Alejandro Guillen

November 19, 2023

IT FDN 110 B

ASSIGNMENT 05: Dictionaries, JSON files, Exception handling and Github.

Introduction

Dictionaries:

A Python dictionary is a collection of key: value pairs. You can think about them as words and their meaning in an ordinary dictionary. Values are said to be *mapped* to keys. For example, in a physical dictionary, the definition *science that searches for patterns in complex data using computer methods* is mapped to the key *Data Science*.

Python dictionaries allow us to associate a value to a **unique** key, and then to **quickly access this value**. It's a good idea to use them whenever we want to find a certain Python object. We can also use lists for this scope, but they are much slower than dictionaries.

JSON files:

JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package called JSON. To use this feature, we import the JSON package in Python script. The text in JSON is done through quoted-string which contains the value in key-value mapping within { }. JSON shows an API similar to users of Standard Library marshal and pickle modules and Python natively supports JSON features.

Exceptions:

In Python, there are several built-in exceptions that can be raised when an error occurs during the execution of a program. Here are some of the most common types of exceptions in Python:

- **SyntaxError:** This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.
- **TypeError**: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.

- **NameError**: This exception is raised when a variable or function name is not found in the current scope.
- **IndexError**: This exception is raised when an index is out of range for a list, tuple, or other sequence types.
- **KeyError**: This exception is raised when a key is not found in a dictionary.
- **ValueError**: This exception is raised when a function or method is called with an invalid argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.
- **AttributeError**: This exception is raised when an attribute or method is not found on an object, such as trying to access a non-existent attribute of a class instance.
- **IOError**: This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.
- **ZeroDivisionError**: This exception is raised when an attempt is made to divide a number by zero.
- **ImportError**: This exception is raised when an import statement fails to find or load a module.

GitHub:

At a high level, GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. To understand exactly what GitHub is, we need to know two connected principles: Version control and Git (is a distributed version control system).

GitHub's interface is user-friendly enough so even novice coders can take advantage of Git. GitHub is so user-friendly, though, that some people even use GitHub to manage other types of projects. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

Step 1: Define Libraries, constants, variables.

First, it was to define the libraries JSON, constants, variables and organize the code according to the template for this module and define the input and outputs. The variables were set to empty string, student_data for row was set as dictionary.

```
import json
from json import JSONDecodeError
```

```
# 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.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[str, str] = {}  # one row of student data
students: list[dict[str, str]] = []  # a table of student data
datamsg: str = ''   # Holds format data before to print
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Step 2: Load data from initial file and exception handling

When the program starts, the program try to open the file: "Enrollments.json". If it exists is automatically read and then hold into the list of dictionary students. The program opens File_NAME object using the open function in mode read.

Also, I have added the exception handled in this part, due to the object FILE_NAME should exist before and it must have data. If the file does not exist in except FileNotFoundError will create the file. After this, if the file is empty the another except JSONDecodeError add {} brackets to initialize the data as well. Also, the last error handle is Exception to catch any other errors and print messages.

```
# When the program starts, read the file data into a list of dictionary rows
(table)
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    print('The file data was loaded successfully')
except FileNotFoundError as e:
    print("The file should exist before running this script!")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep="\n")
    print('Creating file if it does not exist')
    file = open(FILE_NAME,'w')
    json.dump(students, file)    # Loading initial data [] from collection
students
except JSONDecodeError as e:
    print("The file is invalid")
    file = open(FILE_NAME, "w")
    json.dump(students, file)    # Loading initial data [] from collection
students
except Exception as e:
    print('---Technical Information---')
    print(e, e.__doc__, type(e), sep="\n")
finally:
    if not file.closed:
        file.close()
```

Step 3: Create While loop.

Write the code for the different options according to the menu:

> ---- 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
> ----------

1- 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 variables. Also, we catch the errors for student name and last name. I have included validations for these inputs. Before holding on to the variables the program run ValueError if these fields are not alphabetic chars totally and shows a message.

```
# Present and Process the data
while True:
    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")
    # Input user data
    if menu_choice == "1":
        try:
            student_first_name = input("Enter the student's first name: ")
```

```
            if not student_first_name.isalpha():
                raise ValueError('The first name must be alphabetic')
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError('The last name must be alphabetic')
            course_name = input("Please enter the name of the course: ")
            student_data = {'first_name': student_first_name, 'last_name':
student_last_name, 'course_name': course_name}
            # Load it into the collection students (list)
            students.append(student_data)
        except ValueError as e:
            print(e)
            print('---Technical Information---')
            print(e.__doc__, type(e), sep='\n')
```

2- On menu choice 2, the variables hold the input data from dictionary students. I have used a for loop to assign all rows to each student record and then for the output to print each row to the console display.

```
# Present the current data
elif menu_choice == "2":
    # Process the data to create and display a custom message
    print("-"*50)
    for student in students:
        student_first_name = student['first_name']
        student_last_name = student['last_name']
        course_name = student['course_name']
        datamsg = "The student {} {}, has been registered for {}"
        print(datamsg.format(student_first_name, student_last_name,
course_name))
    print("-"*50)
```

3- On menu choice 3, the program opens a file named "Enrollments.json" using the open() function mode write. After this I used the json.dump function to save the content of the students dictionary to the file. Then file is closed using the close() method. Then the program displays the data stored in the file. Also, I have used the handle errors to prevent the program from crashing. In this case I ensure that the data will be stored clean using except TypeError to show messages to user and Execption error for any catch as well.

```
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students,file)
        file.close()
    except TypeError as e:
        print('---Technical Information---')
        print(e.__doc__, type(e), sep='\n')
    except Exception as e:
        print('---Technical Information---')
        print(e.__doc__, type(e), sep='\n')
    finally:
```

```
        if not file.closed:
            file.close()
```

4- On menu choice 4, the program ends using the break function.

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

```
        print('The data has not been recorded')
    continue
```

Summary

After running the program for this Module, the data was loaded using JSON file and dictionary collection, it allows our application to be more flexible to other applications and keep our code clearer and reusable.

This module has been interesting due to the handle of errors properly using try-except blocks, I have learned different types of errors and how to use them to show messages, validate, fix into the code when this happens and prevent the program from crashing.

Also, I have learned to use GigHub and how to collaborate and distribute our code with other developers.