Deborah Briggs

May 18, 2025

Foundations of Programming: Python

Assignment 05

https://github.com/jdbriggs3/IntroToProg-Python-Mod05

Collections of data

Introduction

Assignment 05 builds upon the last assignment. The program on the surface works similarly. The changes are primarily internal – not seen by the user but assists with usability. In the last assignment we created tables of data with lists of lists, this time we create the same table with lists of dictionaries. In the last we retrieved and sent data to CSV files, this time we utilize JSON files. Finally, we add code to consider errors and then open a GitHub account uploading our assignment to the account.

Dictionaries

A dictionary is similar to a list. It is a row of data. A dictionary is differentiated primarily by its key-value structure making it easier to read and understand. In this assignment the Keys are FirstName, LastName, and CourseName. Something we might see as headers in an excel table. The values are the corresponding data, in this case names, input by the user.

JSON Files

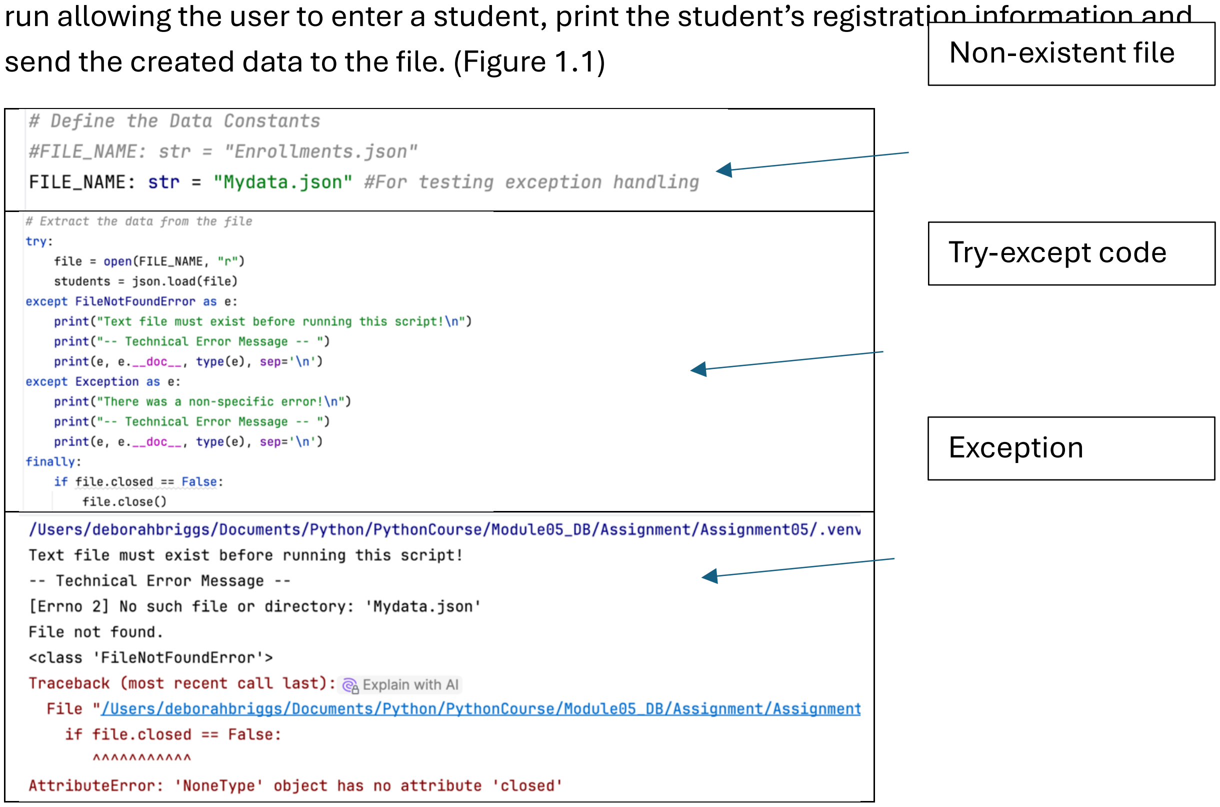
In the assignment we utilized JSON files rather than CSV files. JSON stands for JavaScript Object Notation. Like Python dictionaries it utilizes the key-value pairs. It too is widely used and is known for its ease of reading and understanding.

Exception Handling

The assignment outlined three locations to add exception handling. First when the file is read into the list of dictionary rows, second when the user inputs the student's first/last name, and finally when the dictionary rows are written back into the file. In this assignment we implemented try-except constructs to catch errors. I utilized the code given in the Module 05 Notes pages 25-26.

Read File

To test the exception code when reading the JSON file in, I changed the FILE-NAME to a non-existent file. When running the program, the exception occurred, and the script was not read. (Figure 1) Additionally, I ran the program with the correct file name, but the file was empty. In this case the error was found, properly handled, and the script continued to run allowing the user to enter a student, print the student's registration information and send the created data to the file. (Figure 1.1)

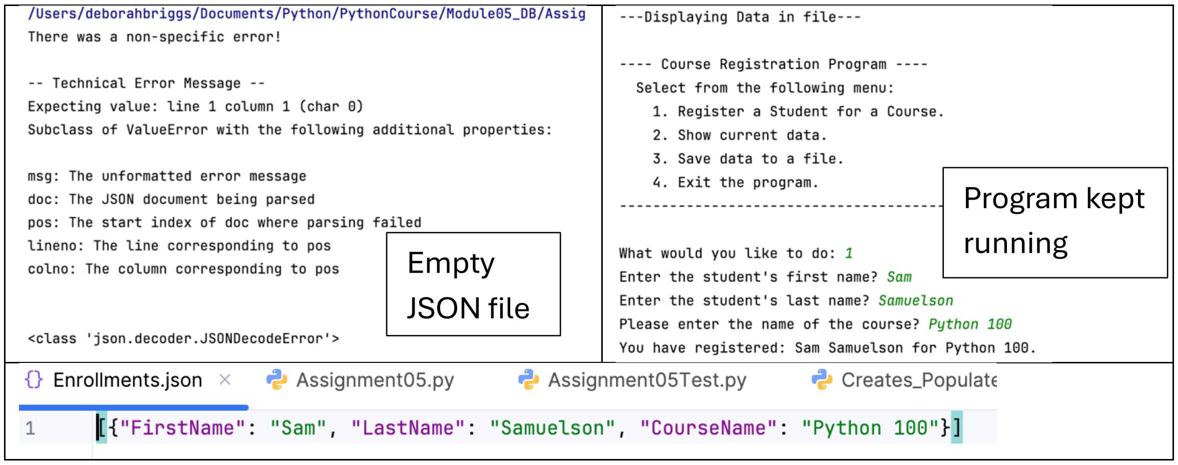


```
# Define the Data Constants
#FILE_NAME: str = "Enrollments.json"
FILE_NAME: str = "Mydata.json" #For testing exception handling
```

```
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
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:
    if file.closed == False:
        file.close()
```

```
/Users/deborahbriggs/Documents/Python/PythonCourse/Module05_DB/Assignment/Assignment05/.venv
Text file must exist before running this script!
-- Technical Error Message --
[Errno 2] No such file or directory: 'Mydata.json'
File not found.
<class 'FileNotFoundError'>
Traceback (most recent call last):
  File "/Users/deborahbriggs/Documents/Python/PythonCourse/Module05_DB/Assignment/Assignment
    if file.closed == False:
       ^^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'closed'
```

Figure 1: Exception handling – JSON file non-existent



```
/Users/deborahbriggs/Documents/Python/PythonCourse/Module05_DB/Assig
There was a non-specific error!

-- Technical Error Message --
Expecting value: line 1 column 1 (char 0)
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'>
```

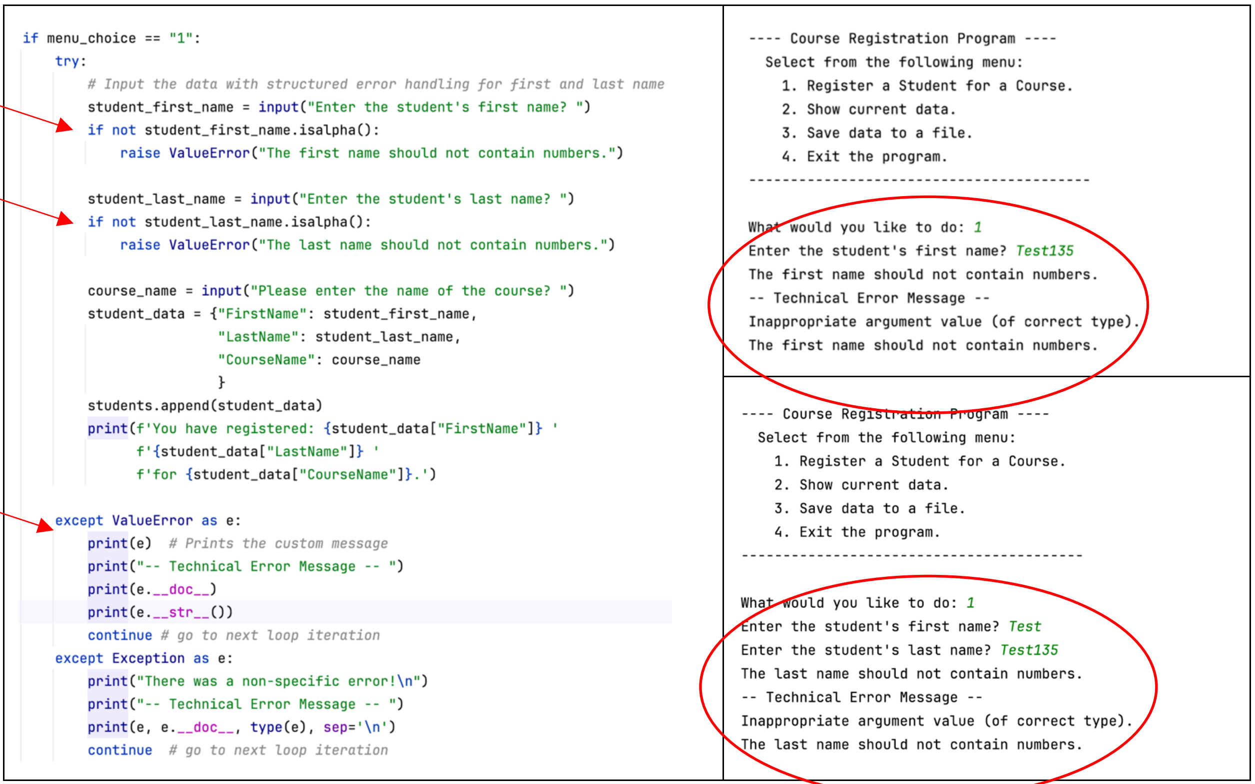
```
---Displaying Data in file---

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

```
[{"FirstName": "Sam", "LastName": "Samuelson", "CourseName": "Python 100"}]
```

Figure 1.1: Exception handling– JSON file empty

User Input

To test if the try-except code worked to catch input errors I ran the code using numbers for both first and last name. (Figure 2) The exception was properly executed in both cases.



```
if menu_choice == "1":
    try:
        # Input the data with structured error handling for first and last name
        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)
        print(f'You have registered: {student_data["FirstName"]} '
              f'{student_data["LastName"]} '
              f'for {student_data["CourseName"]}.')

    except ValueError as e:
        print(e)  # Prints the custom message
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
        continue # go to next loop iteration
    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  # go to next loop iteration
```

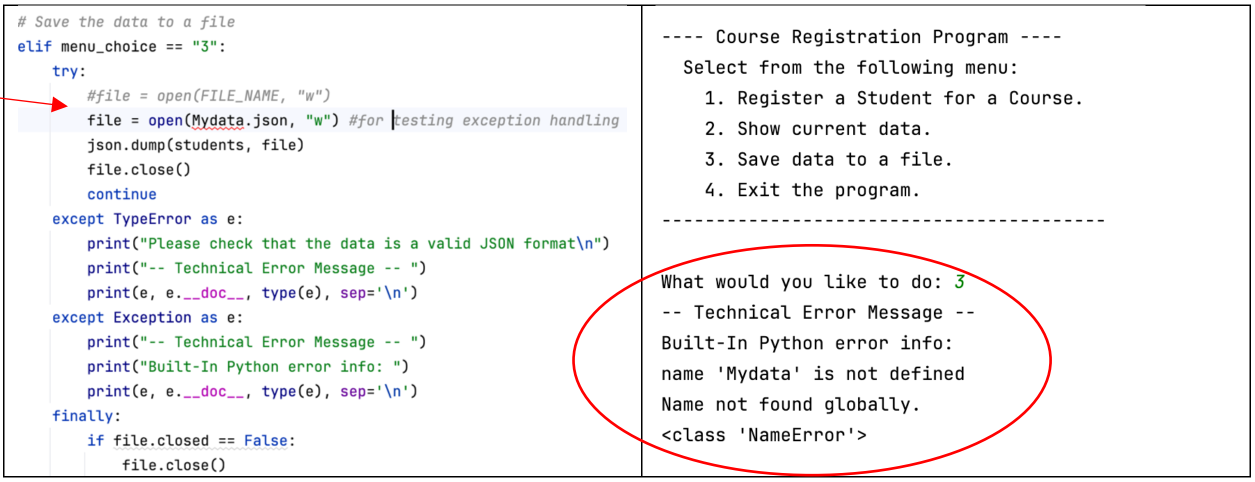
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------

What would you like to do: 1
Enter the student's first name? Test135
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: 1
Enter the student's first name? Test
Enter the student's last name? Test135
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.
```

Figure 2: Exception handling when user inputs first/last name

Write File

To test if the try-except code worked while writing the code to the external file I again put in a non-existent file. (Figure3)



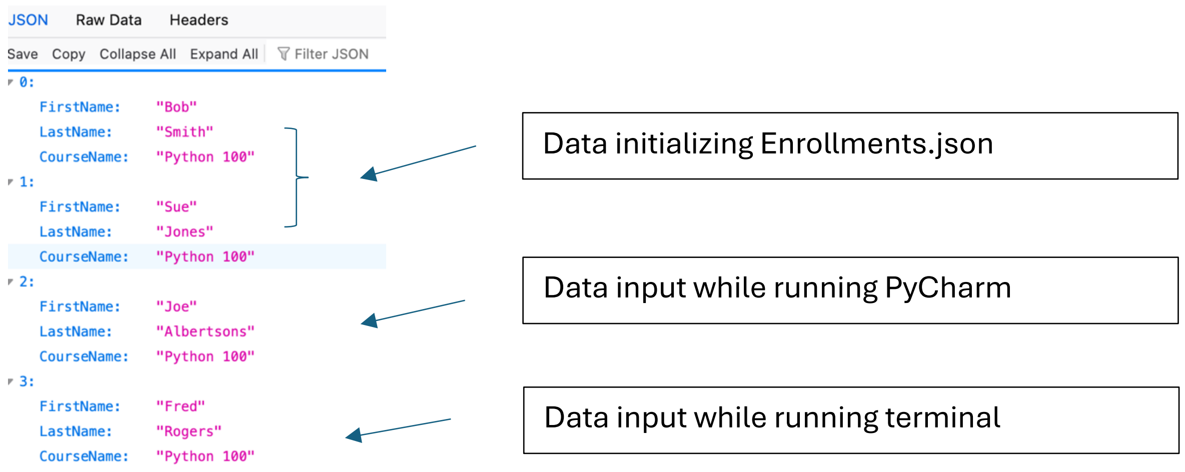
```
# Save the data to a file
elif menu_choice == "3":
    try:
        #file = open(FILE_NAME, "w")
        file = open(Mydata.json, "w") #for testing exception handling
        json.dump(students, file)
        file.close()
        continue
    except TypeError as e:
        print("Please check that 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()
```

```
---- 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
-- Technical Error Message --
Built-In Python error info:
name 'Mydata' is not defined
Name not found globally.
<class 'NameError'>
```

Figure 3: Exception handling when dictionary written to a nonexistent file

Terminal

I ran and tested the program within the terminal with success. The data was read from and written to the Enrollments.json file. (Figure 5)


JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
Data initializing Enrollments.json
Data input while running PyCharm
Data input while running terminal

```
0:
    FirstName:    "Bob"
    LastName:     "Smith"
    CourseName:   "Python 100"
1:
    FirstName:    "Sue"
    LastName:     "Jones"
    CourseName:   "Python 100"
2:
    FirstName:    "Joe"
    LastName:     "Albertsons"
    CourseName:   "Python 100"
3:
    FirstName:    "Fred"
    LastName:     "Rogers"
    CourseName:   "Python 100"
```

Figure 5: Terminal

Summary

Assignment 05 built upon the last assignment. The program on the surface works similarly. The changes were primarily internal – not seen by the user. We implemented dictionaries and utilized JSON files. We also added code to handle errors. Additionally, we opened a GitHub account and uploaded our assignment to our GitHub account.