Christopher Cipolla

2/14/2024

IT FDN 110 A Wi 24: Foundations of Programming: Python

Assignment 05

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

Registering Students with Dictionaries & Lists

Introduction

This paper will discuss the steps used to create, test, and execute a program that reads from a json file, accepts user input, prints the user inputs, and saves all the data to a file. It will cover the program requirements along with its creation testing the program, and finally, the execution and results of the program.

Creating the Program

This week's assignment, similar to last's weeks, but using json files instead of csv files, is to create a program which first reads a json file and store that data in a 2-dimensional list of dictionaries. Next, it will use a while loop to perform several tasks utilizing if/elif/else statements following these options:

1. Enter a student's name and course
2. Print the current data
3. Save data to a file
   Since the file overwrites the previous file, it saves the previous data and new data entered
4. Exit the program

There is also a bit of error handling incase an invalid input is given or if the data from the original file, or data to be written to the file is incorrect.

Writing the Code

Setting up

The first step was setting up the pseudo code to organize the program. This was provided in the starter file. The next step is to define the constants and variables. See Figure 1 for the header, and the initialization and assignment of values. Since json files were being used, the "import json" was called at the beginning.

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   CCipolla, 2/14/2024, Created Script
# ------------------------------------------------------------------------------------------ #

import json  # import code from Python's json module

# 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
# FILE_NAME: str = "Enrollments.json1" # Test to get the FileNotFound error
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.
json_data: str = ''  # Holds combined string data separated by a comma.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
```

Figure 1 – Initialization and Constant & Variable Definition

Reading the Starting file

Next the file was opened, read, and sorted into a list using the json.load() command. This function loads all the data into a list of dictionaries called “students”. In case the incorrect file was called, the statement was put into a try/except/finally block to catch potential errors, such as the FileNotFound error. In this case, a custom error message is displayed followed by the technical information stored in python. Next the file is created and set to an empty list, “students”, which was defined in the variable definition section as an empty list. A catch-all exception is also used. Lastly, the “finally” block will check to see if the file called is closed. If not, it will close it. See Figure 2

```python
# When the program starts, read the file data into a list of
# dictionaries (students)
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)

except FileNotFoundError as e:
    print(f'\nThe file, {FILE_NAME}, does not exist.\nA blank {FILE_NAME} is being created.\n')
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    file = open(FILE_NAME, "w")
    json.dump(students,file)
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 not file.closed:
        file.close()
    print()
```

Figure 2 - Reading the File

Starting the Loop

After the setup, the while loop was then started using the condition "True". With the loop started, the user is presented 4 options and instructed to choose one as shown in Figure 3.

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

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

Figure 3 - Loop Commencement

The next steps go over the menu options (1-4). Any other input gives an error message. The code for this is shown in Figure 7.

Option 1: Data Collection

If the first option is selected, as shown in Figure 1, the user will give input (first/last name, and course name) which will be stored in the defined variables: "student_first_name", "student_last_name", and "course_name". This block of code is also placed in a try/except block to check to see if the first or last name contain anything other than alphabetic characters. If the .isalpha() method returns a false, meaning there are other characters, a ValueError will be raised and a custom message will be displayed. The student's name and course are recorded into a dictionary as seen in Figure 4, and then appended into the list "students". The following code handles the ValueError and any other errors ("Exception") that may occur and tells the user that the error caused the program not to record the data entered.

```python
    # 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():  # if the first name is not all alphabetical characters
                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():  # if the last name is not all alphabetical characters
                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,
                            "Course": course_name}
            students.append(student_data)
            print(f"\nYou have registered {student_first_name} {student_last_name} for {course_name}.")
        except ValueError as e:
            print("\n", "!" * 10, "  ERROR  ", "!" * 10)
            print(e)  # Prints the custom message
            print("-- Technical Error Message -- ")
            print(e.__doc__, e.__str__(), sep='\n')
            print()
            print("No students were registered")
            print()
        except Exception as e:
            print("\n!!!!!!  ERROR!!!!!!")
            print("There was a non-specific error!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
            print()
            print("No students were registered")
            print()
        continue
```

Figure 4 - Option 1: Enter Data

Option 2: Print to Screen

When option 2 is selected, the program will print all the data the list “students” to the screen. This includes the students from the original file as well as any entered in via option 1. I noticed that the starter json file had a key typo so I added error handling for KeyError, telling the user to check the original file. The code is shown in Figure 5.

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

        # Process the data to create and display a custom message
        try:
            print("-" * 50)
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
            print("-" * 50)
        except KeyError as e:
            print(f"\nError found in the original file\nCheck the keys in {FILE_NAME} to ensure they match")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        continue
```

Figure 5 – Option 2: Print to Screen

Option 3: Save to File

Option 3 will overwrite the json file with all the lists in “students” via the json.dump function/method, as shown in Figure 6. If successful, it will print the students and their classes from the “students” list, which, again, includes new and old data. The type of errors expected here are the TypeError, which I don’t understand how to trigger, as well as the KeyError mentioned in the Option 2 section (in this case, the data is saved, but the user still needs to clean up the json file). It also will catch any other errors through the “Exception” code. And lastly, if the file is still open because an error interrupted the program, the “finally” block will close it.

```python
    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file)

            print("\nThe following data was saved to file!")
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
            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 KeyError as e:
            print(f"\nError found in the original file\nCheck the keys in {FILE_NAME} to ensure they match")
            print(f"Data saved, but the keys in {FILE_NAME} need to be cleaned up.")
            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()
```

Figure 6 - Option 3: Save to File

Option 4: Close the Program

Figure 7 shows the code for option 4 that will end the while loop using the “break” function a long with a “feel good” message and a final message letting the user know the program is closing.

```
    # Stop the loop
    elif menu_choice == "4":
        print("\nWe know you have choices when it comes to registering students.\n"
              "Thank you for choosing Assignment05.\n")
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("=" * 25)
print("Program Ended")
print("=" * 25)
```

Figure 7 - Option 4: Close the Program

Invalid Inputs

Finally, in Figure 8 (also in Figure 7), the else statement will catch any input that is not 1, 2, 3, or 4 and inform the user that the input was not valid. At this point, the While Loop will start over and give the user the valid options again.

```
    else:
        print("Please only choose option 1, 2, 3, or 4")
```

Figure 8 - Invalid Input

Testing and Debugging

There was quite a bit of back and forth to get the correct formatting determining what needed to be done in the error handling (such as creating the FILE_NAME when one didn’t exist, as discussed in the labs), as well as reading from the original json file (finding the errors in the starter json file). It took some time to remember that I needed to create the json file (via the open(FILE_NAME, ‘w’) function) in the FileNotFound exception block. Since defining the “file” variable failed in when I tried to open it in read mode, python didn’t know what the “file” variable was and couldn’t define the .closed method in the “finally” block.

Execution and Results

Eventually, the program was executed in both the PyCharm shell and from the command line and printed to the screen and to the json file as expected. Upon running in either application, the prompts and displayed messages were iterated for clarity and effectiveness. Figure 9 through **Error! Reference source not found.** show the results from the PyCharm, the command line, and json files.

```
The file, Enrollments.json1, does not exist.
A blank Enrollments.json1 is being created.

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollments.json1'
File not found.
<class 'FileNotFoundError'>


---- 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 9 – FileNotFound Error

```
---- 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 enrolled in Python 100

Error found in the original file
Check the keys in Enrollments.json to ensure they match
-- Technical Error Message --
'LastName'
Mapping key not found.
<class 'KeyError'>

---- 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: Qwerty
Enter the student's last name: Que
Please enter the name of the course: Eng101

You have registered Qwerty Que for Eng101.

---- 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

Error found in the original file
Check the keys in Enrollments.json to ensure they match
Data saved, but the keys in Enrollments.json need to be cleaned up.
-- Technical Error Message --
'LastName'
Mapping key not found.
<class 'KeyError'>

---- 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 10 – KeyError in the original json file

```
What would you like to do: 1
Enter the student's first name: 1234

 !!!!!!!!!!   ERROR   !!!!!!!!!!
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

No students were registered


---- 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: Bob
Enter the student's last name: 23

 !!!!!!!!!!   ERROR   !!!!!!!!!!
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.

No students were registered
```

Figure 11 - Name Errors (ValueError)

```
---- 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: Qwerty
Enter the student's last name: Sue
Please enter the name of the course: Eng101

You have registered Qwerty Sue for Eng101.

---- 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 enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Qwerty Sue is enrolled in Eng101
--------------------------------------------------

---- 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 Qwerty Sue is enrolled in Eng101

---- 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

We know you have choices when it comes to registering students.
Thank you for choosing Assignment05.

========================
Program Ended
========================

Process finished with exit code 0
```

Figure 12 - Complete Program

Enrollments.json × Assignment05.py Mod05-Lab01-WorkingwithDictionariesa

```
[{"FirstName": "Bob", "LastName": "Smith", "Course": "Python 100"},
 {"FirstName": "Sue", "LastName": "Jones", "Course": "Python 100"},
 {"FirstName": "Qwerty", "LastName": "Sue", "Course": "Eng101"}]
```

Figure 13 - PyCharm json 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: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Qwerty Sue is enrolled in Eng101
--------------------------------------------------

---- 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: Extra
Enter the student's last name: Kid
Please enter the name of the course: Class234

You have registered Extra Kid for Class234.

---- 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 Qwerty Sue is enrolled in Eng101
Student Extra Kid is enrolled in Class234

---- 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.
----------------------------------------

What would you like to do: 4

We know you have choices when it comes to registering students.
Thank you for choosing Assignment05.

=========================
Program Ended
=========================
```

Figure 14 - Command Line Completed Code

```
[{"FirstName": "Bob", "LastName": "Smith", "Course": "Python 100"},
{"FirstName": "Sue", "LastName": "Jones", "Course": "Python 100"},
{"FirstName": "Qwerty", "LastName": "Sue", "Course": "Eng101"},
{"FirstName": "Extra", "LastName": "Kid", "Course": "Class234"}]
```


Summary

This document demonstrated how to write a program to read and write to a json file as well as the how to expect and handle some errors from both the users and developers. The program was tested and ran successfully in both PyCharm and the command line.