

Diego Mejia

November 19, 2024

Course: Introduction to Programming with Python

Assignment06 – Functions and Classes

[dmejia27/IntroToProg-Python-Mod06: Functions and Classes](#)

Functions and Classes for Course Registration

Introduction

The task for this assignment was 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 Assignment05, but It adds the use of functions, classes, and using the separation of concerns pattern.

We were introduced to functions and classes.

Functions

In programming, a function is a reusable block of code that performs a specific task or a set of tasks. Functions are a fundamental concept in most programming languages and serve several important purposes. The two we will focus on for now are:

- **Modularity:** Functions allow you to break down a large program into smaller, more manageable pieces. Each function can focus on a specific task, making the code easier to understand, maintain, and debug.
- **Reusability:** Once you've defined a function, you can use its code multiple times throughout your program without having to rewrite the same code. This promotes code reuse and helps prevent redundancy.

Functions allow you to group a set of programming statements and later reference them by a given name. In Python, you place your functions after the variable declarations since they must be defined in a script before they can be called. The area after the functions is defined is called the "main body" of the script. The statements within the function run later in your program by “calling” the function from the main body.

Classes

Another way to organize your code is by using classes. Classes are a way of grouping functions, variables, and constants by the name of the class. Using classes to group functions is a fundamental concept in programming. Grouping functions within classes create a modular structure, making it easier to manage and maintain code. Classes provide a natural way to organize code by grouping functions and data needed by those functions, making code more structured and readable, especially in larger projects.

Methodology

We begin with a Script Header as we have done in the past with a constant MENU: str that informs the user about the different options they have when running the program. The following is the MENU constant str() used.

```
---- 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 file name used in this assignment is the same as Assignment 03, Enrollments.csv. The variables used in this program are as follows.

Variables:

- **menu_choice: str is set to empty string.**
- **students: list : list is set to and empty list**

Classes:

- The program includes a class named FileProcessor.
- The program includes a class named IO.
- All classes include descriptive document strings.

Functions:

- All functions include descriptive document strings.
- All functions except blocks include calls to the function handling error messages.
- All functions use the @staticmethod decorator.
- The program includes functions with the following names and parameters:
 - output_error_messages(message: str, error: Exception = None)
 - output_menu(menu: str)
 - input_menu_choice()
 - output_student_courses(student_data: list)
 - input_student_data(student_data: list)
 - read_data_from_file(file_name: str, student_data: list):

The input/output(s):

- On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.
- On menu choice 2, the presents a string by formatting the collected data using the print() function.
- Data collected for menu choice 1 is added to a two-dimensional list table (list of dictionaries).
- All data in the list is displayed when menu choice 2 is used.

Error Handling

- The program provides structured error handling when the file is read into the list of dictionary rows.

```

        file_obj.close()
    except FileNotFoundError:
        IO.output_error_messages(f'There is no file {FILE_NAME}')

    file_obj = open(FILE_NAME, 'w')
    except Exception:
        IO.output_error_messages("There was an error opening the file")

```

Figure 1: Inserted a file name that did not exist in the current folder "Enrollments34.csv"

- The program provides structured error handling when the user enters a first name.

```

student_first_name = input("Enter Student's First Name: ")
if not student_first_name.isalpha():
    raise ValueError("First name must contain only letters")

```

Figure 2: Error when first name contains characters other than letters

- The program provides structured error handling when the user enters a last name.

```

student_last_name = input("Enter Student's Last Name: ")
if not student_last_name.isalpha():
    raise ValueError("Last name must contain only letters")

```

Figure 3: Error when last name contains characters other than letters

- The program provides structured error handling when the dictionary rows are written to the file.

```

except IOError:
    IO.output_error_messages(f'Error writing data into {FILE_NAME}')

```

Figure 4: Error when dictionary rows are written to the file

```

What would you like to do? 3
Closing file
Traceback (most recent call last):

  File c:\users\total\assignment06.py:59 in write_data_to_file
    raise KeyError("First name not in dict{}")

KeyError: 'First name not in dict{'

```

Figure 5: Error showing "KeyError"

Testing the Program

The grading criteria for the assignment are as follows:

- The program takes the user's input for a student's first, last name, and course name.

```
In [22]: %runfile C:/Users/total/Assignment06.py --wdir
Closing File

--- 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.
-----

What would you like to do? 1
Enter Student's First Name: Diego
Enter Student's Last Name: Mejia
Enter Course Name: Math 100
All Done!
```

Figure 6: The program displays the user's input for a student's first, last name, and course name

- The program saves the user's input for a student's first, last name, and course name to a coma-separated string file. (Check this in a simple text editor like notepad.)

```
What would you like to do? 2
Current List Table: Full Table of all students registered
[{'FirstName': 'Diego', 'LastName': 'Mejia', 'CourseName': 'Math 100'}]
All Done!

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

Figure 7: the IDE shows the input data and presents that the student was successfully registered

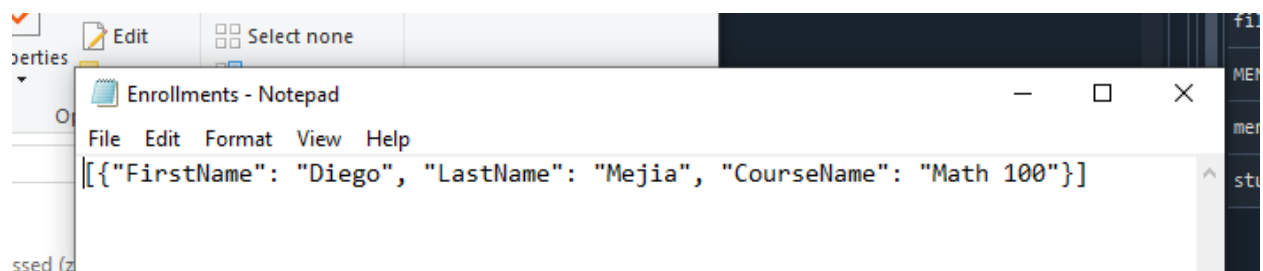


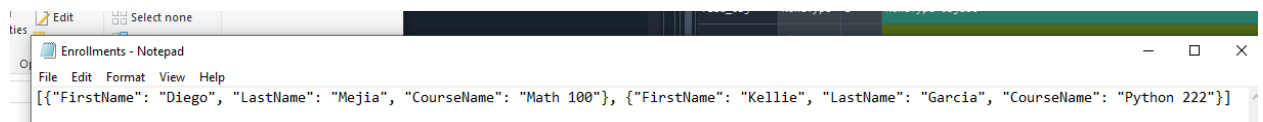
Figure 8: a simple check in a Notepad shows the data was successfully saved in the wanted format

- The program allows users to enter multiple registrations (first name, last name, course name).

```
What would you like to do? 2
Current List Table: Full Table of all students registered
[{'FirstName': 'Diego', 'LastName': 'Mejia', 'CourseName': 'Math 100'},
{'FirstName': 'Kellie', 'LastName': 'Garcia', 'CourseName': 'Python 222'}]
All Done!
```

Figure 9: Multiple registrations are shown, and students were successfully saved

- The program allows users to save multiple registrations to a file (first name, last name, course name).



```
File Edit Format View Help
[{"FirstName": "Diego", "LastName": "Mejia", "CourseName": "Math 100"}, {"FirstName": "Kellie", "LastName": "Garcia", "CourseName": "Python 222"}]
```

Figure 10: Multiple entries saved to a file in json format

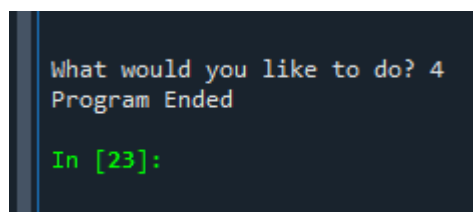
- The program runs correctly in both **the IDE and from the console or terminal.**

```
What would you like to do? 2
Current CSV Data: Last student entered in session
tony,shives,math 100

Current List Table: Full Table of all students registered
[{'FirstName': 'shake', 'LastName': 'shack', 'CourseName': 'burger 100'}, {'FirstName': 'tony', 'LastName': 'shives', 'CourseName': 'math 100'}]
```

Figure 11: the program successfully runs in the console or terminal

To exit the program we input menu item 4 as follows



```
What would you like to do? 4
Program Ended

In [23]:
```

Figure 12: Program ends

Summary

The task for this assignment was 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 Assignment05, but It adds the use of functions, classes, and using the separation of concerns pattern.

We were introduced to functions and classes.