Hwai-Jiang Jong

June 7, 2021

Foundations of Programming: Python

Assignment 09

<https://github.com/uwp-h2021/IntroToProg-Python-Mod08.git>

Working with Modules

# Introduction

This report documents my learning concerning modules, which is demonstrated by a Python script I created for Assignment09. It contains three sections (excluding this introduction):

1. Test Harness Script;
2. Main Module;
3. Summary

# The Test Harness Script

## The TestHarness.py Breakdown

In this script, I am testing the three modules in “DataClasses.py”, “ProcessingClasses.py”, and “IOClasses.py” to make sure they work as intended. The testing script is called “TestHarness.py” as described as follows.

The first portion is the importing of the modules, as shown in Figure 1 .

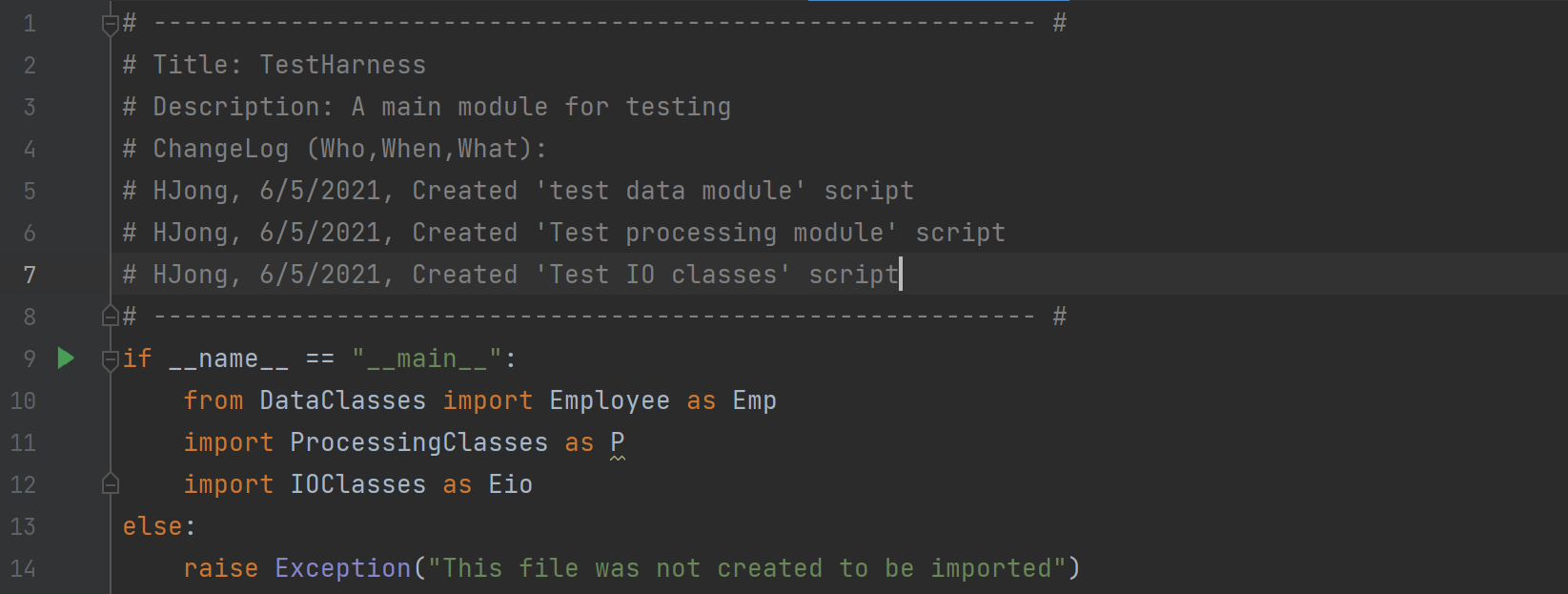


Figure 1 Importing Modules in TestHarness.py

The second portion is the testing of the employee data class, as shown in Figure 2. A list of two employee objects was created using the *Employee* class in the *DataClasses* module.

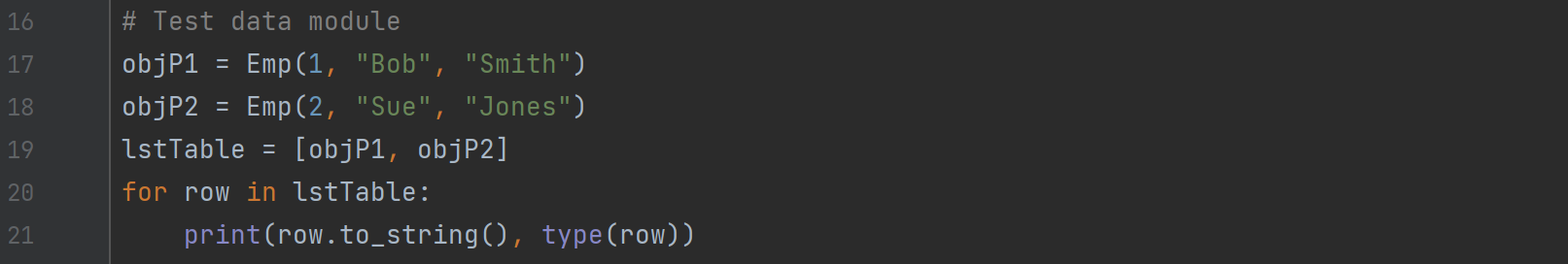


Figure 2 Testing DataClasses Module in TestHarness.py

The third portion of the script is the testing of *FileProcessor* class in the *ProcessingClasses* module, as shown in Figure 3. The list of two employee objects was saved into “EmployeeData.txt”, the employee data were read back from the file, and then the employee data were converted into the original list of employee objects.

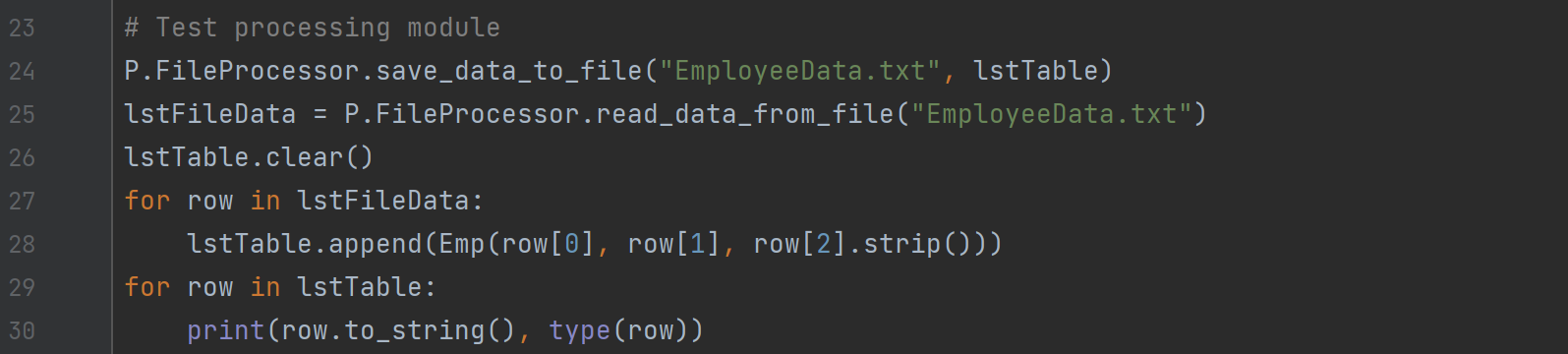


Figure 3 Testing ProcessingClasses Module in TestHarness.py

The fourth portion of the script is the testing of the *IOClasses* module, as shown in Figure 4. The *print\_current\_list\_items()*, *print\_menu\_items()*, *input\_menu\_options()*, and *input\_employee\_data()* functions in the *EmployeeIO* class were tested.

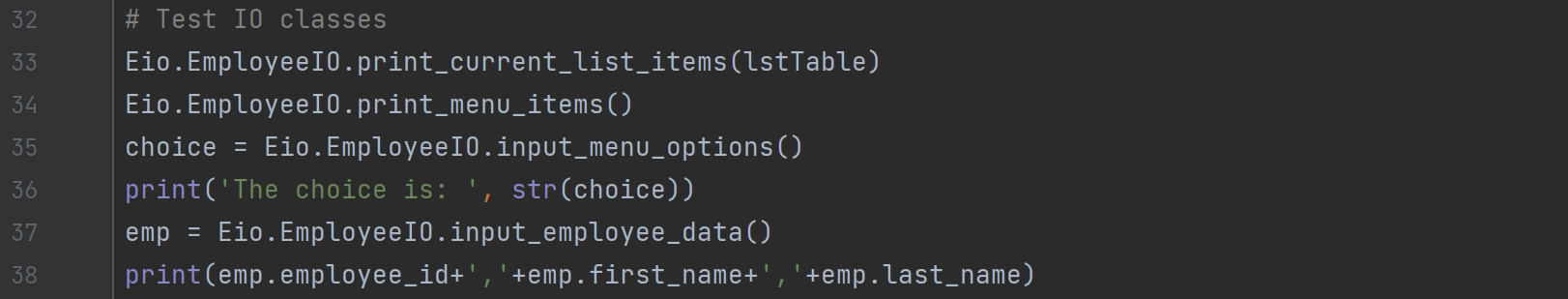


Figure 4 Testing IOClasses Module in TestHarness.py

## Running the TestHarness.py

As shown in Figure 5, the script tested the three modules “*DataClasses.py*”, “*ProcessingClasses.py*”, and “*IOClasses.py*” successfully. It proved that these modules are working as intended and ready for importing in the main module.

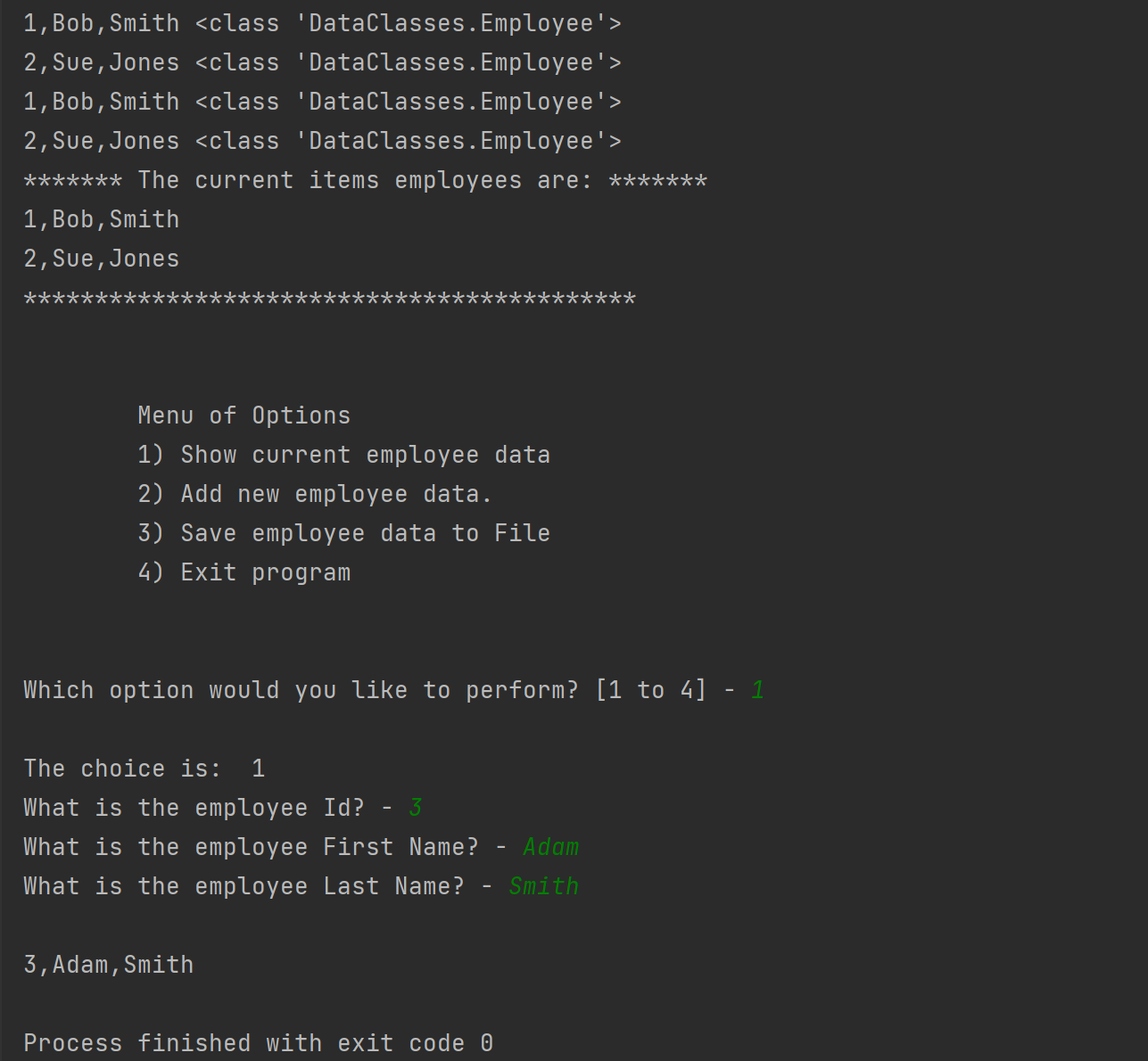


Figure 5 Running the TestHarness.py

# Main Module

The main module is composed of four sections: change log, module import, variable and constants, and main program. As shown in , the

demonstrating a Python script to operate on a class of product object data. The steps in this demonstration are shown below.

*Load the data from “EmployeeData.txt” into a list of employee objects.*

*While True:*

*{*

*Display menu of options to user to choose from.*

*Receive user choice between options 1-4:*

* + *If option 1: Display current employee data*
  + *If option 2: Add new employee data into the list of objects*
  + *If option 3: Save employee data into file*

*Use Exception to handle incompatible type of input data*

* + *If option 4: Exit the program, break*
  + *Else: Exception handling to let user re-enter choice*

*}*

## Writing the Script in PyCharm

I created a new project folder “C:\\_PythonClass\Assignment09\”, in which the script “Main.py” was created and tested. The script was completed with its various sections described in detail as follows.

1. Change History: Figure 6 shows the history in creating the script.

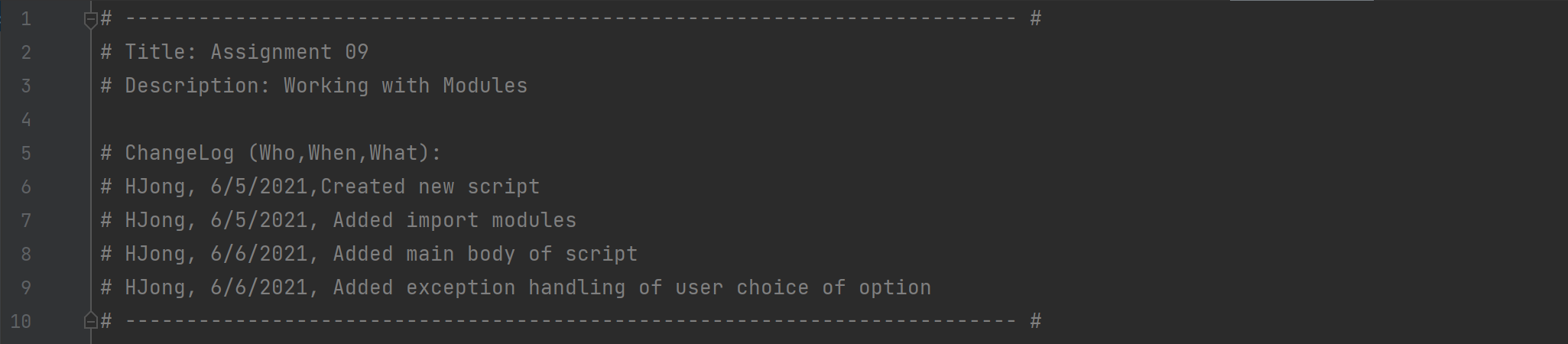


Figure 6 Change Log of the Script

1. Import modules: Figure 7 shows the *Employee* class in the *DataClasses* module, *FileProcessor* class in the *ProcessingClasses* module, and the *EmployeeIO* class in the *IOClasses* module are imported.

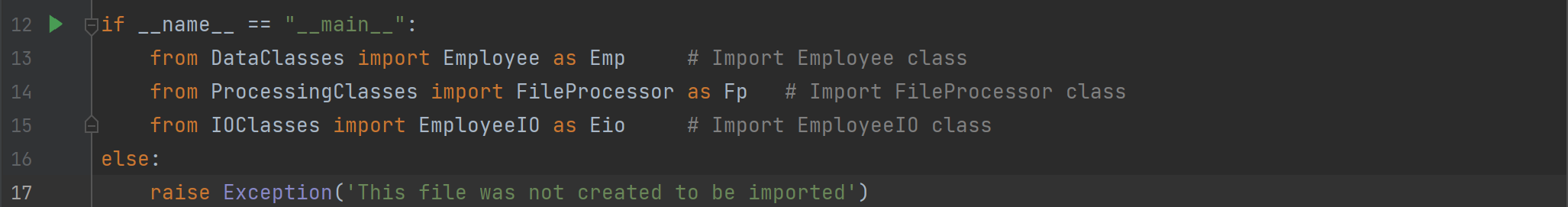


Figure 7 Importing Modules

1. Data definition: Figure 8 shows the variables in the script. The variables are very simple because of the encapsulation of data using classes.

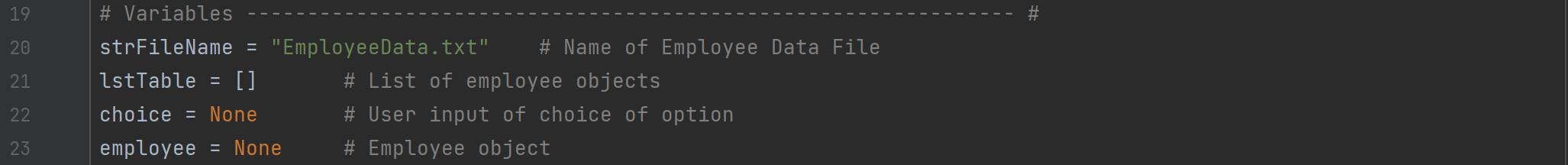


Figure 8 Data Declaration Section of the Script

1. Main program – In the main program, the employee data are read from the file into a list of string data. The employee data are stored into a list of employee data objects. Then the menu of options is displayed for the user to select action. The program performs the action based on the user selection. The menu options are print current employee data, add employee data, save employ data into the text file, and exit the program. The script also handles an error when user enter the choice outside of the [1-4] range. The scrip in the main program is shown in Figure 9.

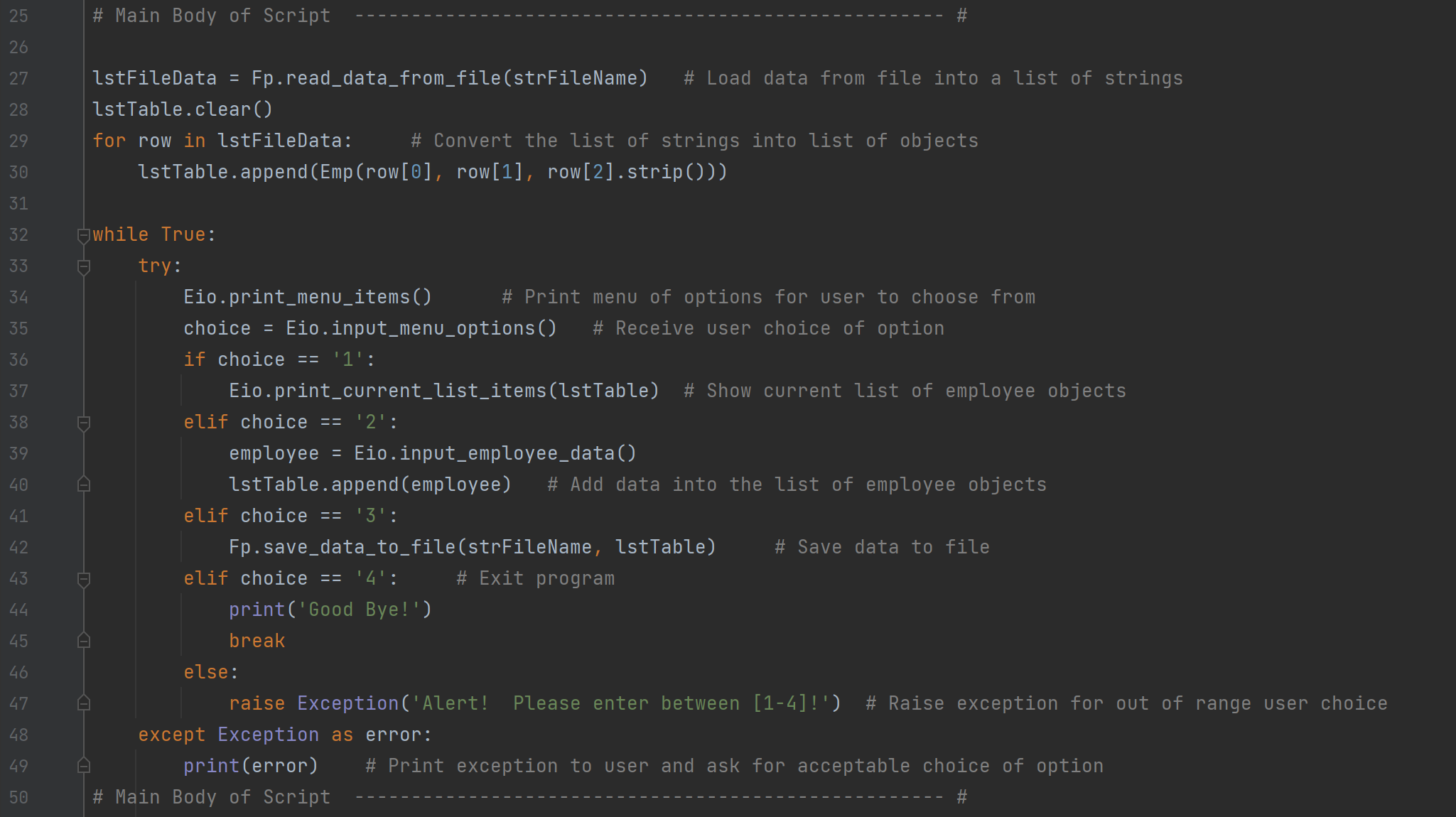


Figure 9 The Main Program

## Executing the Script in PyCharm

Before the run, there was a text data file “*EmployeeData.txt*” that stored two employee data in the hard drive, as shown in Figure 10.

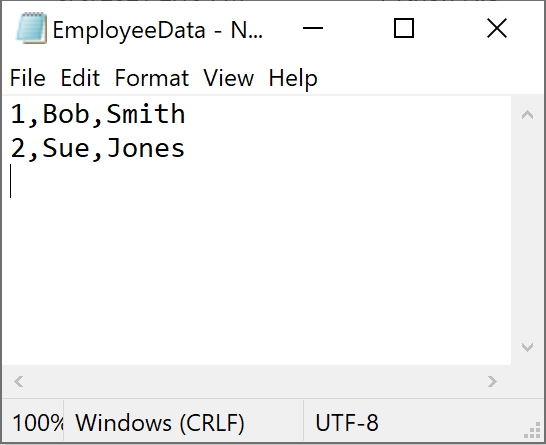


Figure 10 The “EmployeeData.txt” With Its Content

The main module was executed according to the following steps.

1. The program reads the employee data from the file into a list of string.
2. A list of employee objects was created based on the list of string.
3. The program prints out the menu of options for the user to select
4. Select option 1 to display the current employee data
5. Select option 2 to add new employee data (3, Adam, Smith)
6. Try error handling of employee name being numeric
7. Save employee data to file
8. Try error handling of user choice input not in [1-4]
9. Exit the program

Figure 11 shows the run results of steps 1-5.

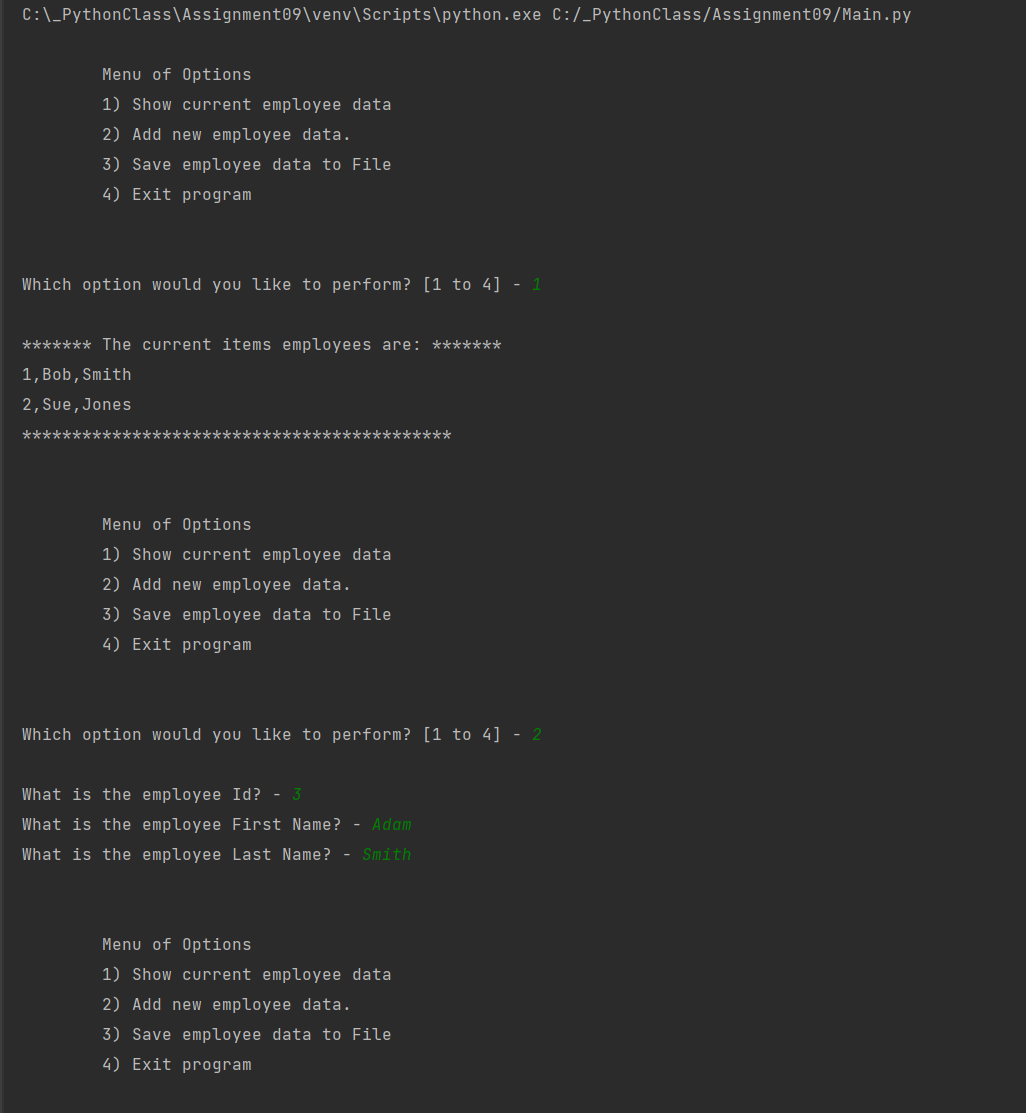


Figure 11 Run Results in PyCharm Part 1: Show Data and Add Data

Figure 12 shows the run results of steps 6-9.

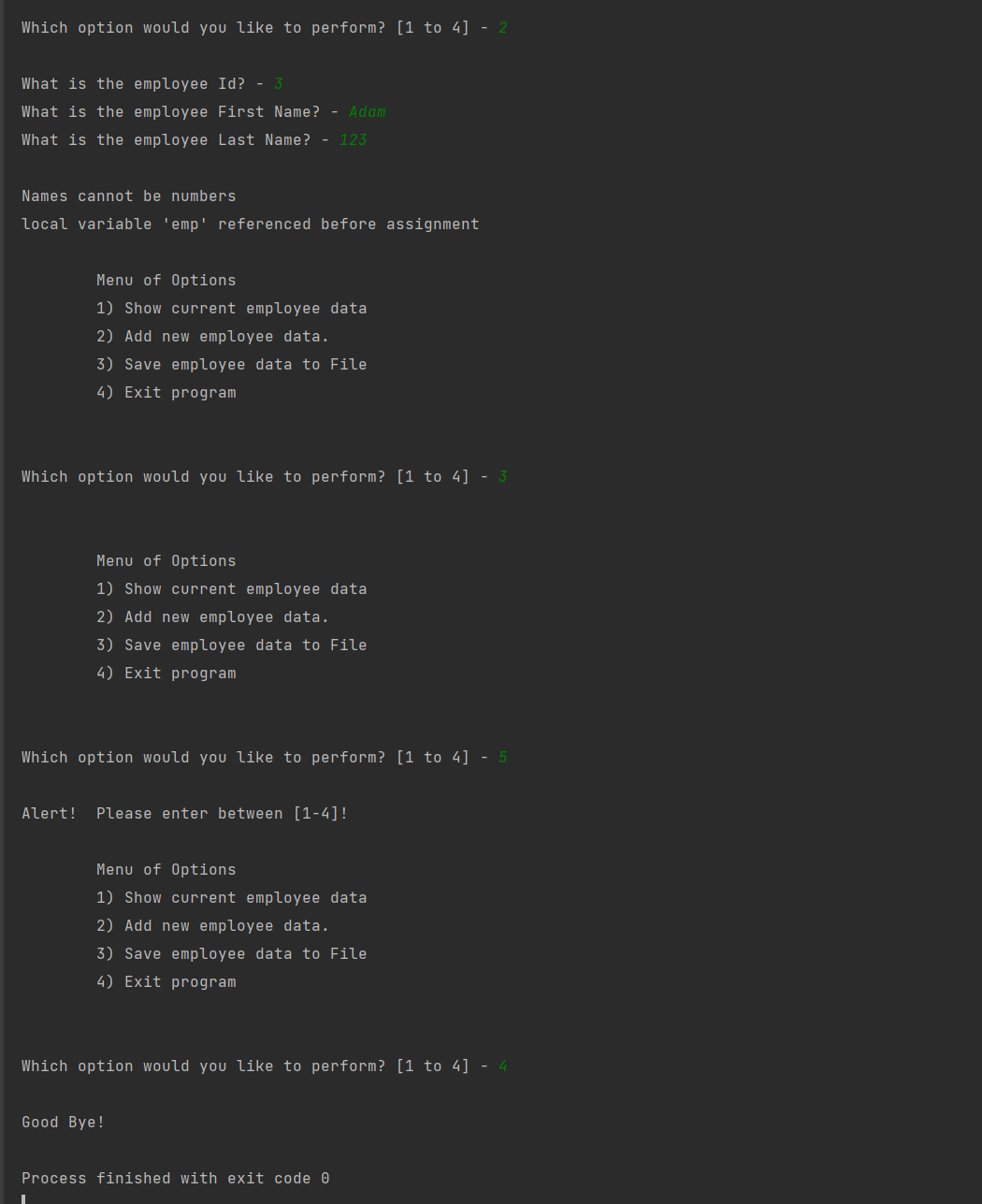


Figure 12 Run Results in PyCharm, Part 2: Save Data, Exception Handling, and Exit

## Checking the Data File After PyCharm Run

Finally, I opened “*EmployeeData.txt”* to verify (3,Adam, Smith) was added, which is shown in Figure 13.

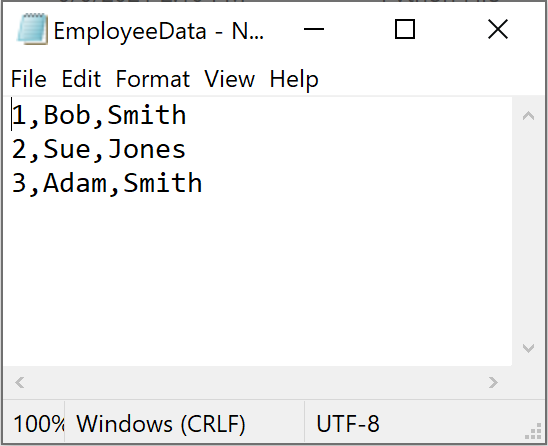


Figure 13 Employee Data in “EmployeeData.txt” After the Run in PyCharm

## Executing the Script in Windows Command Prompt

I went to “C:\\_PythonClass\Assignment09\” and run Main.py in Windows command prompt. Same steps in the PyCharm run were followed, but one more employee data was added (4, John, Doe). I also demonstrated a successful exception handling. The results are shown in the screenshot in Figure 14.

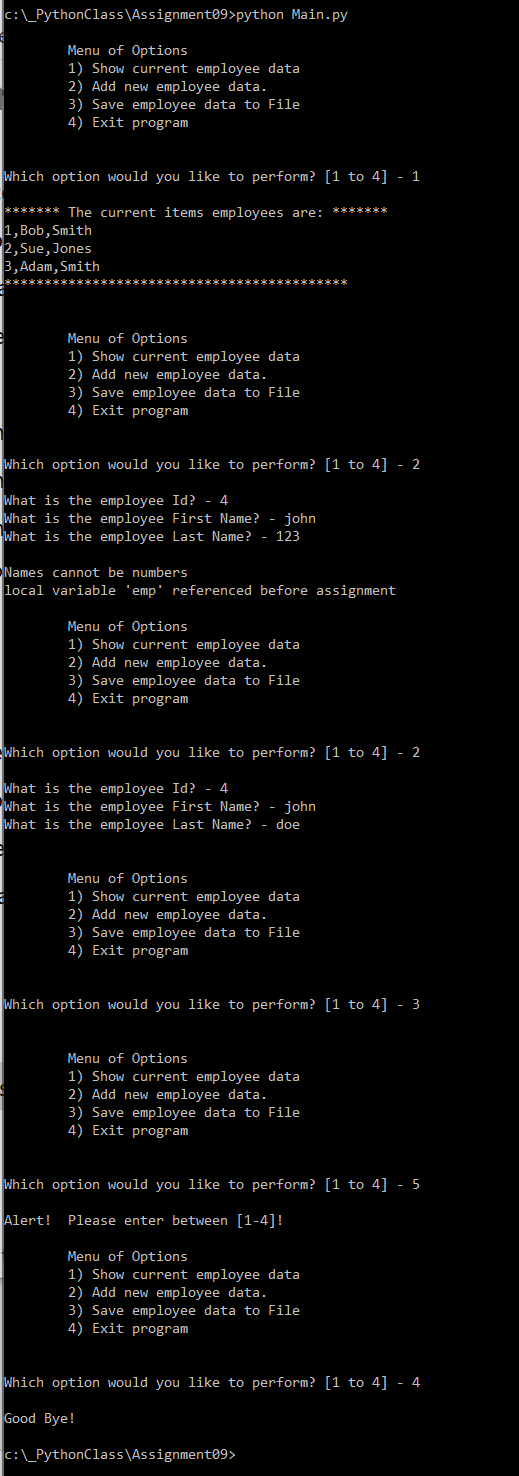


Figure 14 Run Results in Windows Command Prompt

## Checking the Data File After Windows Command Run

After the Windows command run, the file “*EmployeeData.txt*” in the working folder had four employee data, as shown in Figure 15. The script was thus validated in both PyCharm and Windows command.

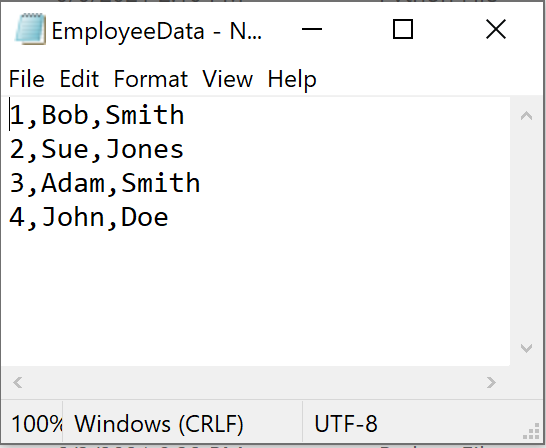


Figure 15 Employee Data in “EmployeeData.txt” After the Run in Windows Command

# GitHub Repository

I dropped the assignment files into the local repository folder as shown in Figure 13.

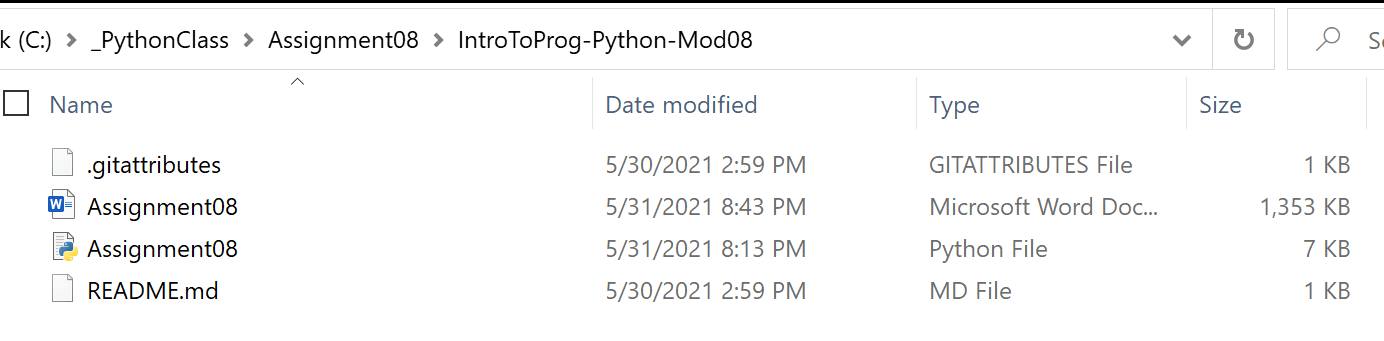


Figure 20 Assignment Files Copied into Local Repository Folder

I used GitHub Desktop to upload the files to the GitHub repository, as shown in Figure 14.

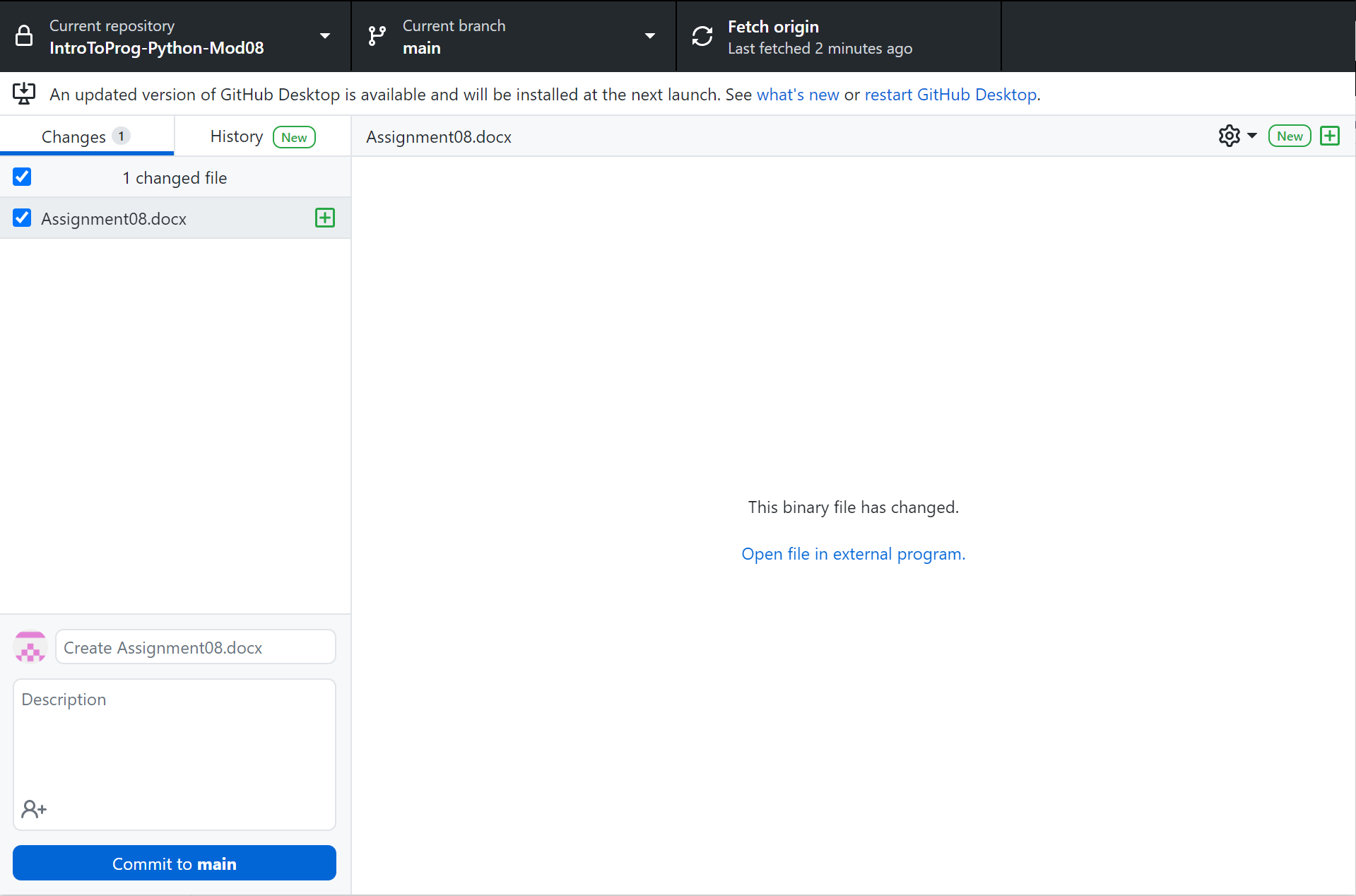


Figure 21 Uploading Assignment Files to GitHub.com

# Summary

This report summarizes the steps I took to implement the script and demonstrate it by test running the code in PyCharm and Windows command. This module has been the most difficult part of this course so far. In particular, the concept of class versus object, field versus attribute, property setter versus property getter, and private attribute versus public attribute requires much practice of many short scripts to solidify the learning. The practice of these short codes to try out the behavior of class objects have shown to be very helpful.