

Denise Albano

3/15/22

Foundations of Programming: Python

Assignment09

<https://github.com/dla425/IntroToProg-Python-Mod09>

Modules in Python

Introduction

This paper will document the steps I took to create three script modules and a main module in Python. In order to complete the assignment, I watched the course video in module 9 by Randall Root, read chapter9 in text book, and reviewed the additional web pages and videos assigned.

Creating the Python Script

To start the assignment, I created a new project in PyCharm that uses the folder `_PythonClass\Assignment09`. Within this project, I created five new python script files, `DataClasses`, `ProcessingClasses`, `IOClasses`, `TestHarness`, and `Main`, and a text file, `EmployeeData.txt` (see figure 1).

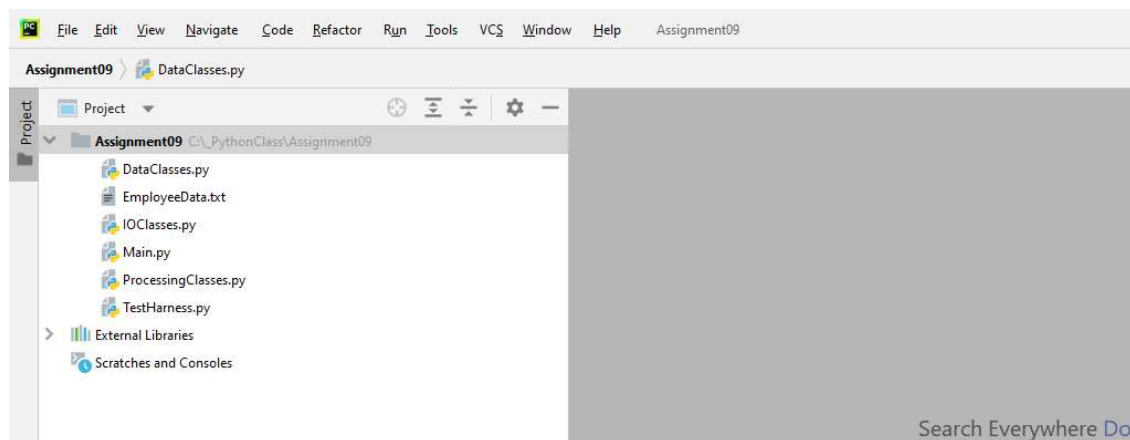


Figure 1: New project and the six files PyCharm

Data Classes

To create the `DataClasses` script, I used listing 6 and 9 from module 09 by Randal Root. These two listings contained the code to create the `Person` and `Employee` classes. To test these classes, I added code from listing 8 and 10 from module 09 by Randal Root to my `TestHarness` script.

Processing Classes

To create the `ProcessingClasses` script, I used listing 7 from module 09 by Randall Root. This listing contained the code to create the `FileProcessor` class. To test this class, I added code from listing 10 from module 09 by Randal Root to my `TestHarness` script.

Input/output Classes

To create the IOClasses script, I used listing 11 from module 09 by Randall Root. This listing contained the code to create the EmployeeIO class. To test this class and all the previous classes, I use the code from listing 12 from module 09 by Randal Root in my TestHarness script.

Main Module

To create the main module, I started with the code from listing 13 from module 09 by Randal Root. First I created an if ... else statement using the `__name__` variable to verify that the main.py script was running as the main program and referencing the three modules, DataClasses, ProcessingClasses, and IOClasses to include in the program. If the script was not running from the main program, an error message will be displayed. Instead of just calling the `import()` function for each module, I choose to use the 'from module_name import as name' method to import just a specific class within the module.

Next I added a try ... except statement to read in the data from the EmployeeData text file. If not exception occurs, then a while loop is executed that calls the `print_menu_items()` and the `input_menu_options()` functions from the IOClasses module. Once the user enters a menu choice, an if ... elif statement is used to call each function from either the ProcessingClasses or IOClasses module that is associated with the menu option. For option 1, the script calls the `print_current_list_items` function from IOClasses module using the parameter `lstTable`. For option 2, the script calls `appends lstTable` variable with new employee data by calling the `input_employee_data()` function from the IOClasses module. For option 3, the script calls the `save_data_to_file` function in the ProcessingClasses module to write the data in `lstTable` to the EmployeeData text file. For option 4, a `print()` function displays goodbye to the user and exits the program.

The final code for the main module is shown below:

```
# ----- #
# Title: Assignment 09
# Description: Working with Modules

# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added pseudo-code to start assignment 9
# DALbano,3.15.22,Modified code to complete assignment 9
# ----- #

# TODO: Import Modules
if __name__ == "__main__":
    from DataClasses import Employee as Emp
    from ProcessingClasses import FileProcessor as Fp
    from IOClasses import EmployeeIO as Eio
else:
    raise Exception("This file was not created to be imported")

# Main Body of Script ----- #
# TODO: Add Data Code to the Main body
# Load data from file into a list of employee objects when script starts
# Show user a menu of options
# Get user's menu option choice
    # Show user current data in the list of employee objects
    # Let user add data to the list of employee objects
    # let user save current data to file
    # Let user exit program
```

```

# Main Body of Script ----- #

lstTable = []

try:
    lstFileData = Fp.read_data_from_file("EmployeeData.txt")
    lstTable.clear()
    for line in lstFileData:
        lstTable.append(Emp(line[0], line[1], line[2].strip()))

    while True:
        # Show user a menu of options
        Eio.print_menu_items()
        # Get user's menu option choice
        strChoice = Eio.input_menu_options()

        if strChoice.strip() == '1':
            # Show user current data in the list of employee rows
            Eio.print_current_list_items(lstTable)
            continue

        elif strChoice.strip() == '2':
            # Let user add data to the list of employees
            lstTable.append(Eio.input_employee_data())
            continue

        elif strChoice.strip() == '3':
            # Let user save current data to file
            Fp.save_data_to_file("EmployeeData.txt", lstTable)
            continue

        elif strChoice.strip() == '4':
            print("Goodbye")
            break

except Exception as e:
    print("There was an error!")
    print(e, e.__doc__, type(e), sep='\n')

```

Running the program:

I first ran the TestHarness script in PyCharm to confirm that my modules DataClasses, ProcessingClasses and IOClasses were working correctly (see figures 2-4). Then I ran the main program in PyCharm and entered in each menu choice to confirm that the script was working. Screenshots of the output in PyCharm is shown in Figure 5 as well as the final contents of the EmployeeData.txt file in Figure 6.

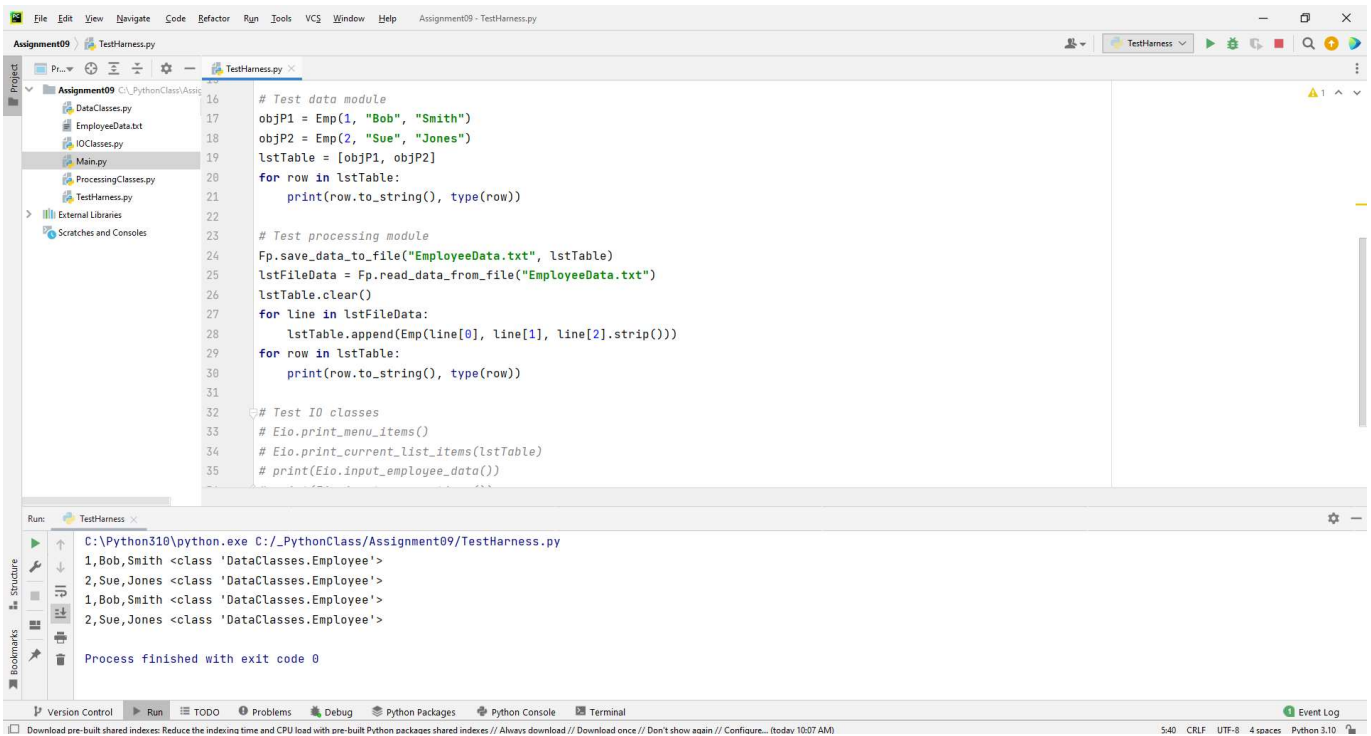


Figure 2: Screenshot of testing the Person, Employee and FileProcessor classes

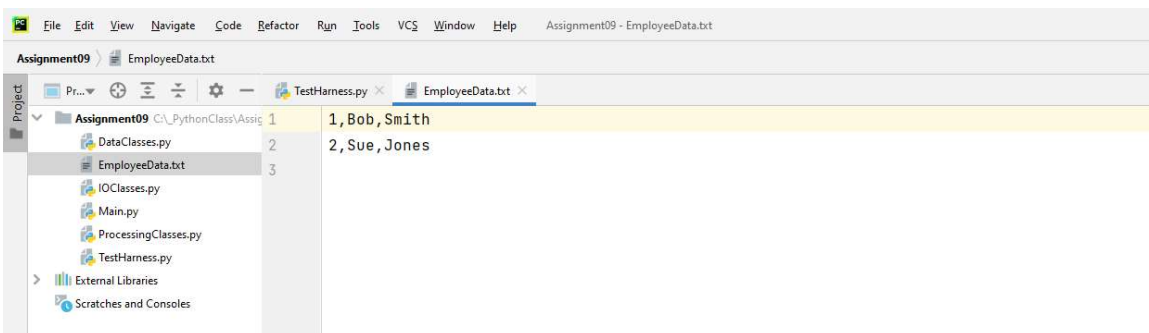


Figure 3: Screenshot of EmployeeData.txt

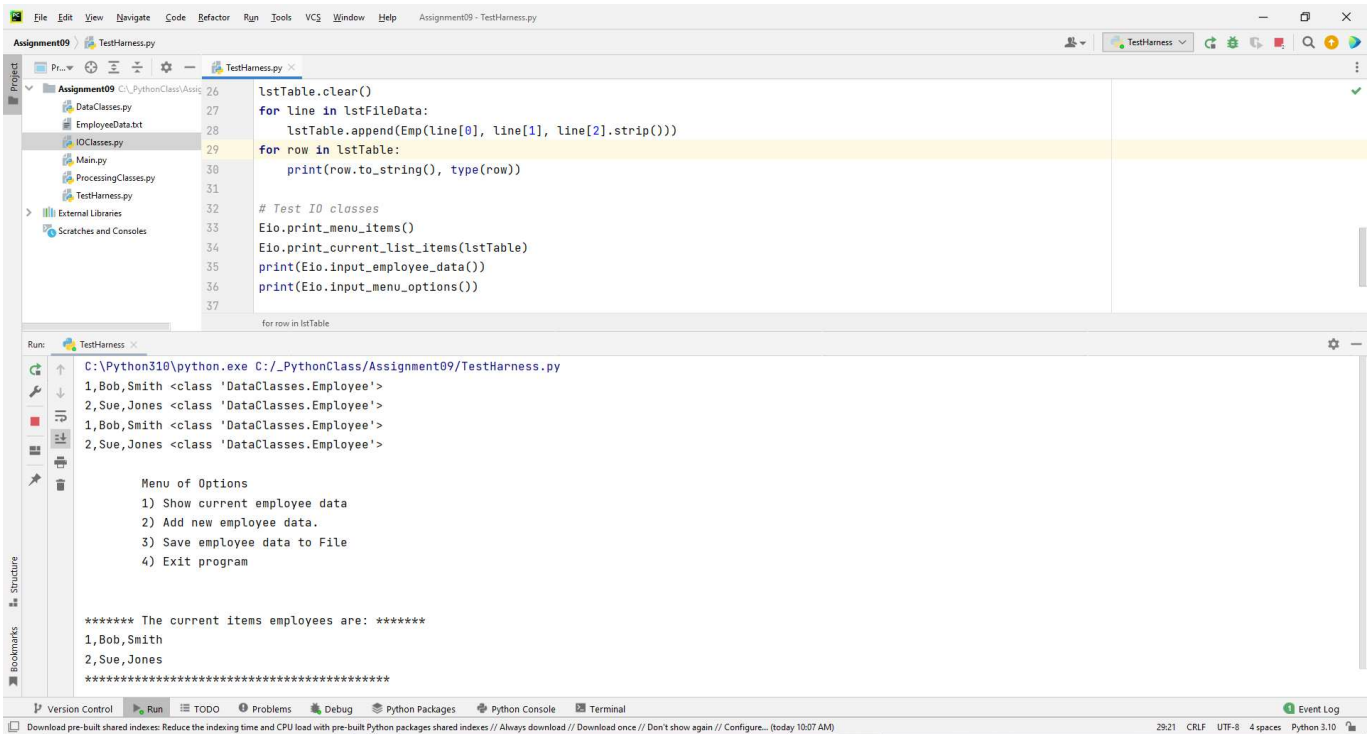
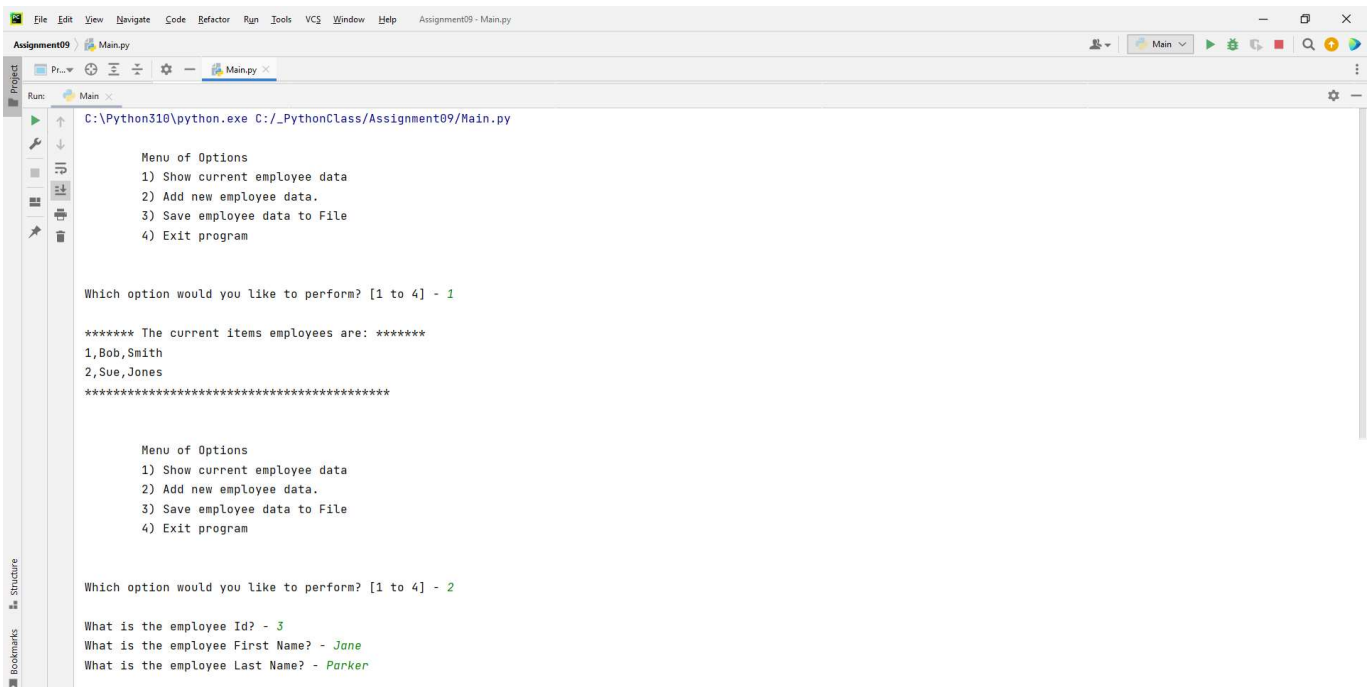


Figure 4: Screenshot of testing the Person, Employee, FileProcessor and Employee IO classes





```
Menu of Options
1) Show current employee data
2) Add new employee data.
3) Save employee data to File
4) Exit program

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

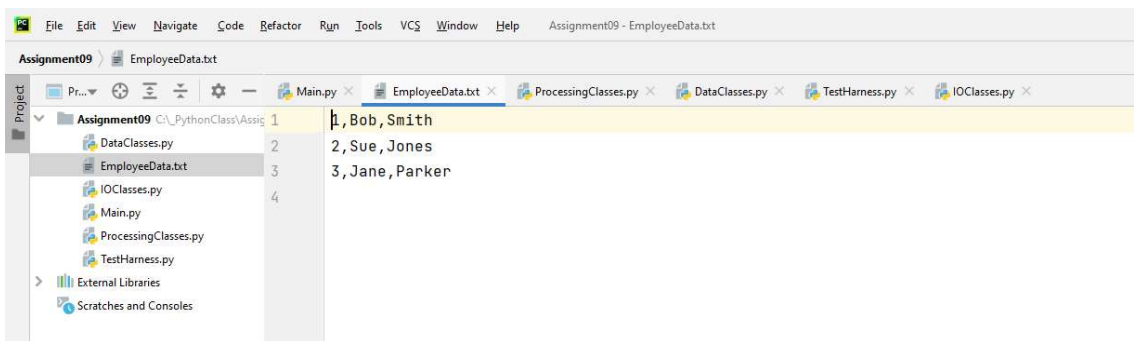
Menu of Options
1) Show current employee data
2) Add new employee data.
3) Save employee data to File
4) Exit program

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

Goodbye

Process finished with exit code 0
```

Figure 5: Screenshot of output from main module



```
1, Bob, Smith
2, Sue, Jones
3, Jane, Parker
4
```

Figure 6: Screenshot of EmployeeData.txt after running main module

Post Files to GitHub

After creating a GitHub repository named “IntoToProg-Python-Mod09” in my account, I uploaded the Assignment09 files and committed the changes.

Figure 7: Screenshot of module 9 files in GitHub

Summary

To complete this assignment I needed to understand the difference between a class and a module. It was also helpful to know what the “main” module and the “__name__” system variable were, and how to import one module into another. I found the examples in the course videos, textbook, and additional webpages and video to be extremely helpful in understanding the concepts needed to create a working program.