

Adnan Dawood

May 29, 2024

Foundations of Programming: Python

Assignment 07

The link to the repository in GitHub: <https://github.com/adnandawood/IntroToProg-Python-Mod06>

## **Classes and Objects**

### **Introduction to Classes and Objects in Python**

In Python, classes and objects are core concepts of object-oriented programming (OOP). Classes serve as blueprints for creating objects, while objects are instances of classes with their own unique attributes and behaviors.

A class defines the structure and behavior of objects by encapsulating data (attributes) and functionality (methods) into a single entity. When we create an object from a class, we're essentially creating a specific instance of that class, inheriting its properties and behaviors.

Classes and objects provide a powerful way to organize and manage code, promoting modularity, reusability, and scalability in software development. They allow developers to model real-world entities and interactions, making it easier to design and maintain complex systems.

In Python, defining a class is straightforward, using the `class` keyword followed by the class name and its attributes and methods. Objects are then created by calling the class constructor, which initializes the object's state.

Overall, understanding classes and objects in Python is essential for building robust and flexible applications, enabling developers to create well-structured and maintainable codebases.

### **Drafting the Code:**

Break down the Python program into its components:

#### **Classes:**

- **Person:**

Represents a person with attributes `first_name` and `last_name`.

Provides methods to convert to JSON format (`to_json`), create a Person object from JSON data (`from_json`), and create a Person object from comma-separated data (`from_csv`).

- Student (Inherits from Person):

Represents a student, inheriting attributes and methods from Person.

Adds an attribute `course_name` to represent the course the student is enrolled in.

Provides methods to convert to JSON format (`to_json`), create a Student object from JSON data (`from_json`), and create a Student object from comma-separated data (`from_csv`).

- FileProcessor:

Handles file processing operations.

Provides methods to read data from a JSON file and load it into a list (`read_data_from_file`), and write data from a list to a JSON file (`write_data_to_file`).

- IO:

Handles input/output operations.

Provides methods to output error messages (`output_error_messages`), output a menu to the console (`output_menu`), input a menu choice from the user (`input_menu_choice`), output student data to the console (`output_student_courses`), and input student data from the user (`input_student_data`).

### Constants:

- MENU: A string representing the menu options for the program.
- FILE\_NAME: The name of the file used for storing enrollment data.

### Variables:

`student_data`: A list to store student enrollment data.

### Main Program Logic:

- Reads initial data from the specified JSON file using `FileProcessor.read_data_from_file`.
- Enters a menu loop where the user can choose from various options:
  1. Register a new student (`IO.input_student_data`).
  2. View current student data (`IO.output_student_courses`).
  3. Save data to a file (`FileProcessor.write_data_to_file`).
  4. Exit the program.
  5. Continues looping until the user chooses to exit.

### Change Log:

- Logs any changes made to the script, including the creator, creation date, and details of changes.

### Program End:

- Displays "Program Ended" when the user chooses to exit the program.

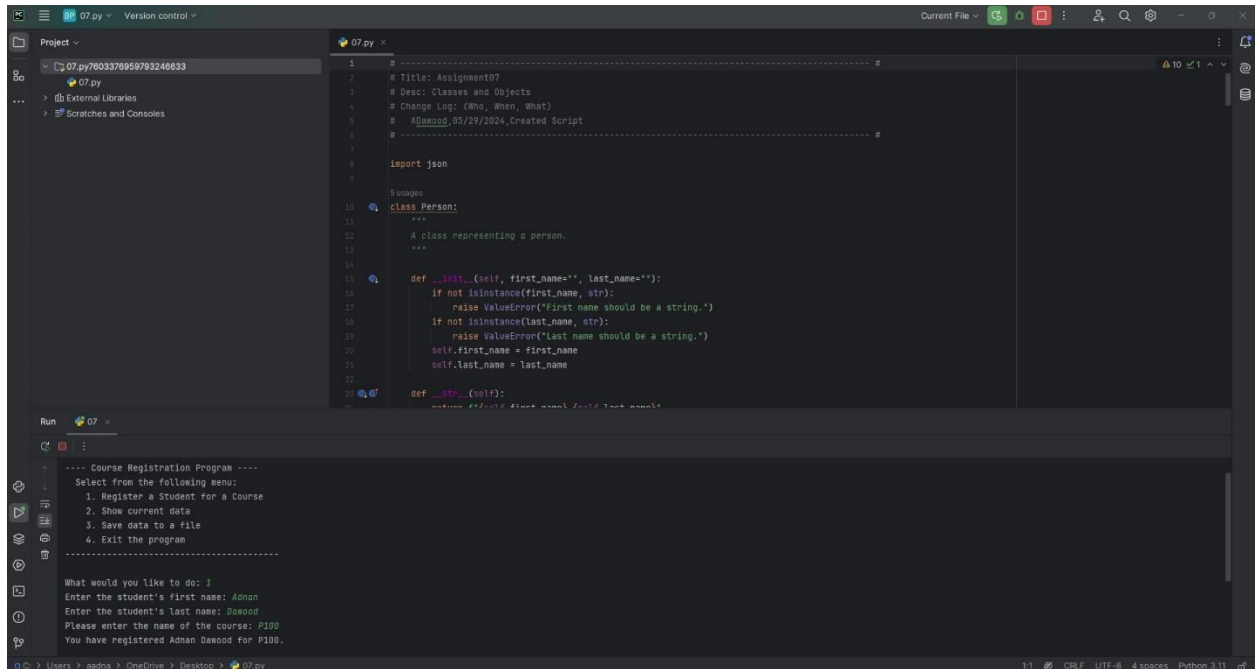
The breakdown provides a clear understanding of how the program is structured and what each component is responsible for.

Testing the script and the findings:

I used PyCharm to evaluate my script. Also, the script was evaluated in terminal.

### 1- Here is how my script looks like and its output in PyCharm:

- a- Running the program and entering 1 for option one “Register a Student for a Course” as shown in the picture below:

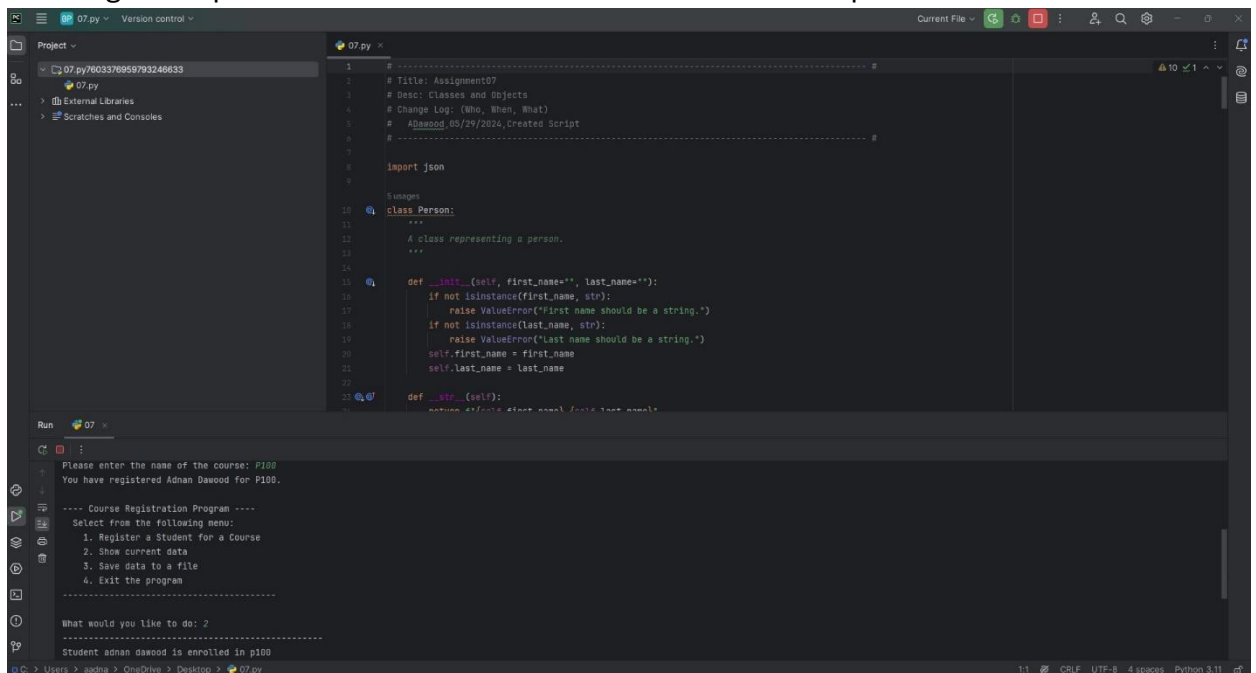


The screenshot shows the PyCharm IDE with a Python script named '07.py' and its output in the Run console. The script defines a 'Person' class with attributes 'first\_name' and 'last\_name', and a method 'register' that takes 'first\_name', 'last\_name', and 'course' as arguments. The output shows the program running and displaying a menu with four options. Option 1, 'Register a Student for a Course', is selected. The user enters 'Adnan' for the first name, 'Dawood' for the last name, and 'P100' for the course. The program outputs: 'You have registered Adnan Dawood for P100.'

```
1 # ----- #
2 # Title: Assignment07
3 # Desc: Classes and Objects
4 # Change Log: (Who, When, What)
5 # Adnan Dawood, 05/29/2024, Created Script
6 # ----- #
7
8 import json
9
10 # Suages
11 class Person:
12     """
13     A class representing a person.
14     """
15     def __init__(self, first_name="", last_name=""):
16         if not isinstance(first_name, str):
17             raise ValueError("First name should be a string.")
18         if not isinstance(last_name, str):
19             raise ValueError("Last name should be a string.")
20         self.first_name = first_name
21         self.last_name = last_name
22
23     def __str__(self):
24         return f"{self.first_name} {self.last_name}"
```

```
---- 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 the student's first name: Adnan
Enter the student's last name: Dawood
Please enter the name of the course: P100
You have registered Adnan Dawood for P100.
```

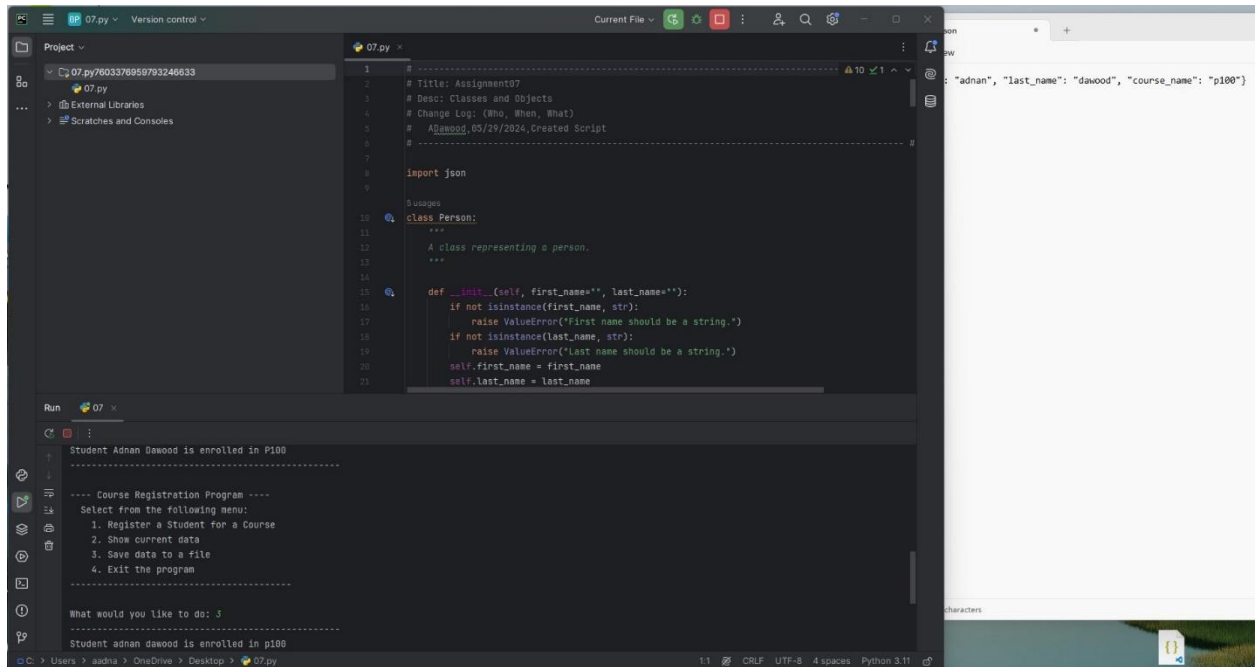
- b- Entering 2 for option two “Show current data” as shown in the picture below:



The screenshot shows the PyCharm IDE with the same Python script '07.py' and its output in the Run console. The output shows the program running and displaying the same menu as in the previous screenshot. Option 2, 'Show current data', is selected. The program outputs: 'Student adnan dawood is enrolled in p100'.1 # ----- #
2 # Title: Assignment07
3 # Desc: Classes and Objects
4 # Change Log: (Who, When, What)
5 # Adnan Dawood, 05/29/2024, Created Script
6 # ----- #
7
8 import json
9
10 # Suages
11 class Person:
12 """
13 A class representing a person.
14 """
15 def \_\_init\_\_(self, first\_name="", last\_name=""):
16 if not isinstance(first\_name, str):
17 raise ValueError("First name should be a string.")
18 if not isinstance(last\_name, str):
19 raise ValueError("Last name should be a string.")
20 self.first\_name = first\_name
21 self.last\_name = last\_name
22
23 def \_\_str\_\_(self):
24 return f"{self.first\_name} {self.last\_name}"

```
Please enter the name of the course: P100
You have registered Adnan Dawood for P100.
---- 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: 2
-----
Student adnan dawood is enrolled in p100
```

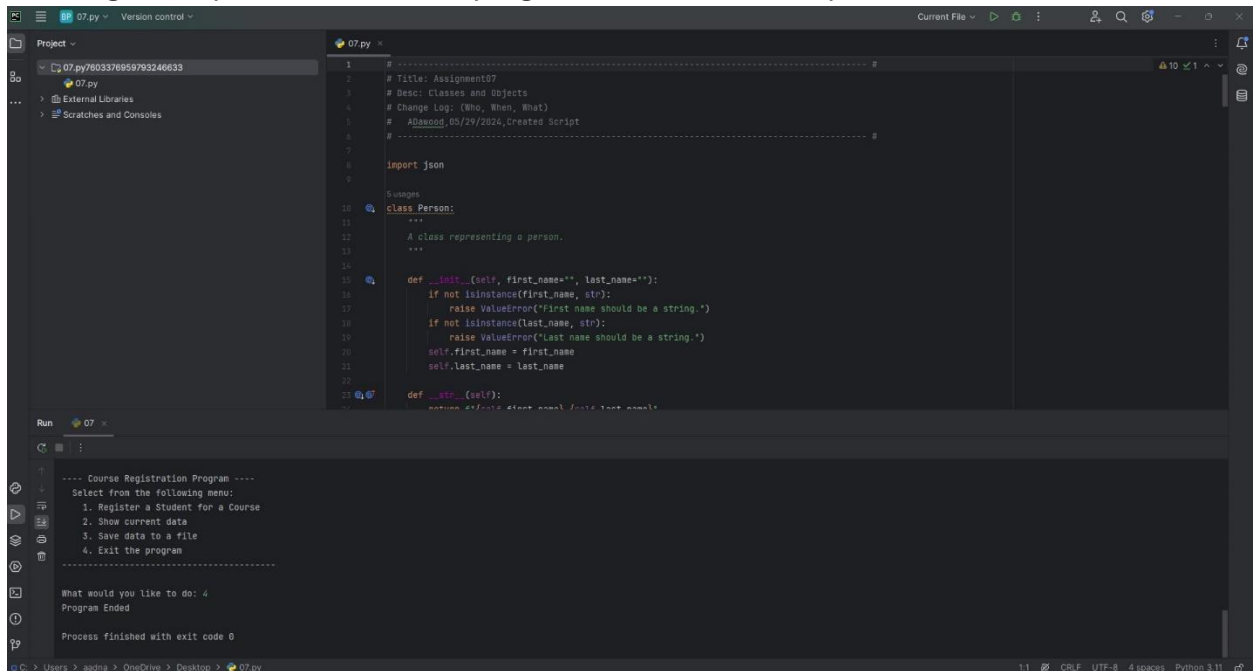
- c- Entering 3 for option three “Save data to a file (check the right side of the picture)” as shown in the picture below:



The screenshot shows a Python IDE with a file named '07.py'. The code defines a 'Person' class with an 'init' method that takes 'first\_name' and 'last\_name' as arguments. The 'init' method includes validation for string input and assigns the values to 'self.first\_name' and 'self.last\_name'. The 'str' method is also defined. The program's output in the Run console shows the following sequence of events:

```
Student Adnan Dawood is enrolled in P100
-----
---- 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: 3
-----
Student adnan dawood is enrolled in p100
```

- d- Entering 4 for option four “Exit the program” as shown in the picture below:

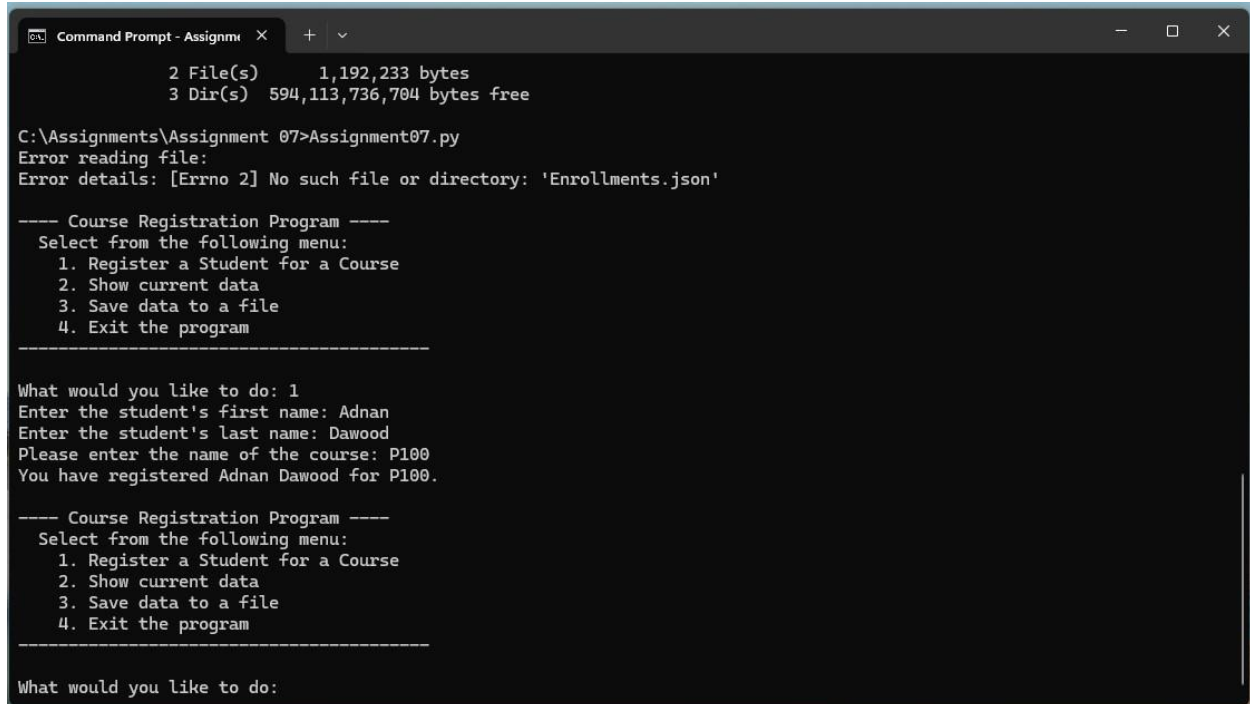


The screenshot shows the same Python IDE with the '07.py' file. The output in the Run console shows the following sequence of events:

```
---- 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: 4
-----
Program Ended
Process finished with exit code 0
```

## 2- Here is how my script looks like and its output in terminal:

- a- Running the program in terminal and entering 1 for option one “Register a Student for a Course” as shown in the picture below:



```
2 File(s)      1,192,233 bytes
3 Dir(s)      594,113,736,704 bytes free

C:\Assignments\Assignment 07>Assignment07.py
Error reading file:
Error details: [Errno 2] No such file or directory: 'Enrollments.json'

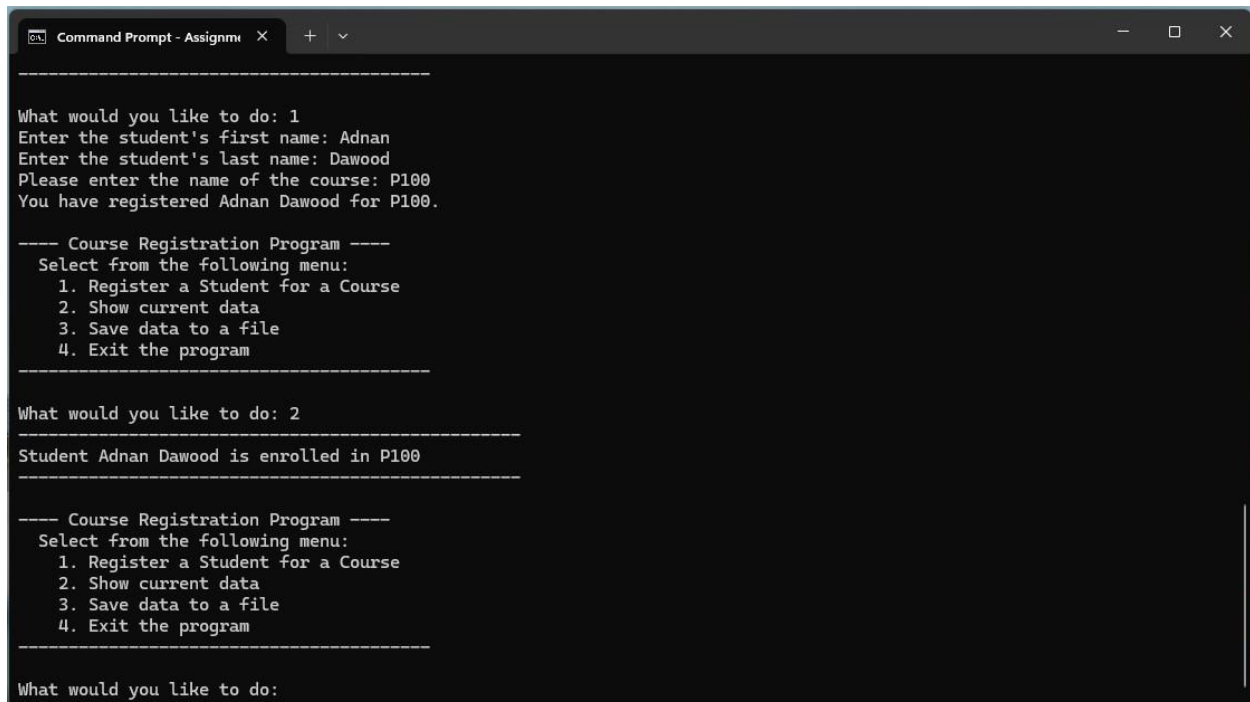
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

What would you like to do: 1
Enter the student's first name: Adnan
Enter the student's last name: Dawood
Please enter the name of the course: P100
You have registered Adnan Dawood for P100.

---- 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:
```

- b- Entering 2 for option two “Show current data” as shown in the picture below:



```
-----

What would you like to do: 1
Enter the student's first name: Adnan
Enter the student's last name: Dawood
Please enter the name of the course: P100
You have registered Adnan Dawood for P100.

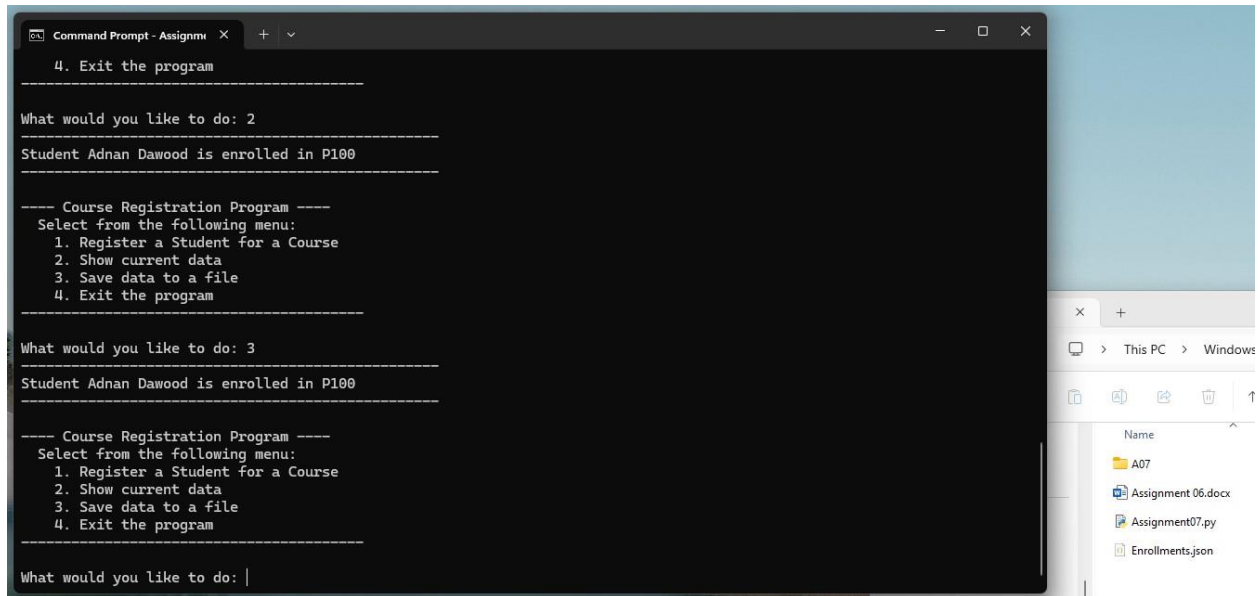
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

What would you like to do: 2
-----
Student Adnan Dawood is enrolled in P100
-----

---- 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:
```

- c- Entering 3 for option three “Save data to a file (check the right side of the picture)” as shown in the picture below:



```
Command Prompt - Assignm...
4. Exit the program
-----
What would you like to do: 2
-----
Student Adnan Dawood is enrolled in P100
-----

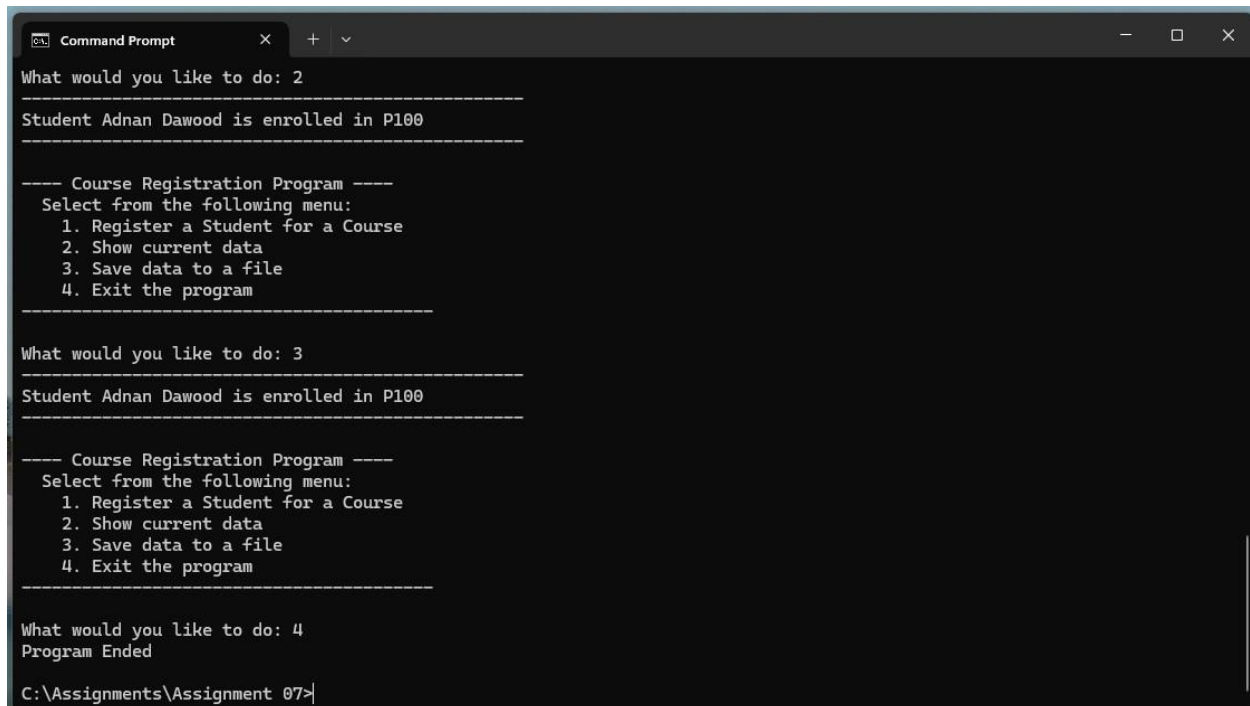
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

What would you like to do: 3
-----
Student Adnan Dawood is enrolled in P100
-----

---- 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: |
```

- d- Entering 4 for option four “Exit the program” as shown in the picture below:



```
Command Prompt
What would you like to do: 2
-----
Student Adnan Dawood is enrolled in P100
-----

---- 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: 3
-----
Student Adnan Dawood is enrolled in P100
-----

---- 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: 4
Program Ended

C:\Assignments\Assignment 07>
```

## **Summary:**

This Python program is designed as a Course Registration Program utilizing classes and objects. It features a comprehensive structure including error handling and data manipulation functionalities. Here's a summary of its key components:

### **Classes and Inheritance:**

The program defines two classes: Person and Student, where Student inherits from Person. These classes represent individuals and students respectively.

### **File Processing:**

It includes a FileProcessor class responsible for reading data from and writing data to a JSON file. This class ensures seamless handling of file operations.

### **Input/Output Operations:**

The IO class manages input/output operations, such as displaying menus, taking user choices, and presenting student data.

### **Error Handling:**

The program incorporates structured error handling mechanisms to handle exceptions gracefully, providing informative error messages to the user.

### **Data Handling:**

Student data is stored in a list of Student objects. The program can read initial data from a JSON file and allows users to register new students, view current data, and save data back to the file.

### **Menu-Driven Interface:**

The program offers a user-friendly menu-driven interface, enabling users to navigate through various options efficiently.

### **Change Log:**

The script includes a change log section documenting the creator, creation date, and any subsequent changes made to the script.