

Advanced Collections and Error Handling

Introduction

This assignment improves a student enrollment program by switching from CSV to JSON for data storage and using dictionaries to better organize student information. It also adds error handling to manage input errors and file issues, making the program more reliable and easier to maintain. This document explains the key changes and concepts applied.

Key Concepts

1. A **dictionary** is “a kind of mutable data structure that stores items in key-value pairs. A key is a unique identifier for an item, and a value is the data associated with that key. Dictionaries often store information such as words and definitions, but they can be used for much more. Dictionaries are also unordered, indicating the items in a dictionary are not stored in any particular order.”¹
2. “Once you have created a dictionary, you can add, remove, or update elements using the methods `dict.update()`, `dict.pop()`, and `dict.popitem()`.”²
3. Although **JSON** is not a built-in Python data type, Python provides the `json` module to work with JSON data. JSON represents data as key-value pairs and ordered lists, similar to Python dictionaries and lists. Using the `json` module, you can convert (serialize) Python objects like dictionaries and lists into JSON-formatted strings with `json.dumps()`, and convert (deserialize) JSON strings back into Python objects using `json.loads()`³.
4. “Python **exceptions** provide a mechanism for handling errors that occur during the execution of a program. Unlike syntax errors, which are detected by the parser, Python raises exceptions when an error occurs in syntactically correct code.”⁴
5. Logic flow of exceptions (figure 1):

¹ Website “Python Dictionary Guide: What It Is & How to Create One”:
<https://www.simplilearn.com/dictionary-in-python-article>

² Website “Python Dictionary Guide: What It Is & How to Create One”:
<https://www.simplilearn.com/dictionary-in-python-article>

³ OpenAI. (2025). Explanation of JSON data in Python [Large language model response]. ChatGPT.
<https://chat.openai.com/>

⁴ Website “Python Exceptions: An Introduction”:
<https://realpython.com/python-exceptions/>

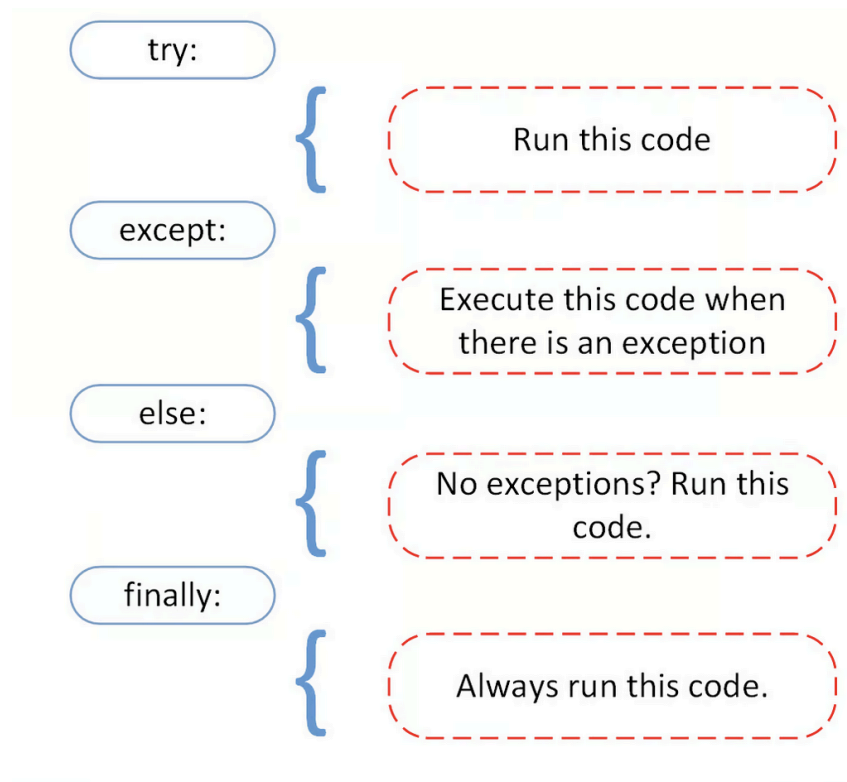


Figure 1. Exception in Python

Script Setup and Constants

To begin the modifications for Assignment 05, I first updated the script header to include the current date and my name, documenting the latest changes in the changelog. I then defined two constants for clarity and maintainability: `MENU`, which contains a multiline string representing the options displayed to the user, and `FILE_NAME`, which is set to "Enrollments.json" to indicate that we are now using a JSON file for saving and loading student data instead of the CSV format used in earlier assignments (figure 2). These changes ensure that the script is clearly marked as using JSON and maintains a consistent user interface.

```

# ----- #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030, Created Script
#   Yuying Xie,5/23/2025, Edited Code
# ----- #

import json
# 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.csv"
FILE_NAME: str = "Enrollments.json"

```

Figure 2. Script Setup and Constants

Import JSON Module

To enable the script to handle JSON data for persistent storage, I imported Python's built-in json module. This module allows for straightforward serialization and deserialization of Python data structures such as dictionaries and lists (Also see figure 2).

Variable Initialization (Change to Dict Data Format)

Next, I revised the variable initialization section to support the new data format. The student-related variables—`student_first_name`, `student_last_name`, and `course_name`—are initialized as empty strings to temporarily store user input. Instead of using a flat list for student data, I redefined `student_data` to use a dictionary structure with three keys: "FirstName", "LastName", and "CourseName" (See figure 3).

```

# 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 (TODO: Change this to a Dictionary)
students: list = [] # a table of student data
csv_data: str = '' # Holds combined CSV data. Note: Remove later since it is NOT needed with the JSON File
file = None # Holds a reference to an opened file.
menu_choice: str # Hold the choice made by the user.

```

Figure 3. Variable initialization

Edit Data Open and Saving Code to Fit for JSON Data Type

I replaced the CSV-based file operations with JSON-compatible code. For reading data, I used an `open(FILE_NAME, "r")` block in combination with `json.load(file)` to load the list of student dictionaries into the `students` list when the program starts. When saving data, I used `json.dump(students, file, indent=2)` to write the list of dictionaries to the file in a formatted way (Figure 4 Part 1 & 4). When updating `student_data`, I use key (dictionary data type) as index instead of number (Figure 4 Part 2 & 3). These modifications allow the program to persist student records using structured JSON, and eliminate the need to manually split or join comma-separated values, as was necessary with CSV.

```
# Extract the data from the file
try:
    file=open(FILE_NAME, 'r')
    students=json.load(file)
```

Figure 4 Part 1: Modify code to fit for json data type

```
else:
    student_data = {"FirstName":student_first_name,"LastName":student_last_name,"CourseName":course_name}
    students.append(student_data)
    print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    continue
```

Figure 4 Part 2: Modify code to fit for json data type

```
# 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 4 Part 3: Modify code to fit for json data type

```
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file, indent=2)
```

Figure 4 Part 4: Modify code to fit for json data type

Add Try-Exception Structure

To make the program more robust and prevent it from crashing due to unexpected file or input errors, I implemented try-except-finally blocks in both the file reading and writing sections. When reading the JSON file, the try block attempts to open and load the file; if the file does not exist or contains malformed JSON, the except block prints a user-friendly error message. Similarly, when writing to the file, I wrapped the write operation in a try-except-finally block to catch potential I/O errors. Additionally, I used try-except to validate that names entered by the user contain only alphabetic characters, displaying an error

message if the input is invalid. This ensures data integrity and improves the user experience by providing clear feedback when things go wrong (See [Assignment04.py](#) file).

Test the Result

After completing all the modifications, I tested the script by running it through Pycharm (Figure 5) and Mac OS terminal (Appendix 1). All test cases ran successfully and showed up in the json file (Figure 6), confirming that the script performs reliably with the new JSON data format and enhanced error handling.

```
---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones 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: 1
Enter the student's first name: Yuying
Enter the student's last name: Xie
Please enter the name of the course: Python 101
You have registered Yuying Xie for Python 101.
```

Figure 5. Test result in Pycharm (Part 1)

```
---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Yuying Xie is enrolled in Python 101
-----

---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Yuying Xie is enrolled in Python 101
```

Figure 5. Test result in Pycharm (Part 2)

```
---- 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: 34232534356
First name must be alphanumeric.
--Technical Error Message--
Inappropriate argument value (of correct type).
First name must be alphanumeric.

---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Yuying Xie is enrolled in Python 101
-----
```

Figure 5. Test result in Pycharm (Part 3)

```
---- 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: Elli
Enter the student's last name: Baker
Please enter the name of the course: Python 101
You have registered Elli Baker for Python 101.

---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Yuying Xie is enrolled in Python 101
Student Elli Baker is enrolled in Python 101
-----
```

Figure 5. Test result in Pycharm (Part 4)


```
---- 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.
-----

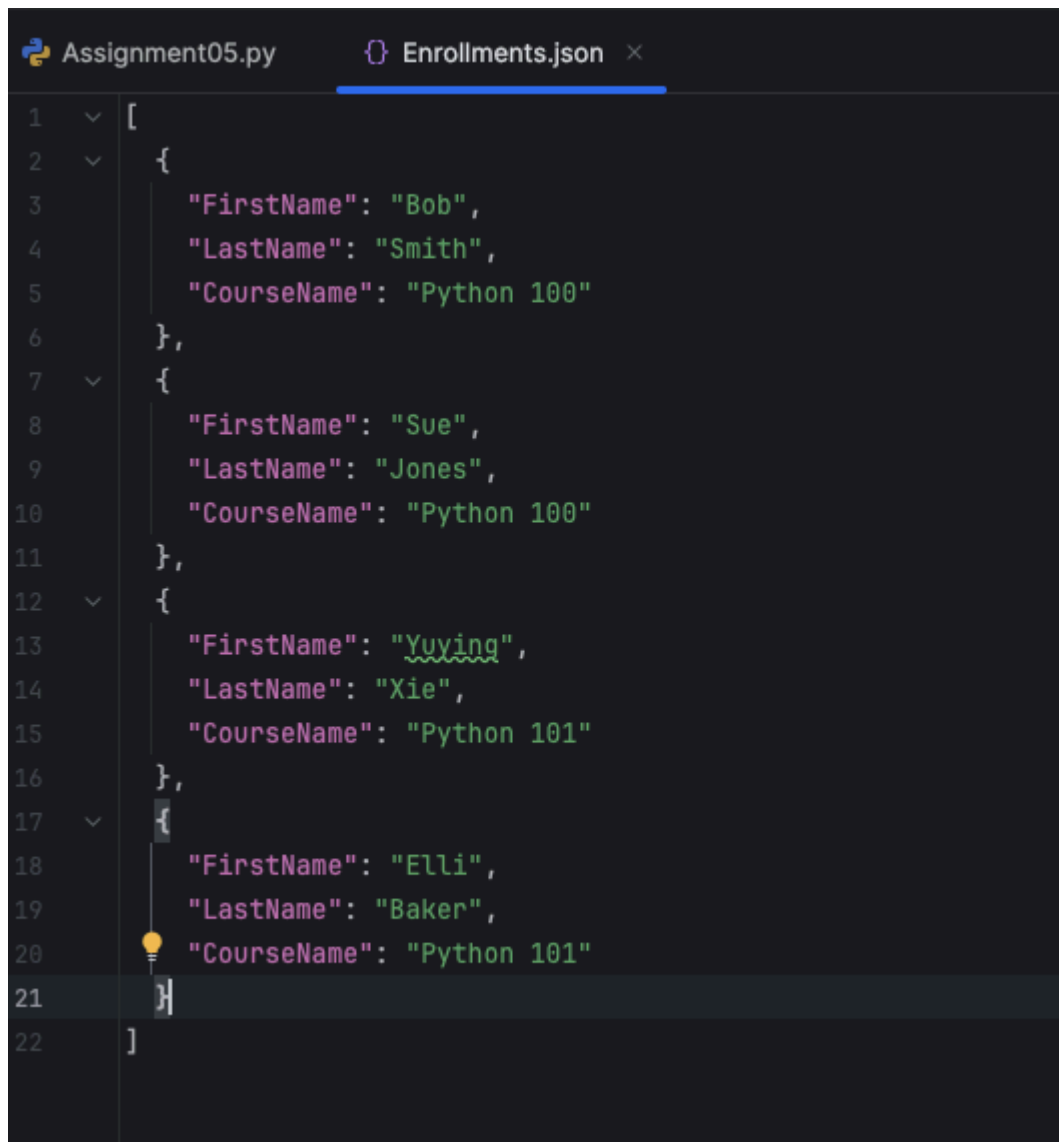
What would you like to do: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Yuying Xie is enrolled in Python 101
Student Elli Baker is enrolled in Python 101

---- 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
```

Figure 5. Test result in Pycharm (Part 5)



```
1  [
2    {
3      "FirstName": "Bob",
4      "LastName": "Smith",
5      "CourseName": "Python 100"
6    },
7    {
8      "FirstName": "Sue",
9      "LastName": "Jones",
10     "CourseName": "Python 100"
11   },
12   {
13     "FirstName": "Yuying",
14     "LastName": "Xie",
15     "CourseName": "Python 101"
16   },
17   {
18     "FirstName": "Elli",
19     "LastName": "Baker",
20     "CourseName": "Python 101"
21   }
22 ]
```

Figure 6. Program result in json file

Summary

In conclusion, I updated the program to use JSON and dictionaries for structured data storage and added try-except blocks for robust error handling. The changes improved data management and program stability. Testing confirmed the program runs smoothly across different environments, reinforcing my skills in Python data handling and exception management.

Appendix 1: test script in terminal

(base) xy@MacBook-Pro-683 Python Foundation101 % python3 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: Yuying Xie

First name must be alphanumeric.

--Technical Error Message--

Inappropriate argument value (of correct type).

First name must be alphanumeric.

---- 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:

Please only choose option 1, 2, 3, or 4

---- 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: Yuying

Enter the student's last name: Xie

Please enter the name of the course: Python 101

You have registered Yuying Xie for Python 101.

---- 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 Bob Smith is enrolled in Python 100

Student Sue Jones is enrolled in Python 100

Student Yuying Xie is enrolled in Python 101

Student Elli Baker is enrolled in Python 101
Student Yuying Xie is enrolled in Python 101

---- 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: Tian

Enter the student's last name: Wang

Please enter the name of the course: Python 101

You have registered Tian Wang for Python 101.

---- 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 Bob Smith is enrolled in Python 100

Student Sue Jones is enrolled in Python 100

Student Yuying Xie is enrolled in Python 101

Student Elli Baker is enrolled in Python 101

Student Yuying Xie is enrolled in Python 101

Student Tian Wang is enrolled in Python 101

---- 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 Bob Smith is enrolled in Python 100

Student Sue Jones is enrolled in Python 100

Student Yuying Xie is enrolled in Python 101

Student Elli Baker is enrolled in Python 101
Student Yuying Xie is enrolled in Python 101
Student Tian Wang is enrolled in Python 101

---- 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

(base) xy@MacBook-Pro-683 Python Foundation101 %