Hannah Clayton

February 20th, 2021

Foundations of Programming: Python

Assignment 06

[GitHub URL](https://github.com/hnolte-grad/ITFnd100-Mod06/blob/main/Assignment06.py)

06\_To Do List/Functions & Arguments

I. Introduction

This document serves as a demonstration of my experience working through this week’s assignment module. In the ‘module objectives’ section, I highlight the intended goals for this week’s reading and instructional video as given by Professor Root, and in my own words communicate my understanding of the objectives. In the ‘assignment work-through’ section I show how I went about approaching the module’s assignment, and any issues or epiphanies I encountered in doing so. The overall intent of this document is to serve as both a reference for myself and others in the future and demonstrate my competency in the module to Professor Root.

II. Module Objectives

1. Objectives

• What is a function?

• What are parameters?

• What are arguments?

• What is the difference between parameters and arguments?

• What are return values?

• What is the difference between a global and a local variable?

• How do you use functions to organize your code?

• What is the difference between a function and a class?

• How do functions help you program using the “Separations of Concerns" pattern?

• How are the debugging tools use in PyCharm?

• What is a GitHub webpage?

1. Objective Summary

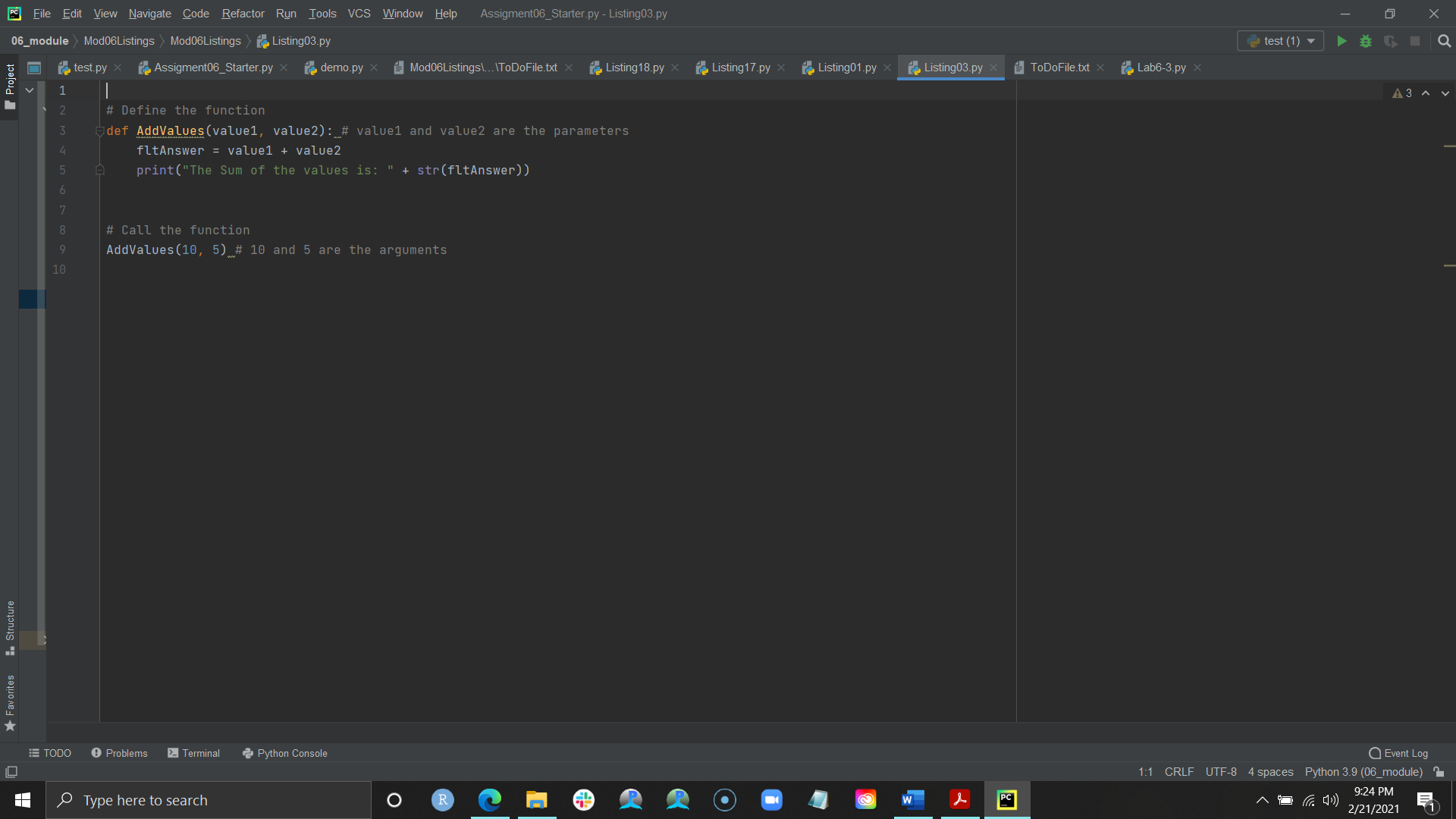
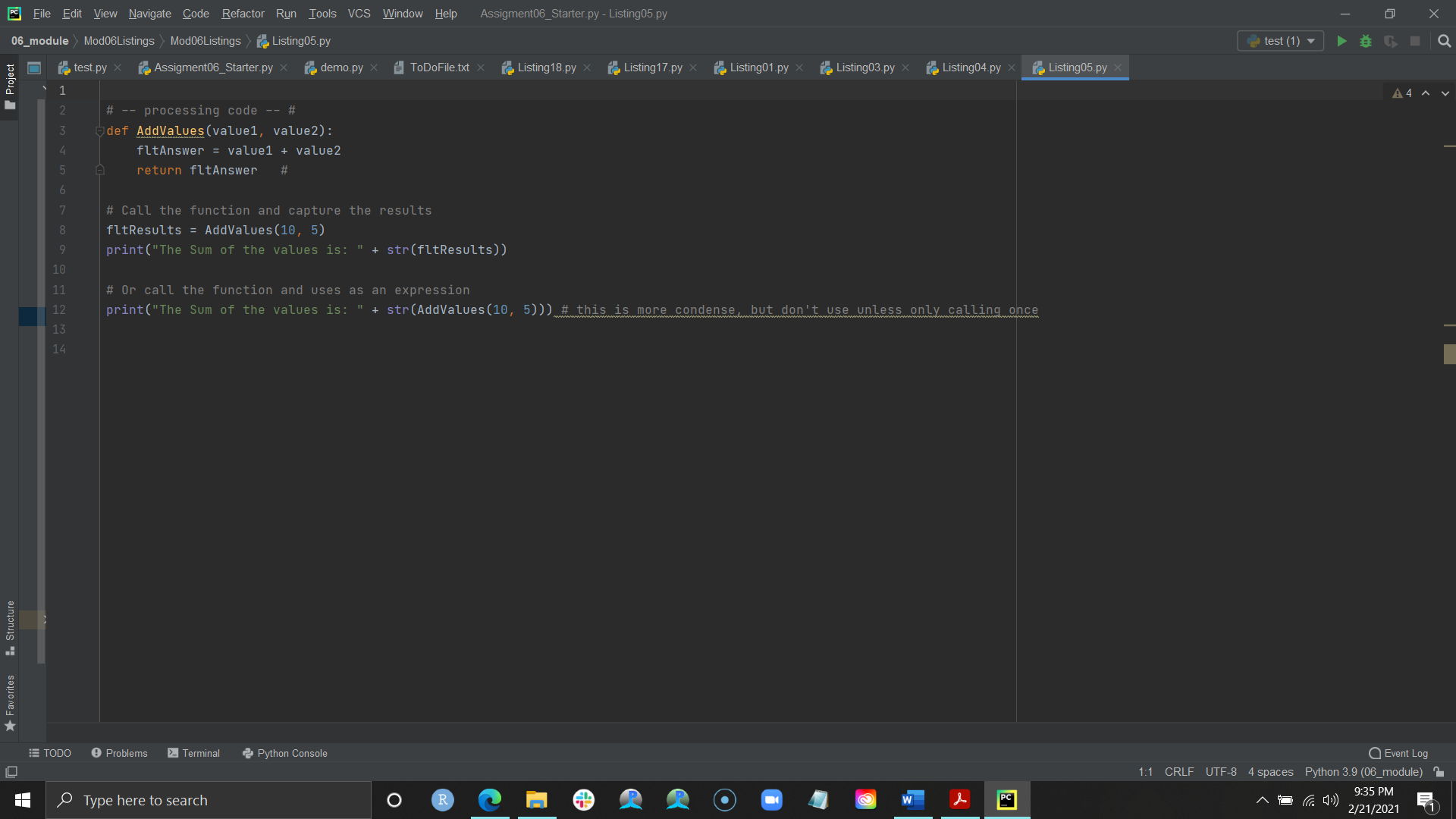
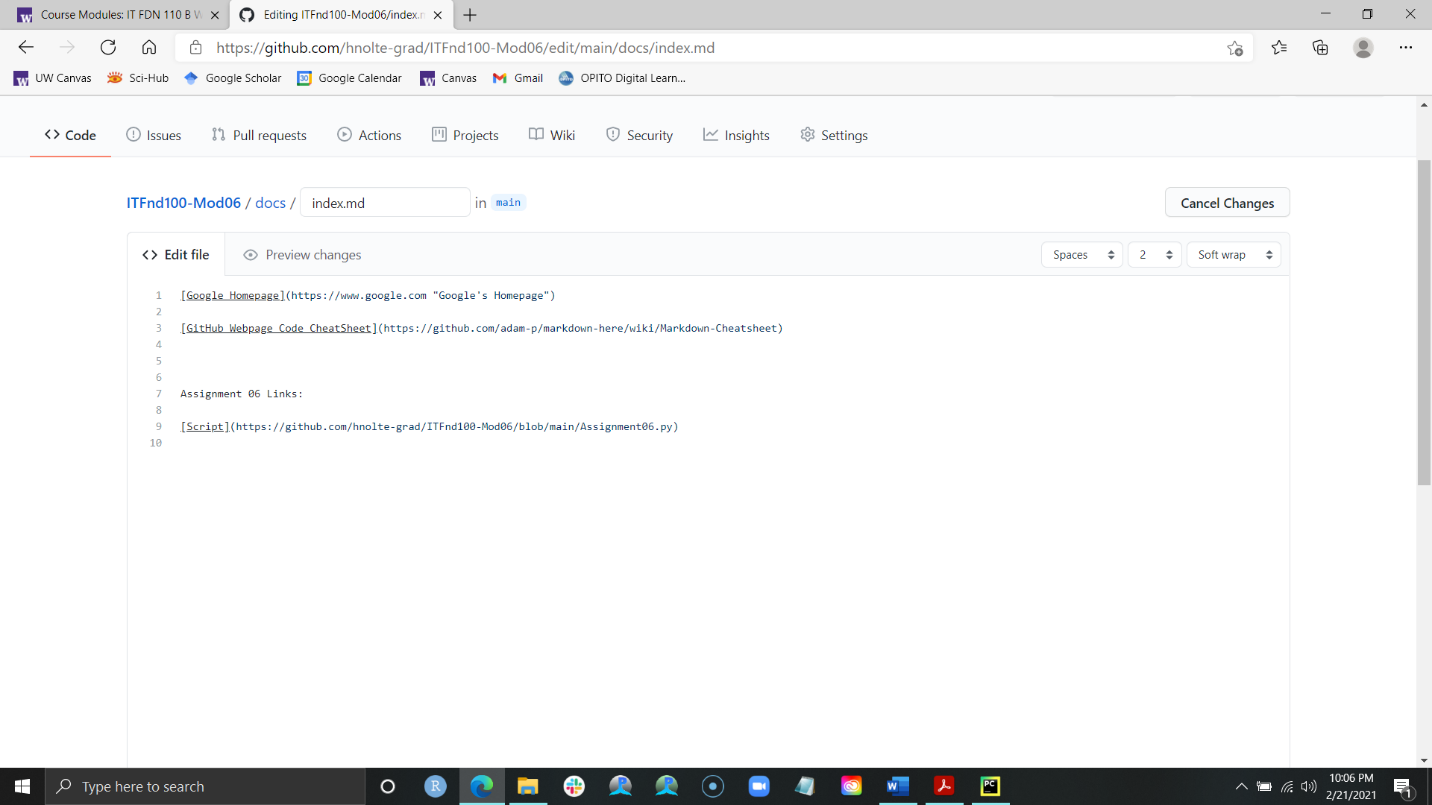
In this week’s module we learned about useful tools such as functions, classes, and arguments to make organizing and trouble shooting our code easier. A *function* is a group of code statements defined by a variable. Defining that function by a variable allows you to later “call” that group of code using the variable name. Being able to call a function allows you to separate your code more easily into data, processing, and presentation sections. To pass values or input into a function Python uses *parameters* in the initial declaration of the function. You can think of parameters as placeholders for the values you would eventually input in the main body of your script. In image 2.1 we can see that ‘value1’ and ‘value2’ are considered our parameters, while ’10’ and ‘5’ are our arguments. While there is no limit to the number of parameters you can introduce to a function, when you go to fill in arguments if you add more or less than the number of parameters you initially input then you will receive an error. Functions would not be as useful if they could not pass on values from processing to other functions, so to do so Python uses the **return** statement. Image 2.2 shows how the return statement is used by designating the **fltAnswer** and **fltResults** variables. Though these two variables look similar, they are two different types of variables. **fltAnswer** is considered a *local variable* meaning it is only intended to be used within that function. To capture values in the main body of your script or perhaps pass them into other functions it is recommended to identify and use *global variables*, such as **fltResults.** Functions, variables, and constants can also be grouped using *classes*, which are created using the ‘**class**’ statement.

Image 2.1 Arguments v. parameters

Image 2.2 the ‘return’ statement

Since starting this class I have been curious about the little green bug in Pycharm and how to utilize it. The debugging tool allows you to perform a variety of different functions to identify problems or ‘bugs’ in your code. Running the debugger allows you to select ‘breakpoint’ lines in you code where you want the program to pause so you can get more information on a particular section. Other useful controls in the debugger allow you to skip over a line of code without examining it (‘Step Over’) and the variables window which shows you all the variables used and their current values within that line or section of code.

We wrapped up this week with an introduction to GitHub websites, which use an entirely different type of programming language. GitHub websites allow you to enhance your repository and is useful for programmers and developers for uploading multiple files. Image 2.3 is an example of what GitHub website code looks like, and what that produces in terms of interface.

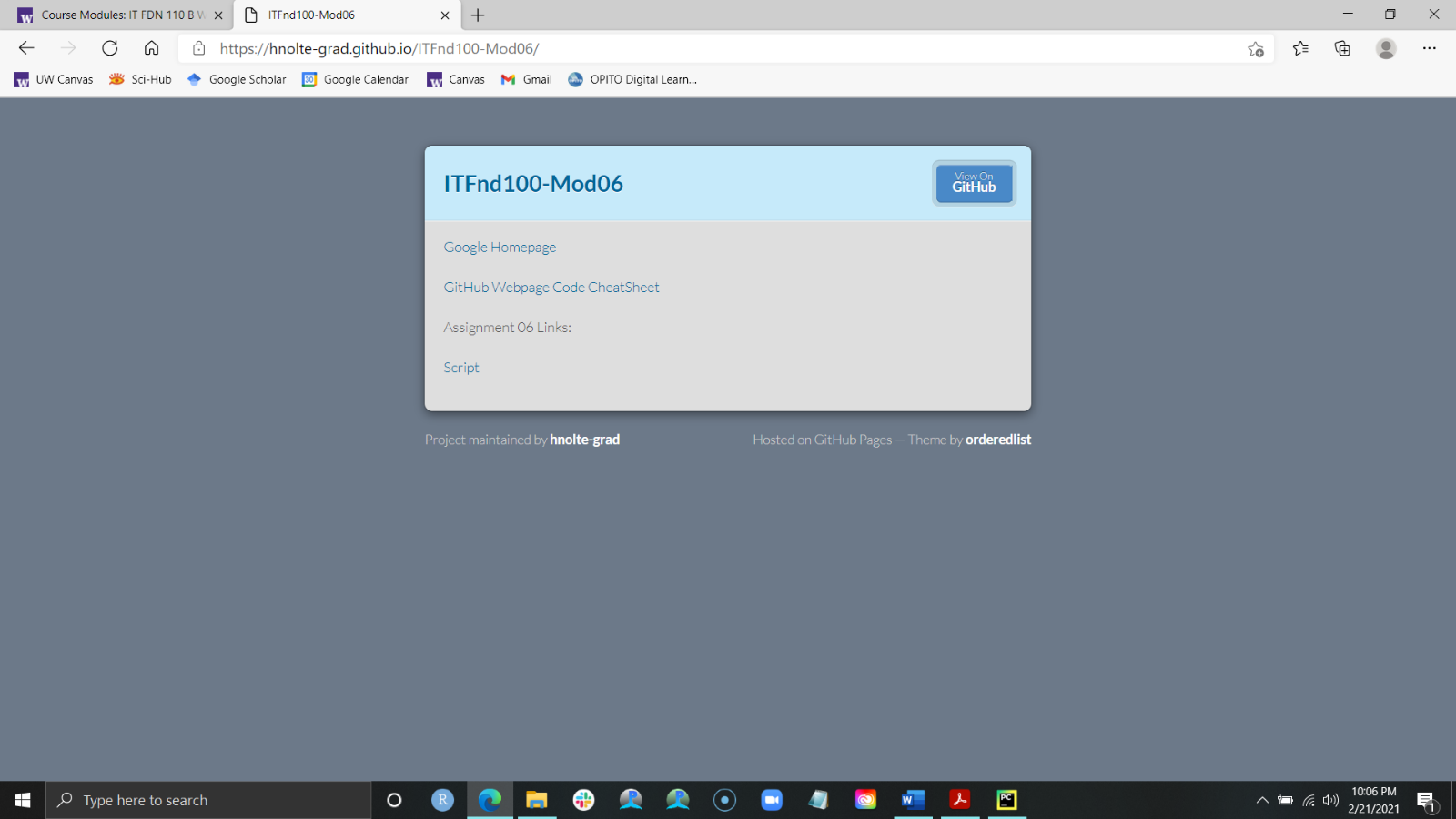


Image 2.3 GitHub websites

III. Assignment Work-through

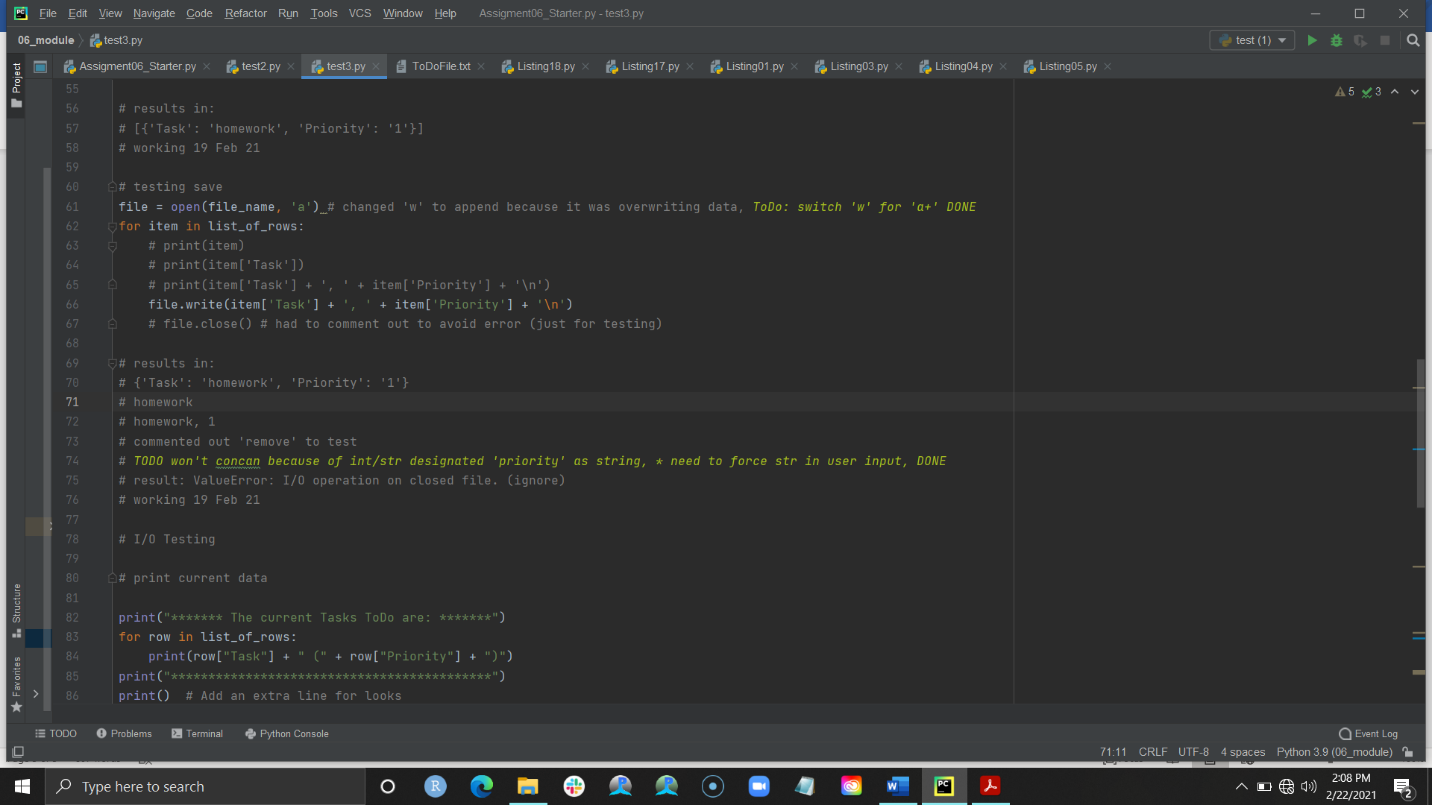
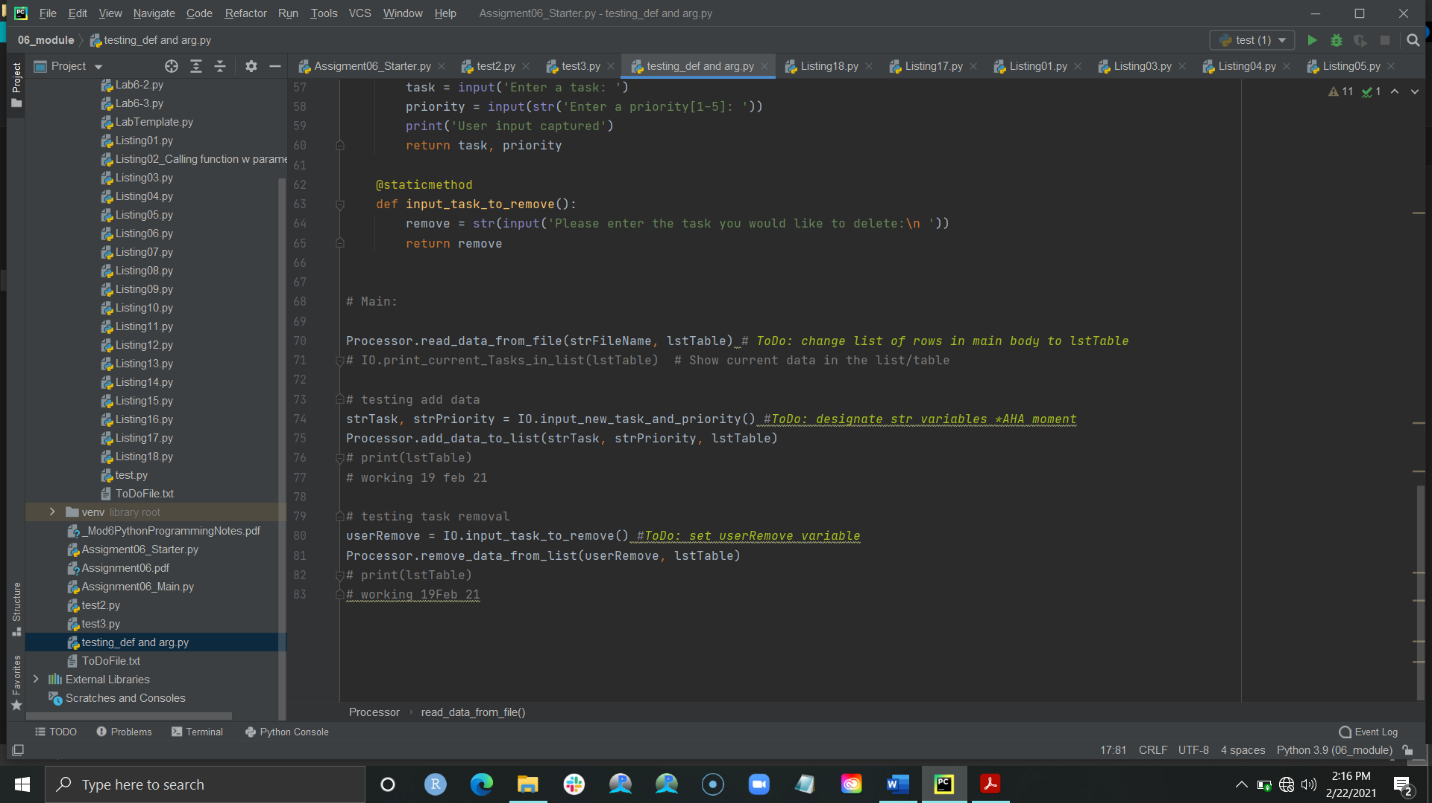
 For this week’s assignment we were required to take a starter code given to us by Professor Root and input code to complete different functions within the code. This went easy for me, as I generally copied my code from last weeks assignments to fill in the gaps and execute tasks such as reading and writing to a file. Like last week, I took the starter code and broke it into sections by the functions to test each independently. Something that seemed to help was adding a lot of comments to help keep me on track and using the ‘ToDo’ highlight function to identify things I needed to add or fix in the main code. Image 3.1 is an example of how I worked on each function and utilized that ‘ToDo’ function to make notes while I was testing.

Image 3.1 Using ‘ToDo’

 My ‘aha!’ moment this week came as I was attempting to figure out why I was getting empty strings returned to me despite the function itself working. Within the add data function itself the program would return the task and priority as expected when I asked the program to print out the variable **dicRow**, but once I got into the main body of the program and tested to see the data in **lstTable** I would get only empty strings.

Of course, the issue was that I was not using global variables to capture the strings being returned by user input, and they therefore were not being transferred to the add data function correctly, resulting in empty strings (Image 3.2). Once I fully comprehended that concept, the rest of the code went easily. I am looking forward to learning more about classes and functions in next week’s module.

Image 3.2 ‘Aha!’ moment