**JSON File**

In Python, a JSON file is a file that contains data in JavaScript Object Notation (JSON) format. JSON is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. JSON is often used for transmitting data between a server and a web application, as it is language-independent and can be easily parsed by a variety of programming languages, including Python.

JSON files can be created and read in Python using the built-in **json** module. This module provides methods for encoding Python objects into JSON format, and decoding JSON data into Python objects.

The advantages of using JSON files in Python include:

1. Easy data exchange: JSON files provide a standardized format for exchanging data between different applications and systems. JSON can be easily parsed by a variety of programming languages, making it ideal for transmitting data across different platforms.
2. Human-readable format: JSON files are easy to read and write, which makes them more user-friendly than other data formats like XML.
3. Compact size: JSON files are lightweight and can be compressed, making them ideal for use in web applications where fast data transfer is important.
4. Versatility: JSON files can store a wide range of data types, including strings, numbers, Booleans, lists, and dictionaries. This makes them suitable for a variety of applications, from storing simple configuration data to complex nested data structures.
5. Easy integration with Python: Python provides built-in support for working with JSON files, making it easy to encode and decode data to and from JSON format.

Overall, JSON files provide a simple, efficient, and flexible way to store and exchange data in Python, making them a popular choice for web developers and data scientists alike.

I will use an example to demonstrate how we can modify a JSON file

Consider a JSON file containing 5 people’s name, ID, and mail address:

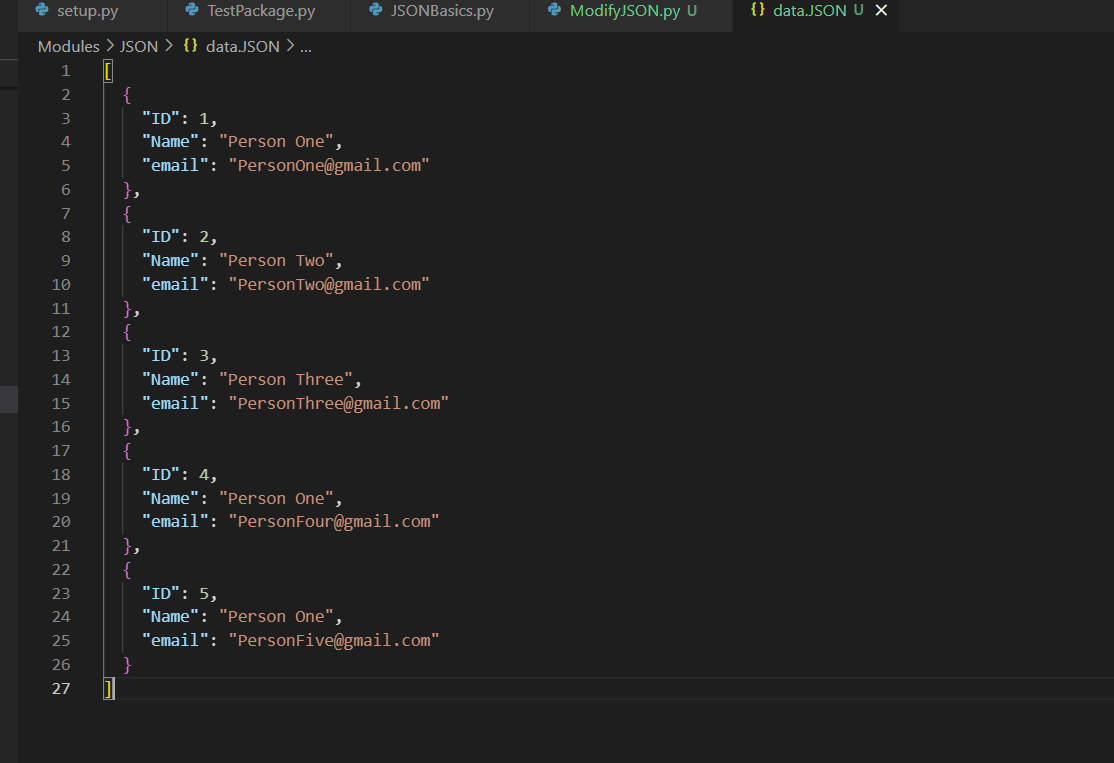


Figure 1 : JSON file containg details of 5 people

It is like a list of dictionaries.

We can modify this file by appending or removing a dictionary to this list or by changing the value inside a dictionary.

To operate on a JSON object, first we have to load it into a python object. The method **loads ()** is used to convert the JSON String document into the Python dictionary. The method **json. load()** is used to read the JSON document from file .

**1.Add to JSON file**

The below function is used to add a new entry into the JSON file containing the ID, Name and mail of 5 people.



Figure 2: Function to add data to JSON file

The ID, Name and mail of the new person is taken as input from CLI. Now, to read the contents of the json file, we open the file as file pointer ‘f’. Now, the data inside the JSON file is read using the json. load (<file pointer>) method. A list of dictionaries is returned, which is saved as data.Now,to this list, we append a new dictionary containing the information of the new person. Now, we need to write this updated list to the original file. We have to open the original file, this time with write permission. Existing data is overwritten.

now, the list we now have is a python object. To save it into a JSON file, we have to convert it into a JSON object first.

Data is written to the file using **the json. Dump (<List of dictionaries>, <file pointer>, <required indentation>)** method. The json.dump() method allows us to convert a python object into an equivalent JSON object and store the result into a JSON file at the current working directory. Indent is added to make the file more readable.

Now, we can see that the JSON file is updated with the new name!

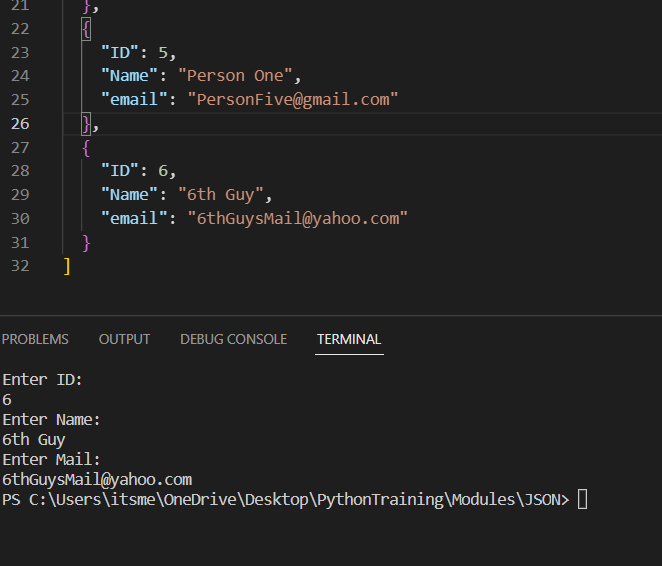


Figure 3 : New data added to JSON file

**2.Changing a particular data**

Consider a scenario in which one of the people whose data is stored in the JSON file wishes to change his mail id. How do we do that?

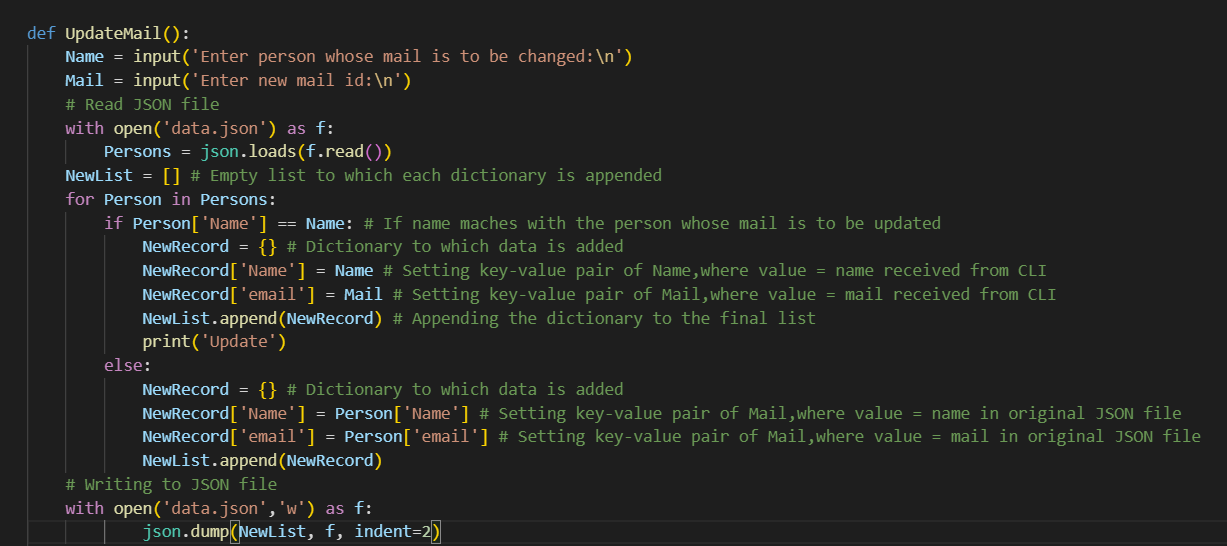


Figure 4: Function to update the details given in JSON file

First, we get the name of the person who needs to change his mail ID and get his new mail ID.

Next, we have to read the JSON file as before, using json.loads() method. This time we save the list as ‘Persons’. Now we iterate through each element in the list using a for loop. The list contains dictionaries. Now we have to check if the name in the dictionary matches the name we got from CLI.This is easy! All we have to do is check if the value associated with the key “Name” in the dictionary is the same as the name we got from CLI.We do this using an if statement. We first create an empty new list. If the values match, we create a new dictionary, and add the name and new mail as key value pairs. This dictionary is appended to the new list. In the else block, we do the same but name and mail in the new dictionary will be the same as the original list.

After the execution of the for loop, the new list is converted from python object to JSON object and written into the file using json. Dumps () method.

Here, we can see that the mail of Person One is changed.

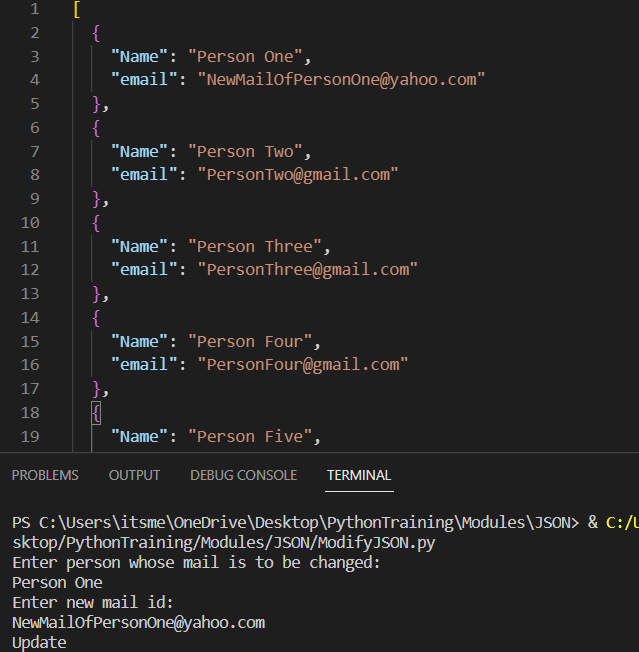


Figure 5: Mail of Person One is modified

**3.Delete a person’s data**

This can be done using a slight modification of the function that updated the mail.

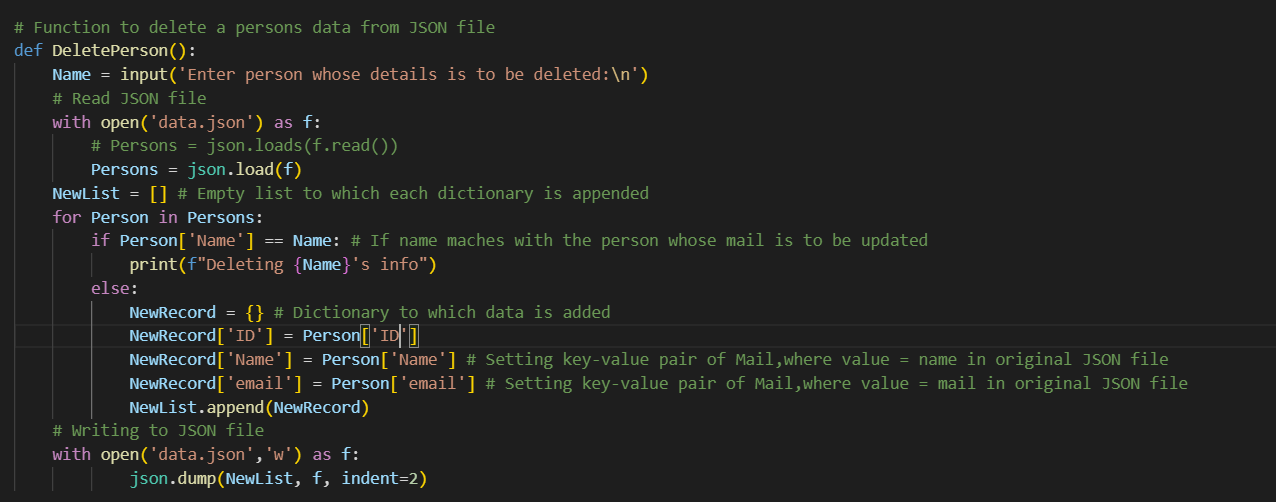


Figure 6: Function to delete a persons data from JSON file

We get the name of the person whose data is to be deleted from CLI. Then check if that name is a value associated with the key ‘Name’ in any of the dictionaries. If it is detected, just print deleting data or something. If the name doesn’t match with the value, we append the dictionary to the final list. After execution of the for loop, we write the list into the file using dumps() method.

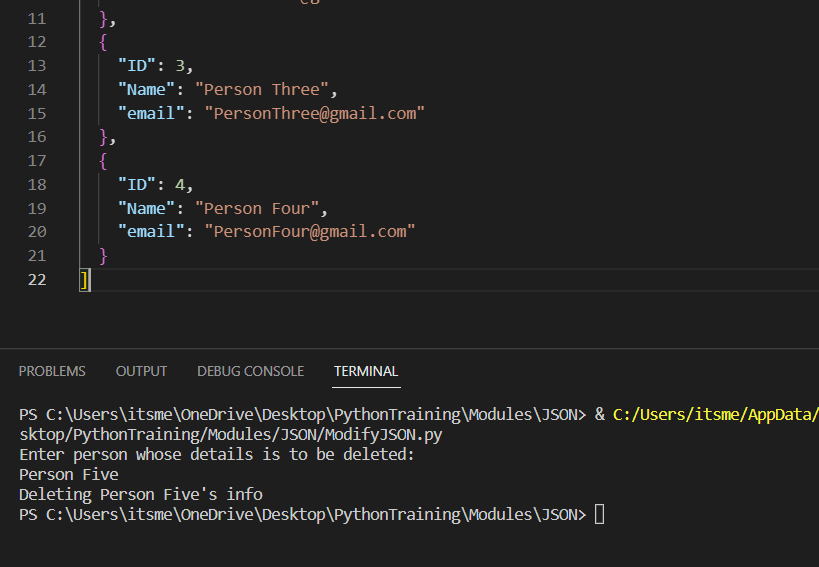


Figure 7: Deleted data of Person 5

Here, we can see that the information of person 5 is deleted.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

From the above examples, we understand that:

**To modify a JSON file there are essentially 3 steps;**

**1.Convert the JSON file into a python object after reading from file.**

**2.Do operations on the python object**

**3. Convert the python object into a JSON file and write to file.**