Pattama Srianomai

Aug 12, 2025

Foundations of Programming: Python

Assignment 05

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

Steps to complete Assignment 05:
Data Processing with Dictionaries, Error Handling, and JSON Files

Introduction

In this paper, I will describe the steps I took to create a Python Script using dictionaries, error handling and JSON files. Guided by the notes and videos provided in Module 05. I will walk through each step beginning with opening and reviewing the starter file Assignment05-Starter.py.

Step by Step Process

Step 1. Opening and reviewing the starter file Assignment05-Starter.py

Opening a file in PyCharm

1. Use the Project Pane on the left side
2. Double-click the Assignment 05-Starter.py
3. It will appear in the editor window

Renaming a file in PyCharm

1. Right-click the Assignment 05-Starter.py on the Project Pane
2. Select Rename
3. Rename the file to Assignment05.py
4. Click Refactor

Step 2. Updating a script header

Update the script header with my name, current date, and change history. *My script header is shown in Figure 1 below.*

```
# ------------------------------------------------------------------------------------------ 3
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   PSrianomai,08/10/2025,Created Script
# ------------------------------------------------------------------------------------------
```

Figure 1: Script Header

Step 3. Defining Data Constants and Variables

Assignment 05 - Task 4 outlined the required constants and variables for this exercise:

Constants:

- The constant **MENU: str** is set to the value:

  ---- 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

- The constant **FILE_NAME: str** is set to the value "Enrollments.json"
- The constant values do not change throughout the program.

Variables:

- **student_first_name: str** is set to empty string.
- **student_last_name: str** is set to empty string.
- **course_name: str** is set to empty string.
- **file** is set to None.
- **menu_choice: str** is set to empty string.
- **student_data: dict** is set to an empty dictionary (This is changed from a list using in assignment04)
- **students: list:** list is set to and empty list

Figure 2 presents the complete set of constants and variables used in my script.

```
#------------------------------------------#
import json #import code from Python's JSON module into my script
# 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:
# set to the value of Enrollments.json file
FILE_NAME: str = "Enrollments.json"

# 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 = {}  # Holds one row of student data (TODO: Change this to a Dictionary)
students: list = [] # hold a list of dictionaries
file = None  # Holds a reference to an opened file.
menu_choice: str = '' # Hold the choice made by the user.
```

Figure 2: Constants and Variables definition

I start my script with *import json* to access Python's built-in JSON module.

Step 4. Presenting and processing the data

This assignment is very similar to Assignment 04, but it utilizes a dictionary for data processing, allowing the program to collect, store, and display multiple records.

Assignment 05 - Task 4 outlined the requirement for each option in the Input/Output, Processing and Error Handling sections:

Error Handling

- The program provides structured error handling when the file is read into the list of dictionary rows.
- The program provides structured error handling when the user enters a first name.
- The program provides structured error handling when the user enters a last name.
- The program provides structured error handling when the dictionary rows are written to the file.

Processing

- When the program starts, the contents of the "Enrollments.json" are automatically read into the **students** two-dimensional list of dictionary rows using the json.load() function. (**Tip:** Make sure to put some starting data into the file or you will get an error!)

Figure 3 shows the script I used to automatically load data into the students two-dimensional list of dictionary rows using json.load() function with error handling.

```
# When the program starts, the contents of the "Enrollments.json" are
# automatically read into the students two-dimensional list of
# dictionary rows using the json.load() function
# read from the json file
try: # use error handling
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
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()
```

Figure 3: Automatically read into the file with error handling when the program is opened

I referred to Mod05-Lab03 and applied its approach to my script by using a try-except block, which allowed me to customize the error message when the file does not exist. I used the open() and json.load() functions to access and read the contents of the JSON file. To ensure the file is properly closed even if an error occurs, I added a finally block.

Input / Output:

- On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.
- Data collected for menu choice 1 is added to a dictionary named student_data. Next, student_data is added to the **students** two-dimensional list of dictionaries rows.

Figure 4 shows my script, which incorporates a while loop, try-catch, print(), input(), a dictionary, and the implementation of menu choice 1.

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

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")
    print() # Adding extra space to make it look nicer.

    # Input user data:
    # On menu choice 1, the program prompts the user to enter the student's
    # first name and last name, followed by the course name, using the input()
    # function and stores the inputs in the respective variables.
    # using try-except to check when users input first names
    if menu_choice == "1":  # This will not work if it is an integer!
        try: # use error handling
            print("-"*50)
            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 mot contain numbers.")
            course_name = input("Please enter the name of the course: ")
            # Data collected for menu choice 1 is added to a dictionary
            # named student_data. Next, student_data is added to the students
            # two-dimensional list of dictionaries rows.
            student_data = {"FirstName": student_first_name, "LastName":
                student_last_name, "CourseName": course_name}
            students.append(student_data) # add row to list of dictionaries 2D
```

```
            print(f"You have registered {student_data["FirstName"]}",
                  f"{student_data["LastName"]} for",
                  f"{student_data["CourseName"]}.")
            print("-" * 50)
        except ValueError as e:
            print(e)  # Prints the custom message
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        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
```

Figure 4: while loop, try-catch, print(), input(), and dictionary, and the implementation of menu choice 1.

I used custom error raising to validate user input for a student's first and last name, ensuring that only alphabetic values are accepted by using the isalpha() function. The student_data dictionary stores three pieces of information: the student's first name, last name, and course name. This dictionary is then added to the students list using the students.append() function.

- On menu choice 2, the presents a string by formatting the collected data using the print() function.
- On menu choice 2, the program uses the print() function to show a string of comma-separated values for each row collected in the **students** variable.

Figure 5 shows my script, which incorporates a for loop statement

```
    # Present the current data:
    # On menu choice 2, the presents a string by formatting the collected
    # data using the print() function.
    elif menu_choice == "2":
        print("-"*50)
        for student in students:
            print(f"{student["FirstName"]},{student["LastName"]},"
                  f"{student["CourseName"]}")
        print("-"*50)
```

Figure 5: Incorporate a for loop to loop through each student record

I applied a for loop statement—for student in students—to iterate through each row (student record). I then formatted the output to display the values as comma-separated strings.

- On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. It writes the contents of the **students** variable to the file using the json.dump() function. Next, the file is closed using the close() method. Finally, the program displays what was written to the file using the **students** variable.

Figure 6 displays my script, using the try-except, open(), jason.dump(), and close () functions.

```
# On menu choice 3, the program opens a file named "Enrollments.json"
# in write mode using the open() function. It writes the contents of
# the students variable to the file using the json.dump() function.
# Next, the file is closed using the close() method. Finally, the
# program displays what was written to the file using the students variable
elif menu_choice == "3":
    try: # Error Handling
        file = open(FILE_NAME,"w") #open the file in write mode
        json.dump(students, file, indent = 2) #write file using json dump
        # print statement inside the try block to only print the
        # statement if writing succeeds
        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"]}")
    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: # ensures the file is closed properly before continue
        if file.closed == False:
            file.close()
    continue
```

Figure 6: using the try-except, open(), jason.dump(), and close () functions..

I used the json.dump() function to write the table list (students), which contains dictionaries, to the JSON file. I used its indent option to add indentation and new lines, making them easier to read.

- On menu choice 4, the program ends.

I used the break command to end the program when users select Option 4. *Figure 7. Ending the program upon user selection of Option 4, with a fallback for invalid input.*

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

print("Program Ended")
```

Figure 7: Ending the program and default case for invalid input.

Step 5. Testing the Script

I tested the script in PyCharm and the Command Prompt. *The results are shown in Figure 8.* After running the script, I verified that the JSON file was saved successfully on my computer *(see Figure 9).*

PyCharm:

```
C:\Python\Python3.13\python.exe C:\Users\taeya\Documents\Python\PythonCourse\_

---- 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: 1
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 ----
```

```
---- 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: Pat
Enter the student's last name: Srianomai
Please enter the name of the course: Python 100
You have registered Pat Srianomai 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

--------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Srianomai,Python 100
Nancy,Wang,Python 100
Taeya,Serm,Python 100
Pat,Srianomai,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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Pat Srianomai is enrolled in Python 100
Student Nancy Wang is enrolled in Python 100
Student Taeya Serm is enrolled in Python 100
Student Pat Srianomai 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

Process finished with exit code 0
```

Command Prompt

```
C:\Users\taeya\Documents\Python\PythonCourse\A05>python 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: Pat
Enter the student's last name: 1
The last name should mot contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should mot 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: Peter
Enter the student's last name: Vu
Please enter the name of the course: Python 100
You have registered Peter 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

--------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Srianomai,Python 100
Nancy,Wang,Python 100
Taeya,Serm,Python 100
Pat,Srianomai,Python 100
Peter,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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Pat Srianomai is enrolled in Python 100
Student Nancy Wang is enrolled in Python 100
Student Taeya Serm is enrolled in Python 100
Student Pat Srianomai is enrolled in Python 100
Student Peter 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
```

Figure 8. Script results

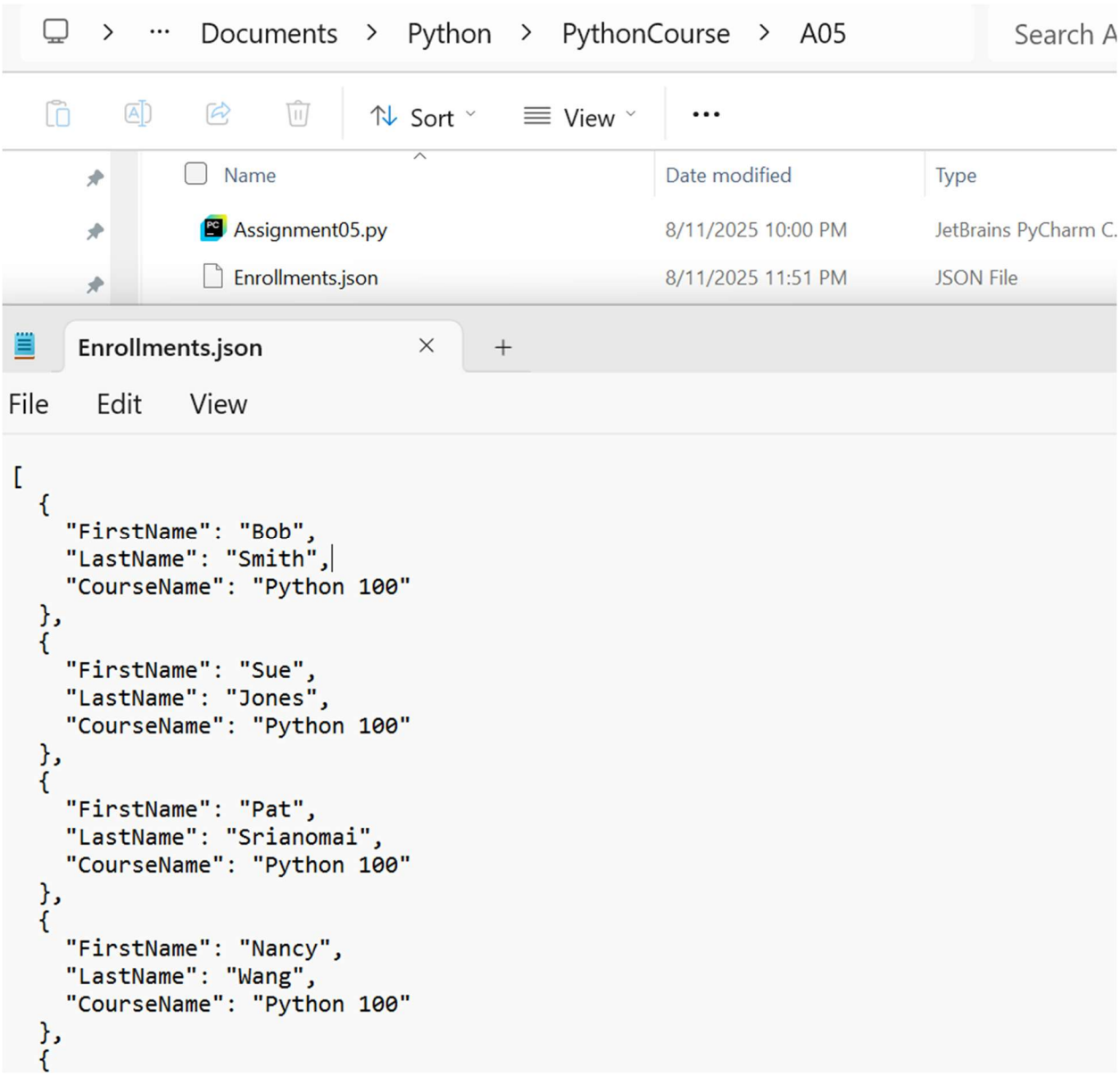

Figure 9. Saved JSON file

Summary

This assignment focused on building a Python script that processes student data using dictionaries, error handling, and JSON file operations. I applied notes and labs from Module 5 to guide my script development. The script includes input validation with isalpha(), error handling using try-except blocks. I read from json file using json.load() and wrote data to json file using json.dump() function.