

<Kseniia Temnikova>

<08/28/2024>

<Programming Basics>

<Assignment_5>

Basic Python program

Introduction

In this assignment, I created a Python program to manage student enrollments for a course. The program allows users to enter a student's first name, last name, and course name, then organizes this information into a structured format using dictionaries and lists. The program can display the current registration data and save it to a CSV file for future reference.

To handle the data, I used dictionaries to store individual student records and lists to manage multiple records efficiently. The program also incorporates essential programming concepts such as constants, variables, user input, and file processing. Additionally, it includes error handling through try-except blocks to ensure the program runs smoothly, even when unexpected issues arise.

This project provided hands-on experience in building a functional Python application that is both easy to understand and use. The code is well-documented, making it straightforward for others to follow along and work with.

Constant

In my assignment, I was required to create constants MENU and FILE_NAME. The MENU constant holds the text for the program's main menu, while FILE_NAME stores the name of the file, "Enrollments.csv", where the data will be saved. I learned that constants are useful for storing values that remain the same throughout the program's execution. This ensures that important values, like file names or menu text, are consistent and easy to manage, without the risk of them being accidentally changed during the program's runtime. (Figure 1)

```
9      # Define the Data Constants
10     MENU: str = '''
11     ---- Course Registration Program ----
12     Select from the following menu:
13         1. Register a Student for a Course.
14         2. Show current data.
15         3. Save data to a file.
16         4. Exit the program.
17     -----
18     '''
19     # Define the Data Constants
20     FILE_NAME: str = "Enrollments.csv"
```

Figure 1: Constants

Input / output

In the Input/Output part of the program, I used lists and dictionaries to handle and show the data clearly. When the user enters a student's first name, last name, and course name, this information is stored in a dictionary, where each key corresponds to a specific piece of data (like "First name", "Last name", and "Course"). This dictionary, representing a single student's record, is then added to a list that holds all the student records.

Whenever the user asks to see the current data, the program goes through the list of student records and displays each one in a readable format. This way, we can manage multiple student registrations and display the data in a structured, easy-to-read manner.

Processing

For the Processing part of the program, I made sure that the data is handled smoothly from start to finish. When the program starts, it automatically reads the contents of the "Enrollments.csv" file. It uses a try-except block to handle any errors that might come up while trying to open or read the file. If there's an error, the program catches it and lets the user know something went wrong. (Figure 2)

The data from the file is split into individual pieces (like first name, last name, and course) and stored in a dictionary. Each of these dictionaries is then added to a list, which holds all the student records. This list helps to keep the data organized and easy to manage.

When the user selects menu option 3, the program opens the "Enrollments.csv" file in write mode. It then goes through the list of student records, writes each one to the file, and makes sure to close the file afterward. It uses a try-except block as well to handle any errors that might come up while trying to open or write to the file. (Figure 3) If there's an error, the program catches it and lets the user know something went wrong. This ensures that everything is saved correctly. The program also shows a confirmation of what was stored in the file.

Finally, if the user chooses menu option 4, the program ends. This controlled exit helps to make sure everything wraps up neatly.

```
try:
    file = open(FILE_NAME, "r")
    for row in file.readlines():
        # Transform the data from the file
        student_data = row.strip().split(',')
        student_dictionary={'First name':student_data[0], 'Last name':student_data[1], 'Course':student_data[2]}
        # Load it into our collection (list of lists)
        students.append(student_dictionary)
    file.close()
except Exception as e:
    print("There is an error while reading document, please fix that")
    print(e, e.__doc__)
```

Figure 2: Try-except block while trying to open or read the file

Using try-except blocks throughout the program helps catch errors that could happen during file reading, writing, or user input. This way, the program can handle unexpected situations without crashing, and it keeps the user informed about any issues that need attention.

```

} elif menu_choice == "3":
}     try:
        file = open(FILE_NAME, "w")
        csv_data = ''
        for student in students:
            csv_data += f"{student['First name']},{student['Last name']},{student['Course']}\n"
        file.write(csv_data)
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student['First name']} {student['Last name']} is enrolled in {student['Course']}")
}     except Exception as e:
        print("There is an error when the dictionary rows are written to the file, please fix that")
}     print(e, e.__doc__)
}     continue

```

Figure 3: Try-except block while trying to open or write to the file

Test

For testing the program, I focused on ensuring it handles user inputs and performs operations correctly. I tested the program's ability to accept and store a student's first name, last name, and course name, verifying that this data is saved properly in a list. I also checked that the program displays the collected information back to the user in a clear, comma-separated format.

I confirmed that the program can save the user's input to a CSV file, ensuring that the data is written correctly and can be viewed in a text editor. I tested the program's capability to handle multiple registrations, verifying that each set of data is stored in a list of dictionaries and displayed accurately when requested. (Figure 4)

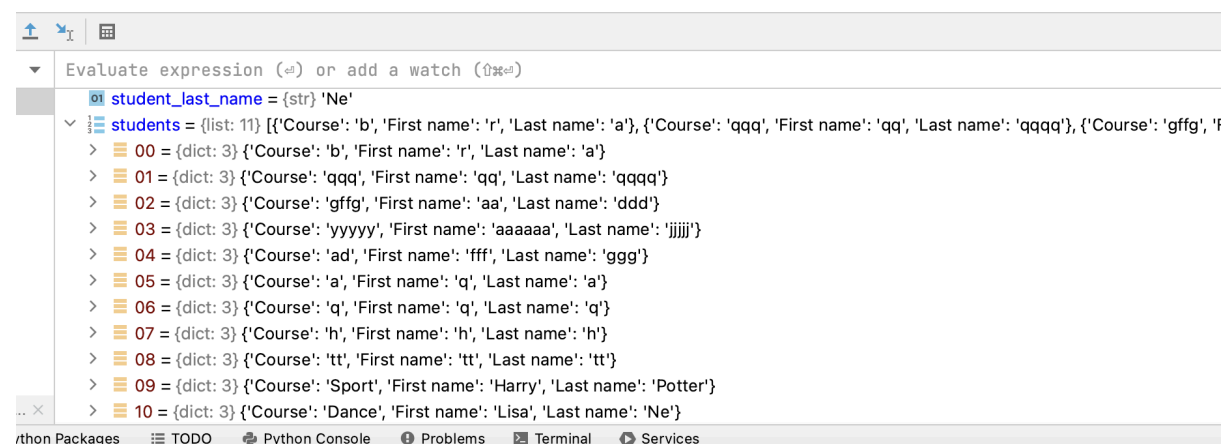
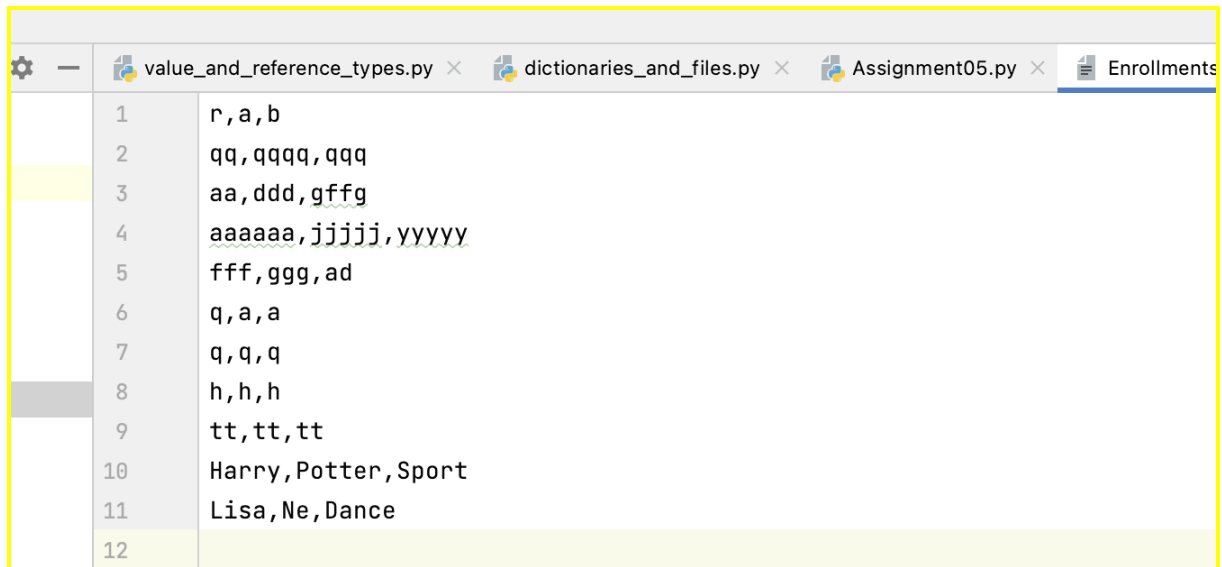


Figure 4: Debugger (data is stored in a list of dictionaries)

Furthermore, I checked that the program correctly saves all registrations to the CSV file, preserving the data for future use. (Figure 5) Finally, I ran tests in both PyCharm and the terminal to ensure the program works consistently and as expected in different environments. (Figure 6)



The screenshot shows a PyCharm editor window with four tabs: 'value_and_reference_types.py', 'dictionaries_and_files.py', 'Assignment05.py', and 'Enrollments.csv'. The 'Enrollments.csv' tab is active, displaying a CSV file with 12 lines of data. The data is as follows:

Line	Data
1	r,a,b
2	qq,qqqq,qqq
3	aa,ddd,gffg
4	aaaaaa,jjjjj,yyyyy
5	fff,ggg,ad
6	q,a,a
7	q,q,q
8	h,h,h
9	tt,tt,tt
10	Harry,Potter,Sport
11	Lisa,Ne,Dance
12	

Figure 5: CSV file

Summary

In this assignment, I developed a Python program to manage student enrollment information. The program gathers a student's first and last names, along with the course name, and organizes this data into a structured format. This information is then saved to a file named "Enrollments.csv."

The program uses constants for predefined course details and handles file operations to ensure data is saved correctly. I tested the program thoroughly in various environments to confirm that it functions as expected and meets all specified requirements. This project provided practical experience in handling user input, formatting strings, and managing files in Python.

```

What would you like to do: 1
Enter the student's first name: Max
Enter the student's last name: New
Please enter the name of the course: Jogging
You have registered r a for b.
You have registered qq qqqq for qq.
You have registered aa ddd for gffg.
You have registered aaaaaa jjjjj for yyyy.
You have registered fff ggg for ad.
You have registered q a for a.
You have registered q q for q.
You have registered h h for h.
You have registered tt tt for tt.
You have registered Harry Potter for Sport.
You have registered Lisa Ne for Dance.
You have registered Max New for Jogging.

```

```

---- 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 r a is enrolled in b
Student qq qqqq is enrolled in qq
Student aa ddd is enrolled in gffg
Student aaaaaa jjjjj is enrolled in yyyy
Student fff ggg is enrolled in ad
Student q a is enrolled in a
Student q q is enrolled in q
Student h h is enrolled in h
Student tt tt is enrolled in tt
Student Harry Potter is enrolled in Sport
Student Lisa Ne is enrolled in Dance
Student Max New is enrolled in Jogging
-----

```

```

---- 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
The following data was saved to file!
Student r a is enrolled in b
Student qq qqqq is enrolled in qq
Student aa ddd is enrolled in gffg
Student aaaaaa jjjjj is enrolled in yyyy
Student fff ggg is enrolled in ad
Student q a is enrolled in a
Student q q is enrolled in q
Student h h is enrolled in h
Student tt tt is enrolled in tt
Student Harry Potter is enrolled in Sport
Student Lisa Ne is enrolled in Dance
Student Max New is enrolled in Jogging

```

```

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.

```

Figure 6: Test in Terminal