

Evan Peper

November 13th, 2024

IT FDN 110 A

Assignment 05

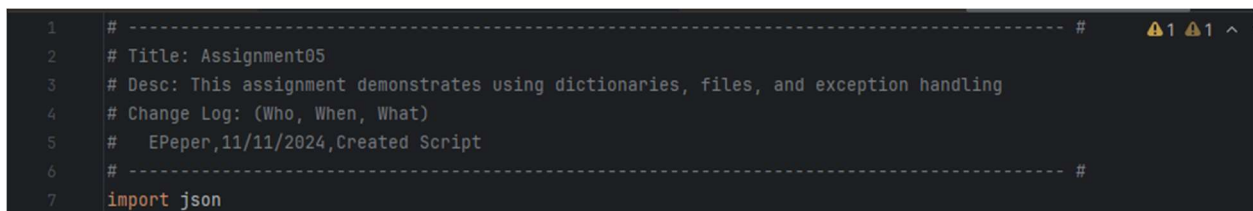
Using Dictionaries, JSON, and Exception Handling in Python

Intro

This week, we learned about dictionaries, JSON and exception handling in Python, focusing on creating an interactive menu that will write to a JSON file, read/interpret data from a JSON file, display in a multidimensional dictionary and contain specific error handling. Using this knowledge, I opened and read a JSON file, generated a script that would create an interactive menu by using a while loop that would allow a user to generate data that would be written to the JSON file for the recording of student enrollment data, and display said data in a multidimensional dictionary while putting in place certain errors to inform the user what the issue could be if they get an error.

Creating the Program and Importing JSON

To start, I created the header for my script on lines 1-6. This is important information for both my future self and other potential programmers who are reviewing my script to know who created the script and what it's used for. I then imported json on line 7 so pycharm could interpret and write to a JSON file.



```
1  # ----- #
2  # Title: Assignment05
3  # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4  # Change Log: (Who, When, What)
5  #   EPeper,11/11/2024, Created Script
6  # ----- #
7  import json
```

Figure 1.1 – Header

After that, I went on to the next portion of the assignment, which was to write the main body of my script. The first step of that was to define my data constants. On lines 9-20, I wrote out the constants “MENU” and “FILE_NAME” in all caps to indicate that they are constants, while also putting type hints to indicate what the data types are. I then set the values to those described in Assignment 05.

```

9      # Define the Data Constants
10     MENU: str = ''
11     ---- Course Registration Program ----
12     Select from the following menu:
13         1. Register a Student for a Course.
14         2. Show current data.
15         3. Save data to a file.
16         4. Exit the program.
17     -----
18     '''
19     # Define the Data Constants
20     FILE_NAME: str = "Enrollments.json"

```

Figure 1.2 – The Constants

I then went on to create my variables on lines 22-30. I wrote out “student_first_name” as my first variable, and gave it a type hint to indicate that it is a string. I then set the value as an empty string. I repeated this process but set the following variable name as “student_last_name”, “course_name”, “json_data” and “menu_choice”. For my final variable, “file”, I set it the value None. Because it is set to None, it does not need a type hint. I also set up my list and dictionaries, “student_data”, and “students” that will hold the string data generated by the user.

```

22     # Define the Data Variables and constants
23     student_first_name: str = '' # Holds the first name of a student entered by the user.
24     student_last_name: str = '' # Holds the last name of a student entered by the user.
25     course_name: str = '' # Holds the name of a course entered by the user.
26     json_data: str = '' # Holds data contained in JSON file
27     file = None # Holds a reference to an opened file.
28     menu_choice: str # Hold the choice made by the user.
29     student_data: dict = {str, str} # one row of student data
30     students: list = [] # a table of student data

```

Figure 1.3 – The Variables

Now that my variables and constants have been created, I opened and read the file “Enrollments.json” on lines 32-48. I set my list “students” equal to the loaded file contents. After, I set a for loop for then item in students and then closed the file. For error handling, I set up a try statement before this section of code and followed it with two exceptions. The first was to throw an error if the files is not found and create a new file so the code could run when restarted. The second exception dealt with an unknown error, which would then rest the roster. Finally, the file would close so the user could rerun if the try fails.

```

32 # When the program starts, read the file data into a list of dictionaries (table)
33 # Extract the data from the file
34 try:
35     file = open(FILE_NAME, 'r')
36     students = json.load(file)
37     for item in students:
38         file.close()
39 except FileNotFoundError as e:
40     print("File not found. Creating...")
41     open(FILE_NAME, 'w')
42 except Exception as e:
43     print('Unknown exception. Resetting roster...')
44     students = []
45     print(type(e), e, sep='/n')
46 finally:
47     if not file.closed:
48         file.close()

```

Figure 1.4 – Reading the file

I then set up a while loop to create my interactive menu on lines 50-55. I started by creating my while loop and set the Boolean value to True. I then set the variable menu_choice equal to the input function of our constant MENU. This will allow our user to choose one of the four choices we have displayed to them.

```

50 # Present and Process the data
51 while True:
52
53     # Present the menu of choices
54     print(MENU)
55     menu_choice = input("What would you like to do: ")

```

Figure 1.5 – Presenting the Menu

I then set up option 1 on lines 57-71, which would allow a user to input data into our program. I set up an if statement, I set the variable menu_choice to a conditional statement of 1. This means if our user enters 1, they will be asked a series of questions so data can be recorded. I then set inputs so the user can enter their data. Lastly, I set the dictionary “student_data” to hold the data from the inputted variables, and appended the it to create a new row. A message will then be displayed informing the user that they have registered the student. Lastly, I used a try statement on line 59 and raised exceptions on lines 62, 65, and 71. This will stop users from incorrectly adding data.

```

57 # Input user data
58 if menu_choice == "1": # This will not work if it is an integer!
59     try:
60         student_first_name = input("Enter the student's first name: ")
61         if not student_first_name.isalpha():
62             raise ValueError('Please, do not enter a number in the name field')
63         student_last_name = input("Enter the student's last name: ")
64         if not student_last_name.isalpha():
65             raise Exception('Please, do not enter a number in the name field')
66         course_name = input("Please enter the name of the course: ")
67         student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course_name}
68         students.append(student_data)
69         print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
70     except ValueError as e:
71         print(e) # Prints the custom message

```

Figure 1.6 – Inputting User Data

Next, for option 2, I set up an elif statement and I set the variable menu_choice to a conditional statement of 2. This menu choice will allow the user to be presented the current data. To do this, I created a print statement to tell the user “The current data is: “. I then set up a for loop to display a statement informing the user the individual student has registered for a course. This is done on lines 73-80.

```
73     # Present the current data
74     elif menu_choice == "2":
75
76         # Process the data to create and display a custom message
77         print("-"*50)
78         for item in students:
79             print(f"Student {item['FirstName']} {item['LastName']} is registered for {item['CourseName']}")
80         print("-"*50)
```

Figure 1.7 – Presenting User Data

For option 3, I created another elif statement and I set the variable menu_choice to a conditional statement of 3. This option saves the data to the file. I set the variable “file” to open the constant “FILE_NAME”, which holds the .JSON file “Enrollments”. I dumped the data from list “students” in Enrollments.json and closed the file. After that, I put a print statement to show what has been recorded. To catch any errors, I set a for statement and threw an exception if there were any errors in saving the file. This is done on lines 82-96.

```
82     # Save the data to a file
83     elif menu_choice == "3":
84         try:
85             file = open(FILE_NAME, "w")
86             json.dump(students, file)
87             file.close()
88             print("The following data was saved to file!")
89             for item in students:
90                 print(f"Student {item['FirstName']} {item['LastName']} is enrolled in {item['CourseName']}")
91         except Exception as e:
92             print('Error saving to the file')
93             print(e)
94         finally:
95             if file and not file.closed:
96                 file.close()
```

Figure 1.8 – Saving User Data

Lastly, we need to stop the loop on lines 98-104. I create an elif statement and I set the variable “menu_choice” to a conditional statement of “4”. I then set my break here to close the loop. I also put an else statement, where if the user inputs any other number at the menu besides 1-4, they will receive an error message.

```
98     # Stop the loop
99     elif menu_choice == "4":
100         break # out of the loop
101     else:
102         print("Please only choose option 1, 2, or 3")
103
104     print("Program Ended")
```

Figure 1.8 – Closing the Loop

Testing the Script

To test the script, I run the command console. I then type in “CD” to change the directory to the location of my script. I type in the “Python” command followed by the name of our script, Assignment05.py. I’m then given the menu options.

```
---- 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:
```

Figure 2.1 – Command Console

I enter “1”. I enter my first name, last name and Python 100. I am then presented with the menu again.

```
C:\Python\PythonCourse>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: Evan
Enter the student's last name: Peper
Please enter the name of the course: Python 100
You have registered Evan Peper for Python 100.
```

Figure 2.2 – Option 1

I then enter “2” at the menu to display my 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
-----

Student Bob Smith is registered for Python 100
Student Sue Jones is registered for Python 100
Student Evan Peper is registered for Python 100
-----
```

Figure 2.3 – Option 2

I then enter “3” at the menu to save the data to the 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: 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 Evan Peper is enrolled in Python 100
```

Figure 2.3 – Option 3

I then check inside my “Enrollments.json” file to show that the data has been recorded.

```
{"FirstName": "Evan", "LastName": "Peper", "CourseName": "Python 100"}]
```

Figure 2.4 – Enrollments.csv

I then enter in a “5” to show my first error message.


```

---- 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: 5
Please only choose option 1, 2, or 3

```

Figure 2.5 – Error Message 1

I then try to run my program without an enrollment.json file available. I get my second error.

```

C:\Python\PythonCourse>Python Assignment05.py
File not found. Creating...
Traceback (most recent call last):
  File "C:\Python\PythonCourse\Assignment05.py", line 47, in <module>
    if not file.closed:
           ^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'closed'

```

Figure 2.6 – Error Message 2

I try to add numbers into the name fields to display another error message.

```

---- 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: 123
Please, do not enter a number in the name field

```

```

What would you like to do: 1
Enter the student's first name: Evan
Enter the student's last name: 123
Traceback (most recent call last):
  File "C:\Python\PythonCourse\Assignment05.py", line 65, in <module>
    raise Exception('Please, do not enter a number in the name field')
Exception: Please, do not enter a number in the name field

```

Figure 2.6 – Error Message 3

And lastly I enter “4” to close my 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
```

Figure 2.6 – Exiting the program

Summary

In summation, I was able to use my readings, module notes and lecture to give me an understanding of dictionaries, JSON and error exceptions. Using this knowledge, I was able to write a script in my IDE to display a menu, open/interpret data from my file “enrollments.json” and use a while loop to create an interactive program where a user can store enrollment data. I then was able to use a try/exception block to create error messages that would allow my user to be able to better understand the errors that might arise.