

James Francisco Jabal

13 Aug 2025

IT FDN 110 A Su 25

Assignment 05

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

Advanced Collections and Error Handling

Introduction

This assignment demonstrates using dictionaries, JavaScript Object Notation (JSON) files, and exception handling by creating a Course Registration Program. The program starts with a menu for multiple functions. A starter enrollment JSON file was included with previous enrollments to be read by modifying a starter program template based on the previous assignment 04. The program would allow the user to add students to the enrollment list, display the enrollment list, and save the enrollment list back to the enrollment JSON file. We were also introduced to GitHub for saving and sharing our files.

Program

This week, I continued my plan to have running working code as soon as possible before the office hours.

Header

I started with the header from the starter file. Eventually, I made several revisions to the code after the two office hours with the instructor assistants Chris Voit (8/8/2025) and Becky Peltz (8/11/2025).

```
# ----- #
# Title: Assignment05
# Desc: Assignment demonstrates using dictionaries, JSON, & exception handling
# Change Log: (Who, When, What)
#   James Jabal,8/7/2025,Created Script. Use dictionaries & JSON module.
#   James Jabal,8/8/2025,Add exception handling
#           ,Change FileNotFoundError -> PermissionError on writing
#           ,Show Data: Remove message string. Use CSV string.
#   James Jabal,8/9/2025,Change PermissionError -> TypeError on writing JSON.
#   James Jabal,8/10/2025,Initialize menu_choice. Update comments for A05
#   James Jabal,8/11/2025,Loop invalid student name input. Use TextIOWrapper.
#           Moved name input reset outside of exception handler
#   James Jabal,8/12/2025,Use constants for repeated error message strings.
#           Remove check for file as None. TextIOWrapper in use.
#           Add Data: Reset names after (in)valid input for multi
#           registrations, and moved outside Exception Handler.
#           Save Data: Change JSON indent 2 -> 4 for readability.
# ----- #
```

Imports

Although list of dictionaries could have been worked with manually, this assignment is an exercise in learning to use JSON files with the `json` module.

Also, in Demo 03, the use of the `io` module was demonstrated for use with JSON files. During the office hours, it was found that this fixed some errors caused by using the `None` data type for the file object.

```
import json # For json file
import io as _io # For file object
```

Constants

I defined the menu, JSON filename, and added error message strings that were repeated in the program.

```
# Define Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    h1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----
'''

FILE_NAME: str = "Enrollments.json"
NON_SPEC_ERROR_MESSAGE: str = "There was a non-specific error!\n"
TECH_ERROR_MESSAGE: str = "-- Technical Error Message --\n"
```

Variables

I used the standard variables from the previous assignment, a dictionary data type (rows) instead of list, and added the proper data type for working with the JSON file, which is essentially a text file wrapper. Also initialized the `menu_choice` variable.

```
# Define Data Variables
student_first_name: str = '' # Holds first name of a student entered by user.
student_last_name: str = '' # Holds last name of a student entered by user.
course_name: str = '' # Holds the name of a course entered by the user.
student_data: dict = {} # One row of student data (Now a Dictionary)
students: list = [] # A table of student data
file = _io.TextIOWrapper # Holds a reference to an opened file.
menu_choice: str = '' # Hold the choice made by the user.
```

Read JSON

I start the program by reading the JSON file with exception handling using `try-except-finally` blocks. The entire file is loaded into the `students` variable as a list of dictionary rows.

The exception handling displays custom error messages for specific & catch-all errors. The `finally` block was simplified using suggestions from PyCharm. This makes sure the file is closed after an error.

```
# When the program starts, read JSON file data into a list of dictionary rows
try: # Use Exception Handling when reading file
    file = open(FILE_NAME, "r")
    students = json.load(file) # Extract the data from the JSON file
    file.close()
# Exception Handling
except FileNotFoundError as e:
    print("Text file must exist before running this script!\n")
    print(TECH_ERROR_MESSAGE)
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print(NON_SPEC_ERROR_MESSAGE)
    print(TECH_ERROR_MESSAGE)
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if not file.closed: file.close() # Ensure file is closed after error
```

Main Menu Loop

The Main Menu loop shows the menu and prompts the user to input a menu choice.

```
# Present and Process the data (Main Menu Loop)
while True:

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

Add Data (Input Loop)

For registering a student for a course, I added an input loop and error handling with `try-except` blocks. This also checks for valid non-empty data entry.

```
# Input user data (Add Data)
if menu_choice == "1": # This will not work if it is an integer!

    while student_first_name == "" or student_last_name == "": # Input loop
        try: # Use Error Handling
```

Data Entry (Name Loops)

Originally, when I used the exception handlers, the program would loop back to the main menu. So, I added internal `while` loops to make sure a valid name was entered before returning to the main menu.

After prompting the user for a first name, the program checks for any non-alphabetic characters. If so, the invalid entry is reset to an empty string, then raises an `ValueError` exception. This skips the rest of the code, and straight to the exception handler (which goes back to input loop for the user to try again).

A similar loop was used for the last name. Also, in this case, the user would not need to re-enter the first name if it was already valid.

The course name did not require any data validation and left as-is from the previous assignment.

```
# First Name
while student_first_name == "":
    student_first_name = input("Enter the student's first"
                               " name: ")
    if not student_first_name.isalpha():
        student_first_name = "" # Reset invalid name input
        raise ValueError("The first name should not contain"
                           " numbers.")

# Last Name
while student_last_name == "":
    student_last_name = input("Enter the student's last"
                              " name: ")
    if not student_last_name.isalpha():
        student_last_name = "" # Reset invalid name input
        raise ValueError("The last name should not contain"
                           " numbers.")

# Course Name
course_name = input("Please enter the name of the course: ")
```

Add to student data

The entered data is added to a dictionary (row). This row is appended to the `student_data` list of dictionaries (loaded JSON 2D table data in memory). A message displays the student course registration.

```
# Add to a dictionary (One row in a JSON file)
student_data = {"FirstName": student_first_name,
                "LastName": student_last_name,
                "CourseName": course_name}

# Add row to list of dictionaries (JSON 2D table)
students.append(student_data)

# Extra print to screen (uses dictionary row data)
print(f"You have registered {student_data['FirstName']}",
      f"{student_data['LastName']} for",
      f"{student_data['CourseName']}.")
```

Exception Handling

The exception handler displays custom error messages for specific & catch-all errors. After the exceptions, the program goes back to the input loop instead of the Main Menu loop. This allows the user to re-enter names for registration.

After the office hours, it was found that the code did not allow for multiple registrations. Debugging showed the previous entered names were still saved in memory and considered valid. So, it was necessary to reset not just the invalid entries, but also to reset the valid input variables after they are added to memory (student_data).

With valid input, the program loops back to the Main Menu.

```
# Exception Handling
except ValueError as e:
    print(e) # Prints the custom message
    print(TECH_ERROR_MESSAGE)
    print(e.__doc__)
    print(e.__str__())
except Exception as e:
    print(NON_SPEC_ERROR_MESSAGE)
    print(TECH_ERROR_MESSAGE)
    print(e, e.__doc__, type(e), sep='\n')
# After Exceptions, Loop back to Input Loop

# After valid name input, reset name to allow multiple registrations
student_first_name = ""
student_last_name = ""

continue # Go back to Main Menu Loop
```

Show Data

For displaying the data, the program reads the students variable, and iterates through each dictionary (row). It prints out each value of the dictionary row, for each key listed. I made sure to change the separator to create a comma-separated values list.

After showing the data, the program loops back to the Main Menu.

```
# Present the current data (Show Data)
elif menu_choice == "2":
    # Process data to create & show a CSV string for each row in students
    print("-"*50)
    for student in students:
        print(student["FirstName"], student["LastName"],
              student["CourseName"], sep = ",")
    print("-"*50)

    continue
```

Save Data

For saving data, the program opens a file to dump (write) the file to a JSON with error handling `try-except` blocks. I also added the `indent` argument to improve human-readability of the JSON file.

After saving, the data saved is printed out as a formatted message. The program reads the `students` variable, and iterates through each item (row) to display a formatted message with values from each row.

```
# Save the data to a file (Save Data)
elif menu_choice == "3":
    try: # Use Exception Handling
        file = open(FILE_NAME, "w")
        json.dump(students, file, indent = 4) # Write to JSON file
        file.close()

        # Display what was written to the file using the students variable
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student['FirstName']} {student['LastName']}",
                  f"is enrolled in {student['CourseName']}")
```

Exception Handling

The exception handler displays custom error messages for specific & catch-all errors. The `finally` block makes sure the file is closed after an error. After the exceptions, this goes back to the Main Menu.

```
# Exception Handling
except TypeError as e: # Data written might not in JSON format
    print("Please check that the data is a valid JSON format\n")
    print(TECH_ERROR_MESSAGE)
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print(NON_SPEC_ERROR_MESSAGE)
    print(TECH_ERROR_MESSAGE)
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if not file.closed: file.close() # Ensure file closed after error

continue # Go back to Main Menu Loop
```

Exit Program

I used the standard `break` function from the starter file to break out of the Main Menu loop, which goes to the end of the program.

```
# Stop the loop (Exit Program)
elif menu_choice == "4":
    break # Break out of the loop
```

Show Valid Options

Any other input displays valid menu options, and loops back to the main menu.

```
else: # Go back to Main Menu Loop for a valid choice
    print("Please only choose option 1, 2, 3, or 4")
```

End of Program

This is now outside of the main menu loop, which is the end of the program.

```
# End of Program (Outside the Main Menu Loop)
print("Program Ended")
```

Testing

I tested the program in both PyCharm & Terminal, while checking PyCharm & Notepad for the data in the saved file “Enrollments.json”. PyCharm & Notepad notified me when the file was changed. Testing made sure the program meets the acceptance criteria:

- Takes user input for student registration
- Displays the data
- Saves the data (Check file in PyCharm & Notepad)
- Allows multiple registrations
- Allows displaying multiple registrations
- Allows saving multiple registrations (Check file in PyCharm & Notepad)
- Run on both PyCharm & Terminal

PyCharm

The program was tested in PyCharm using the starter “Enrollments.json” file. One student was added.

Add Data

```
---- 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: James
Enter the student's last name: Jabal
Please enter the name of the course: Python 100
You have registered James Jabal for Python 100.
```

Show Data

```
---- 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
-----
Bob,Smith,Python 100
Sue,Jones,Python 100
James,Jabal,Python 100
-----
```

Save Data

```
---- 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 James Jabal is enrolled in Python 100
```

Exit Program

```
---- 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
```

JSON in PyCharm

The contents of the “Enrollments.json” file saved was checked after testing in PyCharm.

```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "James",
    "LastName": "Jabal",
    "CourseName": "Python 100"
  }
]
```

Terminal

The program was later tested in the Terminal (PowerShell) using an updated “Enrollments.json” file from the previous PyCharm testing. One more student was added to the list to test for multiple registrations.

Add Data

```
---- 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: Jack
Enter the student's last name: Worthington
Please enter the name of the course: Python 100
You have registered Jack Worthington for Python 100.
```

Show Data

```
---- 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
-----
Bob,Smith,Python 100
Sue,Jones,Python 100
James,Jabal,Python 100
Jack,Worthington,Python 100
-----
```


Save Data

```
---- 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 James Jabal is enrolled in Python 100
Student Jack Worthington is enrolled in Python 100
```

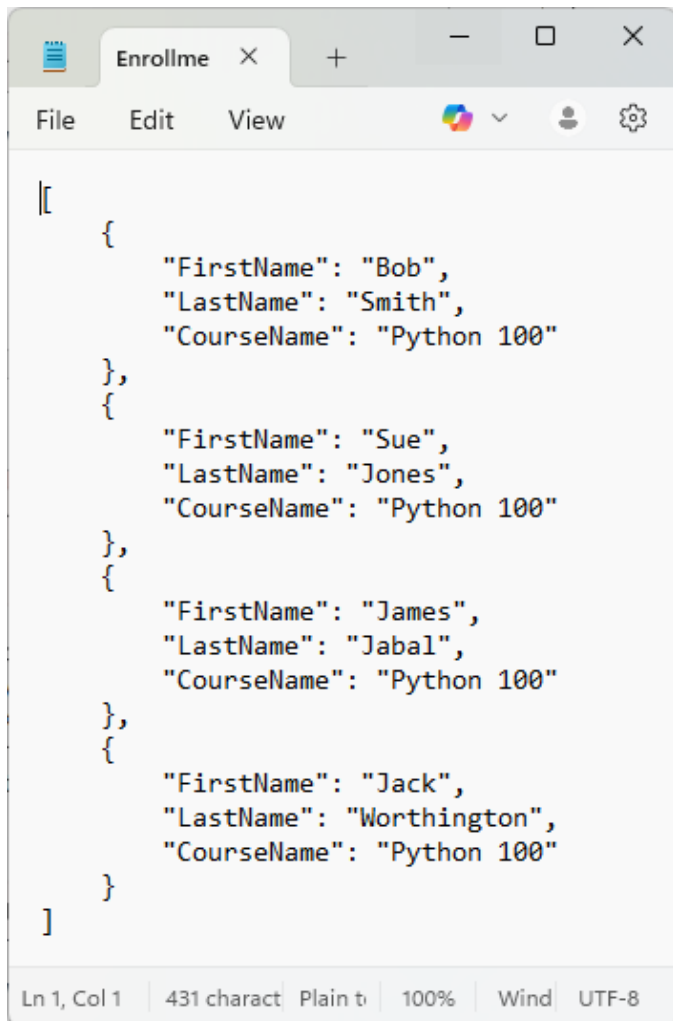
Exit Program

```
---- 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
```

JSON in Notepad

The contents of “Enrollments.json” was checked in Notepad after testing in the Terminal. (Figure 1)

A screenshot of a Notepad window titled 'Enrollme'. The window shows a JSON array with four objects, each representing a student's enrollment. The JSON is formatted with indentation. The status bar at the bottom indicates 'Ln 1, Col 1', '431 charact', 'Plain t', '100%', 'Wind', and 'UTF-8'.

```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "James",
    "LastName": "Jabal",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Jack",
    "LastName": "Worthington",
    "CourseName": "Python 100"
  }
]
```

Figure 1. Final “Enrollments.json” file after testing in both PyCharm & Terminal

Error Handling

The error handling functions were also tested: reading a non-existing JSON file, entering a numeric student name, and writing to a read-only JSON file. The results are as shown below.

FileNotFoundError

```
Text file must exist before running this script!
-- Technical Error Message --

[Errno 2] No such file or directory:
'Enrollments.jsonZZZ'
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: 4
Program Ended
```

Catch-All Exception (Read-Only File)

```
What would you like to do: 3
There was a non-specific error!

-- Technical Error Message --

[Errno 13] Permission denied: 'Enrollments.json'
Not enough permissions.
<class 'PermissionError'>

---- 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
```

Raise ValueError (Numeric Names)

```
Enter the student's first name: James 123
The first name should not contain numbers.
-- Technical Error Message --

Inappropriate argument value (of correct type).
The first name should not contain numbers.
Enter the student's first name: James
Enter the student's last name: Jabal 123
The last name should not contain numbers.
-- Technical Error Message --

Inappropriate argument value (of correct type).
The last name should not contain numbers.
Enter the student's last name: Jabal
Please enter the name of the course: Python 100
You have registered James Jabal for Python 100.
```

Summary

This assignment helped me practice reading/writing JSONfiles, and working with list & dictionary data structures. It reinforces the concept of converting between data formats (JSON → Lists & Dictionaries → JSON) and working with data between file storage and memory (File → Memory → File). It also showed how to use the PyCharm code debugger to see and work with error handling (exception handling) and how to adjust & control the program flow accordingly. I also created my first public repository on GitHub and uploaded my program, JSON file, and this document.