enter a task: read
Enter a priority (Low, Medium, High): HIGH
***** The current tasks To Do are: *****
Read (high)
*****

```

Figure 13: Testing add a new task in PyCharm.

Next, I tested the ability to remove an existing task and save the data to the ToDoList.txt file (Figure 14).

```

Which option would you like to perform? [1 to 4] - 2

Enter the task name to be removed: read
***** The current tasks To Do are: *****
Jog (medium)
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program

Which option would you like to perform? [1 to 4] - 3

Data Saved!
***** The current tasks To Do are: *****
Jog (medium)
*****

```

Figure 14: Removing a task and saving to a file.

Finally, I tested exiting the program. I was pleased to see that the save_reminder function executed as planned (Figure 15).

```

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

Would you like to save before exiting the program? [Y]es or [N]o: y
Data Saved!

Goodbye!

Process finished with exit code 0

```

Figure 15: Save reminder and exit program.

After exiting the program, I found the ToDoList.txt file to confirm my tasks had saved properly (Figure 16).

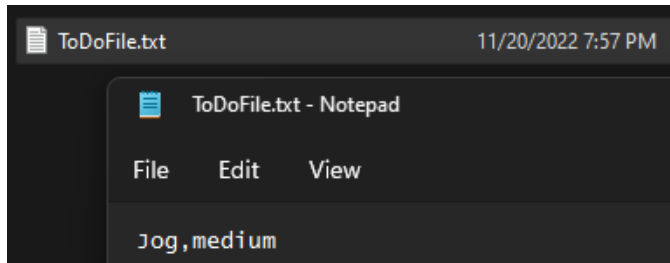


Figure 16: Checking ToDoList.txt.

Command Prompt

After testing in PyCharm I conducted the same tests in the command prompt.

First, I added a new item (Figure 17).

```
No To Do List created yet. Try adding some items!

***** The current tasks To Do are: *****
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program

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

Enter a task: read
Enter a priority (Low, Medium, High): Medium
***** The current tasks To Do are: *****
Read (medium)
*****
```

Figure 17: Adding a task via the command prompt.

Next, I removed a task and saved the data to the ToDoList.txt file (Figure 18).

```
Enter the task name to be removed: read
***** The current tasks To Do are: *****
Jog (medium)
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program

Which option would you like to perform? [1 to 4] - 3

Data Saved!
***** The current tasks To Do are: *****
Jog (medium)
*****
```

Figure 18: Remove a task and save the data.

I then exited the program and again confirmed that the save_reminder function was working (Figure 19).

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

Would you like to save before exiting the program? [Y]es or [N]o: y
Data Saved!

Goodbye!
```

Figure 19: Save reminder and exit the file.

GitHub

In Assignment05 I created a new GitHub account. For Assignment06 I will also be saving my script and knowledge doc to GitHub.

First, I created a new repository (“repo”) so that this assignment’s files are separate from the prior assignment files (Figure 20). GitHub makes this process very simple. I clicked the green “create new repository” button on the home page, added the details in Figure 20, and clicked Create repository.

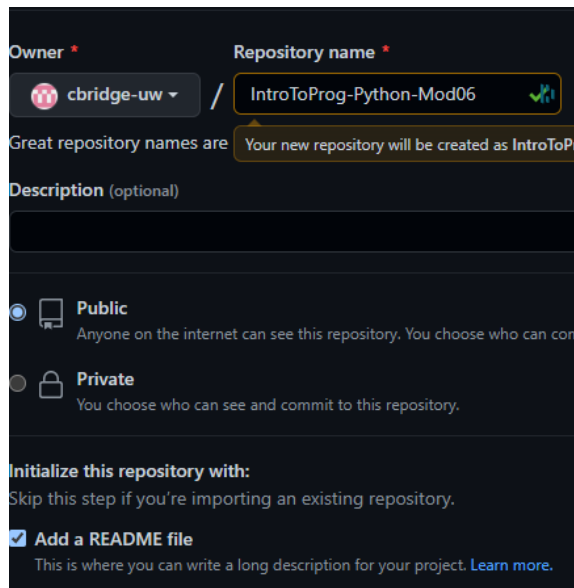
The screenshot shows the GitHub 'Create new repository' form. The 'Owner' field is set to 'cbridge-uw'. The 'Repository name' field is 'IntroToProg-Python-Mod06', which is highlighted with a green checkmark. Below the name, a tooltip says 'Your new repository will be created as IntroToP...'. The 'Description' field is empty. Under 'Visibility', the 'Public' option is selected with a radio button. The 'Initialize this repository with:' section has 'Add a README file' checked with a checkbox. A link 'Learn more.' is provided for the README file.

Figure 20: Creating a new GitHub repository.

I entered the newly created repository, selected Add file, found Assignment06_ToDoListFunctions.py and committed the changes (Figure 21).

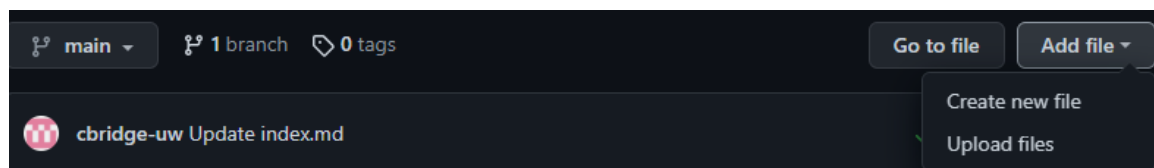
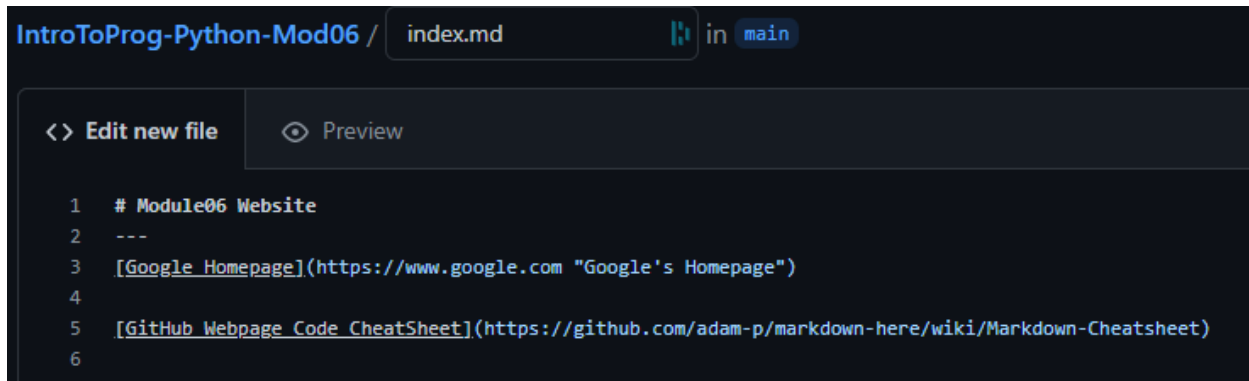


Figure 21: Adding files to GitHub.

Creating a GitHub Web Page

The final requirement of the assignment was to create a GitHub web page. True to character, GitHub makes this extremely easy.

First, I created an index.md file in the Assignment06 repository and added the starter code provided by Professor Root in the assignment brief (Figure 22).



The screenshot shows the GitHub web editor interface for the file 'index.md' in the 'main' branch of the repository 'IntroToProg-Python-Mod06'. The editor has two tabs: 'Edit new file' and 'Preview'. The code in the editor is as follows:

```
1 # Module06 Website
2 ---
3 [Google Homepage](https://www.google.com "Google's Homepage")
4
5 [GitHub Webpage Code CheatSheet](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)
6
```

Figure 22: Creating index.md.

With index.md created, I navigated to the main screen of the Assignment06 repository, selected the Settings tab, and went to Pages (Figure 22).

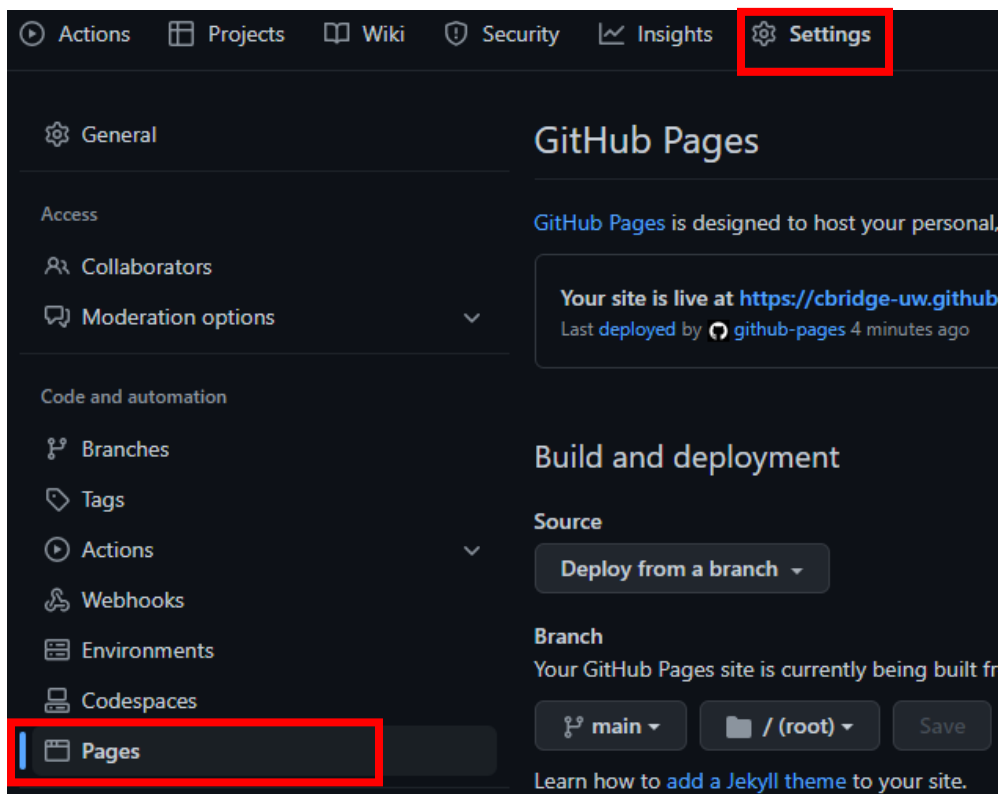


Figure 22: Creating a GitHub Web Page.

I selected the Main branch to build the webpage (Figure 23).

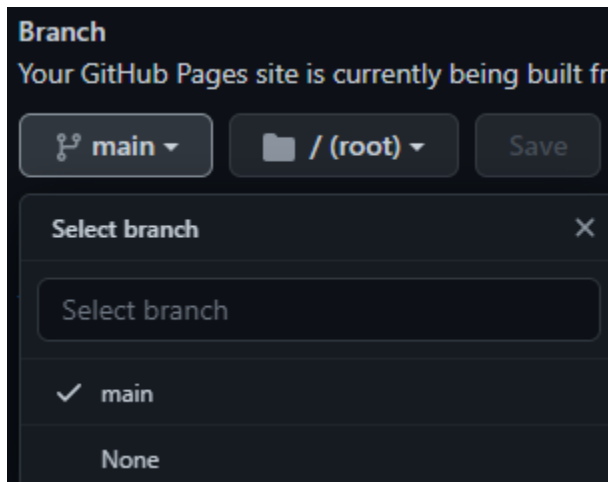


Figure 23: Building the page from main.

Finally, I waited a few minutes and refreshed the page. Upon doing so, I found a link to the new GitHub site (Figure 24).

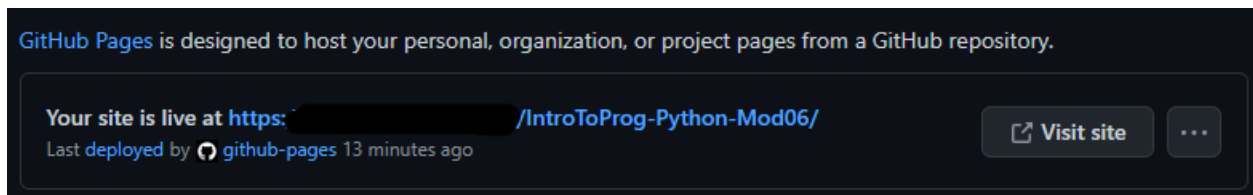


Figure 24: GitHub web page link.

I tested the link to visit the site and it worked as expected. The site is not yet very existing, but it seems that enhancements will be made in coming assignments (Figure 25).

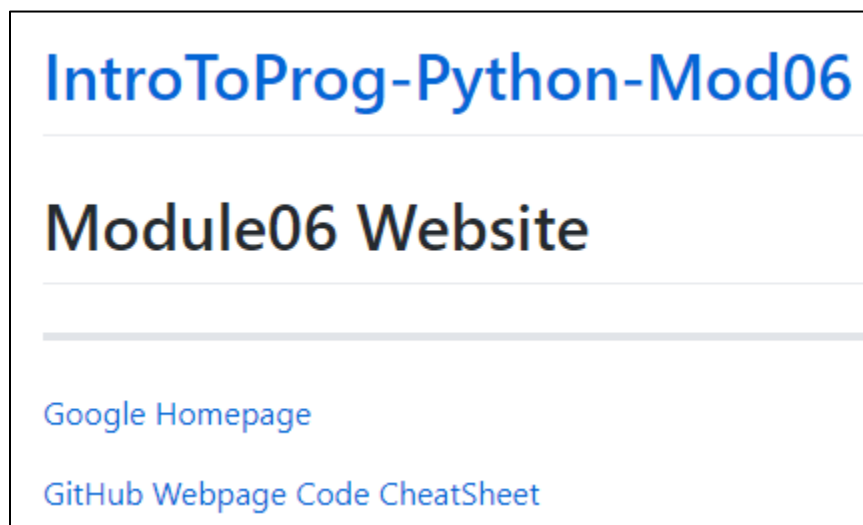


Figure 25: Working GitHub web page.

Summary

In this assignment I worked with a provided script and updated the functions to make an enhanced To Do List program. In addition to modifying the existing functions I created a new one of my own. This assignment helped me to internalize the benefits of using functions and separating code into defined processing, I/O, and main script sections.