

Jeff Howard

11/11/2024

Foundations Of Programming: Python

Assignment 05

# Advanced Collections and Error Handling

## Introduction

In this paper, I will review the steps I took to create a program in Python that presents a menu of options for a user to select, and depending on the user's selection, will either gather their data, present the data back, save the data to a file, or exit the program. This program builds on our 4th assignment by using dictionaries, JSON files and building in exception handling.

## Topic 1: Defining the constants and variables

### SubTopic 1A - Constants

In this assignment, we wanted to present a menu of options for the user to choose from - to be used in our conditional logic later in the code. The first constant is the menu of options and the second constant is the JSON file name where we will be saving the data to (fig 1).

```
# 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"
```

**Figure 1: Defined Constants**

### SubTopic 1B - Variables

The variables for this project are user inputted values - which are the first and last name of the student, the course name, the data to be saved (list and list of dictionaries), the file object (JSON file) and menu choices (Fig 2).

```
# 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: dict = {} # one row of student data
students: list = [] # a table of student data
json_data: str = '' # Holds combined string data separated by a comma.
file = _io.TextIOWrapper # Holds a reference to an opened file.
menu_choice: str # Hold the choice made by the user.
```

**Figure 2: Defined Variables**

## Topic 2: Load JSON into program

### SubTopic 2a - loading data

Once the data was defined, the next step was to load the data that is currently in the Enrollments.json file into the program. I used the 'json.load' function to load in the data after opening in read mode (fig 3).

```
file = open(FILE_NAME, "r")
students = json.load(file)
```

**Figure 3: Load json data into program**

### SubTopic 2a - Error handling

While reading the data into the program, I added a try-except block to create a custom message for errors loading the file. This is to give users a more user-friendly message when an error occurred in addition to the full technical message for developers who may also encounter the error.

```
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)

except FileNotFoundError as e: #custom error message if file name/path not found
    print("File does not exist in this location\n")
    print("Please try again\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')

except Exception as e: #generic message for other types of errors
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')

finally:
    if not file.closed:
```

**Figure 4: Load json data into program**

The first message is used for when a specific error occurs (FileNotFoundError) and the second is for all other types of errors. Finally I used 'finally' to close the file to ensure the file closes regardless of the errors encountered.

## Topic 3: Conditional Logic

### SubTopic 3a - Selection 1

The next step in the program was to start a 'while' loop to cycle through the menu options in our MENU constant. Once in the loop, I presented the MENU to the user and created an 'If' statement to cycle to tell the program what to do based on the user's selections. The first option in the "IF" statement is designed to get the user's information and append it as a list to the 'students' variable. Figure 5 shows the same inputs as in previous modules, with the addition of using error handling. I created an error message if the user entered a number or non-alpha character in their name (a), as well as a generic message (b) if any other type of error occurred during this step in the program.

```
if menu_choice == "1": #register a student for class - gather first, last and class name
    try:
        student_first_name = input("Enter the student's first name: ")
        A if not student_first_name.isalpha(): #check for non-alphabet entry in first name
            raise ValueError("Error - The last name should not contain numbers.\n")

        student_last_name = input("Enter the student's last name: ")
        A if not student_last_name.isalpha(): #check for non-alphabet entry in last name
            raise ValueError("Error - The last name should not contain numbers.\n")

        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_data['first_name']} {student_data['last_name']} "
              f"for {student_data['course_name']}.")

    except Exception as e:
        B print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
```

**Figure 5 : Gathering user input**

### SubTopic 3b - Selection 2

If the user selected menu option #2, (or the first elif in the conditional logic), then I simply printed back their information (Fig 6).

```

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

    # Print all users registered for the class including those recently added
    print("-" * 50)
    for student in students:
        print(f"Student {student['first_name']} {student['last_name']} is enrolled in {student['course_name']}")
    print("-" * 50)
    continue

```

**Figure 6: Presenting back user's input**

### SubTopic 3c - Selection 3

If the user selected menu option #3, then I saved the user's data to the json stored in FILE\_NAME. I used the open() function in 'write' mode to open the json file stored in the location of the script. Then I used the json.dump function to save the data stored in 'students'. In addition to saving the data, I again created custom error messages for the user. In this case, I customized an error message for a specific type of error - the "TypeError" - and another custom message for all other types of errors. After saving I made sure to close the file (Fig 7).

```

# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        print("Your data has been saved successfully.")
        continue

    except TypeError as e: #catch errors where data is not in JSON format
        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: #generic message for other types of errors
        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()
        continue

```

**Figure 7: Saving user's data to Enrollments.csv**

### SubTopic 3d - Selection 4

The last option or the 'else' statement in my conditional logic was to exit the program and break out of the loop. I also printed a message to the user if the user selected anything other than numbers 1-4 (Fig 8).

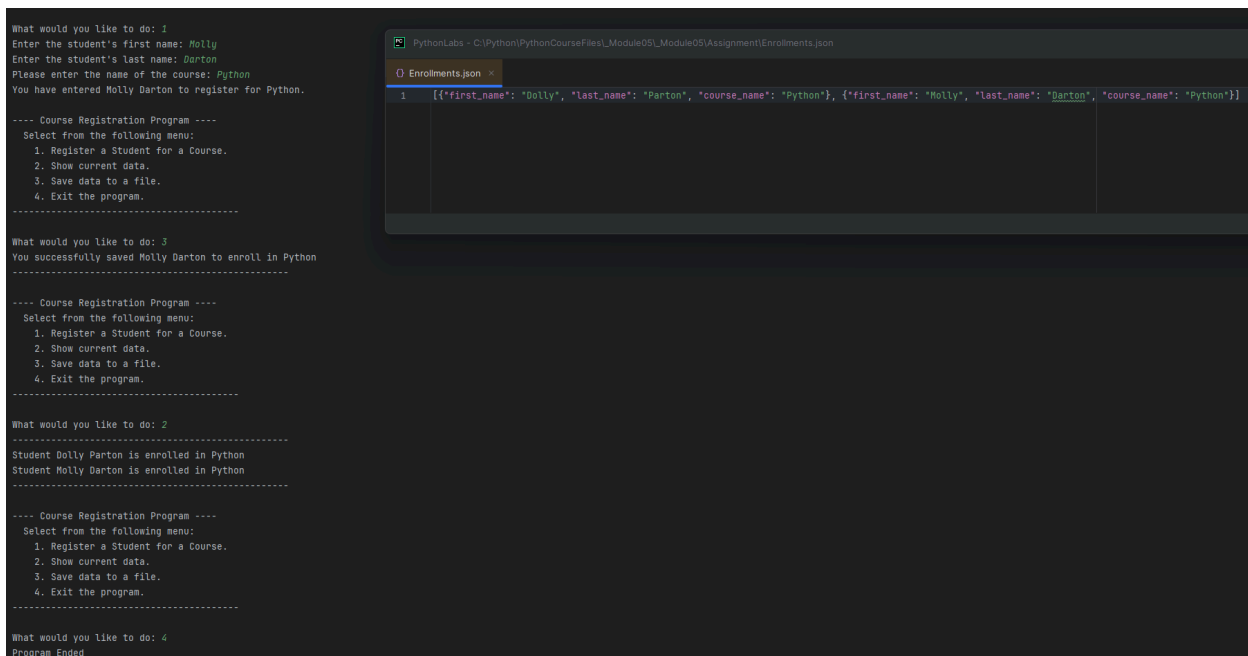
```
# Stop the loop
elif menu_choice == "4":
    break # out of the loop
else:
    print("Please only choose option 1, 2, 3, or 4") #in case user enters anything other than numbers 1-4
print("Program Ended")
```

**Figure 8: Exit the program**

## Topic 4: Testing code

### SubTopic 4a - Testing in PyCharm

To test that the script is working properly in PyCharm and the console, I ran the script and followed each of the prompts to ensure the program was saving the user-inputted data to the Enrollemnts.csv file. (Fig 8 and Fig 9)



```
What would you like to do: 1
Enter the student's first name: Molly
Enter the student's last name: Barton
Please enter the name of the course: Python
You have entered Molly Barton to register for Python.

---- 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 successfully saved Molly Barton to enroll in Python
-----

---- 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 Molly Barton is enrolled in Python
Student Molly Barton is enrolled in Python
-----

---- 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
```

PythonLabs - C:\Python\PythonCourseFiles\Module05\Module05\Assignment\Enrollments.json

0 Enrollments.json

```
1 [{"first_name": "Molly", "last_name": "Barton", "course_name": "Python"}, {"first_name": "Molly", "last_name": "Barton", "course_name": "Python"}]
```

**Figure 9: Testing in PyCharm**

### SubTopic 4b - Testing in Console

```
---- 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 Dolly Parton is enrolled in Python  
Student Molly Darton is enrolled in Python  
-----
```

```
---- 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: Polly  
Enter the student's last name: Barton  
Please enter the name of the course: Python  
You have entered Polly Barton to register for Python.
```

```
---- 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 Dolly Parton is enrolled in Python  
Student Molly Darton is enrolled in Python  
Student Polly Barton is enrolled in Python  
-----
```

```
---- 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.
```

***Figure 10: Testing in Console***

**Summary**

In summary, I reviewed the steps I took to create a program in Python that opens and reads a json file's data into the IDE, presents a menu of options for a user to select, and depending on the user's selection, will either gather their data, present the data back, save the data to a file, or exit the program. In addition, I used dictionaries and error handling learned this week in Module 5.