Fan Yang

August 14, 2025

Course Name: Foundations Of Programming: Python

Assignment 06

GitHub link: https://github.com/Fyang712/IntroToProg-Python-Mod06

Functions With Structured Error Handling

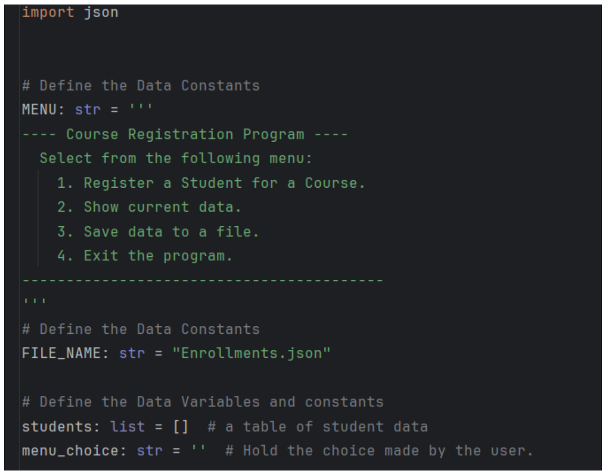
Intro

In this lesson, I learned how to integrate class, functions, parameters, arguments, and the concept of global vs. local variables into my codes.

Programming Steps:

Step 1: Define constants and variables

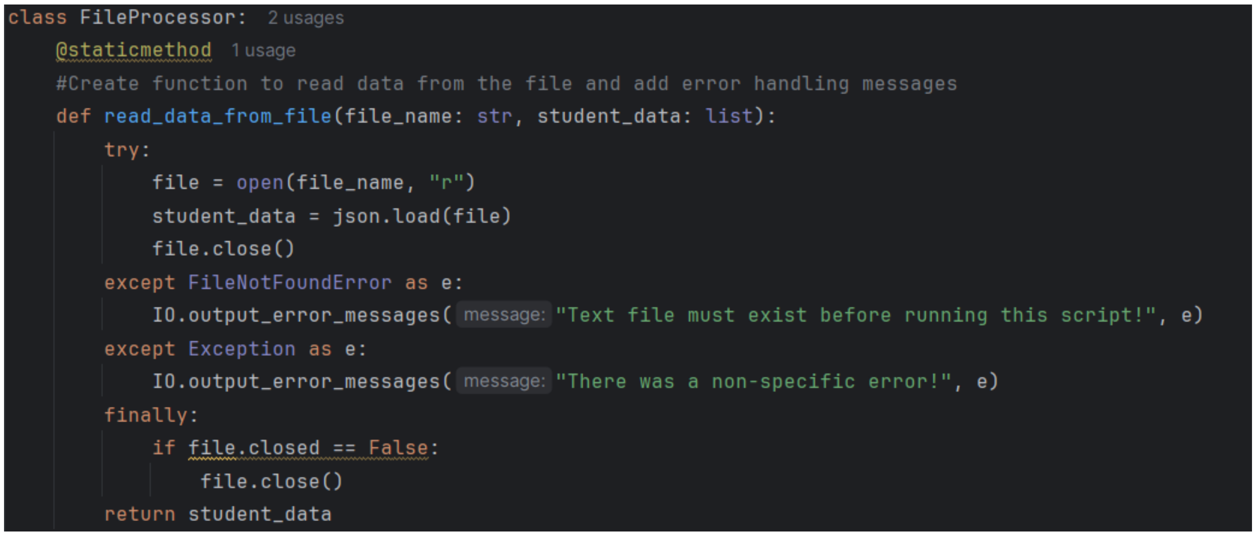
I first defined the data constant and variables. Since this assignment deals with dictionaries and reading and writing data into a json file, I added the "import json" code in the beginning of my codes.



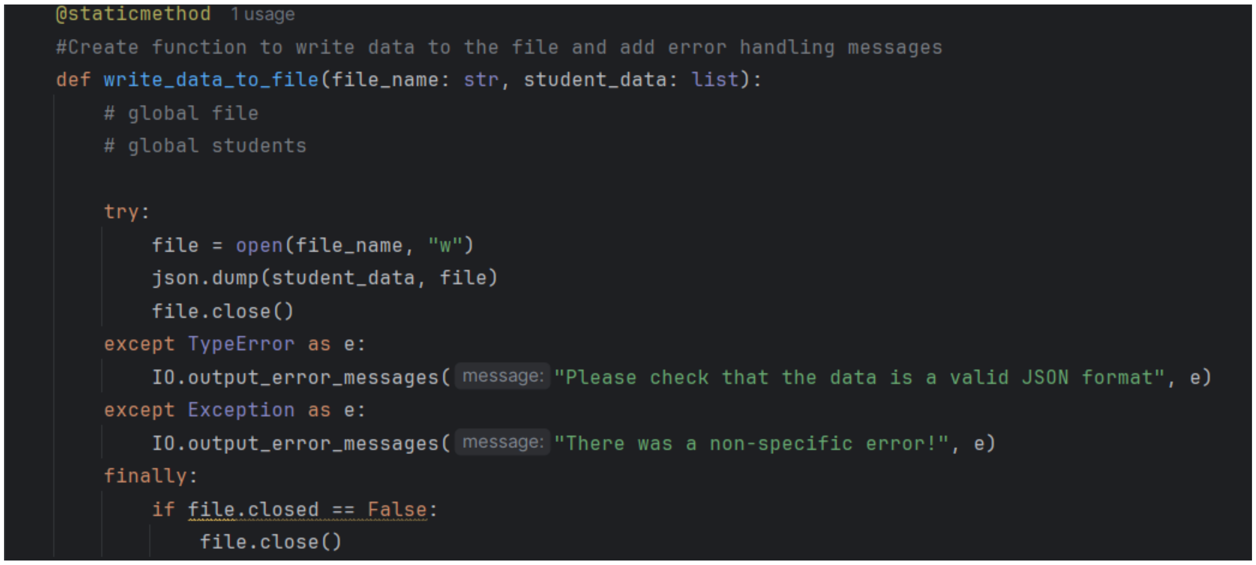
```python
import json


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

# Define the Data Variables and constants
students: list = []  # a table of student data
menu_choice: str = ''  # Hold the choice made by the user.
```

Step 2: Create class to read and write codes into json file

Once the data constants and variables were defined, I create a FileProcessor class code with two functions to read and write data into the json file. I added the error handling features as requested in the assignment.



```
class FileProcessor:  2 usages
    @staticmethod  1 usage
    #Create function to read data from the file and add error handling messages
    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: "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
```

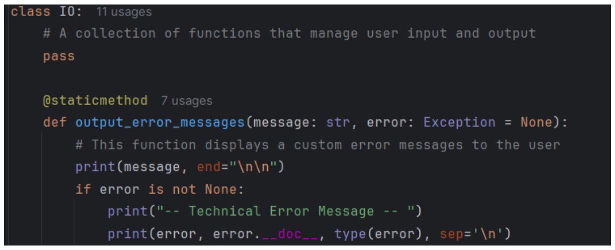


```
    @staticmethod  1 usage
    #Create function to write data to the file and add error handling messages
    def write_data_to_file(file_name: str, student_data: list):
        # global file
        # global students

        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
        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()
```

Step 3: Create functions to present and process the data

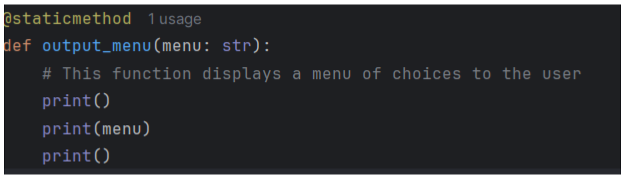
Once the class is created, I wrote a series of functions to present and process the data

output_error_messages function:



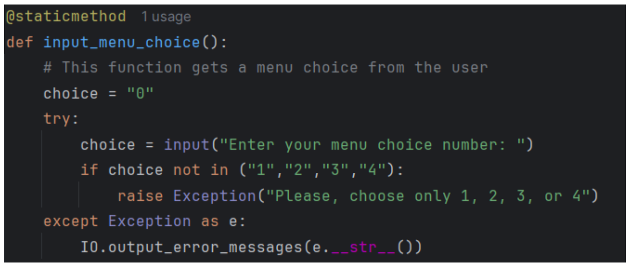
```
class IO:  11 usages
    # A collection of functions that manage user input and output
    pass

    @staticmethod  7 usages
    def output_error_messages(message: str, error: Exception = None):
        # This function displays a custom error messages to the user
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
```

output_menu function:



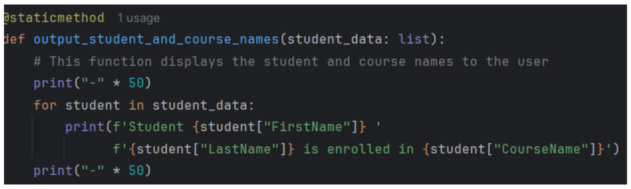
```
@staticmethod  1 usage
def output_menu(menu: str):
    # This function displays a menu of choices to the user
    print()
    print(menu)
    print()
```

input_menu_choice function:



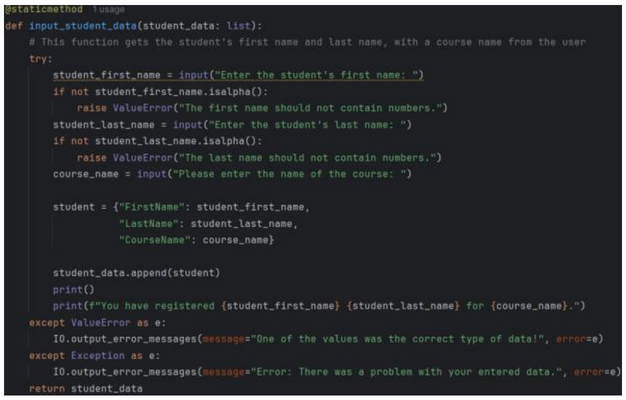
```
@staticmethod  1 usage
def input_menu_choice():
    # This function gets a menu choice from the user
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"):
            raise Exception("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())
```

output_student_and_course_names:



```
@staticmethod  1 usage
def output_student_and_course_names(student_data: list):
    # This function displays the student and course names to the user
    print("-" * 50)
    for student in student_data:
        print(f'Student {student["FirstName"]} '
              f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 50)
```

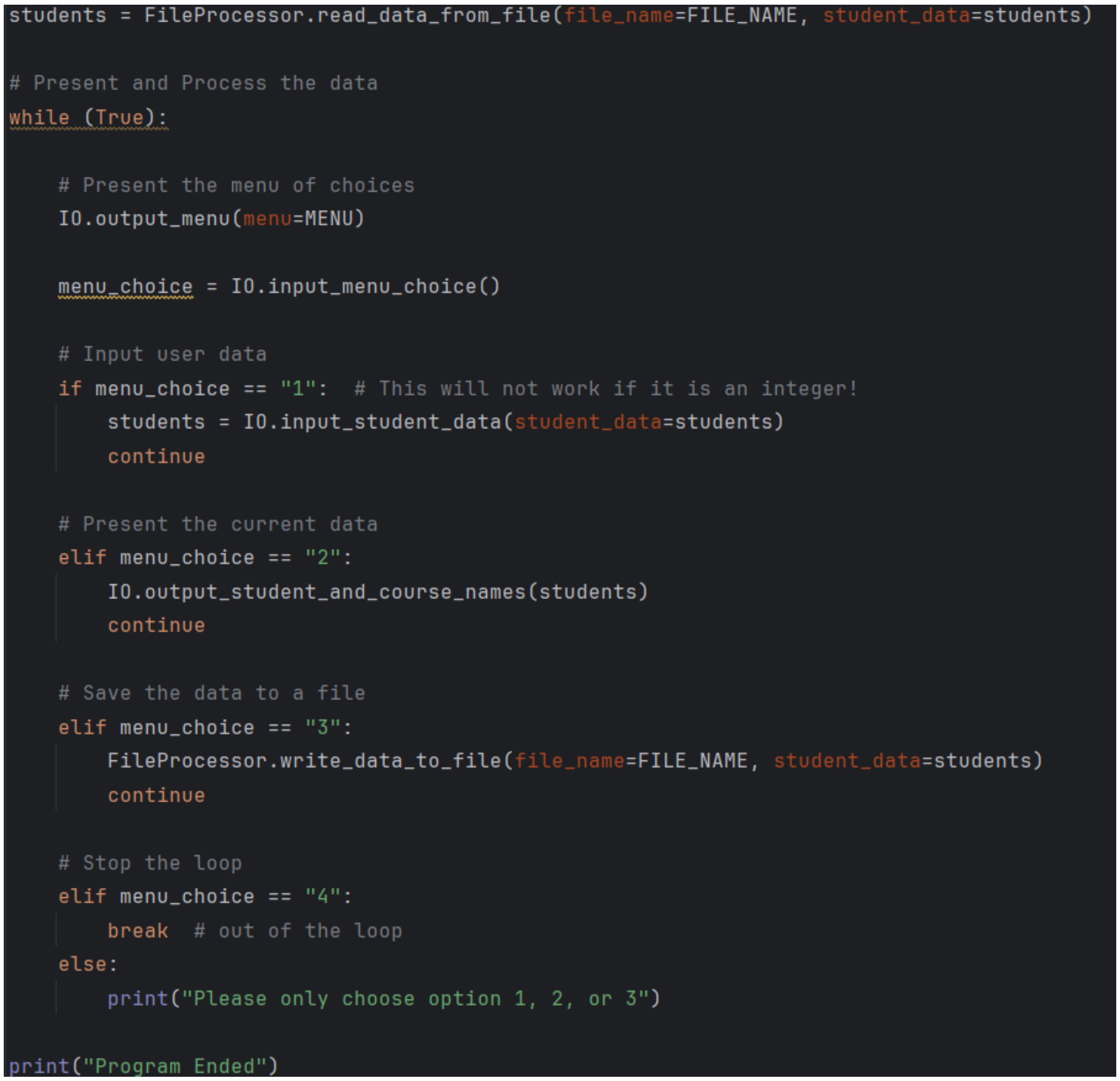
input_student_data:



```
@staticmethod  1 usage
def input_student_data(student_data: list):
    # This function gets the student's first name and last name, with a course name from the user
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first 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()
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was the correct type of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
    return student_data
```

Step 4: Create main script to run the codes

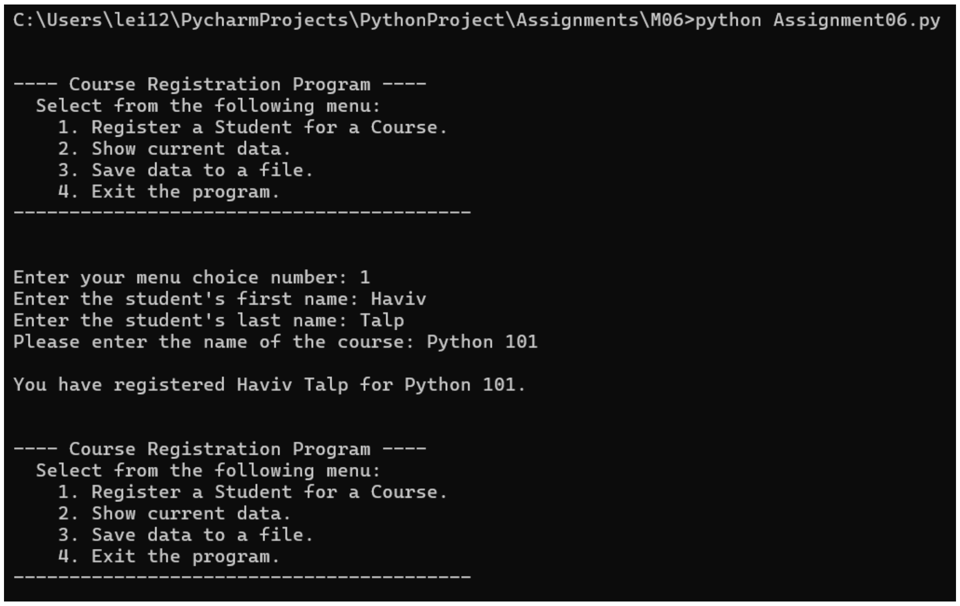
Once the class and functions are defined, I wrote the main script to run the codes.

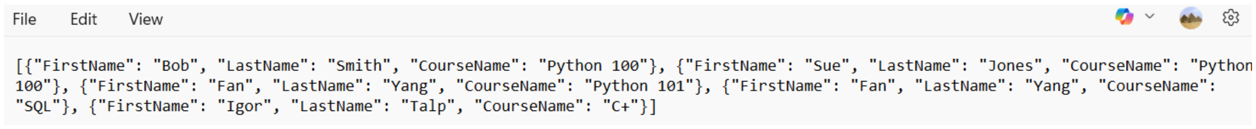


```
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
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_and_course_names(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
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Step 5: Test the codes in both Pycharm and Command Prompt

I tested out the codes in both Pycharm and Command Prompt to test out the codes and it worked both ways.


```
C:\Users\lei12\PycharmProjects\PythonProject\Assignments\M06>python Assignment06.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
Enter the student's first name: Haviv
Enter the student's last name: Talp
Please enter the name of the course: Python 101

You have registered Haviv Talp for Python 101.


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



```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python
100"}, {"FirstName": "Fan", "LastName": "Yang", "CourseName": "Python 101"}, {"FirstName": "Fan", "LastName": "Yang", "CourseName":
"SQL"}, {"FirstName": "Igor", "LastName": "Talp", "CourseName": "C+"}]
```

Summary

This week, I learned about class, functions, parameters, arguments, and the concept of global vs. local variables. The codes built upon the lessons and concepts learned in previous week and expanded the capability by incorporating class and functions.