Olivier Richer
June 2, 2025, IT FDN 110 A Sp 25: Foundations of Programming: Python
GitHub hyperlink: olik2000 · GitHub
Assignment 06- Functions.

Functions and Classes.

Introduction:

Assignment #6 scope of work is to 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 Assignment#05, but **it adds the use of functions, classes, and using the separation of concerns pattern.** Lastly, this report will be divided into three parts (the script, the execution of the program, and the debugging).

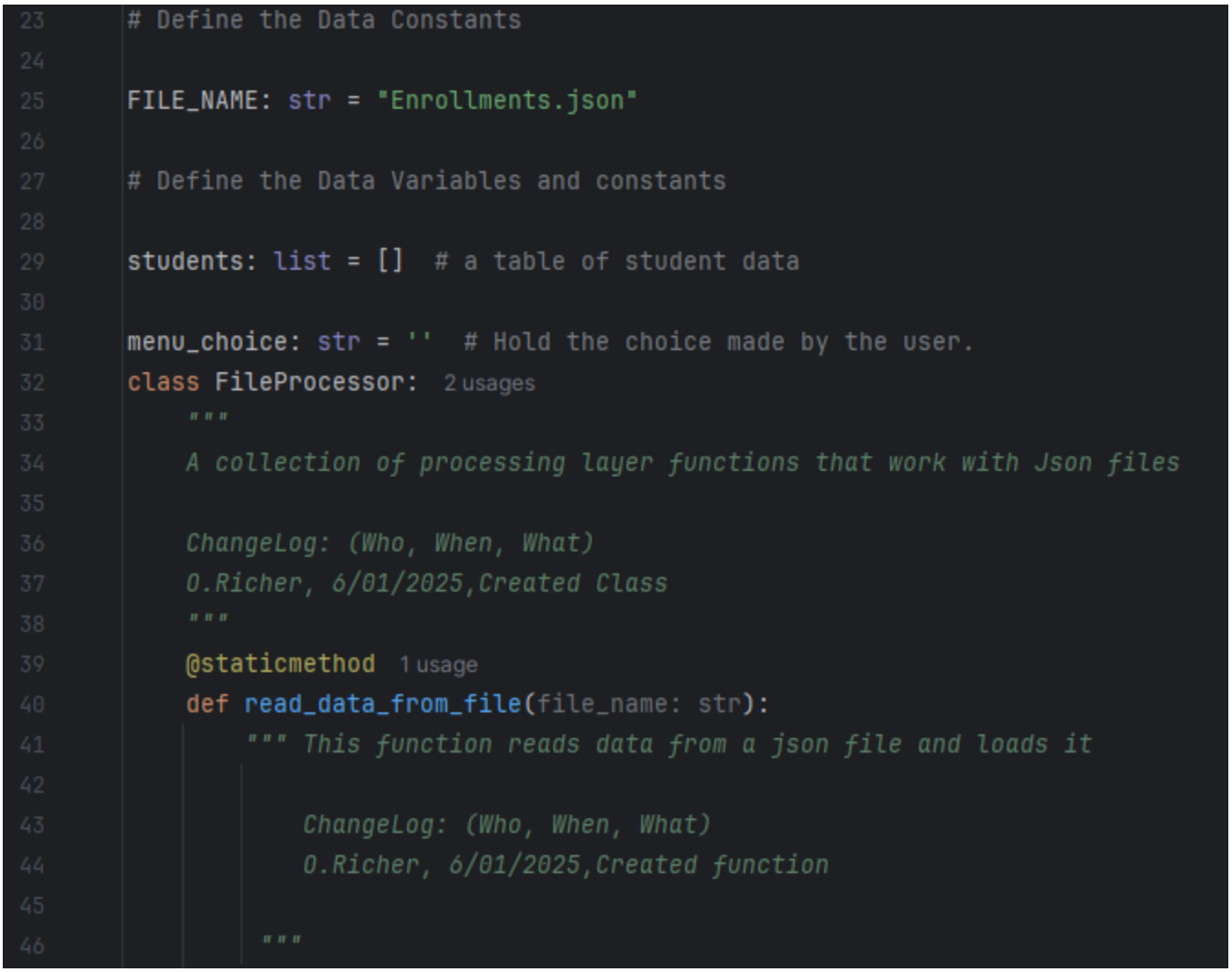
1. The script.

Beginning with the assignment 06 starter.py, one filled out the script with the functions and classes names and task as shown on assignment#06 notes. In brief, we will be using our code from assignment#05 and break it in blocks of function and classes (see fig1 through fig10).

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Olivier Richer, 6/01/2025, created script
#   <Your Name Here>,<Date>,<Activity>
# ------------------------------------------------------------------------------------------ #
# importing the out of the box json capabilities
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.
-----------------------------------------
'''
```

Fig1



```
# 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.
class FileProcessor:  2 usages
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    O.Richer, 6/01/2025,Created Class
    """
    @staticmethod  1 usage
    def read_data_from_file(file_name: str):
        """ This function reads data from a json file and loads it

            ChangeLog: (Who, When, What)
            O.Richer, 6/01/2025,Created function

         """
```

Fig2

```python
class FileProcessor:  2 usages
    def read_data_from_file(file_name: str):

# Extract the data from the file
        try:
            file = open(FILE_NAME, "r")
            student_data = json.load(file)
            file.close()

        except Exception as e:
            IO.output_error_messages( message: "this file doesn't exist! Trying to open it again after creating... ", 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):
        """This function writes  data into a json file
        ChangeLog: (Who, When, What)
            O.Richer, 6/01/2025,Created function
        """

```

Fig3

```python

        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file, indent=2)
            file.close()
            print("The following data was saved to file!")
            for student in students:
                print(f'Student {student["FirstName"]} '
                      f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        except Exception as e:
            if file.closed == False:
                file.close()
            IO.output_error_messages( message: "Please check that the file is not open by another program.", e)


class IO:  9 usages
    """ A collection of presentation layer functions that manage user input and output
            ChangeLog: (Who, When, What)
            O.richer, 6/01/2025
    """
```

Fig4

```python

    @staticmethod  5 usages
    def output_error_messages(message: str, error: Exception = None):
            """ This function displays the custom error messages to the user
                ChangeLog:(Who, When, What)
                O.Richer, 6/01/2025, 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')

    @staticmethod  1 usage
    def output_menu(menu: str):
                    """This function print out the menu
                        ChangeLog:(Who, When, What)
                        O.richer,6/01/2025, Created function
                        Return: None
                    """
                    print()
                    print(menu)
                    print()
```

Fig5

```
113
114     @staticmethod  1 usage
115     def input_menu_choice():
116         """ This function gets a menu choice from the user
117             ChangeLog:(Who, When, What)
118             O.Richer, 6/01/2025, Created function
119             Return: string with the users choice
120         """
121         choice = "0"
122         try:
123             choice = input("Enter your choice: ")
124             if choice not in ("1", "2", "3", "4"):
125                 raise Exception("Please only choose option 1, 2, 3, or 4")
126         except Exception as e:
127             IO.output_error_messages(e.__str__())
128         return choice
129
130     @staticmethod  1 usage
131     def input_student_data(student_data:list):
132         """ This function takes student first and last name + the course  input from the user
133             ChangeLog:(Who, When, What)
134             O.richer,6/01/2025, Created function
135             Return: student registered data
136         """
```

Fig6

```
137         try:
138             student_first_name = input("Enter the student's first name: ")
139             if not student_first_name.isalpha() and student_first_name.find(" ")== -1:
140                 raise ValueError("The first name can only have alphabetic character.")
141             student_last_name = input("Enter the student's last name: ")
142             if not student_last_name.isalpha() and student_last_name.find(" ") == -1:
143                 raise ValueError("The last name can only have alphabetic character.")
144 
145             course_name = input("Please enter the name of the course: ")
146             student_data = {"FirstName": student_first_name,
147                             "LastName": student_last_name,
148                             "CourseName": course_name}
149             students.append(student_data)
150             print(f"You have registered {student_first_name} {student_last_name} for {course_name}. ")
151         except ValueError as e:
152             IO.output_error_messages( message: "Please do not enter numbers.", e)
153         except Exception as e:
154             IO.output_error_messages( message: "errors.", e)
155         return students
156 
```

Fig7

```
157 
158 
159 
160     @staticmethod  1 usage
161     def output_student_courses(student_data:list):
162         """ This function process the data to create and display a custom message
163             ChangeLog:
164             O.richer,6/01/2025, Created function
165             Return: student registered data
166         """
167         print("-" * 50)
168         print("\nThe current data is:")
169         for student in students:
170             print(f'Student {student["FirstName"]} '
171                   f'{student["LastName"]} is enrolled in {student["CourseName"]}')
172 
173         print("-" * 50)
174 
```

Fig8

```
175
176 # Main Program
177 students = FileProcessor.read_data_from_file(FILE_NAME)
178 while True:
179     # Present the menu of choices
180     IO.output_menu(menu=MENU)
181     menu_choice = IO.input_menu_choice()
182
183     # Input user data
184     if menu_choice == "1":  # This will not work if it is an integer!
185         students = IO.input_student_data(student_data = students)
186         continue
187
188     # Present the current data
189     elif menu_choice == "2":
190         IO.output_student_courses(student_data = students)
191         continue
192
193     # Save the data to a file
194     elif menu_choice == "3":
195         FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
196         continue
```

Fig9

```
197
198     # Stop the loop
199     elif menu_choice == "4":
200         break  # out of the loop
201
202
203 print("Program Ended")
```

Fig10

2. Running/Executing the script.

I tried to run different scenarios. From the well behave case to cases where the user input does not behave according to the menu. I also, to the best of my ability, tried to correct some deficiencies from my previous assignment. Of course, some creative user input can probably challenge this assignment (fig 1 through fig11)

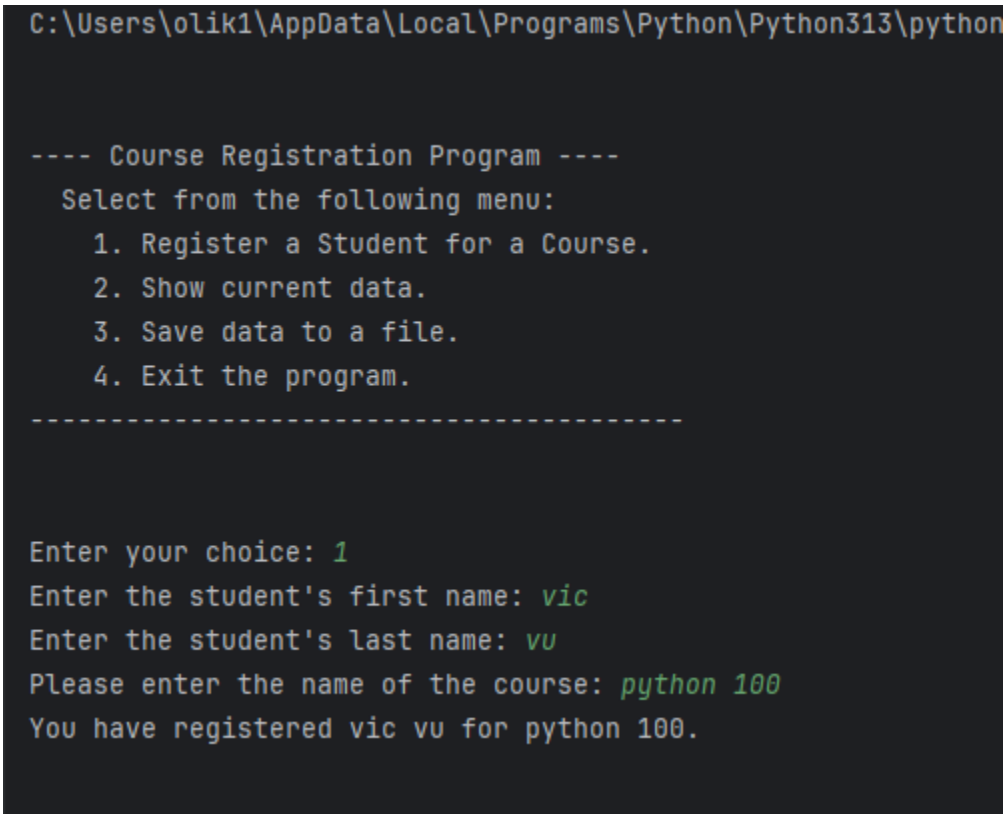


Fig1

```
Enter your choice: 1
Enter the student's first name: alain
Enter the student's last name: delon
Please enter the name of the course: python 100
You have registered alain delon 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.
----------------------------------------
```

Fig2

```
Enter your choice: 2
--------------------------------------------------

The current data is:
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student vic vu is enrolled in python 100
Student alain delon is enrolled in python 100
--------------------------------------------------
```

Fig3

```
Enter your choice: 2
-------------------------------------------------

The current data is:
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student vic vu is enrolled in python 100
Student alain delon is enrolled in python 100
-------------------------------------------------
```

Fig4

```
Enter your choice: 1
Enter the student's first name: jean paul
Enter the student's last name: belmondo
Please enter the name of the course: python 100
You have registered jean paul belmondo for python 100.
```

Fig5

```
Enter your choice: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student vic vu is enrolled in python 100
Student alain delon is enrolled in python 100
Student jean paul belmondo is enrolled in python 100
```

Fig6

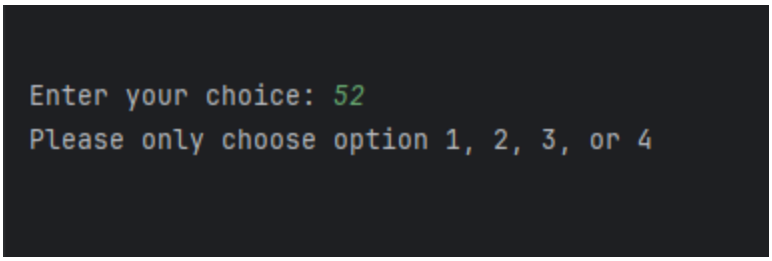


Fig7

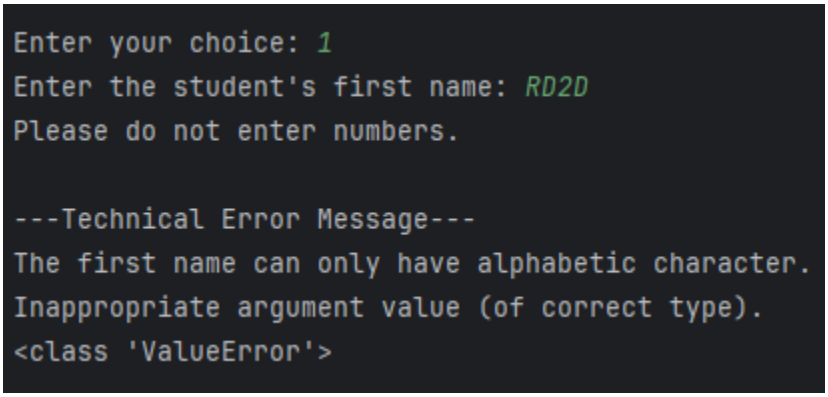


Fig8

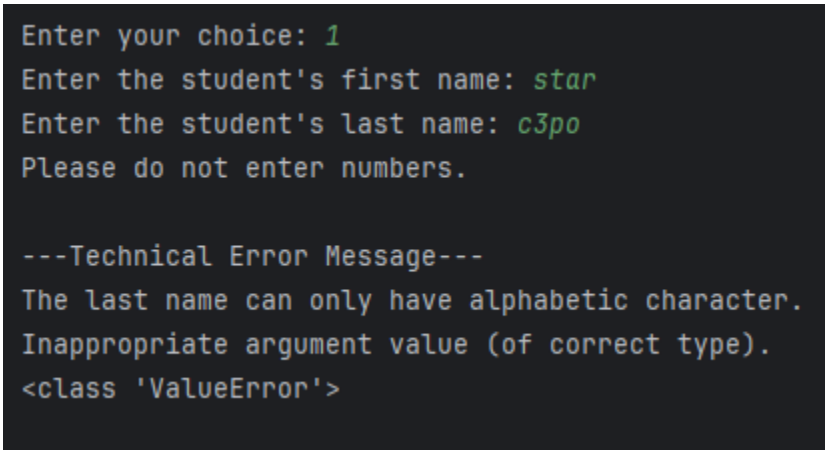


Fig9

```
---- 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 choice: 4
Program Ended
```

Fig10


```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "vic",
    "LastName": "vu",
    "CourseName": "python 100"
  },
  {
    "FirstName": "alain",
    "LastName": "delon",
    "CourseName": "python 100"
  },
  {
    "FirstName": "jean paul",
    "LastName": "belmondo",
    "CourseName": "python 100"
  }
]
```


Fig11

debugging.

Below are the following lessons I learnt:

- Indentations did create a lot of frustrations. Pycharm editor did help out to figure out the right format. Got 34 issues on that one.
- Got some issues with some parameters used on some functions.
- Some concepts are still nebulous, and sometimes you remove some line of code, and it works.

Conclusion.

The use of functions and classes were a novelty from previous assignments. I can see a good use working on a team trying to build a complex script. While this encapsulation of tasks within a class is an interesting concept, I will need some time to digest the concept let alone the possible pitfalls which come with it.

.