

Jeff Nelson

June 15, 2020

IT FDN 100 A

Assignment #9

<https://github.com/jnelson22/IntroToProg-Python-Mod09>

OOP Programming and Modules

Intro

In this paper, I will go over a few interesting and important items to remember that I learned from this week's lecture. Lastly, I will show a program uses the knowledge from this week to gather user input and create an employee database.

Week 8 Learnings

This week I learned about modules and few more concepts classes. Modules allow for an easy way to organize and re-use your code. Each module can have as many classes as you want. Modules can simply be used within your program by calling them with the import command. these modules are not intended to be run directly but rather called form a "main" module. Many time modules are used to preform different functions like data, processing or presentation. One big advantage to classes is easy of reusing code through the ability to inherit code from another class.

Product and Price Python Program Using Classes

This program's problem statement is to use modules and classes to create a database of employee names.

First, I started out with an outline and a "Test Harness" that allows for testing of each of the modules. Before creating the main module.

```
try:
    objP1 = D.Person(1, "Smith")
    objP2 = D.Person("Sue", "Jones")
    lstTable = [objP1, objP2]
    for row in lstTable:
        print(row.to_string(), type(row))
except Exception as e:
    raise print(e)
```

Figure 1: Person class test harness

Below is the output from the test in PyCharm.

```
Bob,Smith <class 'DataClasses.Person'>
Sue,Jones <class 'DataClasses.Person'>
```

Figure 2: Test output

Next was to test the employee class with the code below

```
# Test employee class
objP1 = Emp(1, "Bob", "Smith")
objP2 = Emp(2, "Sue", "Jones")
lstTable = [objP1, objP2]
for row in lstTable:
    print(row.to_string(), type(row))
```

Figure 3:Employee class test code

Testing the file processing class was completed with the code below.

```
# Test Employee processing module
Fp.save_data_to_file("EmployeeData.txt", lstTable)
lstFileData = Fp.read_data_from_file("EmployeeData.txt")
lstTable.clear()
for line in lstFileData:
    lstTable.append(Emp(line[0], line[1], line[2].strip()))
for row in lstTable:
    print(row.to_string(), type(row))
```

Figure 4: File processing testing code

Below is the out from employee class and file processing code.

```
1,Bob,Smith <class 'DataClasses.Employee'>
2,Sue,Jones <class 'DataClasses.Employee'>
1,Bob <class 'DataClasses.Person'>
2,Sue <class 'DataClasses.Person'>
1,Bob,Smith <class 'DataClasses.Employee'>
2,Sue,Jones <class 'DataClasses.Employee'>
```

Figure 5: Test code output

Last test harness code was created for the IO class. Calling the IO class to print menu, list of current data, the user input and then display the newly entered data.

```
# Test IO classes
eIO.print_menu_items()
eIO.print_current_list_items(lstTable)
print(eIO.input_employee_data())
print(eIO.input_menu_options())
```

Figure 6: IO class testing code

Below are the output of the test harness in PyCharm.

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

***** The current items employees are: *****
1, Bob, Smith
2, Sue, Jones
*****

What is the employee Id? - 3
What is the employee First Name? - jeff
What is the employee Last Name? - nelson

3,Jeff,Nelson
Which option would you like to perform? [1 to 4] - 2

2
```

Figure 7: IO class output

Now that all the classes have been tested it was time to create the main code that would call all the classes and allow the users to interact with the program. I started with an outline from my assignment 8 code with I copied over. Below you can see that initial outline.

```
# Main Body of Script ----- #

while(True):
    eIO.print_menu_items() # show user the menu
    strChoice = eIO.input_menu_options()

    # Process user's menu choice
    if strChoice.strip() == '1': # Show Current list of products
        continue # to show the menu

    elif strChoice == '2': # Add new employee data

        data_flag = 2 # set flag to double check if you want to save the file
        continue # to show the menu

    elif strChoice == '3': # Save Data to File

        data_flag = 3 # set flag that the new data has been saved
        continue # to show the menu

    elif strChoice == '4': # Exit Program
        if data_flag == 2:
            strChoice = IO.input_yes_no_choice("You have not saved new data. Are you sure you want Exit? (y/n) - ")
            if strChoice.lower() == 'y':
                print("Goodbye!")
                break # and Exit
            else:
                continue # to shoe the menu
        elif data_flag != 1:
            print("Goodbye!")
            break # and Exit
```

Figure 8: Outline of the main code

The first step was to read in the data from the file and create a list of objects. Below is the code that completes that task.

```
# Read in current file and build a list of objects
lstFileData = Fp.read_data_from_file(strFileName)
lstOfEmployeeObjects.clear()
for line in lstFileData:
    lstOfEmployeeObjects.append(Emp(line[0], line[1], line[2].strip()))
```

Figure 9: Initial data read in code

If the user selected 1 then they would be displayed the current data. This was done by calling the IO class to print the current list of items from the list of objects.

```
# Process user's menu choice
if strChoice.strip() == '1': # Show Current list of products
    eIO.print_current_list_items(lstOfEmployeeObjects)
    continue # to show the menu
```

Figure 10

Below is an example of the output from PyCharm of the current data.

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

***** The current items employees are: *****
1, Bob, Smith
2, Sue, Jones
*****
```

Figure 11

If the user selects 2 they will then be prompted to input and ID, first and last name of the employee. The data will then be passed to the data class where the object are then appended to the list.

```
elif strChoice == '2': # Add new employee data
    lstOfEmployeeObjects.append(eIO.input_employee_data())
    # data_flag = 2 # set flag to double check if you want to save the file
    continue # to show the menu
```

Figure 12

Here is an example of the output from PyCharm.

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

What is the employee Id? - 3
What is the employee First Name? - jeff
What is the employee Last Name? - nelson

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] - 1

***** The current items employees are: *****
1, Bob, Smith
2, Sue, Jones
3, Jeff, Nelson
*****
```

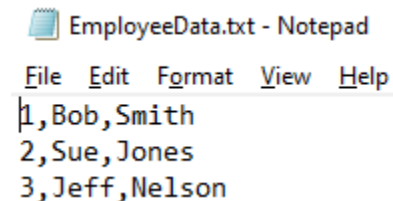
Figure 13

The 3rd choice the users has is to save the data to a file. This is completed by passing the file name and list of objects into the file processing class. Inside the class the code loop through adding the object rows to the text file.

```
elif strChoice == '3': # Save Data to File
    Fp.save_data_to_file(strFileName, lstOfEmployeeObjects)
    data_flag = 3 # set flag that the new data has been saved
    continue # to show the menu
```

Figure 14

Here is the output of the data that has been entered saved to a text file.



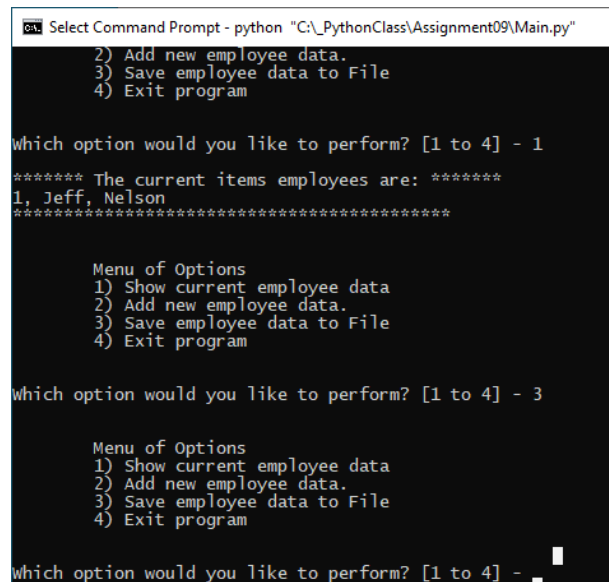
EmployeeData.txt - Notepad

File Edit Format View Help

1,Bob,Smith
2,Sue,Jones
3,Jeff,Nelson

Figure 15

Below is a screenshot of the program also running the windows command prompt.



```
Select Command Prompt - python "C:\_PythonClass\Assignment09\Main.py"
2) Add new employee data.
3) Save employee data to File
4) Exit program

Which option would you like to perform? [1 to 4] - 1
***** The current items employees are: *****
1, Jeff, Nelson
*****

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] -
```

Figure 16

Summary

In summary, I learned about modules and few more concepts classes. I went through the process of how I created the employee id, first and last name storage program. I found this assignment much easier than week eight. The ability to create modules and inherit classes allows for much better organization and re-use of code. These principles are the foundation of modern OOP. Overall, I'm happy with the way this assignment turned out and the new information I learned.