Introduction to Programming with Python

Assignment 06

https://github.com/Mossy-Squatch/IntroToProg-Python

Module 06 – JSON, Classes and Functions

Introduction

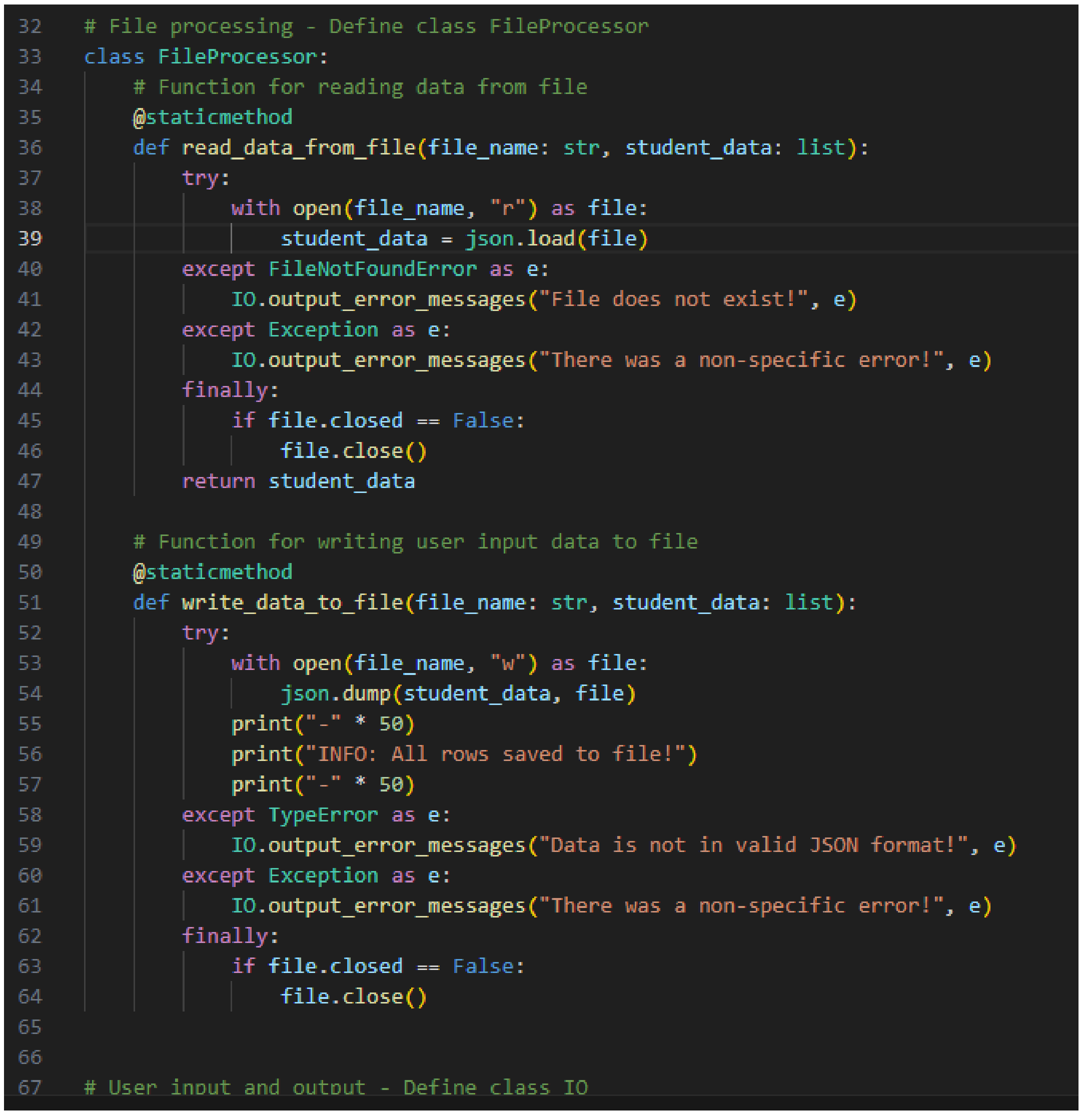
In this document I will describe the steps I took to create a Python script that reads a JSON file, then stores that data. It then presents the user with menu choices, takes the user's selection and performs a task based on that input. User provided information can then be stored with the data read from the JSON file. This process loops until the user selects to exit the program. This assignment expands on the assignment from Module 05, now we define classes and functions and call them from the main program. We also continue to work with structured error handling within the script.

Creating the script

I began by opening Visual Studio Code and reviewing the starter file for this assignment and the instructions in the Mod06-Assignment file. This assignment focuses on using JSON files, creating classes and functions.

- Before defining any constants or variables I used "import json" and "from sys import exit".
- Under "# Define the Data Constants" I created a "MENU" constant that is used to present the user with their options, I also created a "FILE_NAME" constant that is used for the JSON file.
- Under "# Define the Data Variables" I created Variables for menu_choice and students. All other variables are now defined locally in the functions where they are used.
- Under "# File processing" I defined the class "FileProcessor" and then created functions for reading and writing to the JSON file. Each function is converted to a static function that belongs to the class FileProcessor by using @staticmethod. Error handling is also incorporated in these

functions. This can be seen in the screenshot below.

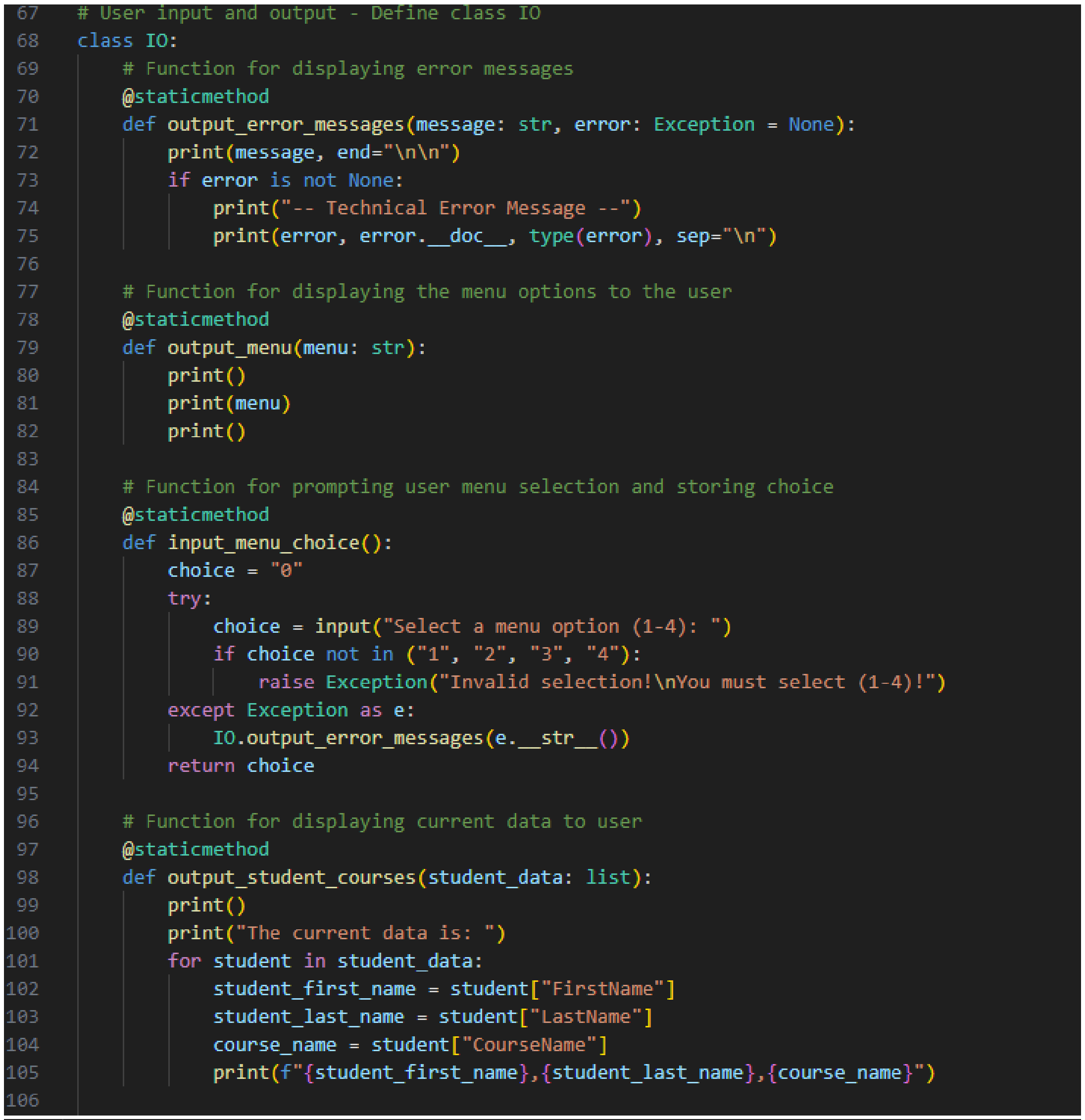


```
# File processing - Define class FileProcessor
class FileProcessor:
    # Function for reading data from file
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        try:
            with open(file_name, "r") as file:
                student_data = json.load(file)
        except FileNotFoundError as e:
            IO.output_error_messages("File does not exist!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    # Function for writing user input data to file
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        try:
            with open(file_name, "w") as file:
                json.dump(student_data, file)
            print("-" * 50)
            print("INFO: All rows saved to file!")
            print("-" * 50)
        except TypeError as e:
            IO.output_error_messages("Data is not in valid JSON format!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()


# User input and output - Define class IO
```

- Next I created the class IO. Here I created functions for handling error messages, prompting user menu choices, gathering student information, displaying the data, saving the data and exiting the program. The classes, related functions and error handling can be seen in the screenshot below.



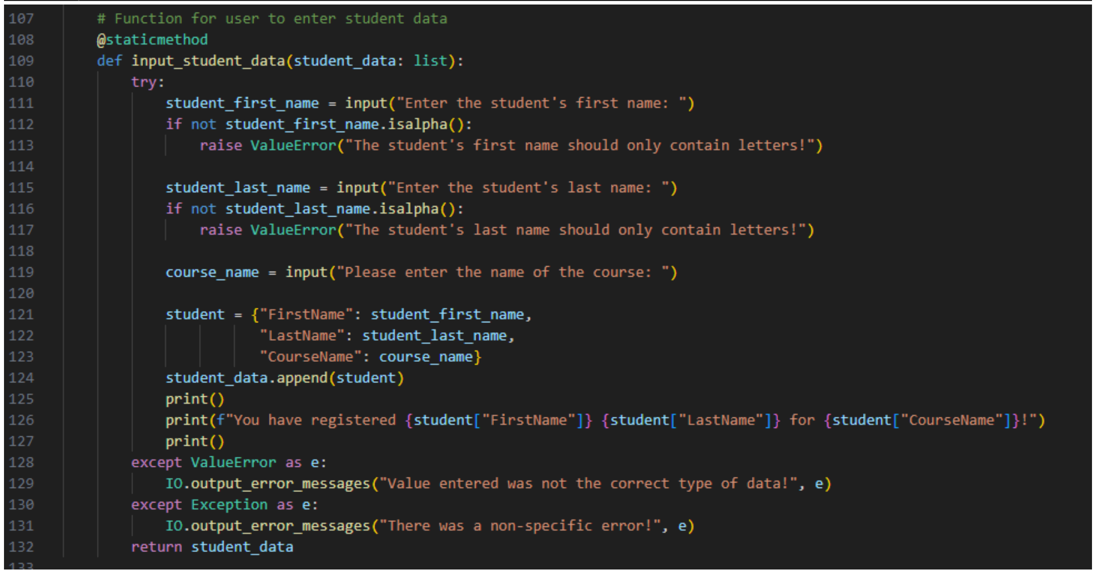
```python
# User input and output - Define class IO
class IO:
    # Function for displaying error messages
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message --")
            print(error, error.__doc__, type(error), sep="\n")

    # Function for displaying the menu options to the user
    @staticmethod
    def output_menu(menu: str):
        print()
        print(menu)
        print()

    # Function for prompting user menu selection and storing choice
    @staticmethod
    def input_menu_choice():
        choice = "0"
        try:
            choice = input("Select a menu option (1-4): ")
            if choice not in ("1", "2", "3", "4"):
                raise Exception("Invalid selection!\nYou must select (1-4)!")
        except Exception as e:
            IO.output_error_messages(e.__str__())
        return choice

    # Function for displaying current data to user
    @staticmethod
    def output_student_courses(student_data: list):
        print()
        print("The current data is: ")
        for student in student_data:
            student_first_name = student["FirstName"]
            student_last_name = student["LastName"]
            course_name = student["CourseName"]
            print(f"{student_first_name},{student_last_name},{course_name}")

```



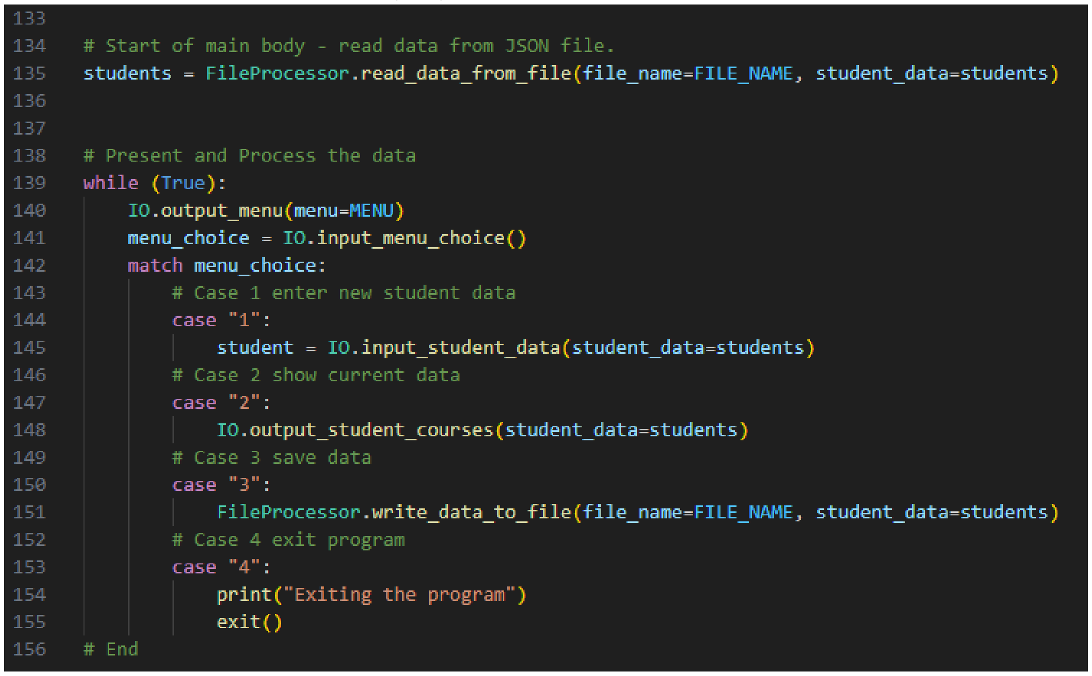
```python
    # Function for user to enter student data
    @staticmethod
    def input_student_data(student_data: list):
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The student's first name should only contain letters!")

            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The student's last name should only contain letters!")

            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["FirstName"]} {student["LastName"]} for {student["CourseName"]}!")
            print()
        except ValueError as e:
            IO.output_error_messages("Value entered was not the correct type of data!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        return student_data
```

- Once the file processing and IO had all been defined, the main body of the program was created. It starts with reading the data from the file using FileProcessor.read_data_from_file and filling students with the data found there. After that I use a match case loop to run the pre-defined functions based on user choices (1-4). This is shown below.



```python

# Start of main body - read data from JSON file.
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)


# Present and Process the data
while (True):
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()
    match menu_choice:
        # Case 1 enter new student data
        case "1":
            student = IO.input_student_data(student_data=students)
        # Case 2 show current data
        case "2":
            IO.output_student_courses(student_data=students)
        # Case 3 save data
        case "3":
            FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        # Case 4 exit program
        case "4":
            print("Exiting the program")
            exit()
# End
```

Testing the script

- I ran the script from Visual Studio Code and tested for the following.
    - Takes user input for first name, last name and course name.
    - Program displayed this input.
    - The program saved this to the correct JSON file, this was verified by opening the file and reviewing the contents.
    - I was able to enter multiple registrations.
    - I was able to display multiple registrations.
    - I was able to save multiple registrations to file.

Summary

Using the class demonstrations, my assignment from module 5 and documents provided for module 6 I was able to create a script that takes user input, presents it to the user and saves it into a JSON file. I expanded my ability to create a loop, take a user's input, save it to file and add to that file as the user inputs more data. I also learned how to check for certain error conditions and present the error information to the user.