Alvin Jappeth Isip
November 27, 2024
IT FDN 110 A – Foundations of Programming: Python
Assignment 06

FUNCTIONS

Introduction

On this module, it was discussed regarding organizing code in python. As we continue to learn more about Python programming, scripts will be more complex and become more large, and without proper organization of the code/script, the one writing or the one reading the script can be confused on the flow of the program.

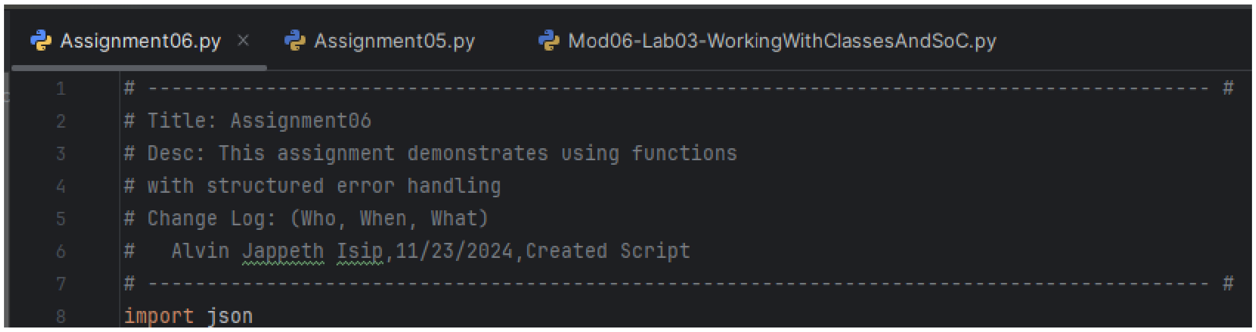
Module 6 Requirements

I did open the module 6 notes and assignment to see the requirements for this module. The assignment is very different from assignment 5 since on this module, we'll be using proper organization of codes but there are variables/constants/requirements that are the same with other assignments.

1. The program includes a class named FileProcessor.
2. The program includes a class named IO.
3. The program includes functions with the following names and parameters:
   - output_error_messages(message: str, error: Exception = None)
   - output_menu(menu: str)
   - input_menu_choice()
   - output_student_courses(student_data: list)
   - input_student_data(student_data: list)
   - read_data_from_file(file_name: str, student_data: list):
   - write_data_to_file(file_name: str, student_data: list):

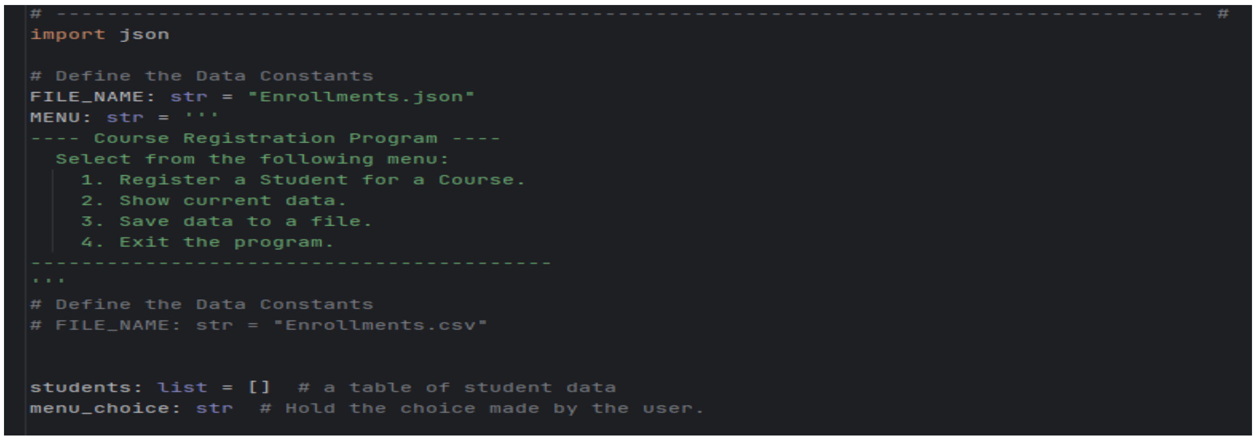
Writing the script

To start my coding, I opened the Assignment06-Starter.py and saved it as Assignment06.py. I did changed the header script to reflect what I'll be doing for this assignment.


Assignment06.py × Assignment05.py Mod06-Lab03-WorkingWithClassesAndSoC.py

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Alvin Jappeth Isip,11/23/2024,Created Script
# ------------------------------------------------------------------------------------------ #
import json
```

Then I added the needed variables and constants. So compared with other assignments, now there are few global variables since we'll be using functions, most of the variables will be local.



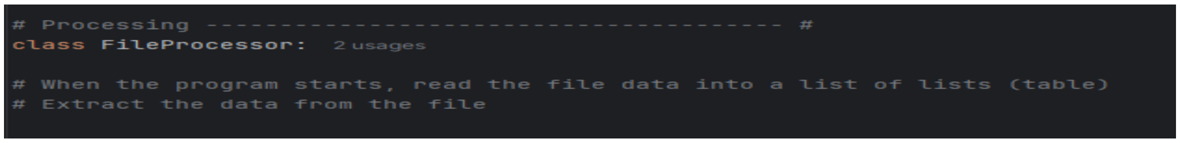
```
# ------------------------------------------------------------------------------------------ #
import json

# Define the Data Constants
FILE_NAME: str = "Enrollments.json"
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.
-----------------------------------------
'''
# Define the Data Constants
# FILE_NAME: str = "Enrollments.csv"


students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```

As the requirement on the assignment6, we need to include the following functions:

- output_error_messages(message: str, error: Exception = None)
- output_menu(menu: str)
- input_menu_choice()
- output_student_courses(student_data: list)
- input_student_data(student_data: list)
- read_data_from_file(file_name: str, student_data: list):
- write_data_to_file(file_name: str, student_data: list):

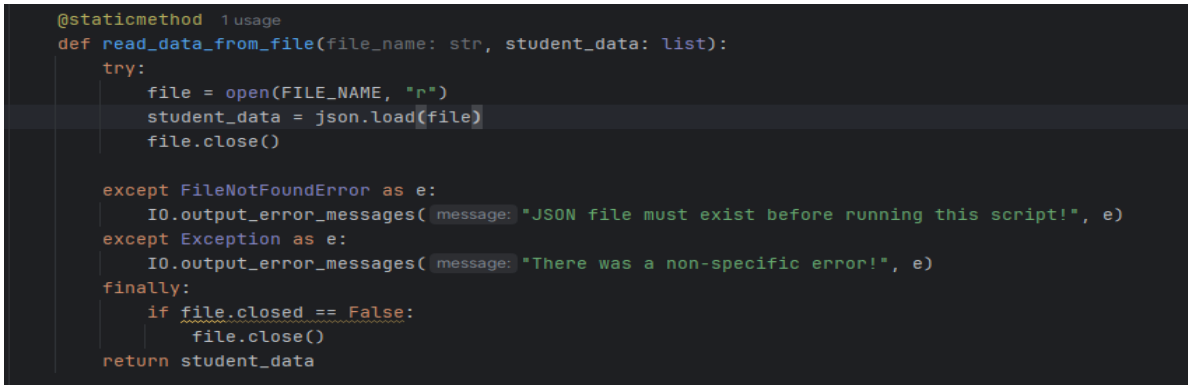
Aside from the function, we need to have two classes:
- FileProcessor
- IO

To code the class FileProcessor, wrote Class and the FileProcessor as seen on the screenshot below:

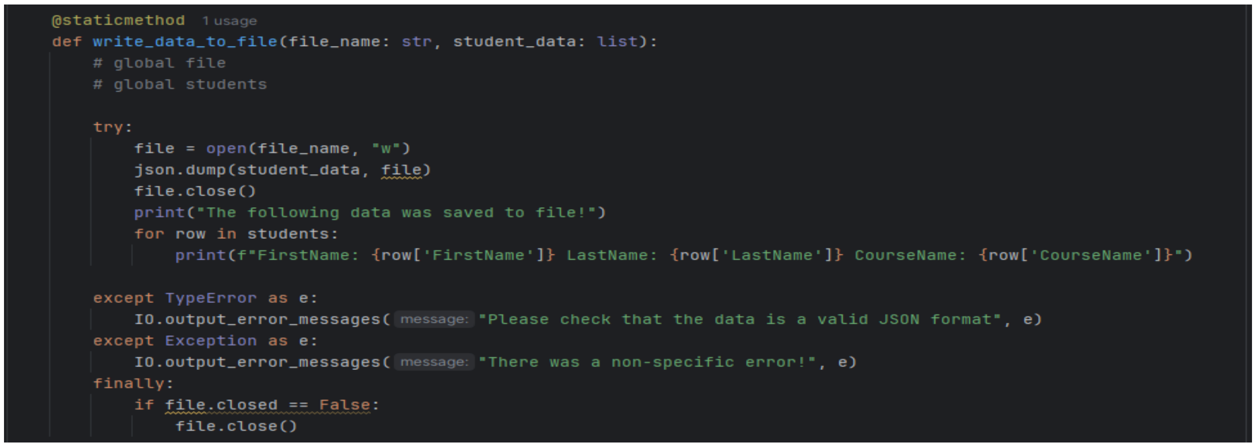


```
# Processing --------------------------------------- #
class FileProcessor:  2 usages

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
```

And under the FileProcessor, I've written the functions read_data_from_file and write_data_to_file. To define the function, we need to start if with word def and then the name of the function that you want to name.

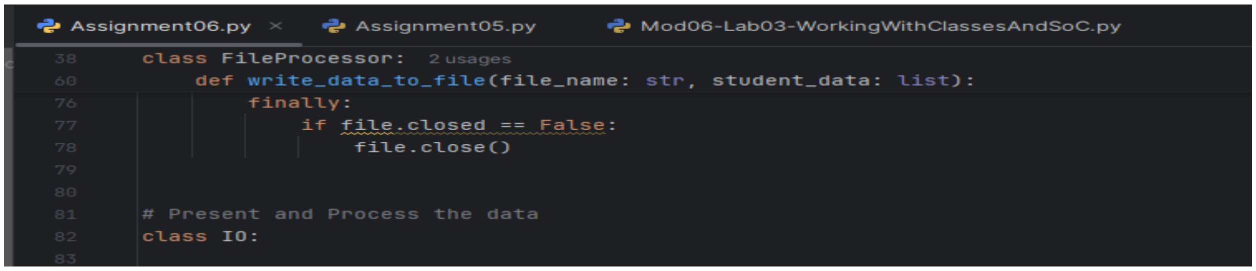


```
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        try:
            file = open(FILE_NAME, "r")
            student_data = json.load(file)
            file.close()

        except FileNotFoundError as e:
            IO.output_error_messages( message: "JSON 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):
        # global file
        # global students

        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            print("The following data was saved to file!")
            for row in students:
                print(f"FirstName: {row['FirstName']} LastName: {row['LastName']} CourseName: {row['CourseName']}")

        except TypeError as e:
            IO.output_error_messages( message: "Please check that the data is a valid JSON format", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
```

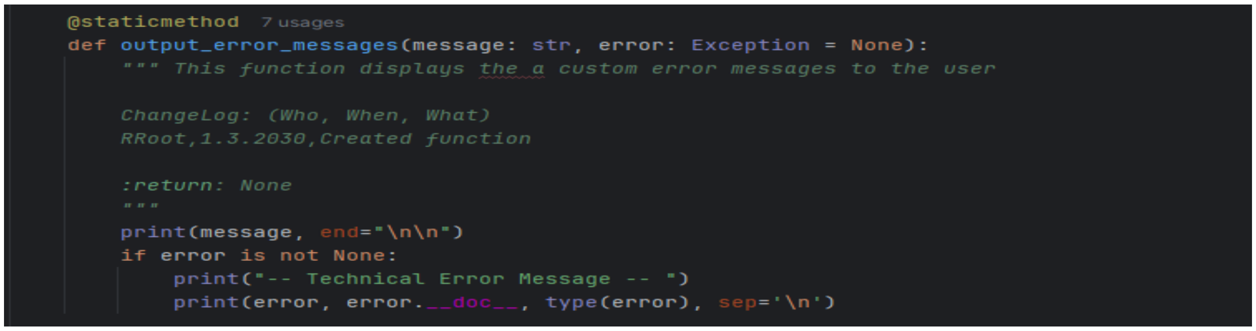
To code the class IO, wrote Class and the IOas seen on the screenshot below:



```
class FileProcessor:  2 usages
    def write_data_to_file(file_name: str, student_data: list):
        finally:
            if file.closed == False:
                file.close()


# Present and Process the data
class IO:
```

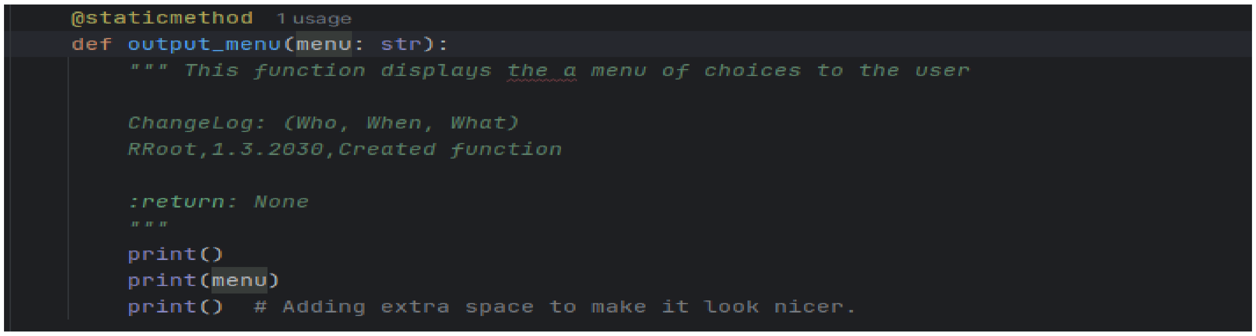
Under the class IO, have to define output_error_messages, output_menu, input_menu_choice, output_student_courses, input_student_data.

output_error_messages



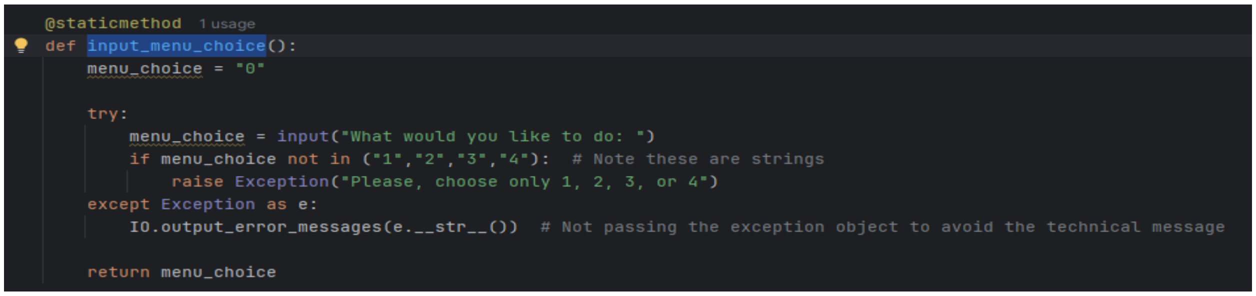
```
    @staticmethod  7 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays the a custom error messages to the user

        ChangeLog: (Who, When, What)
        RRoot,1.3.2030,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')
```

output_menu



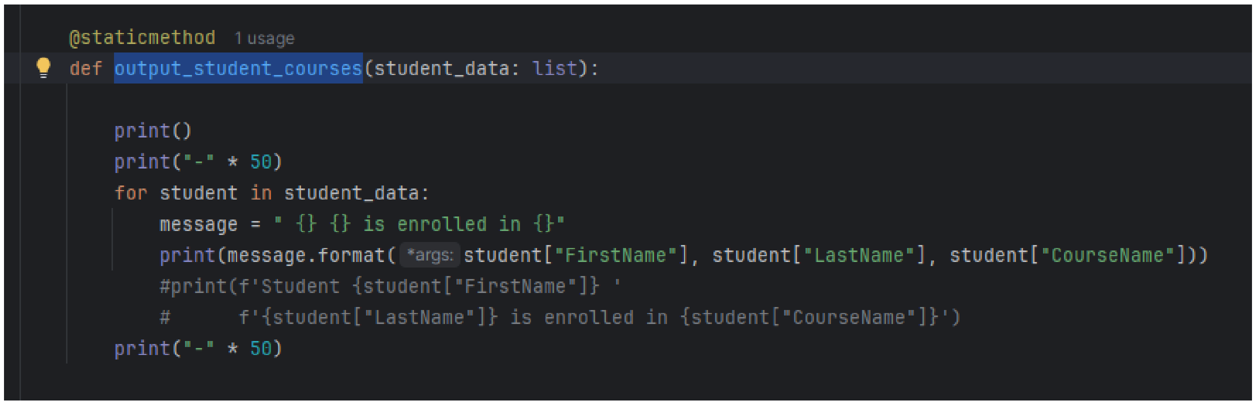
```
    @staticmethod  1 usage
    def output_menu(menu: str):
        """ This function displays the a menu of choices to the user

        ChangeLog: (Who, When, What)
        RRoot,1.3.2030,Created function

        :return: None
        """
        print()
        print(menu)
        print()  # Adding extra space to make it look nicer.
```

input_menu_choice



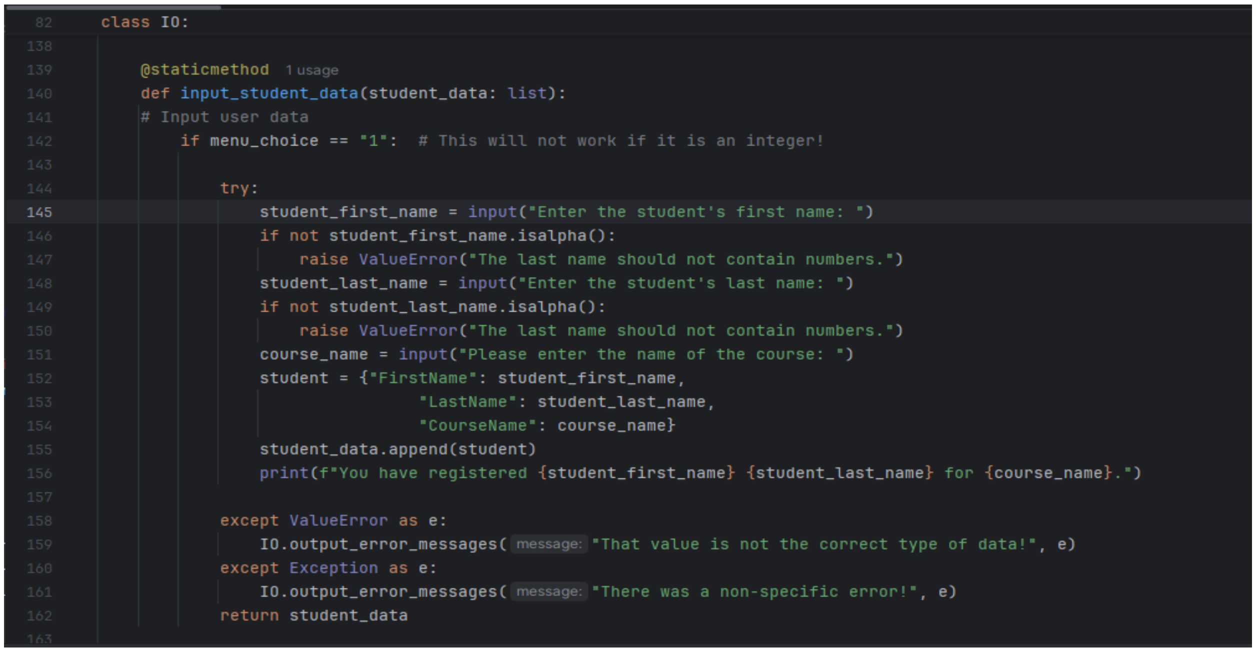
```
    @staticmethod  1 usage
    def input_menu_choice():
        menu_choice = "0"

        try:
            menu_choice = input("What would you like to do: ")
            if menu_choice not in ("1","2","3","4"):  # Note these are strings
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages(e.__str__())  # Not passing the exception object to avoid the technical message

        return menu_choice
```

output_student_courses



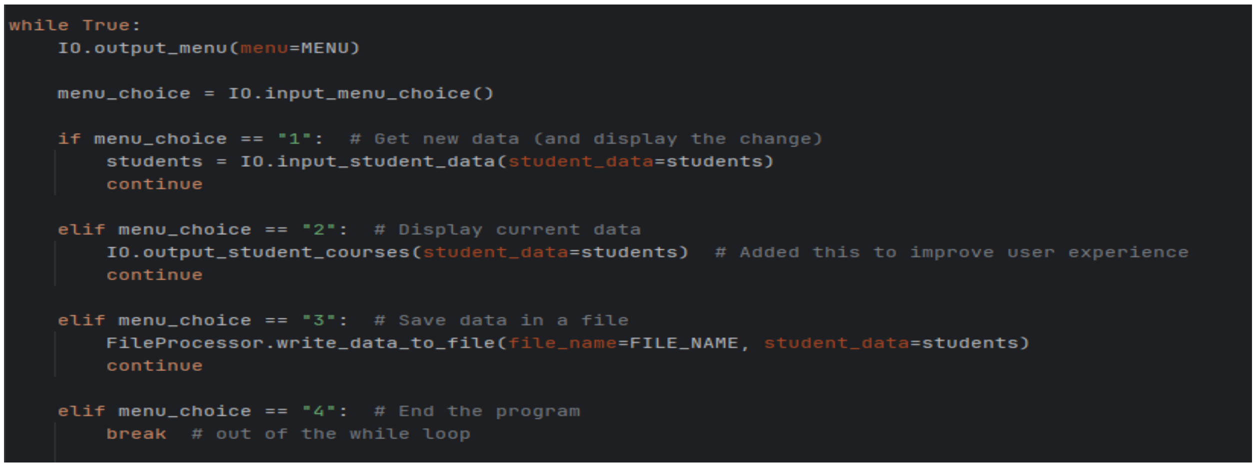
```
    @staticmethod  1 usage
    def output_student_courses(student_data: list):

        print()
        print("-" * 50)
        for student in student_data:
            message = " {} {} is enrolled in {}"
            print(message.format( *args: student["FirstName"], student["LastName"], student["CourseName"]))
            #print(f'Student {student["FirstName"]} '
            #      f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)
```

input_student_data



```
class IO:

    @staticmethod  1 usage
    def input_student_data(student_data: list):
    # Input user data
        if menu_choice == "1":  # This will not work if it is an integer!

            try:
                student_first_name = input("Enter the student's first name: ")
                if not student_first_name.isalpha():
                    raise ValueError("The last name should not contain numbers.")
                student_last_name = input("Enter the student's last name: ")
                if not student_last_name.isalpha():
                    raise ValueError("The last name should not contain numbers.")
                course_name = input("Please enter the name of the course: ")
                student = {"FirstName": student_first_name,
                                "LastName": student_last_name,
                                "CourseName": course_name}
                student_data.append(student)
                print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")

            except ValueError as e:
                IO.output_error_messages( message: "That value is not the correct type of data!", e)
            except Exception as e:
                IO.output_error_messages( message: "There was a non-specific error!", e)
            return student_data
```

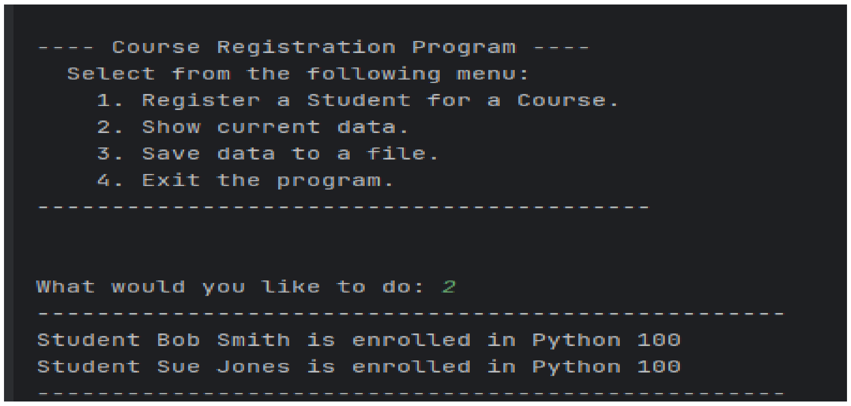
Then for the menu, instead of having all those code, we are just calling the functions we made:



```
while True:
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    if menu_choice == "1":  # Get new data (and display the change)
        students = IO.input_student_data(student_data=students)
        continue

    elif menu_choice == "2":  # Display current data
        IO.output_student_courses(student_data=students)  # Added this to improve user experience
        continue

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

    elif menu_choice == "4":  # End the program
        break  # out of the while loop
```

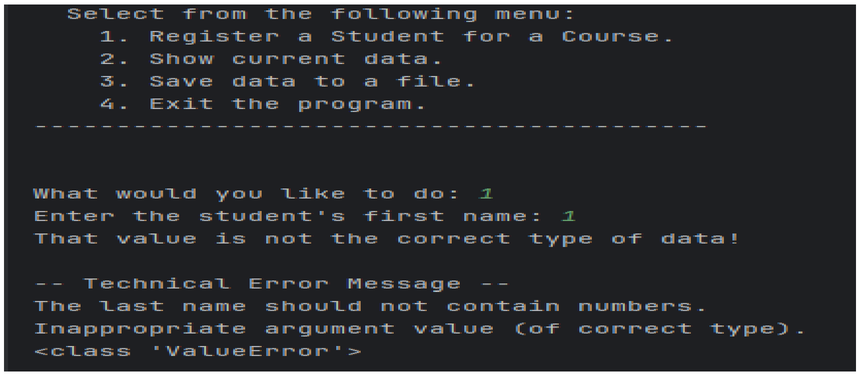
Testing – PyCharm

After coding, I then proceeded with the testing. I tested first using PyCharm. Tested first menu 2 to display the current data on the json file.

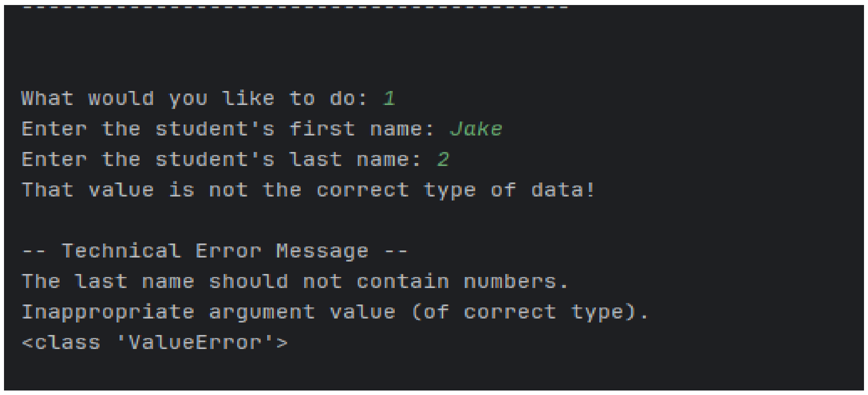


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
--------------------------------------------------
```

Then added record. Have to make sure that the error handling is working properly so I inputted number on both first name and last name and it works.

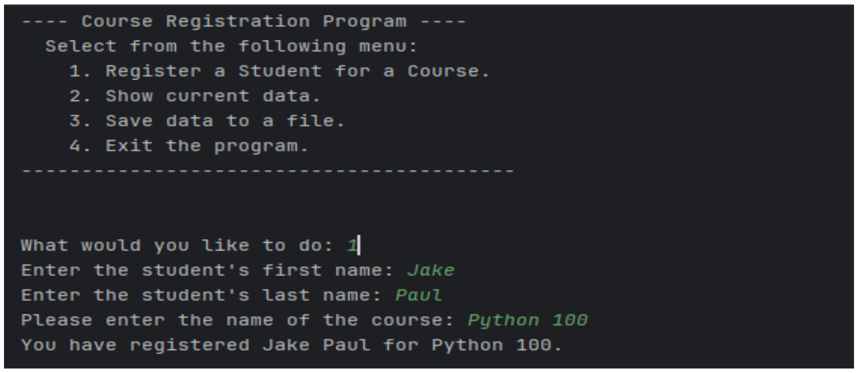


```
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 1
Enter the student's first name: 1
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'>
```

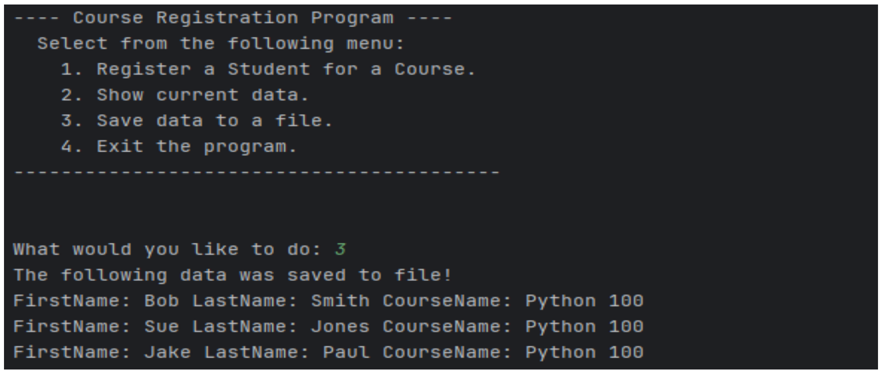


```
-----------------------------------------


What would you like to do: 1
Enter the student's first name: Jake
Enter the student's last name: 2
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'>
```

After making sure that the error handling is working properly, I added a new record and save it on the file.



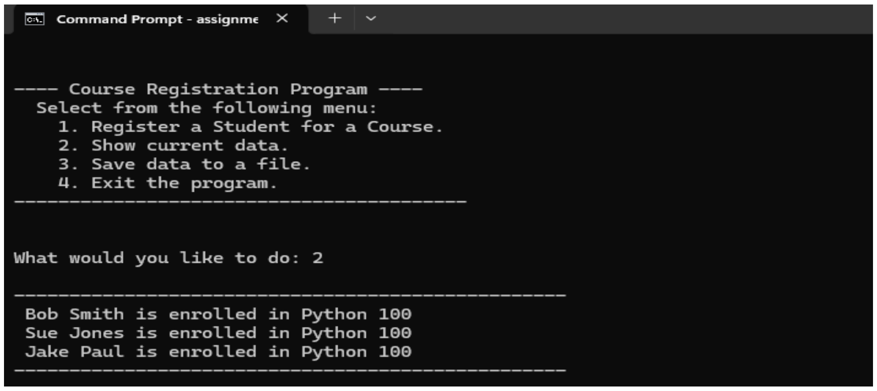
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 1
Enter the student's first name: Jake
Enter the student's last name: Paul
Please enter the name of the course: Python 100
You have registered Jake Paul for Python 100.
```



```
---- 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.
-----------------------------------------


What would you like to do: 3
The following data was saved to file!
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 100
```

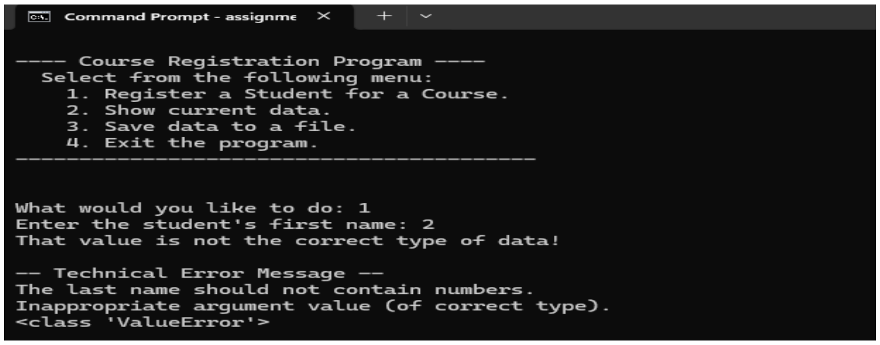
Testing – CMD

After testing it in Pycharm, I tested the python script in the CMD. Tested first that the record I added before is in the json file.

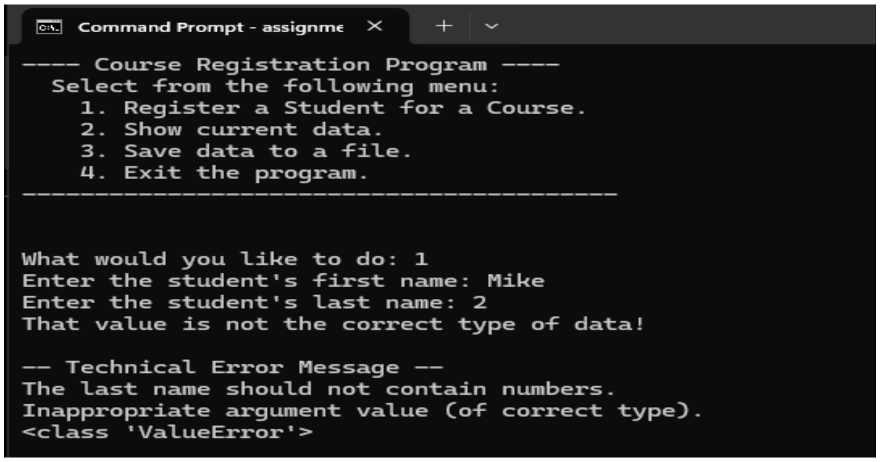


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 2

--------------------------------------------------
 Bob Smith is enrolled in Python 100
 Sue Jones is enrolled in Python 100
 Jake Paul is enrolled in Python 100
--------------------------------------------------
```

Then tried to add a new record. Have to make sure that the error handling is working properly again.

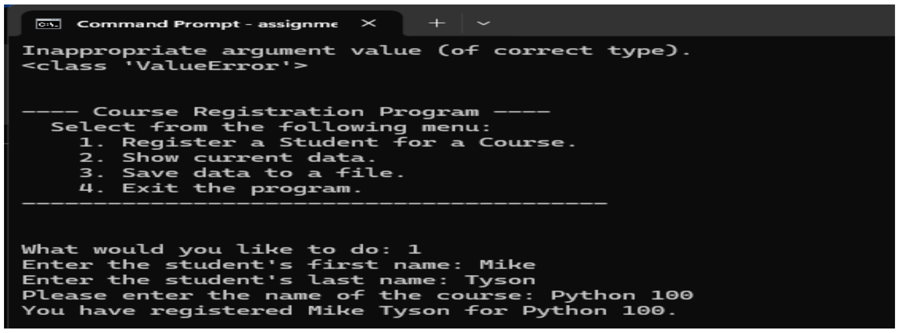


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 1
Enter the student's first name: 2
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'>
```

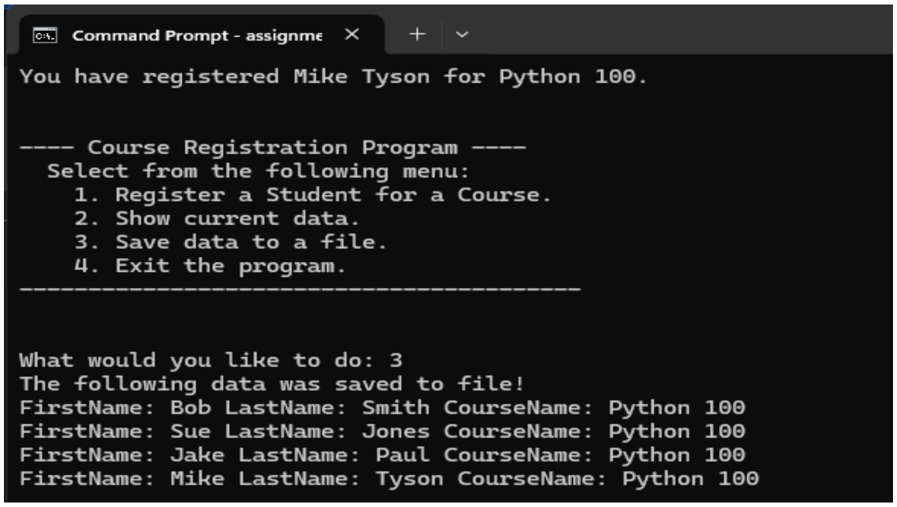


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do: 1
Enter the student's first name: Mike
Enter the student's last name: 2
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'>
```

Once I made sure that it's working, I then proceeded on adding new record.



```
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.
-----------------------------------------


What would you like to do: 1
Enter the student's first name: Mike
Enter the student's last name: Tyson
Please enter the name of the course: Python 100
You have registered Mike Tyson for Python 100.
```



```
You have registered Mike Tyson for Python 100.


---- 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.
-----------------------------------------


What would you like to do: 3
The following data was saved to file!
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 100
FirstName: Mike LastName: Tyson CourseName: Python 100
```

Source Control:

This is the link to my GitHub repository:

https://github.com/AJRIsip/IntroToProg-Python-Mod06

Conclusion:

For the conclusion of this module, it is very important to organize your code/script. In python programming, a function is a reusable block of code that performs specific tasks or set of tasks. Functions are used for reusability and modularity. Aside from functions, we can use classes as well. Functions are used to group specific tasks or set of tasks, classes are used to group functions.