

Allan Duldulao

May 17, 2025

Foundations of Programming: Python

Assignment 05

GitHub: <https://github.com/AllanDuldulao/IntroToProg-Python-Mod05>

Advance Collections and Error Handling

Introduction

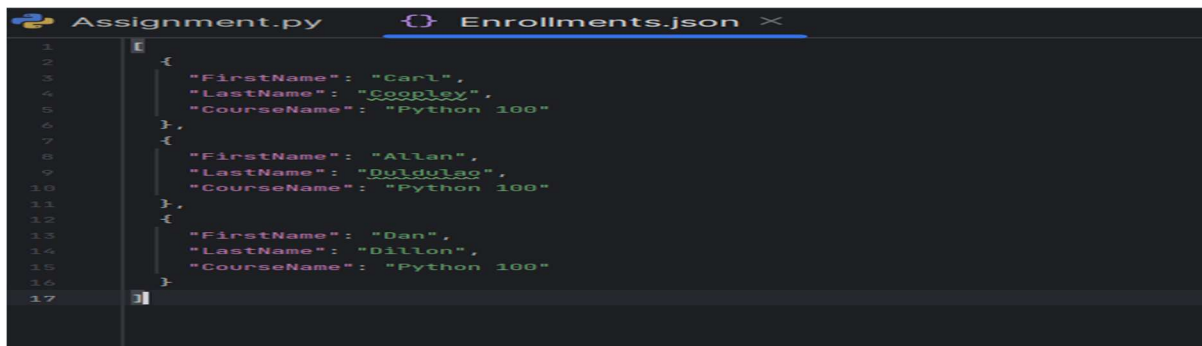
As we get deeper on the course. We have learned additional programming tools and techniques. In this module we are going to delve on advance collections and error handling. We will write a script and save the data collected into a dictionary using JSON. We'll also navigate on the importance of Error Handling by using Try-Except approach. After we've done writing the script, we will by sharing our code to a code storage GitHub. With these the public can look on to our code.

List and Dictionary

Lists are ordered collections of items that can be of different data types. List is mutable. Can store elements of different types (e.g. int, float, str, object, bool). List are indicated by square brackets []. On a list you can use index to access element in ordered sequence. An integer will be use to achieve this. Dictionaries are key-value pairs, where each value is associated with a unique key. They are used for mapping between keys and values and are often used for data that needs to be looked up quickly. Dictionaries are also created using curly braces {}, but when Python finds data that includes a ":" separator, it knows you want dictionary instead of a set. The Keys in the dictionary are case sensitive, so a programmer must be careful to always match the key's name exactly or you will get errors. There's a lot differences between List and Dictionaries. Data organization, accessing data, Data structure choice, Data validation, Data transformation, and field names are the primary differences between them.

JSON Files

JavaScript Object Notation is a lightweight data-interchange format that is easy for both humans to read and write and for machines to parse and generate. The basic structure of JSON file consists of key-value pairs, keys are strings and enclosed in double quotes, values can be string, numbers, objects, arrays, Booleans, or null. Data is organized using curly braces {} for objects and square brackets [] for arrays. Fig. 1. Sample of JSON.



```
1 {  
2   "FirstName": "Carl",  
3   "LastName": "Cooley",  
4   "CourseName": "Python 100"  
5 },  
6 {  
7   "FirstName": "Allan",  
8   "LastName": "Duldasog",  
9   "CourseName": "Python 100"  
10 },  
11 {  
12   "FirstName": "Dan",  
13   "LastName": "Dillon",  
14   "CourseName": "Python 100"  
15 }  
16 }  
17 }
```

Fig. 1: Example of JSON file

The uses of JSON vary. JSON is commonly used to exchange data between client and a web server in web applications. It's a format that both front-end and back-end systems can easily work with. It also used in File configurations, data storage and API responses. Is it suitable for NoSQL databases that store data in document-oriented format. JavaScript based applications often use JSON for data storage and communication.

When writing a script, it is very important to remember to import a JSON file before writing the body of the script. Importing a JSON file before starting a script is often necessary when the script relies on data, settings, or configurations stored in the JSON file to function correctly. Some reasons it is typically done at the start of the scripts are, load required data, initialize configurations, set up application state, avoid runtime errors, enable dynamic behavior, and preprocessing or validation. Importing and parsing the JSON at the start ensures the credentials are available before attempting the connection.

Structured Error Handling (Try-Except)

Exception handling is Python's way of responding to unexpected situations in your program. Instead of crashing when error occurs, your program can catch these exceptions and respond appropriately. When you are programming, you fix your bugs immediately and make sure the code runs smoothly. However, it often happens that other people introduce a new bug when they use your program. Error handling improves your scripts by managing errors you may not have control over in any other way. Python and most modern languages allow you to trap errors in your program using Try-Except construct. The try-except construct in programming, primarily used in languages like Python, is a mechanism for handling exceptions (errors) that may occur during code execution. It allows you to manage errors without crashing the program. Fig. 2 is example of try-except that I used in my code.

```

try:
    file = open(FILE_NAME, "w")
    json.dump(students, file, indent=1) # Added the indent so that my datas are easily readable.
    file.close()
    print("The following data was saved to file!")
    print(f"Student {student_first_name} {student_last_name} is enrolled in {course_name}")
    continue
except TypeError as e:
    print("Please make sure 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()

```

Fig. 2: Try-Except and its different components

Exception handling starts with the **“try”**, it contains code that might raise an exception. **“except”** specifies how to handle the specific exceptions (or general one). You can catch specific exception types (e.g. `TypeError`, `ValueError`) or use a broad **“Exception”** class. And **“finally”** is to execute regardless or whether an exception occurred or not, often used in closing files. How they work? When the **“try”** block is executed, an exception occurs, exception jumps to the matching except block, and **“finally”** always runs even if exception is unhandled. Some specific points to remember when using try-except, use specific exceptions for precise error handling, multiple **“except”** blocks can handle different exceptions, and you can also use raise an exception manually with **“raise”**, (Fig. 3) to trigger handling. This construct promotes robust error handling, ensuring programs can recover from or gracefully handle unexpected issues. The exception block is not just to displaying messages. It can also use to avoid errors later in the code.

```

try:
    student_first_name = input("Enter the student's first name: \n")
    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: \n")
    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: \n")
    student_data = {"FirstName": student_first_name,
                    "LastName": student_last_name,
                    "CourseName": course_name}

```

Fig. 3: Using raise to raise an exception manually

Network and Cloud Sharing of Files

Saving and sharing a code file are essential practices that foster collaboration, code quality, knowledge sharing, code reusability, error detection, documentation, version control, back up and recovery, remote accessibility, open-source development, and overall efficiency in software development.

Network file sharing is like a shared folder on a computer network. Developers often work on a project as a team and when several programmers work together, it's important to have a central place where they can share and store their code. Using network sharing. Developers can collaborate more effectively because they can easily access and modify the same set of files.

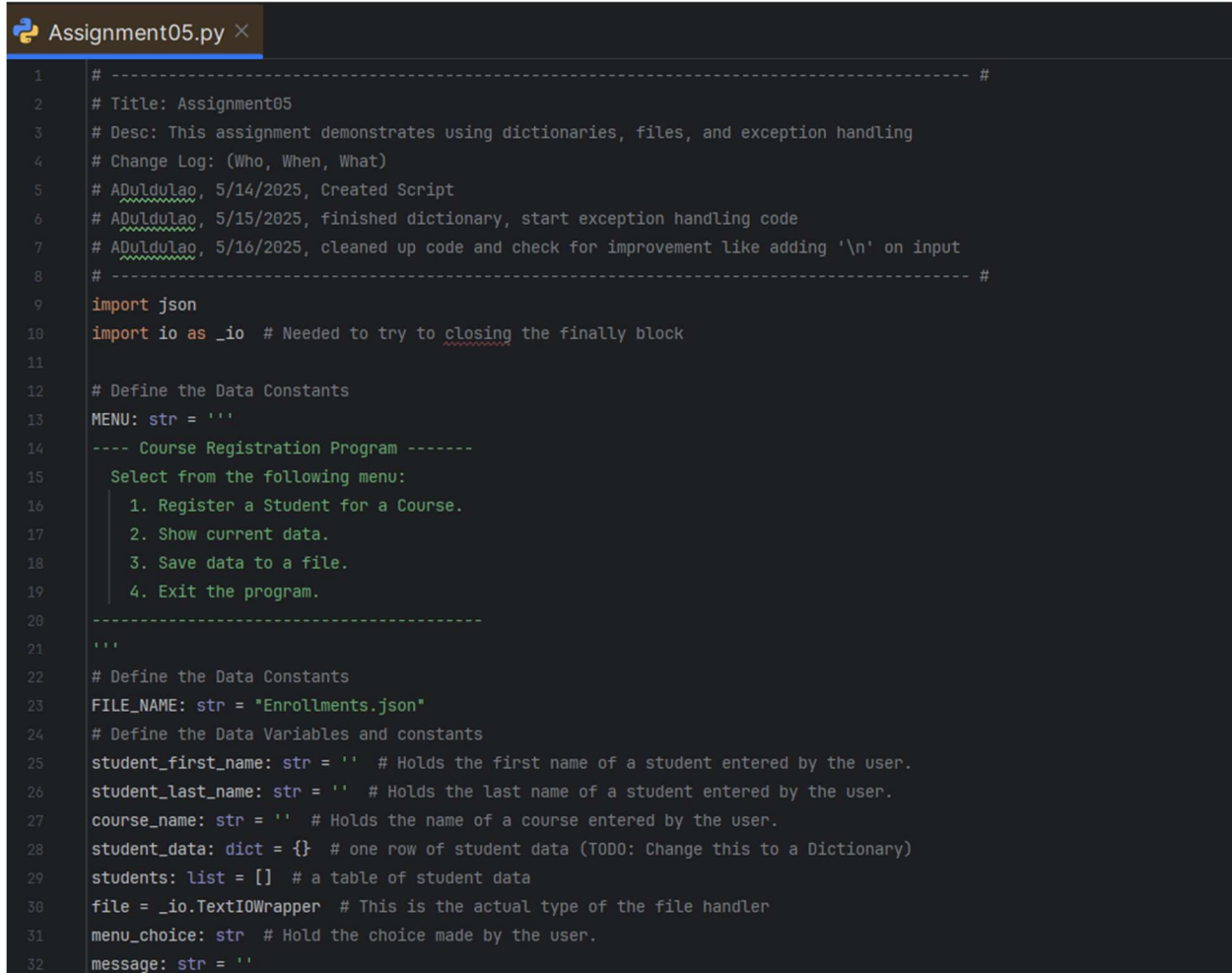
Cloud sharing files is the modern way of sharing and accessing files or folders. Like network sharing, it allows multiple users or collaborators to access and collaborate on the same file or documents from different devices and locations. Cloud storage refers to the practice of storing digital files, data, documents or any type of information on remote servers hosted by cloud service providers. These servers are typically accessible via internet. The benefits of cloud sharing are, they are cost effective, eliminate the need of local storage, supports collaboration and remote work, and automatic updates and maintenance by providers. With these advantages there are some drawbacks also. It requires internet, and potential security/privacy concerns. One of the popular cloud-based platforms is GitHub.

GitHub

GitHub is a cloud-based platform for version control, code collaboration, and software development, primarily using Git. It enables multiple programmers to work on projects simultaneously, track changes, and manage code repositories. GitHub is free for private/public repositories. To get started using GitHub, you will need an account. This process is like creating most web software account and is tied to email account. You need to follow the instruction to create an account. <https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github>. In this course we need to use our UW email. After creating an account, next step is to make a repository. Just follow the link to create one. <https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository>. A repository is the most basic element of GitHub. It's a place where you can store your code, your files, and each file's revision history. Repositories can have multiple collaborators and can be either public or private.

Module 5 Assignment Script

After learning about JSON files, dictionaries, error handling, managing code files whether network or cloud sharing file. Its time do the assignment and share the code in our newly created GitHub account.



```
1 # ----- #
2 # Title: Assignment05
3 # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4 # Change Log: (Who, When, What)
5 # ADuldulao, 5/14/2025, Created Script
6 # ADuldulao, 5/15/2025, finished dictionary, start exception handling code
7 # ADuldulao, 5/16/2025, cleaned up code and check for improvement like adding '\n' on input
8 # ----- #
9 import json
10 import io as _io # Needed to try to closing the finally block
11
12 # Define the Data Constants
13 MENU: str = ''
14 ---- Course Registration Program -----
15     Select from the following menu:
16         1. Register a Student for a Course.
17         2. Show current data.
18         3. Save data to a file.
19         4. Exit the program.
20 -----
21 '''
22 # Define the Data Constants
23 FILE_NAME: str = "Enrollments.json"
24 # Define the Data Variables and constants
25 student_first_name: str = '' # Holds the first name of a student entered by the user.
26 student_last_name: str = '' # Holds the last name of a student entered by the user.
27 course_name: str = '' # Holds the name of a course entered by the user.
28 student_data: dict = {} # one row of student data (TODO: Change this to a Dictionary)
29 students: list = [] # a table of student data
30 file = _io.TextIOWrapper # This is the actual type of the file handler
31 menu_choice: str # Hold the choice made by the user.
32 message: str = ''
```

Fig. 4: Start of the script

The entire script for this assignment was derived from the starter code that Rroot gave us and slowly make way to get a working script that will satisfy the requirements for this assignment. Started the script with the usual header, defining the constants and variables. On line 9, this code won't work without importing JSON. We need to import JSON because python's standard library doesn't natively handle JSON without the JSON module. Line 10, `_io` is also imported. "io" module provide tools for handling streams of data, such as files, in-memory buffers or other input/output operations. On line 23, since we are working with JSON file, we need our file name to be a JSON. Line 28 was defined as dictionary; it was list on the starter. Line 30, it was None, and defined it as "`_io.TextIOWrapper`". "io" operations may raise `IOError` or `ValueError` for stream issues.

```
Assignment05.py X
36 try:
37     file = open(FILE_NAME, "r")
38     students = json.load(file)
39     file.close()
40 except FileNotFoundError as e:
41     print("\nText file must be available before starting the script!\n")
42     print("!!!PLEASE PRESS 4 TO EXIT. OTHERWISE THIS TRANSACTION WILL CREATE A NEW FILE!!!\n")
43     print("-----Technical Error Message-----")
44     print(e,e.__doc__, type(e), sep='\n')
45 except Exception as e:
46     print("\nThere was a non-specific error!\n")
47     print("-----Technical Error Message-----")
48     print(e,e.__doc__, type(e), sep='\n')
49
50 finally:
51     if file.closed == False:
52         file.close()
```

Fig. 5: Start of the main script

On Fig. 5, we started the script by opening and reading a JSON file. If the file name is not yet created, the program will crash immediately. That's where the try-except come into place and still manage to continue running the script. And because of this, whatever the file name is assigned to FILE_NAME it will create. You probably don't like the file name so you have the chance to just choose to exit the program and start with the proper file name. Added the all-caps message to ask the user to exit.

```
Assignment05.py X
52     file.close()
53 # Present and Process the data
54 while (True):
55     # Present the menu of choices
56     print(MENU)
57     menu_choice = input("What would you like to do: \n")
58     # Input user data
59     if menu_choice == "1": # This will not work if it is an integer!
60         try:
61             student_first_name = input("Enter the student's first name: \n")
62             if not student_first_name.replace(" ", "").isalpha():
63                 raise ValueError("The first name should not contain numbers.")
64             student_last_name = input("Enter the student's last name: \n")
65             if not student_last_name.replace(" ", "").isalpha():
66                 raise ValueError("The last name should not contain numbers.")
67             course_name = input("Please enter the name of the course: \n")
68             student_data = {"FirstName": student_first_name,
69                             "LastName": student_last_name,
70                             "CourseName": course_name}
71             students.append(student_data)
72             print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
73         except ValueError as e:
74             print(e) # Prints the custom message
75             print("-----Technical Error Message-----")
76             print(e.__doc__)
77             print(e.__str__())
78         except Exception as e:
79             print("There was a non-specific error!\n")
80             print("---Technical Error Message---")
81             print(e, e.__doc__, type(e), sep='\n')
82             continue
```

Fig. 6: User was presented with the options

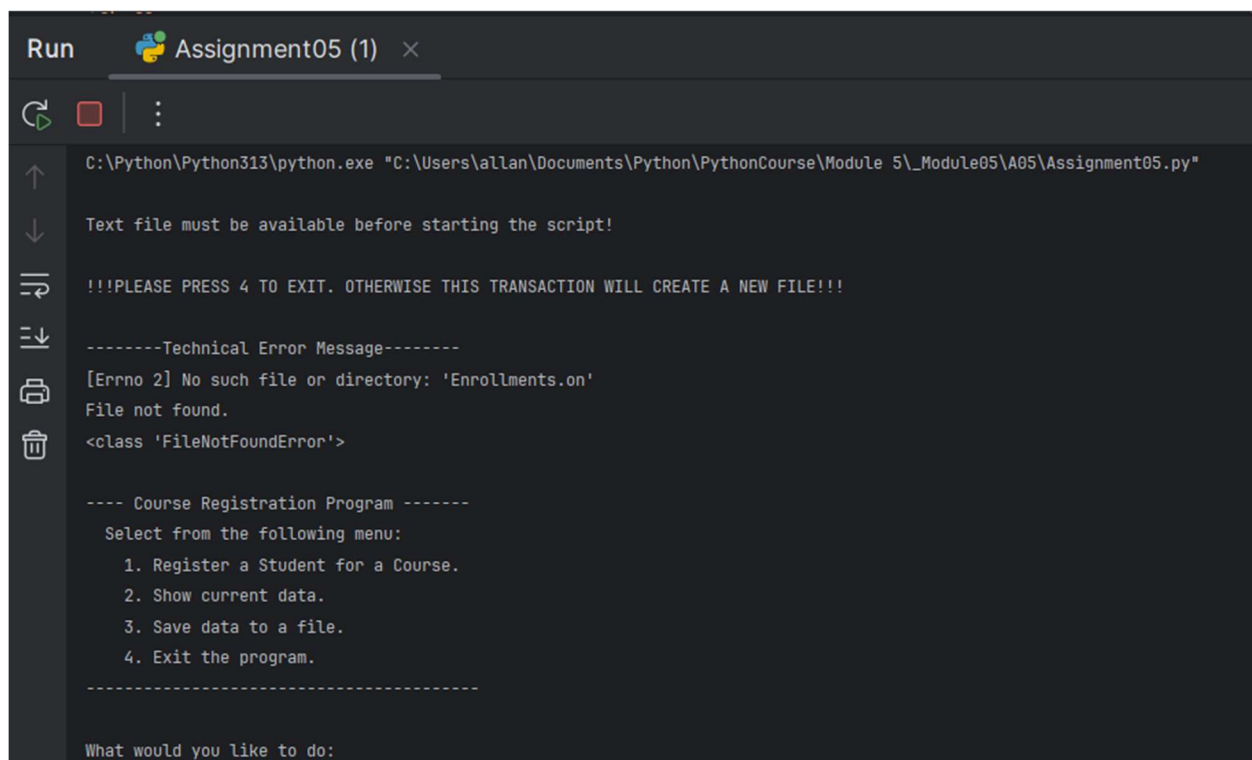
On figure 6, we started the script with “while” loop. If the user choose menu 1, the part of the script will run, where it asks the users the needed input. Once the user is done typing the inputs, the script will update the registration list by using the append command and print out a confirmation message. Since the script is looking for alphabets, the user needs to type in only alphabets, once it uses numbers or some of it are, the program will crash immediately. This where the try-except handling comes to rescue, and give the user chance to correct its mistake. I found out also while trying the code that isalpha will also fail when a space in the name is used. That’s the reason to include (“replace”“;”). Used the “raise” to manually raise that it’s a “ValueError” and print some additional details about the error.

```
Assignment05.py X
79         continue
80         # Present the current data
81     elif menu_choice == "2":
82         # Process the data to create and display a custom message
83         print(" The current registration data:")
84         print("-"*50)
85         for student in students:
86             message = "{},{},{}"
87             print(message.format(*args: student["FirstName"],student["LastName"],student["CourseName"]))
88         print("-"*50)
89         continue
90     # Save the data to a file
91     elif menu_choice == "3":
92         try:
93             file = open(FILE_NAME, "w")
94             json.dump(students, file, indent=1) # Added the indent so that my datas are easily readable.
95             file.close()
96             print("The following data was saved to file!")
97             print(f"Student {student_first_name} {student_last_name} is enrolled in {course_name}")
98             continue
99         except TypeError as e:
100             print("Please make sure that the data is a valid JSON format!\n")
101             print("---Technical Error Message---")
102             print(e, e.__doc__, type(e), sep='\n')
103         except Exception as e:
104             print("---Technical Error Message---")
105             print("Built-In Python error info: ")
106             print(e, e.__doc__, type(e), sep='\n')
107         finally:
108             if file.closed == False:
109                 file.close()
110     # Stop the loop
111     elif menu_choice == "4":
112         break # out of the loop
113     else:
114         print("Please select only 1,2, and 3. Select 4 to exit")
115 print("Program Ended")
```

Fig. 7: Final part of the code

At this part of the code (Fig. 7), as still part of the “while” loop, we have the options 2, 3 and 4. On menu choice 2, when the user chooses this option, we will print out the data of the registration in CSV format. And in menu choice 3, is where we gave the option to the user to save the data to a file by using write command (‘w’). I added indent on the JSON file in order for me to read the information on JSON file easily. We have also the try-except handling here to make sure that the file we are working is a valid JSON format. And again, print messages to the users on what error occurred. The user now once done with registration has an option 4, to end the registration. This where the “while” loop breaks with a “break”.

Test and Results



```
Run Assignment05 (1) x
C:\Python\Python313\python.exe "C:\Users\allan\Documents\Python\PythonCourse\Module 5\_Module05\A05\Assignment05.py"
Text file must be available before starting the script!
!!!PLEASE PRESS 4 TO EXIT. OTHERWISE THIS TRANSACTION WILL CREATE A NEW FILE!!!
-----Technical Error Message-----
[Errno 2] No such file or directory: 'Enrollments.on'
File not found.
<class 'FileNotFoundError'>

---- 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:
```

Fig. 8: Running the script without a valid file name

I intentionally change the file name constant and this is the error that occurs. With the try-except handling, user can still continue but they will be using different file name or they can exit the program gracefully.

On the next figure, I intentionally put a number on the first name. The error showed up but the script still running and even gave the user a chance to correct it. This message will also occur in the other input, which is for the last name. Fig. 9 is the try-except handling when the user enters a non-alphabet. Later during the test that spaces will also fail on isalpha, and cant include second name, so I decided to include (“ .replace.” “,” “”).


```
What would you like to do:
1
Enter the student's first name:
Se7en
The first name should not contain numbers.
---Technical Error Message---
Inappropriate argument value (of correct type).
The first name should not contain numbers.

---- 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:
|
```

Fig. 9: Try-except error handling on non-alphabet input

```
C:\Python\Python313\python.exe C:\Users\allan\Documents\Pythonsample\fish.py

There was a non-specific error!

---Technical Error Message---
Expecting ',' delimiter: line 7 column 1 (char 80)
Subclass of ValueError with the following additional properties:

msg: The unformatted error message
doc: The JSON document being parsed
pos: The start index of doc where parsing failed
lineno: The line corresponding to pos
colno: The column corresponding to pos

<class 'json.decoder.JSONDecodeError'>

---- 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:
```

Fig. 10: Intentionally edit my JSON file to test try-except handling

On figure 10, I edited my JSON file and removed one curly bracket to test this exception. This message come out and still managed to do a graceful exit.

```
C:\Windows\system32\cmd.e: X + v

C:\Users\allan\Documents\Python\PythonCourse\Module 5\_Module05\A05>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:
Vic
Enter the student's last name:
Vu
Please enter the name of the course:
Python 100
You have registered Vic Vu for Python 100.

---- 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
The current registration data:
-----
Andy,Bernard,Python 100
Vic,Vu,Python 100
-----

---- 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 Vic Vu is enrolled in Python 100

---- 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\allan\Documents\Python\PythonCourse\Module 5\_Module05\A05>pause
Press any key to continue . . . |
```

Fig. 11: Result in Command prompt

```
Run Assignment05 (1) x
C:\Python\Python313\python.exe "C:\Users\allan\Documents\Python\PythonCourse\Module 5\_Module05\A05\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:
Jim
Enter the student's last name:
Halpert
Please enter the name of the course:
Python 100
You have registered Jim Halpert for Python 100.

---- 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
The current registration data:
-----
Andy,Bernard,Python 100
Vic,Vu,Python 100
Jim,Halpert,Python 100
-----

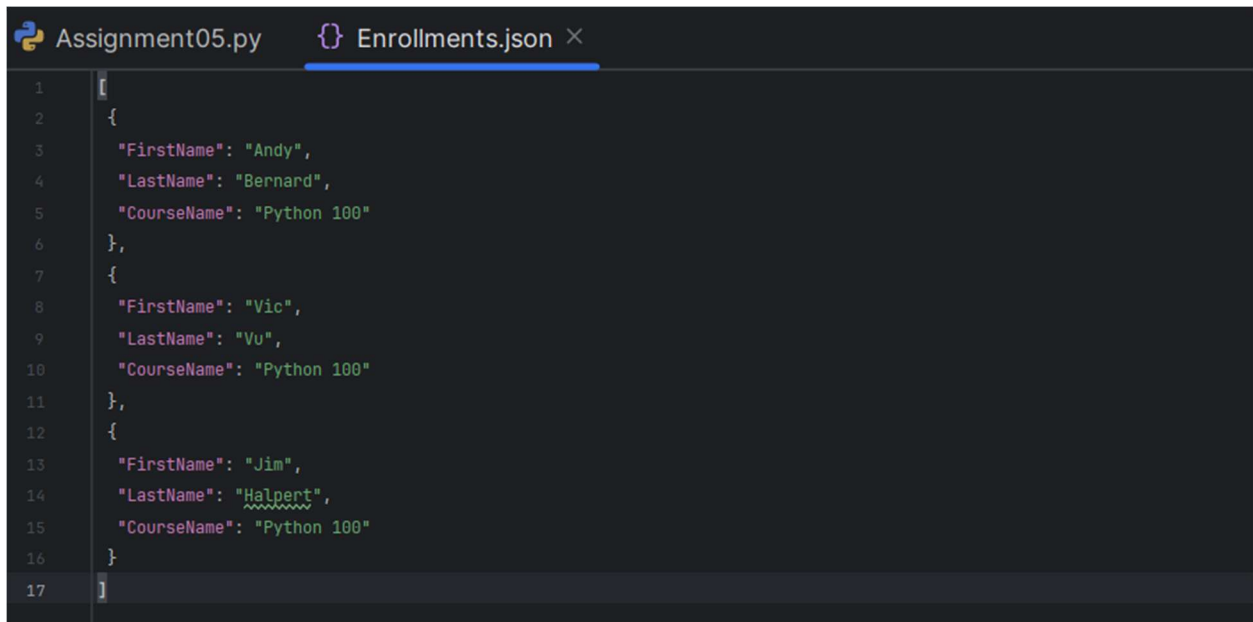
---- Course Registration Program -----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do:
3
The following data was saved to file!
Student Jim Halpert is enrolled in Python 100

---- 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
```

Fig. 11: Result in PyCharm



```
1  [
2    {
3      "FirstName": "Andy",
4      "LastName": "Bernard",
5      "CourseName": "Python 100"
6    },
7    {
8      "FirstName": "Vic",
9      "LastName": "Vu",
10     "CourseName": "Python 100"
11   },
12   {
13     "FirstName": "Jim",
14     "LastName": "Halpert",
15     "CourseName": "Python 100"
16   }
17 ]
```

Fig. 12: Data saved in JSON file. Used indent to show all of the data on the screen

Summary

In this module, we learned about data collections like list and dictionaries their similarities and differences, and saving them into a JSON file. We have learned about their characteristics, limitations, advantages and disadvantages. Getting to learn these things in python makes us prepare more to writing advanced python scripts. Knowing about error exception handling will make us more understand the script that we wanted to build and make our end user happier. Writing the assignment script gets really challenging since the script getting longer and longer. I get to read and watch the module multiple times just to come out with the assignment objective. Plus, watching and reading external sources. Doing these things make me more understand the code that I wanted to build. On this module, we also get introduced on how to manage code files. We learned network file sharing and cloud sharing, and how they work. We managed to create a GitHub account to share our script.