Name: Gustavo Limongi Araujo

Date: 02/26/2024

Course: Introduction to Python

Assignment 07: Introduction to Programming with Python

Intro

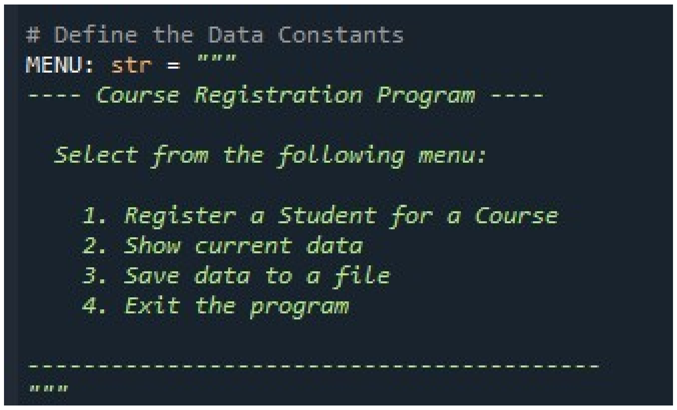
Create a Python program that demonstrates using constants, variables, and print statements to display a message about a student's registration for a Python course. This program is very similar to Assignment05, but **It adds the use of functions, classes, and using the separation of concerns pattern.**

Note: Start by opening and reviewing the starter file Assignment07-Starter.py!

Topic

Using PyCharm as an IDE, the header is the following:

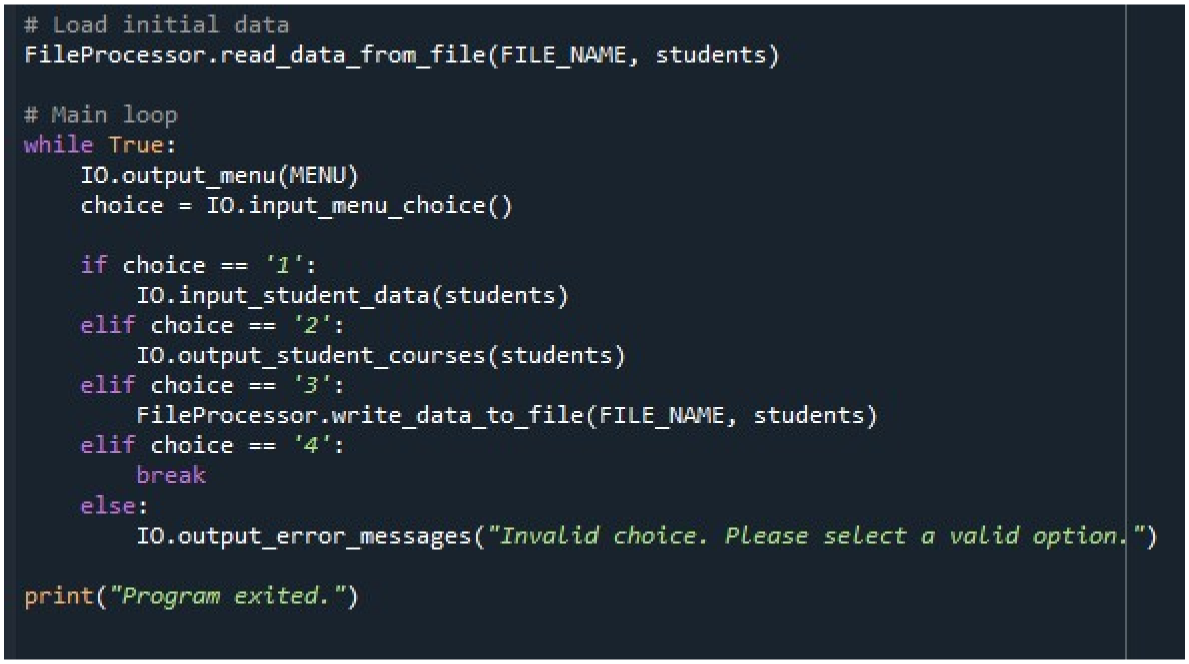
The menu choice and the file name were defined below the header:



```
# 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

-----------------------------------------
"""
```

The While loop is used in conjunction with 3 functions to call options 1 to 3:

1) Function **input student data**: read data from student using the *input* function
2) Function **show student data**: print what was stored in the variables above.
3) Function **save_data_to_file**: save the data in the FILE_NAME in the json format file



```python
# Load initial data
FileProcessor.read_data_from_file(FILE_NAME, students)

# Main loop
while True:
    IO.output_menu(MENU)
    choice = IO.input_menu_choice()

    if choice == '1':
        IO.input_student_data(students)
    elif choice == '2':
        IO.output_student_courses(students)
    elif choice == '3':
        FileProcessor.write_data_to_file(FILE_NAME, students)
    elif choice == '4':
        break
    else:
        IO.output_error_messages("Invalid choice. Please select a valid option.")

print("Program exited.")
```

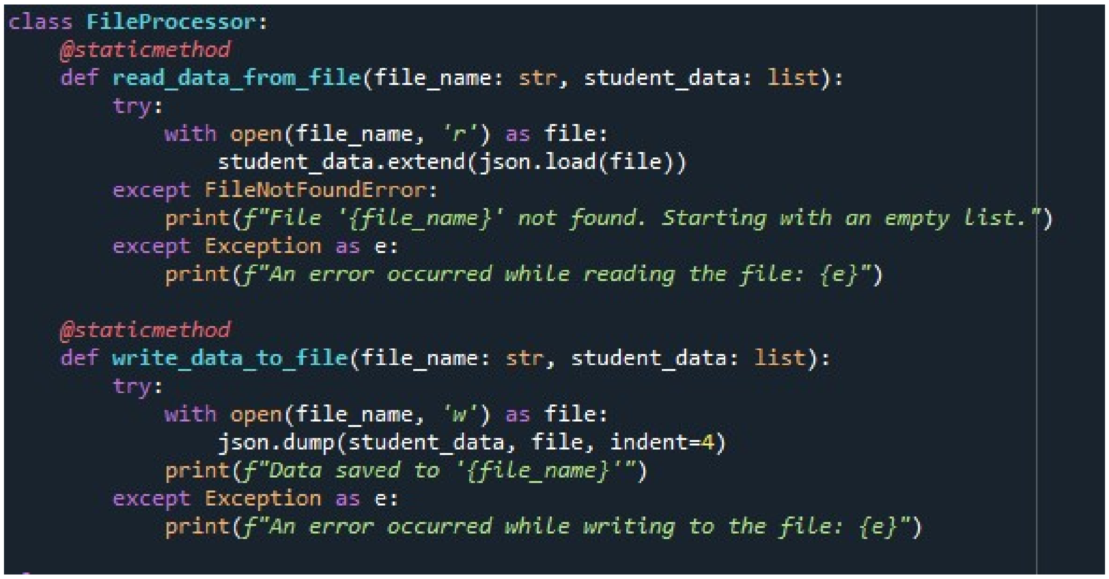
The static decorator was used before each function as instructed and 2 classes were used, one called File Processor to read and write data in the json file and the other one called class IO to call the functions to display the menu choice, read data (option 1), show the data in a dictionary format (option 2) and save the data (3).

There is error handling for readind and writing the data as well as if the input names are empty.

A class named Person and a class named Student was created.

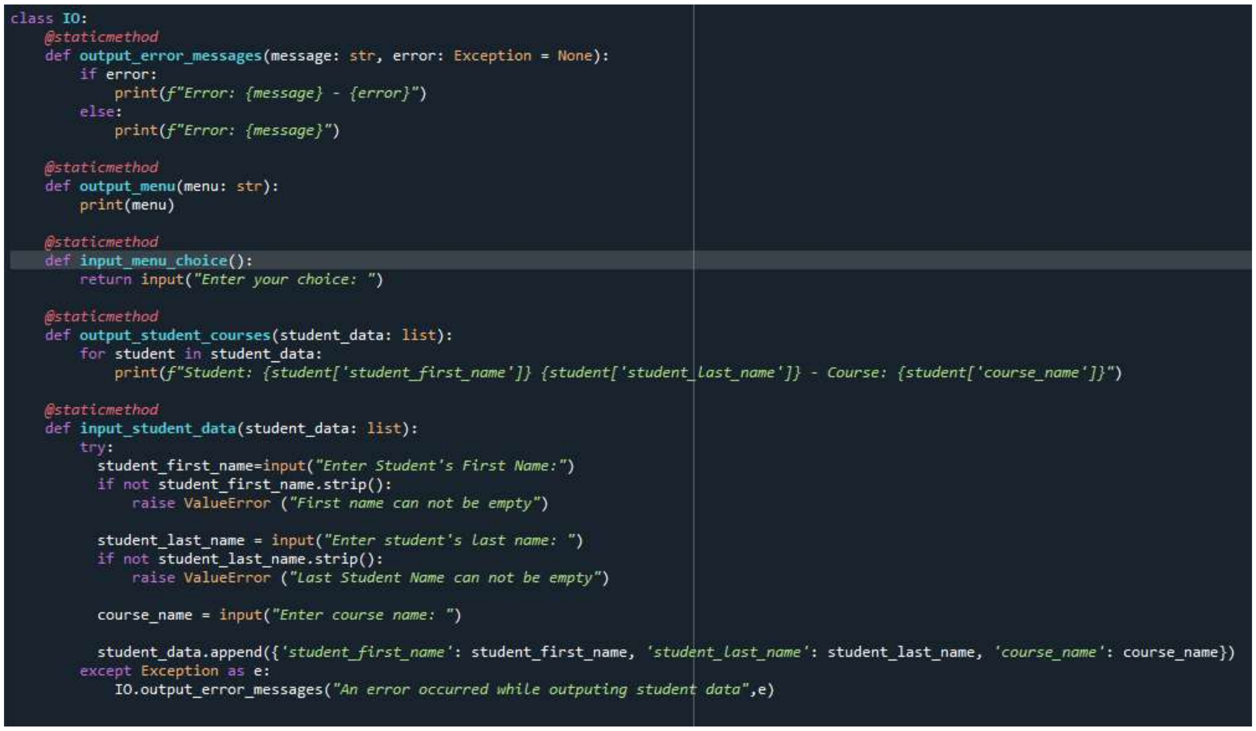
The program includes a method to extract comma separated data from each data class.

CLASS FileProcessor (read and write with error handling):



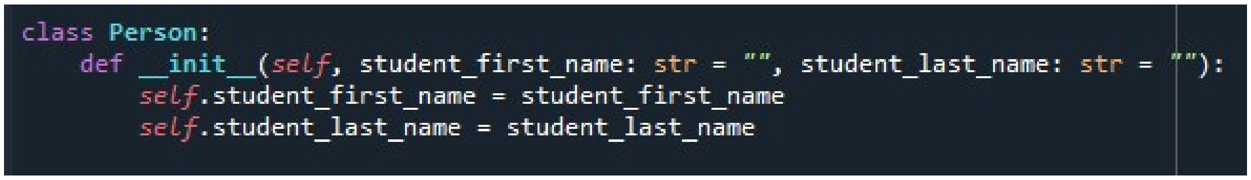
```python
class FileProcessor:
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        try:
            with open(file_name, 'r') as file:
                student_data.extend(json.load(file))
        except FileNotFoundError:
            print(f"File '{file_name}' not found. Starting with an empty list.")
        except Exception as e:
            print(f"An error occurred while reading the file: {e}")

    @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, indent=4)
            print(f"Data saved to '{file_name}'")
        except Exception as e:
            print(f"An error occurred while writing to the file: {e}")
```

Class IO:



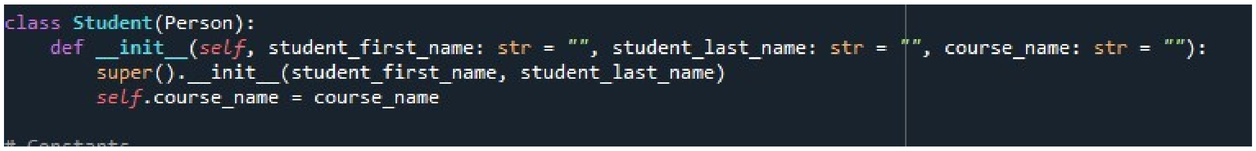
```python
class IO:
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        if error:
            print(f"Error: {message} - {error}")
        else:
            print(f"Error: {message}")

    @staticmethod
    def output_menu(menu: str):
        print(menu)

    @staticmethod
    def input_menu_choice():
        return input("Enter your choice: ")

    @staticmethod
    def output_student_courses(student_data: list):
        for student in student_data:
            print(f"Student: {student['student_first_name']} {student['student_last_name']} - Course: {student['course_name']}")

    @staticmethod
    def input_student_data(student_data: list):
        try:
          student_first_name=input("Enter Student's First Name:")
          if not student_first_name.strip():
              raise ValueError ("First name can not be empty")

          student_last_name = input("Enter student's last name: ")
          if not student_last_name.strip():
              raise ValueError ("Last Student Name can not be empty")

          course_name = input("Enter course name: ")

          student_data.append({'student_first_name': student_first_name, 'student_last_name': student_last_name, 'course_name': course_name})
        except Exception as e:
            IO.output_error_messages("An error occurred while outputing student data",e)
```

Class Person w/ constructor:



```python
class Person:
    def __init__(self, student_first_name: str = "", student_last_name: str = ""):
        self.student_first_name = student_first_name
        self.student_last_name = student_last_name
```

Class Student w/ constructor:



```python
class Student(Person):
    def __init__(self, student_first_name: str = "", student_last_name: str = "", course_name: str = ""):
        super().__init__(student_first_name, student_last_name)
        self.course_name = course_name

# Constants
```

The program was tested in IDE and Gitbash and worked as intended.

It is shared in Github and link posted in the module 07.

Summary

The videos and course notes of Lesson 07 were used to help write this code.

The code was tested with PyCharm and gitbash and worked as intended.