Anna-Sofiia Mykhalevych

August 27, 2024

IT FDN 110 B

https://github.com/AnnaSofiia/IntroToProg-Python

Assignment 05

Advanced Collections and Error Handling

Introduction

For this assignment, you are required to develop a Python script that uses constants, variables, and print statements to manage and display information about a student's registration for a Python course. This script should extend the functionalities introduced in "Assignment 04" by integrating data manipulation using dictionaries and implementing exception handling to manage errors.

Defining the Data Constants and Variables

The **MENU** and **FILE_NAME** constants are key players here (Fig 1) —they stay the same throughout the program and help manage how users interact with the menu and where the data gets saved. The variables like **student_first_name**, **student_last_name**, and **course_name** are placeholders that will hold the information the user enters, **csv_data** is ready to take that information and format it neatly for storage, **file_obj** is there to handle the actual saving of data to a file, and **menu_choice** keeps track of what the user wants to do next. The **students_**list is also set up to store multiple records, allowing the program to manage and keep track of more than one student's information at a time.

```
# 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.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments05.csv"

# Data Variables and constants
# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: list = []  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Figure 1. *Defining the Data Constants and Variables*

Initiating file reading

In this part of the script, I practiced handling exceptions while trying to read data from a file. I used a try block to attempt to open and read the file, processing each line to extract student details. If the file wasn't found, I caught the FileNotFoundError to provide a clear message to the user about the missing file. Additionally, I included a catch-all Exception handler to address any other errors that might occur, which helps in debugging by printing out detailed error messages. This exercise reinforced my understanding of error management in Python, teaching me how to make my scripts more reliable and user-friendly.

```
# Initial file read
try:
    with open(FILE_NAME, "r") as file:
        for row in file:
            student_data = row.strip().split(',')
            students.append(student_data)
except FileNotFoundError:
    print("Text file must exist before running this script")
    print("-- Technical Error Message -- ")
    print("Please make sure the file 'Enrollments05.csv' is in the current directory.")
except Exception as e:
    print("There was a non-specific error!")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
```

Figure 2. *Initiating file reading*

Menu presentation and processing the data

In this part of my Python script, I implemented a menu-driven interface that processes user choices for registering a student for a course. When a user selects the option to register a student, they're prompted to enter the student's first and last names and the course name. I added error handling to ensure that the names don't contain any numbers or special characters by using the isalpha() method, which checks if the string contains only alphabetic characters. If the input is invalid, the script raises a ValueError and provides a specific error message.

```
# Menu presentation and processing data
while True:
    print(MENU)
    menu_choice = input("What would you like to do: ").strip()

    if menu_choice == "1":
        try:
            student_first_name = input("Enter the student's first name: ").strip()
            if not student_first_name.isalpha():
                raise ValueError("The first name should not contain numbers or special characters.")
            student_last_name = input("Enter the student's last name: ").strip()
            if not student_last_name.isalpha():
                raise ValueError("The last name should not contain numbers or special characters.")
            course_name = input("Please enter the name of the course: ").strip()
            students.append([student_first_name, student_last_name, course_name])
            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        except ValueError as ve:
            print(ve)
        except Exception as e:
            print("An unexpected error occurred:", e)
```

Figure 3. *Menu presentation and processing the data(a)*

In this section of my Python script **(Fig 4)**, I focused on displaying the current data from the list of registered students. This is triggered when a user selects the option "2" from the menu. To visually separate the content, I used a line of dashes before and after displaying the data. For each student in the list, the script prints out a formatted string that includes the student's first name, last name, and the course they are enrolled in.

```
elif menu_choice == "2":
    print("-" * 50)
    for student in students:
        print(f"Student {student[0]} {student[1]} is enrolled in {student[2]}")
    print("-" * 50)
```

Figure 4. *Menu presentation and processing the data(b)*

In this part of the script **(Fig 5)**, when the user selects the option "3" from the menu, I implemented a function to save the current student data to a file. I used a try block to handle potential issues during file operations. Here, the script opens the specified file in write mode and iterates over each student in the students list. It formats their information into a comma-separated string and writes this directly to the file. Each entry is written on a new line to keep the data organized.

After writing, the script confirms the successful save with a message "Data has been saved to the file." If any error occurs during the file operation, the script catches the exception, prints an error message "Failed to save data to file," and outputs the specific error details.

```
elif menu_choice == "3":
    try:
        with open(FILE_NAME, "w") as file:
            for student in students:
                csv_data = f"{student[0]},{student[1]},{student[2]}\n"
                file.write(csv_data)
        print("Data has been saved to the file.")
    except Exception as e:
        print("Failed to save data to file:")
        print(e, e.__doc__, type(e), sep='\n')
```

Figure 5. *Menu presentation and processing the data(c)*

In this segment of my script **(Fig 6)**, I managed the termination of the program and input validation for the menu selection. When the user chooses the option "4," the script prints

"Program Ended" and exits the loop with a break statement, effectively closing the program. This gives users a clear exit point and ensures the program terminates gracefully.

For other inputs, I included an else clause to handle any user entries that don't match the valid options ("1", "2", or "3"). This clause alerts the user with the message "Please only choose option 1, 2, or 3," guiding them back to the valid menu choices.

```
elif menu_choice == "4":
    print("Program Ended")
    break

else:
    print("Please only choose option 1, 2, or 3")
```

Figure 6. *Menu presentation and processing the data(d)*

Summary

In this Python project, I developed a course registration system that allows user interaction through a menu. The program handles user inputs with validation, displays registered student data, saves data to a CSV file, and provides options to exit or notify users of invalid inputs.

Key learnings include:

- **Input Validation:** Ensuring data integrity by checking for non-alphabetic characters.
- **Data Display and File Handling:** Looping through lists to display data and practicing safe file operations with exception handling.
- **Control Flow:** Using menu selections to manage program operations and user interactions efficiently.

This project significantly enhanced my understanding of Python by integrating fundamental programming concepts into a cohesive application. I have uploaded the files to the GitHub repository to share the project.