Alan Martin

02/12/2024

Foundations of Python

Assignment_05

Dictionaries, JSON files, Data Handling

Introduction

In this document I'll be walking though creating python script that uses a loop to cycle through selections on our menu until we decide to quit our program, reaching the break statement. We'll first use selection #1 to take user input, selection #2 will print the input receive from user, selection #3 will open-save-close file with user data, selection #4 will break the loop and exit the program. This week also introduced dictionaries and reading file data into a list of lists in the program. We learned how to read that data in with JSON files using **json.load()** how to save a JSON file with **json.dump()**. Data handling was also introduced using the **try:**, **except:**, and **finally:** statements. Lastly, we started using the cloud to save our program with githubs repositories,

Setting python script header

Setting the script header from the start helps document your work. Included I have my title, a description, python version, change log describing who, what, when. I added Python version so I can become more aware of how a new version may affect my program.

```
1  # ------------------------------------------------------------------------------------------ #
2  # Title: Assignment05
3  # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4  # Python 3.12.1
5  # Change Log: (Who, When, What)
6  #   RRoot,1/1/2030,Created Script
7  #   Alan Martin, 2/10/2024, Created Script
8  # ------------------------------------------------------------------------------------------ #
```

Import libraries

Since we'll be dealing with Json files, we needed to import the Json library. I saw during the examples the from io import "TextIOWrapper", but can't quite figure out what this does.

```
9
10  import json
11  from io import TextIOWrapper
12
```

Defining the Constants

The constants in this program were predefined. The MENU constant will become the visual instruction for the user. The FILE_NAME constant is set to "Enrollments.json" and will end up saved in the same folder as my program.

```
12
13  # Define the Data Constants
14  MENU: str = '''
15  ---- Course Registration Program ----
16    Select from the following menu:
17      1. Register a Student for a Course.
18      2. Show current data.
19      3. Save data to a file.
20      4. Exit the program.
21  ----------------------------------------
22  '''
23
24  FILE_NAME: str = "Enrollments.json"
25
```

Defining the Variables

These variables names have been predefined, but their value have not been set. Student_first_name, student_last_name, and course_name will take user input. json_data are both part of calling the file function. menu_choice from user directs the loop and from our MENU constant. We use the dictionary student_data variable to hold user input information of each student (student_first_name, student_last_name, course_name), and students list [ ] to hold each student defined. File variable is left open and the TextIOWrapper from imports is shown again here.

```
26  # Define the Data Variables and constants
27  student_first_name: str = ''  # Holds the first name of a student entered by the user.
28  student_last_name: str = ''  # Holds the last name of a student entered by the user.
29  course_name: str = ''  # Holds the name of a course entered by the user.
30  menu_choice: str  # Hold the choice made by the user.
31  student_data: dict = {}  # one row of student data
32  students: list = []
33  file:TextIOWrapper = None
```

Read existing json file data into Python table

At the top of our program, before we begin our while loop, we need to extract the data from our Json file and save the data into our list of lists (table). Opening our enrollments.json in read mode, we'll transform the data it holds and load it into our list of lists. Loading json file is a little different than loading a csv file. Here, we'll identify the variable "students" to load our json file into using the json.load() function. We also see the data handling with the try:, except:, and finally statements. **Try:** runs first, if the code succeeds without failure, the program skips to the **finally: statement,** but if there is an error in that **try:**, the program skips to the first except statement, and so on. There is a whole table of exceptions that you can use to show the error to the user in a much friendly, easier to read way. **except FileNotFoundError:** is very specific and only executes when that error occurs, where **except Exception as e:** covers all errors and and reports the specifics back to the user by way of print(e, e.__doc__, type(e), sep='\n'.

```
36  # When the program starts, read the file data into a list of lists (table)
37  # Extract the data from the file
38  try:
39      file = open(FILE_NAME, "r")
40      students = json.load(file)
41  except FileNotFoundError:
42      print("Sorry, this file does not exist")
43  except Exception as e:
44      print("There was a non-specific error!\n")
45      print("-- Technical Error Message -- ")
46      print(e, e.__doc__, type(e), sep='\n')
47  finally:
48      print("Closing File")
49      file.close()
```

While Loop – Presenting the menu of choices.

We start our loop using the while statement. This remains true until we tell it otherwise and we end the loop. Our menu is listed below the While statement ensuring that it's incorporated in our loop. Our first user input asks which option they would like to select from the menu, controlling the flow of the loop.

```
52    # Present and Process the data
53    while True:
54
55        # Present the menu of choices
56        print(MENU)
57        menu_choice = input("What would you like to do: ")
58
```

Option #1 Register a Student for a course.

Option #1 uses the if-statement and allows the user to register a student for the course. It does this by asking the user questions sequentially and assigning those input values to our variables. We'll also want to save that input user data into our student_data dict and add/append it to our list of lists, students, each time selection 1 is chosen. The append is also used to add to the list of students that was loaded when we read the csv file into the program. The continue statement takes us to the beginning of our while loop. We also see more data handling here in the **if not and raise statements. If not** student_first_name.isalpha(), means if there is any characters other than alphanumeric, the **raise statement** reads, alerting the user the error **ValueError,** with a print statement explaining the error so its easy for the user to understand. . The continue statement takes us to the beginning of our while loop.

```
59    # Input user data
60    if menu_choice == "1":  # This will not work if it is an integer!
61        try:
62            student_first_name = input("Enter the student's first name: ")
63            if not student_first_name.isalpha():
64                raise ValueError("Student first name can only contain alphabetic characters.")
65            student_last_name = input("Enter the student's last name: ")
66            if not student_last_name.isalpha():
67                raise ValueError("Student last name can only contain alphabetic characters.")
68            course_name = input("Please enter the name of the course: ")
69            student_data = {'firstname': student_first_name, 'lastname': student_last_name, 'course': course_name}
70            students.append(student_data)
71            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
72        except ValueError as e:
73            print("User entered invalid information, please try again.")
74            continue
75
```

Option #2 – Present the current data.

Option #2 uses the else-if statement, (user must enter “2” to enter this loop). In this option we take the data saved in our students list and print it for the user to view. Using the for statement, we’ll list each student in the students table. Using the f-string format, ‘firstname’, ‘lastname’, and ‘course’ save in student_data and inserts into the string. The continue statement takes us to the beginning of our while loop.

```
75      # Present the current data
76      elif menu_choice == "2":
77
78          # Process the data to create and display a custom message
79          print("-"*50)
80          for student in students:
81              print(f"Student {student['firstname']} {student['lastname']} is enrolled in {student['course']}")
82          print("-"*50)
83          continue
```

Option #3 – Save the data to our file.

Option #3 uses the else-if statement. In this option we **try:** assigning file to open our Enrollments.json file in write mode. We then use the json function json.dump to dump the file contents into the students list[]. If this fails we use the broad exception **except Exception as e:** printing that there was an error and printing on the next line the error and the error defining the documentation with __.doc__ . **finally:** we file.close, saving the data. The continue statement takes us to the beginning of our while loop.

```
85      # Save the data to a file
86      elif menu_choice == "3":
87          try:
88              file = open(FILE_NAME, "w")
89              json.dump(students, file)
90          except Exception as e:
91              print("There was an error writing to the file.")
92              print(e, e.__doc__)
93          finally:
94              file.close()
95              continue
96
```

Option #4 – Stop the loop.

Option #4 uses the else-if statement. Selecting option #4 triggers a print statement “Program ended”, and uses a Break statement to kill the loop.

Selecting anything other than 1, 2, 3, 4 will prompt the user, “Please only choose option 1, 2, or 3”

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

Summery

In this assignment our constants were predefined, and our variable names were given but not assigned. We read a json file into a list in our program, allowing us to build onto an existing file. Option 1 allowed use to receive user input and collect data that we used to append our list. Option 2 read the students list back to us, ensuring that data was collected. Option 3 allowed us to save data that was collected and option 4 allowed us to save and close the program. We learned how to load and dump json files and exception handling using the try, except, and finally statements.

References

Common Header Format in Python | Delft Stack

Reading and Writing to text files in Python - GeeksforGeeks

thenewboston - YouTube

Python if-else Statement (tutorialspoint.com)

Writing Professional Papers (youtube.com)

Python Tutorial for Beginners 5: Dictionaries - Working with Key-Value Pairs (external site)

What Is JSON | Explained (external site)

Exceptions in Python - Python Tutorial (external site)

GitHub Tutorial - Beginner's Training Guide (external site)

What Is A Dictionary In Python? (external site)

Python Exceptions: An Introduction (external site)

On-premise vs Cloud-based File Sharing: Pros and Cons (external site)