Erik Duong

November 14, 2024

Course: Python IT FDN 110

Assignment: 05

Create Course Registration Program

Introduction

This document outlines the steps taken 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 will **add the use of data processing using dictionaries and exception handling.**

Topic

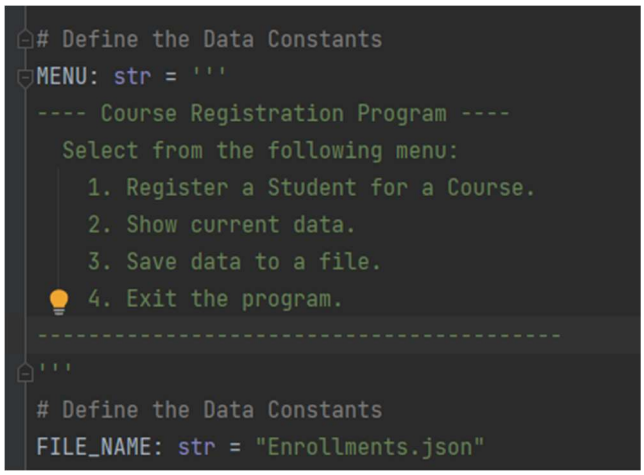
1. Understanding the Requirements

To begin, I thoroughly reviewed the assignment prompt to identify specific requirements, including the use of constants, variables, user input, string formatting, and file handling. The program needed a menu-driven structure, allowing for student registration, data display, data saving to a CSV file, and program exit.

2. Defining Constants

I defined two constants, MENU and FILE_NAME, to ensure consistency:

- MENU: Holds the program menu options.
- FILE_NAME: Holds the file name "Enrollments.json", which I need to update from csv file to json file to avoid further issue in the code.



3. Declaring Variables

The program defines a variety of variables, including student_first_name, student_last_name, and course_name, to hold user input during the execution of the program. The variable students is a list that stores the student data as dictionaries, while student_data holds a single student's data temporarily before it is added to the students list. Furthermore, I created a temporary variable called data to streamline my code.

```
# 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.
json_data: str = ''# Holds combined string data.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data.
data: list = []
```

4. File Handling and Exception Handling

Reading from a File: The program opens a file (Enrollments.json) to load existing student data. The file is read line by line, and each line is processed by splitting the comma-separated values into the corresponding fields (first name, last name, and course name). This data is then stored in a dictionary, which is added to the students list.

```
file = open(FILE_NAME, "r")
for row in file.readlines():
    # Transform the data from the file
    data = row.strip().split(',')
    student_first_name = data[0]
    student_last_name = data[1]
    course_name = data[2]
    student_data = {'first_name':student_first_name,'last_name':student_last_name,'course_name':course_name}
    # Load it into our collection (list of lists)
    students.append(student_data)
file.close()
```

Exception Handling: The use of try, except, and finally blocks is essential for handling potential errors. For instance:

- **File reading**: If the file does not exist or there's an issue reading the file, the program will print an error message but continue running.

- **Input validation**: When registering a new student, input is checked to ensure that names are alphabetic using str.isalpha(). If the user provides invalid input (non-alphabetic names), the program raises an error and prompts for the input again.

```
try:
    file = open(FILE_NAME, "r")
    for row in file.readlines():
        # Transform the data from the file
        data = row.strip().split(',')
        student_first_name = data[0]
        student_last_name = data[1]
        course_name = data[2]
        student_data = {'first_name':student_first_name,'last_name':student_last_name,'course_name':course_name}
        # Load it into our collection (list of lists)
        students.append(student_data)
    file.close()
except Exception as e:
    print('Unknown exception', e)
finally:
    if not file.closed:
        file.close()
```

The program also demonstrates how to collect input from users and validate it before proceeding. For example, user inputs for first and last names are validated using the str.isalpha() method to ensure that only alphabetic characters are accepted. This validation prevents issues with invalid data that could cause errors later in the program.

```
if menu_choice == "1":  # This will not work if it is an integer!
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError('First name must be alphabetic. Please try again.')
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError('Last name must be alphabetic. Please try again.')
        course_name = input("Please enter the name of the course: ")
        student_data = {'first_name':student_first_name,'last_name':student_last_name,'course_name':course_name}
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        print(e)
```

Lastly, the program demonstrates how to save structured data (student information) back into a file. After registering students, the data is written to a file (Enrollments.json) in CSV format, where each student's data is written as a comma-separated string. This file can be opened later to retrieve and process the data.

```python
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        for student in students:
            json_data = f"{student['first_name']},{student['last_name']},{student['course_name']}\n"
            file.write(json_data)
        file.close()
    except Exception as e:
        print('Error saving data to file')
        print(e)
    finally:
        if file and not file.closed:
            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_name']}")
```