

Course Registration - Dictionaries and Exceptions

Introduction

Assignment 05 involves creating a Python script which gives the user the following options:

1. Register a student for a course
2. Show current data
3. Save data to a file
4. Exit the program

In addition to the above actions, the program will initially open up an existing file named **"Enrollments.json"**, store the existing data in a variable, and write it back to the truncated file with any additional data the user provides.

Code

The decision making process of the code allowing the user to have multiple options is handled through **if** statements. The ability given to the user to make multiple choices is handled through an infinite looping process using the **while TRUE** statement until the user decides to close the program.

```
# -----  
# ----- H E A D E R -----  
# -----  
# Title: Assignment05  
# Desc: This assignment demonstrates using dictionaries, files, and exception handling  
# Change Log: (Who, When, What)  
#   Jason Noumeh,11/20/2023,Created Script  
#   <Your Name Here>,<Date>, <Activity>
```

Figure 1: Script Header

Initialization

The code begins with the initialization of constants and variables. Constants are set with predefined values which do not change, and variables are initialized to empty values with a string or list data type. Note the usage of triple apostrophe for the purpose of preserving the format of multi-line strings.

```
# 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.
-----
'''

# Define the Data Constants
FILE_NAME: str = 'Enrollments.json'

# 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.
csv_data: str = '' # Holds combined string data separated by a comma.
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

file = None # Holds a reference to an opened file.
```

Figure 2: Initialization of Constants and Variables

Body of Code

The first functionality of the code is to read the data in the existing file “**Enrollments.json**” and add the information to a list of dictionaries called “**students**”. This is accomplished by using the **.load** method associated with the json module. The **json** module is brought into the script with the **import json** function in the “Setup Code” section of the script.

```

# When the program starts, read the file data into a list of dictionaries (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, 'r')
    students = json.load(file)
    file.close()

except FileNotFoundError as e:
    print('Text file must exist before running this script!\n')
    print('-- Technical Error Message -- ')
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print('There was a non-specific error!\n')
    print('-- Technical Error Message -- ')
    print(e, e.__doc__, type(e), sep='\n')
finally:
    # Check if file is not None before attempting to close it
    if file is not None and not file.closed:
        file.close()

```

Figure 3: Reading Initial Data

Figure 3 displays try/except blocks to provide structured error handling while reading the **json** file. Any data initially in “**Enrollments.json**” will be stored in the “**students**” list variable. Since the data will be in json format, “students” will be a list of dictionary rows.

The user interaction portion of the code first begins with presenting the user with the menu of options. Since the intent of the code is to be recurring, meaning the code will continue until instructed otherwise, a **while** loop will be used to accomplish this. The loop will move forward with the code as long as the condition is true. By using **while True**, the code will loop indefinitely allowing the user to perform multiple actions until it’s desired to close the program.

```

# Present and Process the data
while (True):

    # Present the menu of choices
    print(MENU)
    menu_choice = input('What would you like to do: ')

```

Figure 4: While Statement

The user will be presented with the menu and choose a desired option. The program is able to determine what the user decides by allocating a numerical number to each option. The user is instructed to enter the numerical value using the **input()** function. The code reads this value and

uses a series of “if” and “else if” statements to move forward with the appropriate portion of code to execute. An example of this structure is given in **Figure 5**.

```
# Option 1 requires user input, this block of the while loop will also assign/format the data
if menu_choice == "1":
    student_first_name = input("Enter the student's first name: ")
    student_last_name = input("Enter the student's last name: ")
    course_name = input("Please enter the name of the course: ")
    csv_data = f"{student_first_name},{student_last_name},{course_name}"
    students.append(csv_data.split(",")) # appends the table created from existing file to add additional user input data

# Present the current data
# Option 2 prints the data back to the user
elif menu_choice == "2":
    print("The following data has been stored:")
    if student_first_name == "": # initial check to see if any current data is to be displayed
        for student in students:
            print(student)
        print("\nNo current data entered during this session.")
    else: # otherwise there is current data input from user during this session
        for student in students:
            print(student)
        print("\nThe current data for the most recent user entry is:")
        print(csv_data)

# Save the data to a file
# Option 3 saves the data to a csv file
elif menu_choice == "3":
    if student_first_name == "": # initial check to see if any data is to be appended to file
        print("No current data entered\nPlease either (1) enter data or (4) exit the program")
    else:
        file = open(FILE_NAME, "w")
        for student in students:
            file.write(f"{student[0]},{student[1]},{student[2]}\n")
            print(f"{student[0]},{student[1]},{student[2]}")
        file.close() # required to save the file
        print(f"\nThe above Data has been saved in {FILE_NAME}")

# Stop the loop
# Option 4 ends the program
elif menu_choice == "4":
    break # ends the while loop
```

Decision process
through “if” and “else
if” statements

Figure 5: Example of General Flow of Code

If the user specifies option one, a series of input() functions will be presented requesting the individual pieces of data and storing these in their respective variables. Expectation blocks have been added to this section to provide structured error handling while data is being inputted by the user. The pieces of data are appended to the “**students**” list with care taken to maintain the key structure in the json file. Note, when this process is repeated, the variables used in this block of the “if” statement will be reassigned to the new values. This allows data for multiple students to be appended to the list.

```

# Input user data
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('The first name should not contain numbers.')
        student_last_name = input('Enter the student\'s last name: ')
        if not student_last_name.isalpha():
            raise ValueError('The last name should not contain numbers.')

        course_name = input('Please enter the name of the course: ')
        student_data = {'FirstName': student_first_name,
                        'LastName': student_last_name,
                        'CourseName': course_name}
        students.append(student_data)
    except ValueError as e:
        print(e) # Prints the custom message
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    except Exception as e:
        print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')

    continue

```

Figure 6: Option One Code Block: Input Data

If the user specifies option two, the stored information will be displayed to the user. A **for** loop is used to parse out each element in “**students**” and print the data back to the user. This data includes the data read from the json file in addition to any user entered data.

```

# Present the current data
elif menu_choice == '2':

    # Process the data to create and display a custom message
    print("-"*50)
    for student in students:
        print(f'Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}')
    print("-"*50)
    continue

```

Figure 7: Option Two Code Block: Present Data

If the user specifies option three, the most current stored list in the variable “**students**” (containing pre-existing data and newly added data) will be written to a json file named “**Enrollments.csv**”. This will truncate any pre-existing data on the file. In addition, the program will display all the data being stored on the file assuring the user. Again, an exception block is added to provide structured error handling.

```
# Save the data to a file
elif menu_choice == '3':
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        continue
    except TypeError as e:
        print("Please check that the data is a valid JSON format\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    except Exception as e:
        print("-- Technical Error Message -- ")
        print("Built-In Python error info: ")
        print(e, e.__doc__, type(e), sep='\n')
    finally:
        if file.closed == False:
            file.close()

    print('-'*50)
    print('You have chosen to save the following data to "Enrollements.json":')
    for student in students:
        csv_data = f'{student['FirstName']},{student['LastName']},{student['CourseName']}'
        print(csv_data)
    continue
```

Figure 8: Option Three Code Block: Save Data

As stated before, the code will continue to run based on how the while loop is set up. Option four terminates the program by use of the **break** statement.

```
# Stop the loop
elif menu_choice == '4':
    break # out of the loop
else:
    print('Please only choose option 1, 2, or 3')

print('Program Ended')
```

Figure 9: Option Four Code Block: Close Program

Summary

The Python script is able to read a json file with existing data, save that data to a list, add additional data per user input and finally write this information back to the same file and save it. The following attached pages display the successful run of the code within both the Pycharm IDE and Command Prompt.

Initial JSON File



Pycharm Run

```
---- 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: Roasted
Enter the student's last name: Turkey
Please enter the name of the course: Python100

---- 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 Smoked Ham is enrolled in Python100
Student Mashed Potatoes is enrolled in Python100
Student Roasted Turkey is enrolled in Python100
-----
```

```
---- 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
-----

You have chosen to save the following data to "Enrollements.json":
Smoked,Ham,Python100
Mashed,Potatoes,Python100
Roasted,Turkey,Python100

---- 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: Pumpkin
Enter the student's last name: Pie
Please enter the name of the course: Python100
```


Pycharm Run

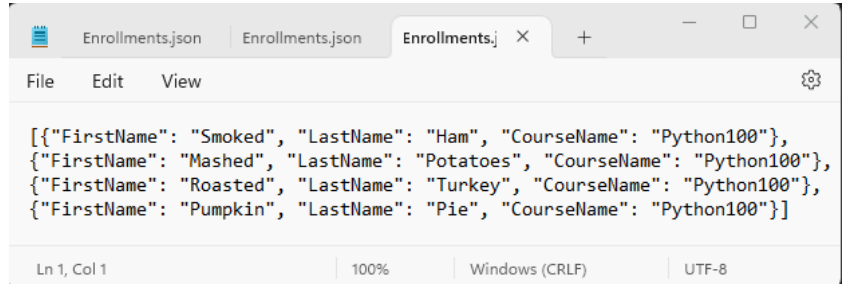
"Enrollments.csv"
after code is ran

```
---- 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 Smoked Ham is enrolled in Python100
Student Mashed Potatoes is enrolled in Python100
Student Roasted Turkey is enrolled in Python100
Student Pumpkin Pie is enrolled in Python100
-----

---- 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
-----
You have chosen to save the following data to "Enrollements.json":
Smoked,Ham,Python100
Mashed,Potatoes,Python100
Roasted,Turkey,Python100
Pumpkin,Pie,Python100
```



The screenshot shows the PyCharm IDE with a file named "Enrollments.json" open. The file contains a JSON array of four student records, each with "FirstName", "LastName", and "CourseName" fields. The status bar at the bottom indicates the cursor is at line 1, column 1, the zoom is at 100%, the line endings are Windows (CRLF), and the encoding is UTF-8.

```
[{"FirstName": "Smoked", "LastName": "Ham", "CourseName": "Python100"},
{"FirstName": "Mashed", "LastName": "Potatoes", "CourseName": "Python100"},
{"FirstName": "Roasted", "LastName": "Turkey", "CourseName": "Python100"},
{"FirstName": "Pumpkin", "LastName": "Pie", "CourseName": "Python100"}]
```

```
---- 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
```

Command Prompt Run

```

C:\> Command Prompt
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Laptop>cd C:\Users\Laptop\Documents\IT FDN 110 A - Foundations of Programming - Python\Module 5\PythonLabs

C:\Users\Laptop\Documents\IT FDN 110 A - Foundations of Programming - Python\Module 5\PythonLabs>python Assignment05.py

---- 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: Peacan
Enter the student's last name: Pie
Please enter the name of the course: Python100

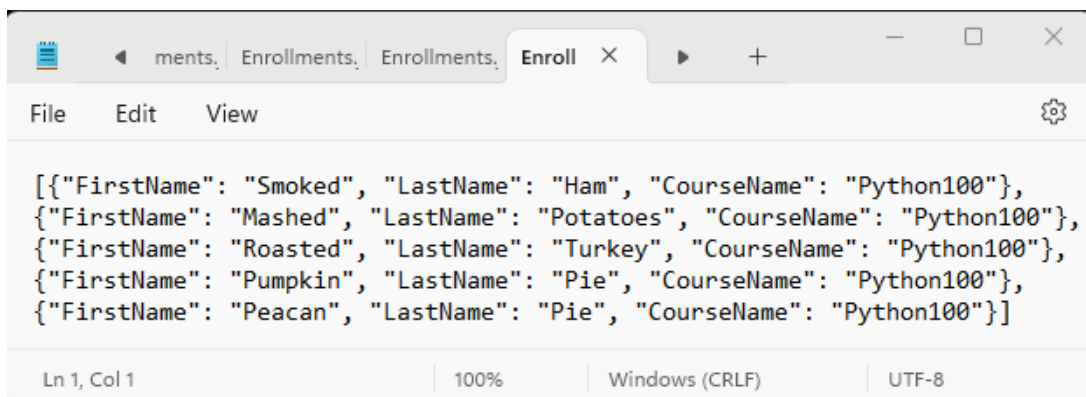
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 3
-----
You have chosen to save the following data to "Enrollements.json":
Smoked,Ham,Python100
Mashed,Potatoes,Python100
Roasted,Turkey,Python100
Pumpkin,Pie,Python100
Peacan,Pie,Python100

---- 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:\Users\Laptop\Documents\IT FDN 110 A - Foundations of Programming - Python\Module 5\PythonLabs>
```



The screenshot shows a code editor window with a tab titled "Enrollments". The editor displays a JSON array containing five objects, each representing a student's enrollment. The objects are: {"FirstName": "Smoked", "LastName": "Ham", "CourseName": "Python100"}, {"FirstName": "Mashed", "LastName": "Potatoes", "CourseName": "Python100"}, {"FirstName": "Roasted", "LastName": "Turkey", "CourseName": "Python100"}, {"FirstName": "Pumpkin", "LastName": "Pie", "CourseName": "Python100"}, and {"FirstName": "Peacan", "LastName": "Pie", "CourseName": "Python100"}. The editor interface includes a menu bar with "File", "Edit", and "View", a status bar at the bottom showing "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8", and a settings gear icon in the top right corner.

```
[{"FirstName": "Smoked", "LastName": "Ham", "CourseName": "Python100"},
{"FirstName": "Mashed", "LastName": "Potatoes", "CourseName": "Python100"},
{"FirstName": "Roasted", "LastName": "Turkey", "CourseName": "Python100"},
{"FirstName": "Pumpkin", "LastName": "Pie", "CourseName": "Python100"},
{"FirstName": "Peacan", "LastName": "Pie", "CourseName": "Python100"}]
```