Giovanna Torres

3/5/2025

IT FDN 110 A Wi 25: Foundations Of Programming: Python

Assignment 6

GitHub Link: https://github.com/giozoar/IntrotoProg-Python-Mod06

Functions

Introduction

This week as a part of my Foundations of Programming course, I learned how to work with functions and classes, which allow me to organize my script in a logical, modular fashion and increase maintainability. I also learned about the Separation of Concerns pattern to know where to immediately find certain script functions by grouping them in logical containers of my script's functionality.

Creating the Script

After reading the acceptance criteria described in the Mod06-Assignment file, I began to work on my script using the PyCharm Community Edition IDE. I reused the header from the Assignment06-Starter.py file included in the module materials to display the necessary information.

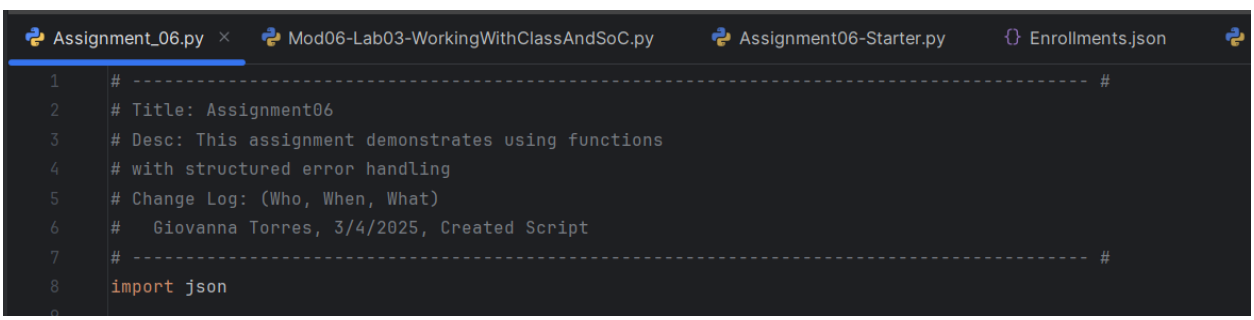

Figure 1 - Script Header

Afterwards, I added the ‘import json’ statement to my code, and copied the variables from the starter file, and verified the variables were consistent with components of the script body required to complete the assignment acceptance criteria.

```python
# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------
'''

FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Figure 2 - Declaring constants and variables

I then got to work on adding specific components called out in the acceptance criteria, such as creating two classes called ‘IO’ and ‘FileProcessor’ with descriptive document strings, and moving

script functions from the previous module underneath each class to act as methods for that class grouping.

Assignment_06.py × Mod06-Lab03-WorkingWithClassAndSoC.py Assignment06-Starter.py Enrollments.json Assignment05.py Mod0

```python
class IO:  11 usages
    """ This class groups the inputs and output functions called in this script.

    ChangeLog: (Who, When, What)
    Giovanna Torres, 3/4/2025, Created class

    :return: None
    """
    @staticmethod  7 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user

        ChangeLog: (Who, When, What)
        Giovanna Torres, 3/4/2025, Created function

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod  1 usage
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

        ChangeLog: (Who, When, What)
        Giovanna Torres, 3/4/2025, Created Function

        :return:
        """
        print(menu)
```

```python
        return student_data

class FileProcessor:  2 usages
    """ This class groups the file and data processing functions called in this script.

    ChangeLog: (Who, When, What)
    Giovanna Torres, 3/4/2025, Created class

    :return: student_data
    """
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """This function reads the data from an existing JSON file into a variable.
            ChangeLog: (Who, When, What)
            Giovanna Torres, 3/4/2025, Created function

            :return: student_data
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages( message: "Text file must exist before running this script!", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data
    @staticmethod  1 usage
    def write_data_to_file(file_name: str, student_data: list):
```

Figure 3 – Classes and methods

Per the acceptance criteria, I made sure to change my function code to include the output_error_messages function defined in the IO class to display my relevant error messages as a result of the try-except handling in the functions.

I then created a while loop to run my program infinitely until the user decides to break, and ask the user to enter student first name, last name, and registration course to register a student using the functions defined in the class layers at the beginning of the script. The user then prompts the program to display or save the data to an existing 'Enrollments.json' file, then exit out of the loop.

Much of this script reused existing statements from Module 5, so the behavior had been defined. The work required was mostly inserting the class methods in the right locations within the loop, as shown in the image below. This condensed the main body down quite a bit thanks to the previously defined functions.

```python
class FileProcessor:  2 usages
                file.close()

# Beginning of the main body of this script
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
while (True):

    # Present the menu of choices
    IO.output_menu(menu=MENU)
    menu_choice=IO.input_menu_choice()
    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        students=IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_courses(student_data=students)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop

print("Program Ended")
```

Figure 4 – Main body of script using class functions.

I then proceeded to test my code, discussed in the next section.

Testing the Script

After setting up my initial 'Enrollments.json' data file with data from module 4 and running the script in the IDE to ensure that the correct outputs were being displayed and created in the directory, I proceeded to run the script in the Command Prompt terminal window. I changed the directory over to my 'A06' file within my PythonCourse directory and ran the script.

I tested multiple cases, including adding multiple names, and adding characters that weren't the numerical in the student name fields, and 1-4 menu option values. As I expected, the script would prompt the user to select another option if the input was invalid. It would also display multiple entries of student registrations and write them to the file as expected. The outputs were the same in either window, as seen below.

Assignment_06.py × Mod06-Lab03-WorkingWithClassAndSoC.py Assignment06-Starter.py Enrollments.

Run Assignment_06 ×

```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 2

-------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000
Student John Wick is enrolled in Philosophy 1000
-------------------------------------------------

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number:
```

ythonCourse > A06 > Assignment_06.py

Assignment_06.py × Mod06-Lab03-WorkingWithClassAndSoC.py Assignm

Run Assignment_06 ×

```
-----------------------------------------

Enter your menu choice number: 1

What is the student's first name? Max
What is the student's last name? Koszlov
Enter the course name. Python 1001

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------

Enter your menu choice number: 2

--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000
Student John Wick is enrolled in Philosophy 1000
Student Max Koszlov is enrolled in Python 1001
--------------------------------------------------
```

onCourse > A06 > Assignment_06.py

Assignment_06.py × Mod06-Lab03-WorkingWithClassAndSoC.py Assignment06-Starter.py

Run Assignment_06 ×

```
Enter your menu choice number: 1

What is the student's first name? 879y4r
That value is not the correct type of data!

-- Technical Error Message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 1

What is the student's first name? John
What is the student's last name? j984wtfrh
That value is not the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Assignment_06.py × Mod06-Lab03-WorkingWithClassAndSoC.py Assignment06-Starter.py

Run Assignment_06 ×

```
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000
Student John Wick is enrolled in Philosophy 1000
Student Max Koszlov is enrolled in Python 1001
-------------------------------------------------

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 3


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number:
```

```
Enter your menu choice number: 0

Please, choose only 1, 2, 3, or 4


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 4

Program Ended
```

PythonCourse > A06 > Assignment_06.py

```
Enter your menu choice number: 3

The following students have been registered:
-------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000
Student John Wick is enrolled in Philosophy 1000
Student Max Koszlov is enrolled in Python 1001
Student Greg Yarros is enrolled in Python 2000
-------------------------------------------------

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 4

Program Ended

Process finished with exit code 0
```

Figure 5 - Testing inputs in PyCharm IDE.

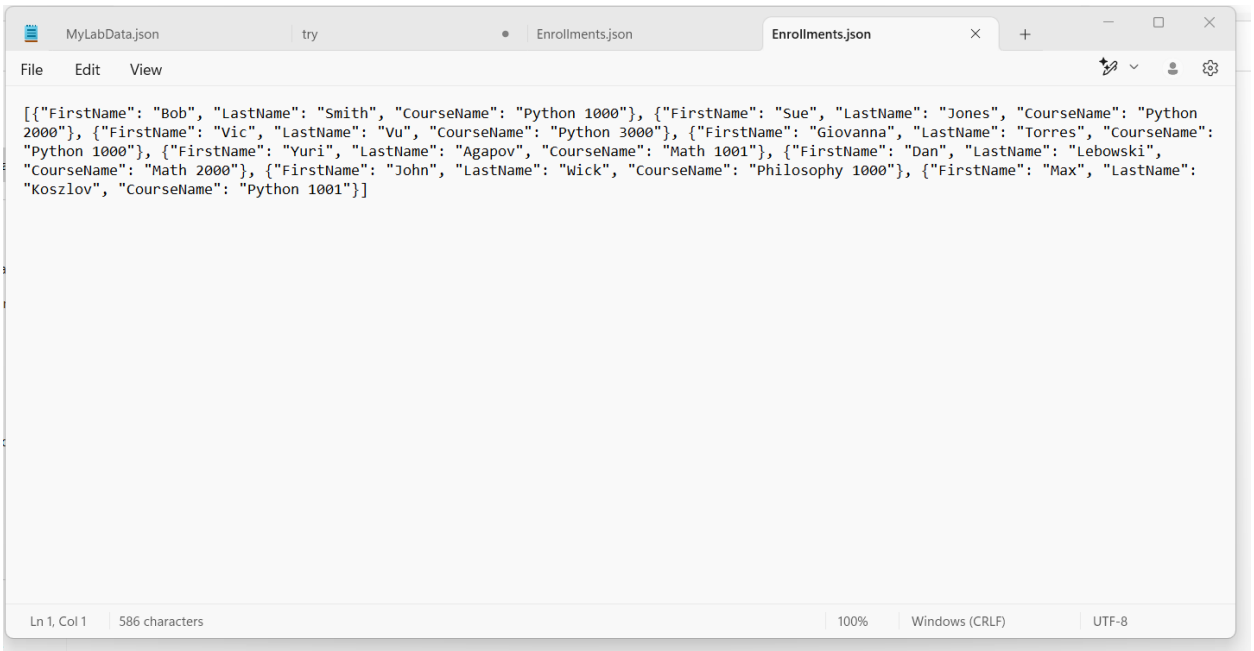

Figure 6 - Outputs in JSON file.

Command Prompt - python A

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\giova>cd C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A06

C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A06>python Assignment_06.py

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 1

What is the student's first name? nf937r
That value is not the correct type of data!

-- Technical Error Message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 1

What is the student's first name? nivsdncw
What is the student's last name? cnfc934i7
That value is not the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
```

Command Prompt - python /

```
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 1

What is the student's first name? Greg
What is the student's last name? Yarros
Enter the course name. Python 2000

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------

Enter your menu choice number: 2

--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000
Student John Wick is enrolled in Philosophy 1000
Student Max Koszlov is enrolled in Python 1001
Student Greg Yarros is enrolled in Python 2000
--------------------------------------------------

---- Course Registration Program ----
```

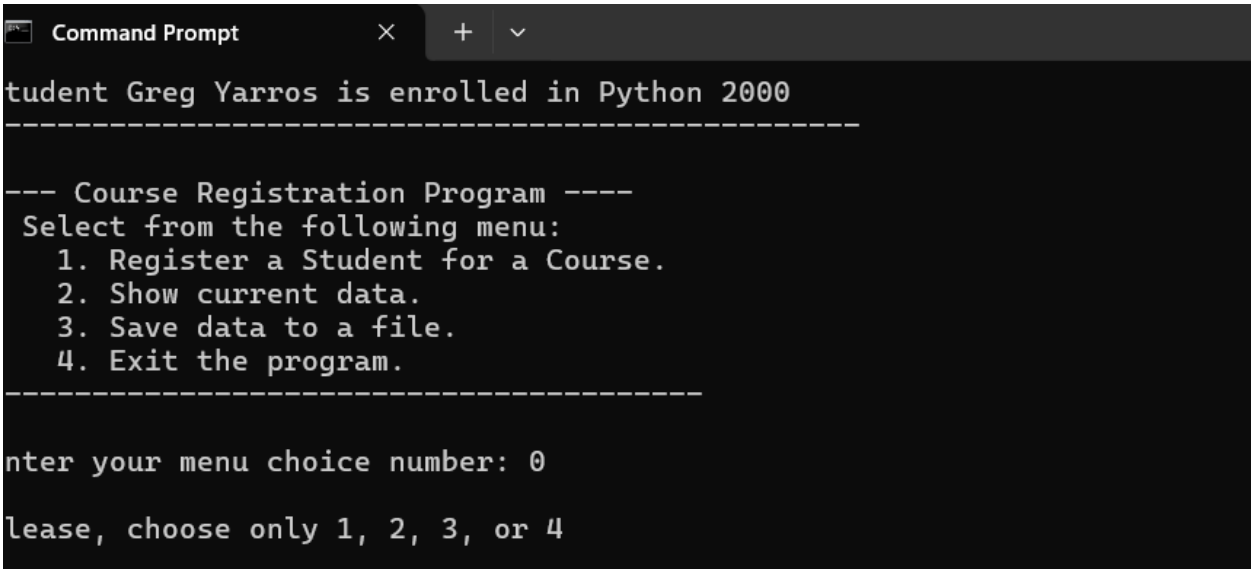

Figure 7 - Testing in Command Prompt window.

Summary

During this assignment, I learned about using classes and functions to organize my data per a Separation of Concerns pattern. The biggest learning curve I've found is in calling the class and function with the **correct parameter arguments.** Understanding the relationship between parameters and variables used throughout the script was difficult to wrap my head around, but once I realized that a parameter is local to the function and you can pass your global variables to them as necessary, it made it easier to grasp the concept. I still have gaps in understanding the @staticmethod decorator and the 'self' parameter, which I'm hoping to be able to understand a bit more in the coming modules.