Ryan Fear

12/13/2021

IT FDN 110 A

Assignment 09

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

Modules and Main Program

# Introduction

This document will discuss how to recursively display an option menu for the user to navigate. The user will be able to repeatedly show the current data, add new data, and save data to a text file. The programming example in this document introduces a new concept of creating modules to handle different aspects of the program. This concept is fundamental to python programming and allows for more control when working with large project files.

# Problem Statement

The goal of this program is to load data, work with data, and save the data to a file. The data to handle will be class objects. The program must first read in data from the text file ‘EmployeeData.txt”. Then the program must recursively print a menu of options to the user. From the displayed menu, the user selects an option then the program performs the selected operation. The user can select from the following four options. One, show the current data. Two, add employee ID, a first name, and a last name as objects. Three, save the data to a file. Four, exit the program. The menu system is recursively printed to the screen after each menu option execution. Only when option four is selected will the program stop executing and exit.

# Data Classes Module

The DataClasses.py module contains two classes, Person and Employee. The Employee class inherits the Person class. The Person class is shown in Figure 1, note the document string is hidden for clarity. The Employee class is shown in Figure 2 and the document string is also hidden for clarity.

The Person Class has two attributes, first\_name and last\_name. The property definition does not allow for numbers, so if a name is trying to be defined as an integer an exception is raised and the user will have to enter data in the correct format.

The Employee Class has three attributes, two of which are inherited from the Person class. The employee\_id attribute is unique to this class. The property definition of the employee ID checks to see that if an integer has been entered, if not an exception will be raised. In the main program, discussed later, a try/except block checks if an employee ID is already in use or not. If the employee ID is a string or the employee ID is already in use, then the user will have to reenter the data again.



Figure : Person Class

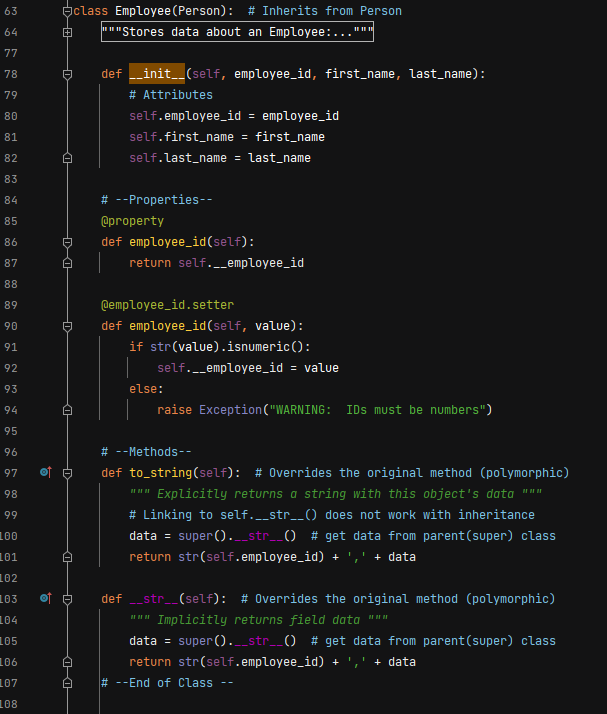


Figure : Employee Class

# ProcessingClasses Module

The ProcessingClasses.py contains two static methods, save\_data\_to\_file and read\_data\_from\_file. The entire module is shown in Figure 3 with document strings hidden for clarity. The read\_data\_to\_file method is used to read data into the program when the main program initially runs. The save\_data\_to\_file method is used when the user selects menu option 3 and decides to save the data to a file.

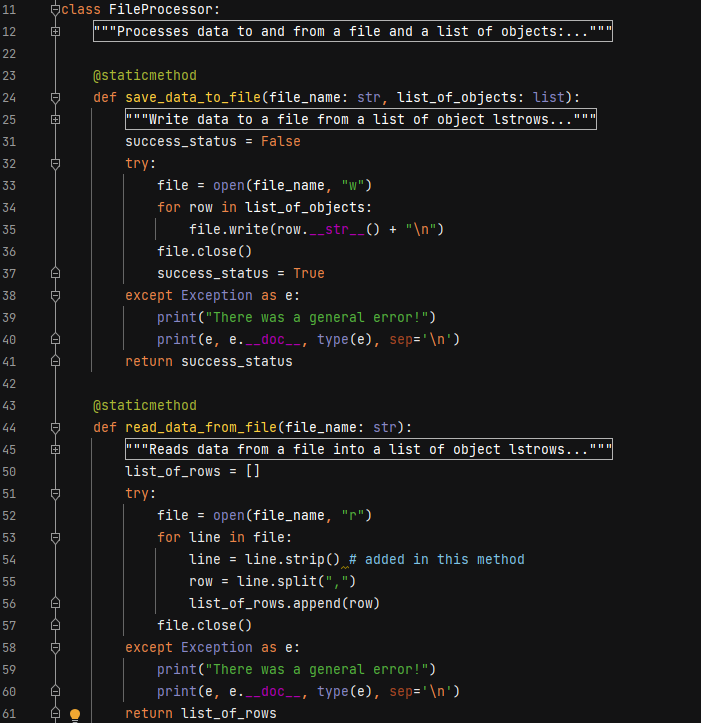


Figure : Processing Classes Module

# Input/Output Module

The IOClasses.py contains four static methods, print\_menu\_items, input\_menu\_options, print\_current\_list\_items, and input\_employee\_data. The entire module is shown in Figure 4 with documentation strings hidden for clarity. The print\_meu\_items is used to repeatedly print the list of menu options to the user. The input\_menu\_options method is used to gather the user input on the menu item they would like to use. The print\_current\_list\_items method is used to print all of the current data in the data list to the screen. The input\_employee\_data method allows additional employee data to be added to the list of objects. This method calls the DataClases.py module to define a new employee object.

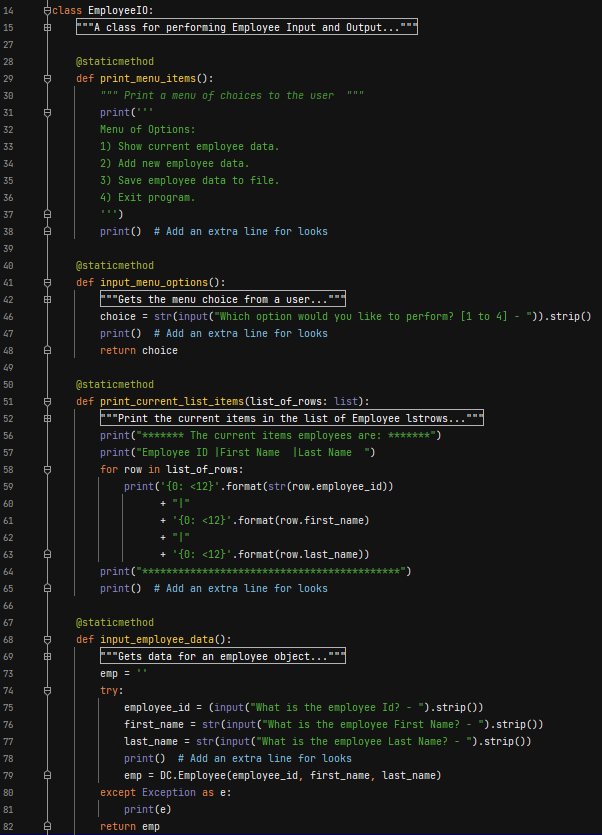


Figure : IOClasses Module

# Main Body of Script

Before the main script is run the previous modules are imported, see Figure 5. The modules are imported on lines 11-13 with given aliases. Global variables to be used through the main script are defined on lines 18 and 19. The Main.py script is shown in Figure 6. First the data is read in line by line from the file ‘EmployeeData.txt’ and printed to the screen this is completed using the code in lines 23-27.

Next the menu system is printed to the screen in line 30 and ask the user for their input on line 31. If the user choice is 1, the current list of items is printed to the screen (line 34).

If the user choice is 2, the code in lines 36-52 executes. Here a new employee is defined within a try/except block. The try/except block is set up to handle errors when a new employee ID entered already exist in the list of data. The try/except block also handles exceptions raised from the Person and Employee class. If the user tries to enter a string for employee ID or integers for a first or last name an exception is raised and the user will have to input correct data.

If the user choice is 3, the code on line 55 executes and the data is saved to the file ‘EmployeeData.txt’. A message is then printed to the screen.

If the user choice is 4, the message on line 59 is printed to the screen and the program exits.

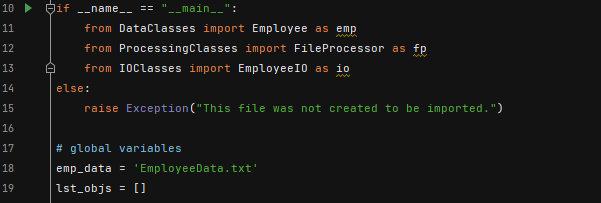


Figure : Import Module and Global Variables



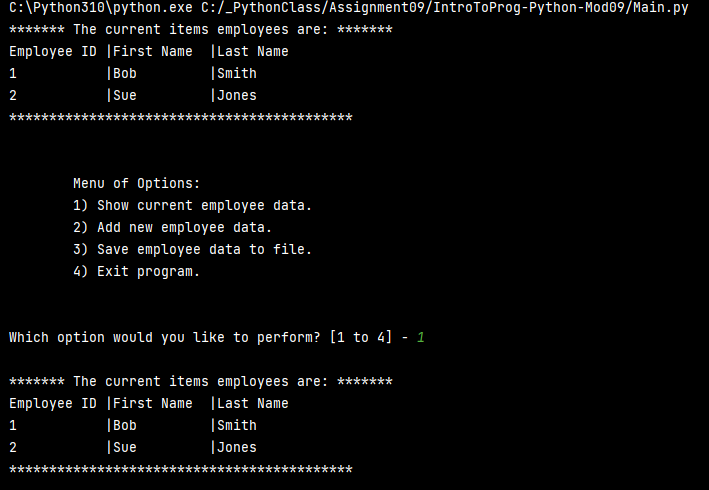
Figure : Main Script

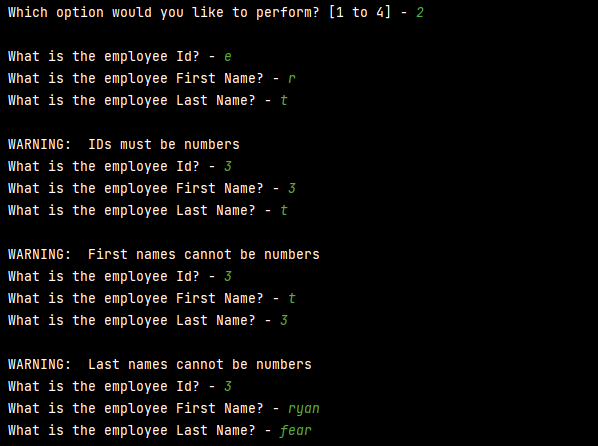
# Code Testing

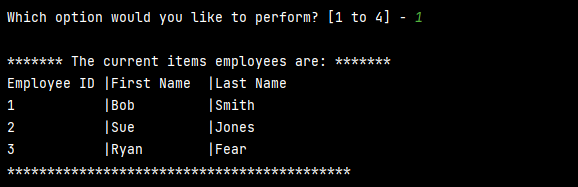
The code was tested both in the PyCharm environment and the Command Window (CMD). See the respective sections for example of code execution.

## PyCharm

The PyCharm execution of ‘Main.py’ is shown in Figure 7. The text file ‘EmployeeData.txt’ after code execution is shown in Figure 8.







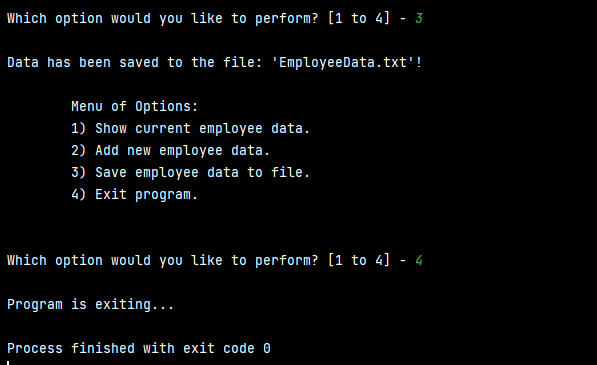


Figure : PyCharm Code Execution

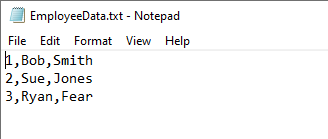
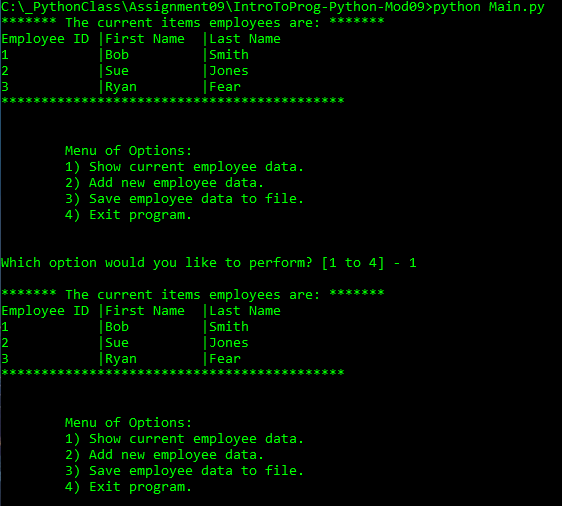
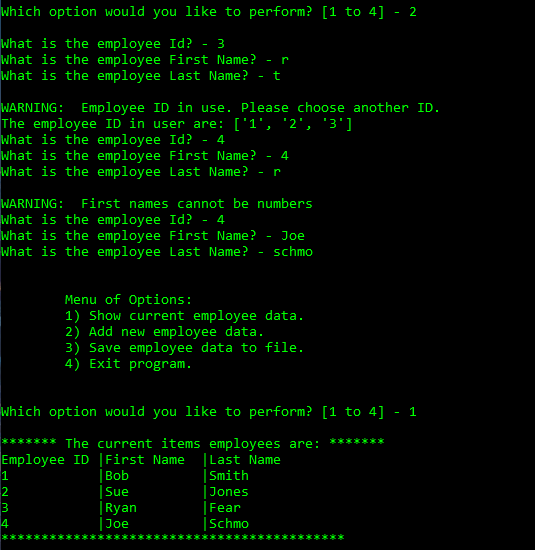


Figure : EmployeeData.txt After PyCharm Execution

## Command Window

The Command Window execution of ‘Main.py’ is shown in Figure 9. The text file ‘EmployeeData.txt’ after code execution is shown in Figure 10.





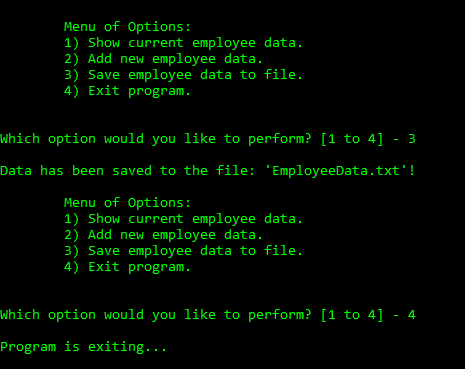


Figure : Command Window Code Execution

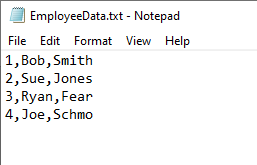


Figure : EmployeeData.txt After Command Window Execution

# Summary

In this document an example of how to recursively display an option menu for the user to navigate was discussed. First the problem statement of the example was discussed. The remainder of the document explained each of the concerns addressed using different modules. Finally code execution examples within PyCharm and the Windows OS command window were given. With the addition of the skills gained from this exercise a newer python programmer can gain greater control over their code flow and make more complex programs.