FERNANDO TAMAYO GRADOS
November 19, 2023
IT FDN 110 B
Assignment 6
https://github.com/fernandotg123/IntroToProg-Python-Mod06

Assignment 6

1. Intro

This week we learned about functions and classes in Python. Functions are reusable blocks of code to perform a specific task. Within the functions, we have parameters in order to encapsulate the data required for the particular job. While a class is a group of functions, variables and constants by the name of the class. We are also organizing the code based on Concerns, separating the application into distinct layers.

2. Problem Statement

The assignment requires creating a program that asks a user to select one of the four menu options in order to register, visualize and store details of a student. The assignment also requires to use functions and classes, and call them throughout the main body of the code. Structured error handling is required throughout the program.

3. JSON set up

To kick off the code, I stated “import json” in the Python script, so that it is easier to work with JSON files.

4. Define data constants and variables

Then, I continued the assignment by defining the data constants. I made sure that I created these constants in CAPITAL LETTERS, as well as added the data types for easy understanding of the reader. Since the first string is large, I used “”” “”” in order to incorporate the full menu. See Figure 1.

```
# 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"
```
Figure 1: Data constants

When I finished with the data constants, I created the empty variables, a meny choice, as well as a list of dictionaries called students. See Figure 2.

```
# Define the Data Variables
menu_choice: str  # Hold the choice made by the user.
students: list = []  # a table of student data
```
Figure 2: Data variables

5. Data access and processing layer

Then, I separated the code into three big sections: 1. Data access and processing layer, 2. Presentation layer and 3. Main Body. For the first two, I provided context on the purpose of the layer and the functions that they contain. For the data access layer, I created two functions: read_data_from_file and write_data_to_file. The first one reads data from a JSON file and loads it into a list. The second function writes rows from a list into a JSON file. See the code below:

```
class FileProcessor:
    """
    This class is a collection that opens, reads and writes JSON files. It
has two functions:
        - read_data_from_file - this function reads data from a JSON file and
loads it into a list.
        - write_data_to_file - this function writes rows from a list into a
JSON file.
    ChangeLog:
    Fernando Tamayo Grados, 11/18/2023, Created Class
    """
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
    # Extract the data from the file
        try:
            file = open(file_name, "r")
            students = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("JSON file must exist before running the
script",e)
        except Exception as e:
            IO.output_error_messages("Something went wrong",e)
        finally:
            if file.closed == False:
                file.close()
        return students

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
    # Write inputs to file
        try:
            file = open(file_name, "w")
            json.dump(student_data,file)
            file.close()
            print("The following data was saved to file!")
            IO.output_student_courses(student_data=student_data)
        except Exception as e:
            IO.output_error_messages("Something went wrong", e)
        finally:
            if file.closed == False:
                file.close()
```
Figure 3: Data Access Layer

6. Presentation layer

For the presentation layer, I had five functions:

- output_error_messages, which shows a custom error message to the user,
- output_menu - this function displays the menu of choices to the user
- input_menu_choice - this function gets a menu choice from the user
- output_student_courses - this function shows students' data to the user
- input_student_data - this function gets the first name, last name and course name from the user

See the code below:

```
class IO:
    """
    This class is a collection of functions that manages user inputs and
outputs.
        - output_error_messages - this function shows a custom error messages
to the user
        - output_menu - this function displays the menu of choices to the
user
        - input_menu_choice - this function gets a menu choice from the user
        - output_student_courses - this function shows students' data to the
user
        - input_student_data - this function gets the first name, last name
and course name from the user
    ChangeLog:
    Fernando Tamayo Grados, 11/18/2023, Created Class
    """
    @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')

    @staticmethod
    def output_menu(menu: str):
        print()  # Adding extra space to make it look nicer.
        print(menu)
        print()  # Adding extra space to make it look nicer.

    @staticmethod
    def input_menu_choice(menu: str):
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if 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 e to avoid
the technical message

        return choice

    @staticmethod
    def output_student_courses(student_data: list):
        print()
        print("-" * 50)
        for student in student_data:

print(f"{student['FirstName']},{student['LastName']},{student['CourseName']}"
)
        print("-" * 50)
        print()

    @staticmethod
    def input_student_data(student_data: list):
        try:
            # Input the data
            student_first_name = input("What is the student's first name? ")
            if not student_first_name.isalpha():
                raise ValueError("The first name should not contain
numbers.")

            student_last_name = input("What is 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"The system has registered the student")
        except ValueError as e:
            IO.output_error_messages("That value is 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
```

Figure 4: Presentation Layer

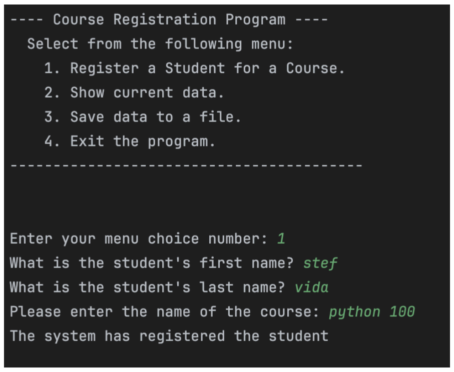
7. Main body

Finally, I used both classes to come up with the main body of the code:

```
#Extract data from file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME,
student_data=students)
print(students)
# Present and Process the data
while (True):

    # Present the menu of choices
    IO.output_menu(menu=MENU)
    # Ask user for input
    menu_choice = IO.input_menu_choice(menu="")

    # 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
    elif menu_choice == "2":
        IO.output_student_courses(students)
        continue
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME,
student_data=students)
        continue
    elif menu_choice == "4":
        break
    else:
        print("")

print("Program Ended")
```

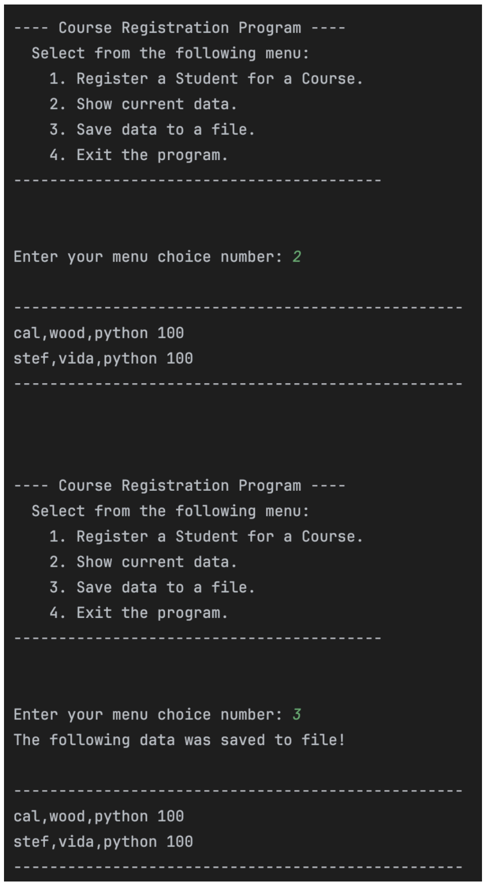
Figure 5: Main Body

I was able to validate adding new students in the code:



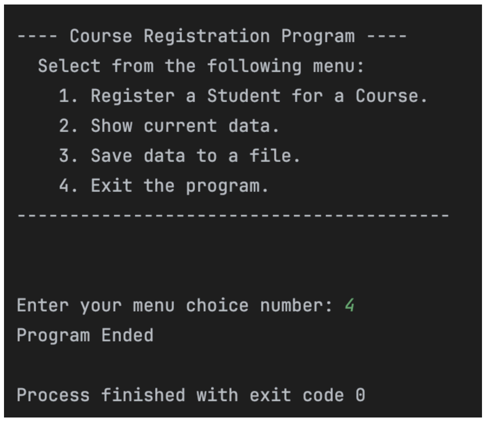
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 1
What is the student's first name? stef
What is the student's last name? vida
Please enter the name of the course: python 100
The system has registered the student
```

Figure 6: User input



```
---- 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

--------------------------------------------------
cal,wood,python 100
stef,vida,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.
-----------------------------------------



Enter your menu choice number: 3
The following data was saved to file!

--------------------------------------------------
cal,wood,python 100
stef,vida,python 100
--------------------------------------------------
```

Figure 7: Input registration



```
---- 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 8: Program end

8. Conclusion

Overall, the code ran as expected without delivery any major incidences. To validate my work through the assignment, I printed the values in different moments of the code in order to find any possible errors.