Mo Farhat

Feb 8, 2026

IT FDN 110 GP Wi 26: Foundations Of Python Programming

Assignment 5 - Advanced Collections and Error Handling

Introduction

This document details the steps taken to complete Assignment 05, "Advanced Collections and Error Handling," for the Introduction to Programming with Python course. The assignment required reviewing module resources on advanced collections (Dictionaries and JSON), file handling, structured error handling, and Git/GitHub. The primary goal of the module assignment was to create an updated menu-driven Python script which builds on previous work by demonstrating the use of constants, variables, user input (input()), print statements (print()), a primary while loop for the menu, and conditional logic. The program now manages student registration data using a two-dimensional list of dictionaries for in-memory storage and achieves data persistence by using the json module to read from and save data to a JSON file named "Enrollments.json," all while incorporating structured error handling (try-except) to manage common exceptions.

Assignment Completion Steps

The completion of this assignment followed the nine tasks outlined in the "Mod05-Assignment.docx" document.

1. Read Module Notes and Watch Module Videos

The initial step involved reading the Module 05 Notes document and watching all associated module videos to establish a foundational understanding of data structures and file handling in Python. Key concepts included the difference between Lists and Dictionaries (Indexes vs. Keys), reading and writing data to a file using Dictionaries, JavaScript Object Notation (JSON) and Python's json module, structured Error Handling using Try-Except and the use of GitHub for code storage and sharing.

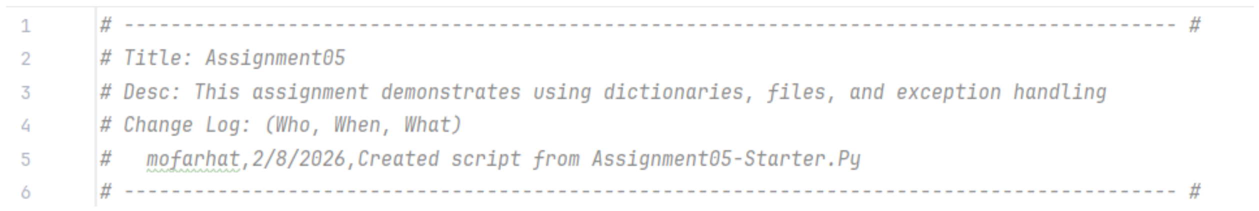
2. Watch Assignment Videos and Read About Module Topics

In addition to the core module materials, external articles were reviewed and supplemental videos were watched to cover specific practical topics for this assignment.

3. Create the Program (Assignment05.py)

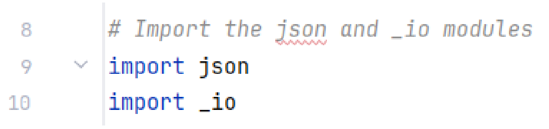
I created a Python program named **Assignment05.py** as specified in the assignment document. The program script was developed and tested using the PyCharm IDE and console, and includes the following features to meet the acceptance criteria:

- **File Name and Script Header:** The file was named "Assignment05.py," and the script header was updated with my name and the current date.



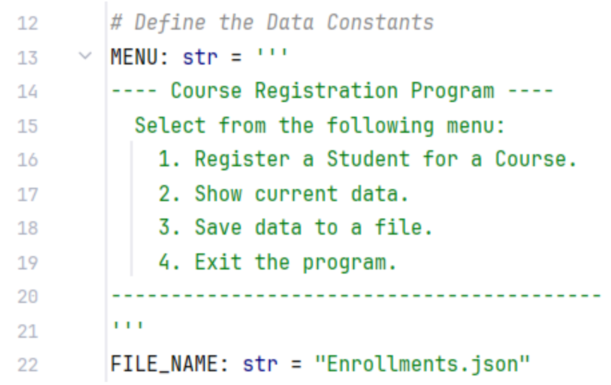
```python
# ----------------------------------------------------------------------------------------- #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   mofarhat,2/8/2026,Created script from Assignment05-Starter.Py
# ----------------------------------------------------------------------------------------- #
```

- **Imports:** The `json` module and the `_io` module were imported.



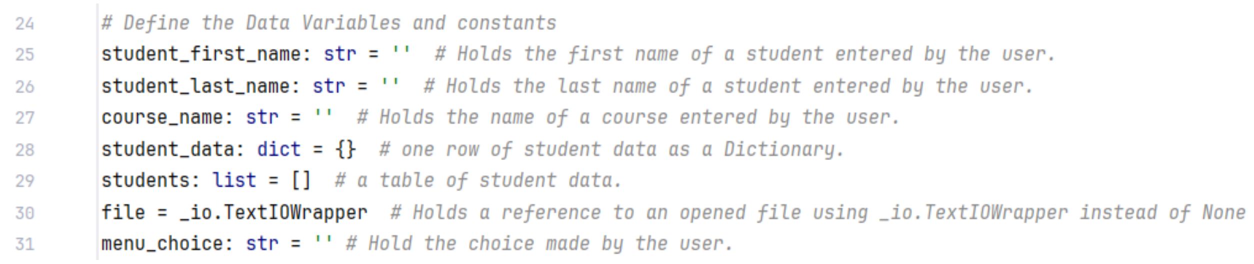
```python
# Import the json and _io modules
import json
import _io
```

- **Define Constants:** The required constants were defined and remain unchanged throughout the program:
  - `MENU: str` set to the four-choice menu string for registration.
  - `FILE_NAME: str` set to "**Enrollments.json**".



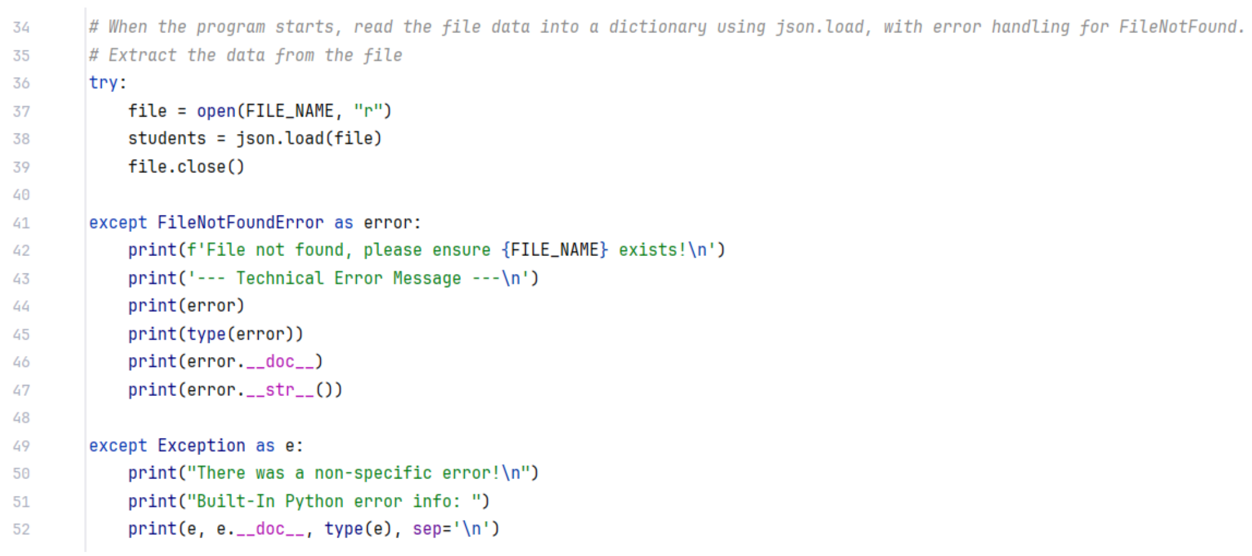
```python
# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''
FILE_NAME: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

- **Define Data Variables:** The required variables were initialized, including:
  - `student_data: dict` was initialized to an **empty dictionary** (`{}`).
  - `students: list` was initialized to an **empty list** (`[]`) to serve as the two-dimensional list of dictionary rows.



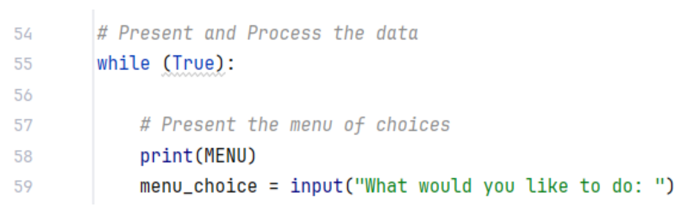
```python
# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data as a Dictionary.
students: list = []  # a table of student data.
file = _io.TextIOWrapper  # Holds a reference to an opened file using _io.TextIOWrapper instead of None
menu_choice: str = '' # Hold the choice made by the user.
```

- **Processing (Program Startup - Reading Existing Data):** When the program starts, the contents of the "**Enrollments.json**" file are automatically read into the `students` list using the `json.load()` function.



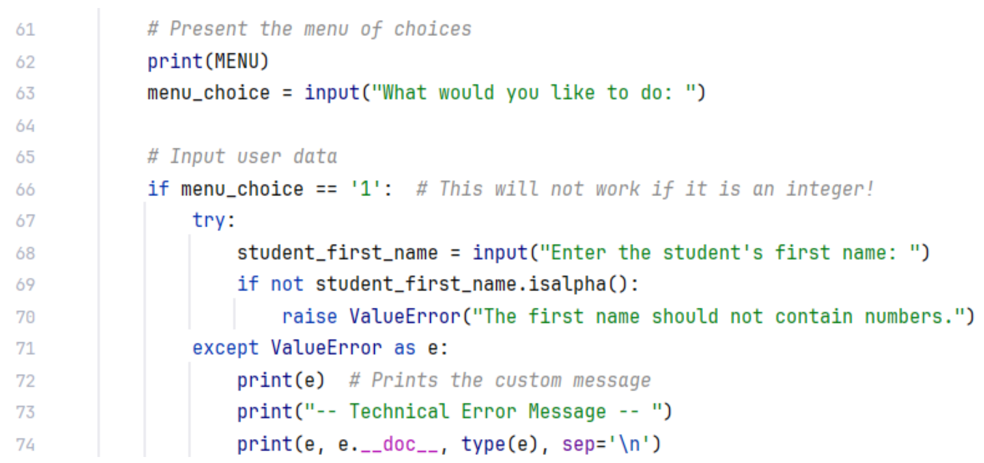
```python
# When the program starts, read the file data into a dictionary using json.load, with error handling for FileNotFound.
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()

except FileNotFoundError as error:
    print(f'File not found, please ensure {FILE_NAME} exists!\n')
    print('--- Technical Error Message ---\n')
    print(error)
    print(type(error))
    print(error.__doc__)
    print(error.__str__())

except Exception as e:
    print("There was a non-specific error!\n")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
```

- **Program Flow:** A primary `while (True):` loop controls the menu flow.

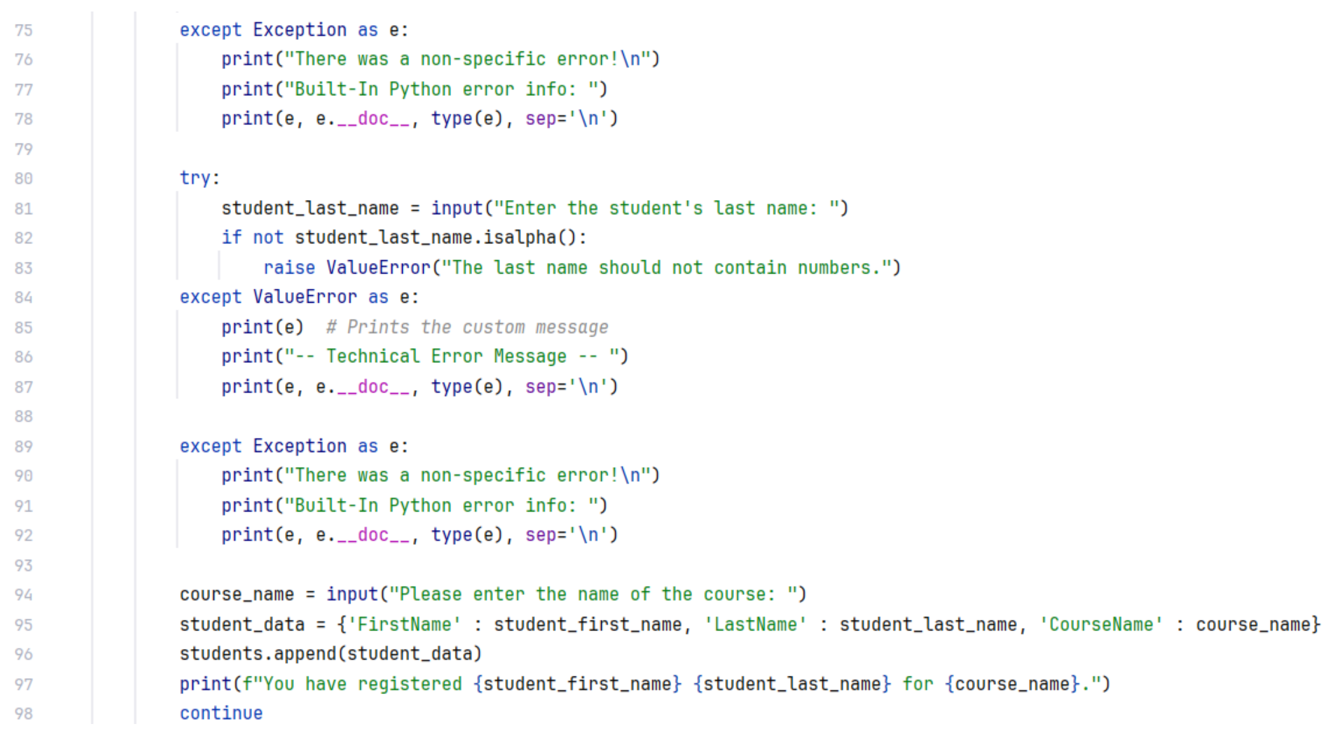


```python
# Present and Process the data
while (True):

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

  - **Menu Choice 1 (Register a Student):** Prompts the user for the student's first name, last name, and course name. This data is collected, formatted into a new `student_data` dictionary, and then added to the `students` list.

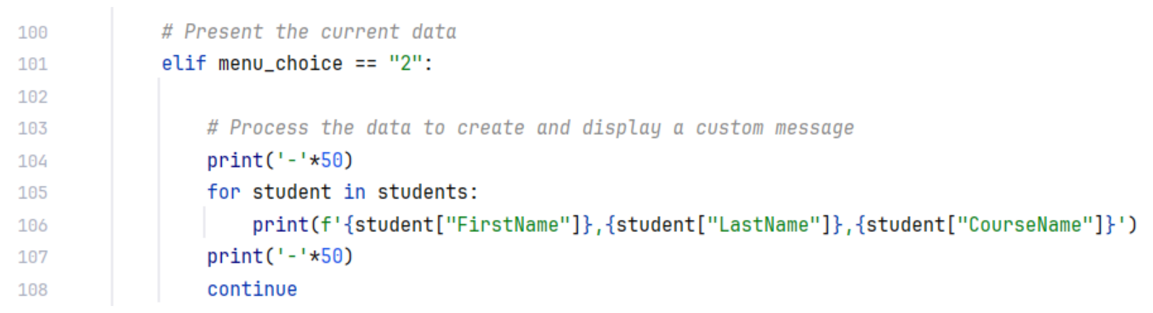


```python
    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")

    # Input user data
    if menu_choice == '1':  # This will not work if it is an integer!
        try:
            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.")
        except ValueError as e:
            print(e)  # Prints the custom message
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
```



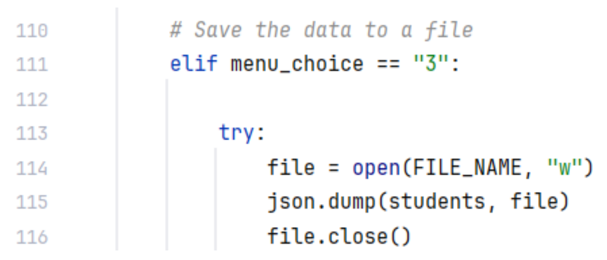
```
except Exception as e:
    print("There was a non-specific error!\n")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')

try:
    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.")
except ValueError as e:
    print(e)  # Prints the custom message
    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("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')

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_first_name} {student_last_name} for {course_name}.")
continue
```

- **Menu Choice 2 (Show current data):** Iterates through the `students` list and uses the `print()` function to display a comma-separated string for each student record.

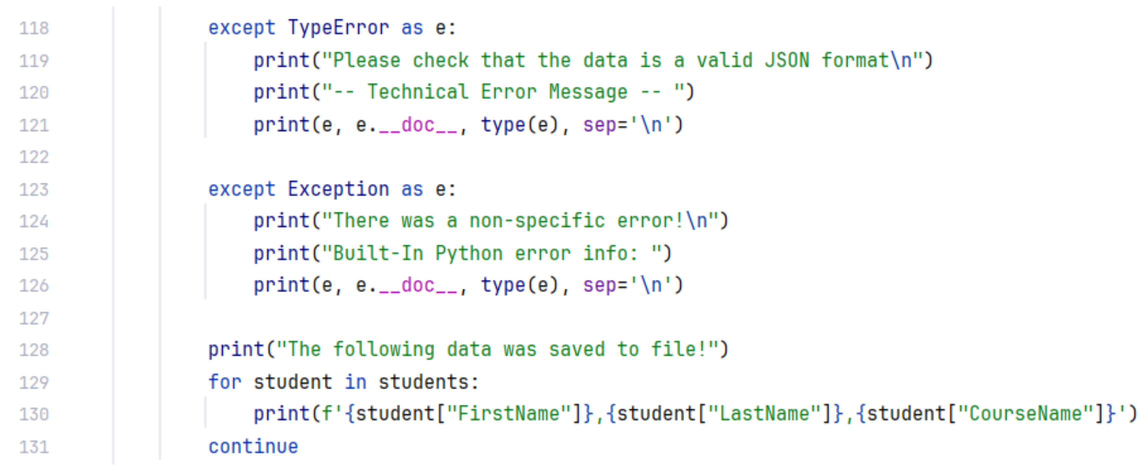


```
# Present the current data
elif menu_choice == "2":

    # Process the data to create and display a custom message
    print('-'*50)
    for student in students:
        print(f'{student["FirstName"]},{student["LastName"]},{student["CourseName"]}')
    print('-'*50)
    continue
```

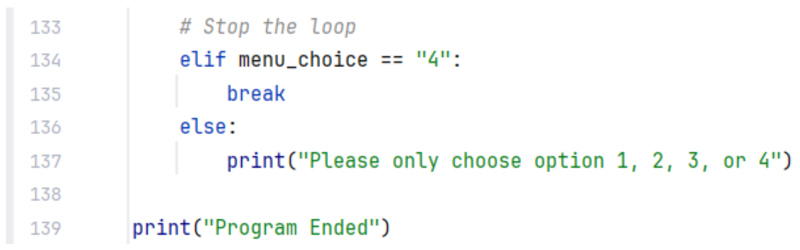
- **Menu Choice 3 (Save data to a file):** Opens "**Enrollments.json**" in write mode, uses the `json.dump()` function to write the entire `students` list to the file, and then closes the file. It displays the saved data.



```
# Save the data to a file
elif menu_choice == "3":

    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
```


```python
        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("There was a non-specific error!\n")
            print("Built-In Python error info: ")
            print(e, e.__doc__, type(e), sep='\n')

        print("The following data was saved to file!")
        for student in students:
            print(f'{student["FirstName"]},{student["LastName"]},{student["CourseName"]}')
        continue
```

- ○ **Menu Choice 4 (Exit the program):** Uses the `break` statement to end the program.


```python
    # Stop the loop
    elif menu_choice == "4":
        break
    else:
        print("Please only choose option 1, 2, 3, or 4")

print("Program Ended")
```

- **Error Handling:** Structured error handling with `try-except-finally` blocks was implemented for four scenarios:
    - When the file is read into the list of dictionary rows (`json.load()`).
        - Checked for a `FileNotFoundError` passing through other `Exception` errors.
    - When the user enters a first name (`input()`).
        - Checked for a `ValueError` passing through other `Exception` errors.
    - When the user enters a last name (`input()`).
        - Checked for a `ValueError` passing through other `Exception` errors.
    - When the dictionary rows are written to the file (`json.dump()`).
        - Checked for a `TypeError` passing through other `Exception` errors.

The program was tested in both PyCharm and directly from the terminal and was confirmed to run correctly, meeting all acceptance and test criteria

4. Document Knowledge

This document was created to describe the steps and process of completing Assignment 05, saved as Assignment05_MoFarhat.pdf.

5. Source Control and Submission

The program files and documentation were posted to the required public GitHub repository and submitted to the course management system:

- **GitHub Repository Creation:** A public repository named "IntroToProg-Python-Mod05" was created on GitHub, accessible on https://github.com/theFarhat/IntroToProg-Python-Mod05
- **File Posting:** The three required files—the knowledge document (Assignment05_MoFarhat.pdf), the Python script (Assignment05.py), and the data file (Enrollments.json with sample data)—were uploaded to the repository and the changes were committed.
- **Link Posting:** The URL for the new GitHub site was copied into the knowledge document and will be posted to the "Assignment 05 Documents for Review!" discussion board on Canvas for peer review.
- **Final Submission:** The three files were placed into a folder named A05, compressed into a ".zip" file, and uploaded to the class assignment page on Canvas.

Summary

Assignment 05 transitioned the student registration application to use dictionaries for structured data storage within a list. The assignment also switched from a comma-delimited file to a JSON file using Python's built-in json module. It introduced structured error handling for issues like file-not-found errors or incorrect user input. Finally, it set up a foundation for the use of GitHub for source control and code sharing.