JSON(JavaScript Object Notation) File Handling

Sidheswar Routray
Department of Computer Science & Engineering
School of Technology

JSON file structure

- JSON stands for JavaScript Object Notation.
- JSON is lightweight data-interchange format.
- JSON is language independent.
- JSON supports array, object, string, number and values.
- Web applications commonly use JSON to exchange data between each other.

```
{"employees":[
    {"name":"Sunny", "email":”sunny@gmail.com"},
    {"name":"Rahul", "email":"rahul32@gmail.com"},
    {"name":"John", "email":"john32bob@gmail.com"}  ]}
```

JSON syntax is derived from JavaScript object notation syntax:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

Characteristics of JSON

- **Human-readable and writable**: JSON is easy to read and write.
- **Lightweight text-based data interchange format**: JSON is simpler to read and write when compared to XML.
- **Widely used**: JSON is a common format for data storage and communication on the web.
- **Language-independent**: Although derived from JavaScript, JSON can be used with many programming languages.

JSON Data types

Data Type	Description	Example
String	A string is always written in double-quotes. It may consist of numbers, alphanumeric and special characters.	"student", "name", "1234", "Ver_1"
Number	Number represents the numeric characters.	121, 899
Boolean	It can be either True or False.	true
Null	It is an empty value.	

JSON object

- JSON objects refer to dictionaries, which are enclosed in curly braces, i.e., { }.
- A JSON object is a collection of key/value pairs. The keys are strings, and the values can be strings, numbers, objects, arrays, true, false, or null.

```
{"name" : "Jack", "employeeid" : 001, "present" : false}
```

```
{
    "employee": {
        "name":       "sonoo",
        "salary":     56000,
        "married":    true
    }
}
```

JSON Array

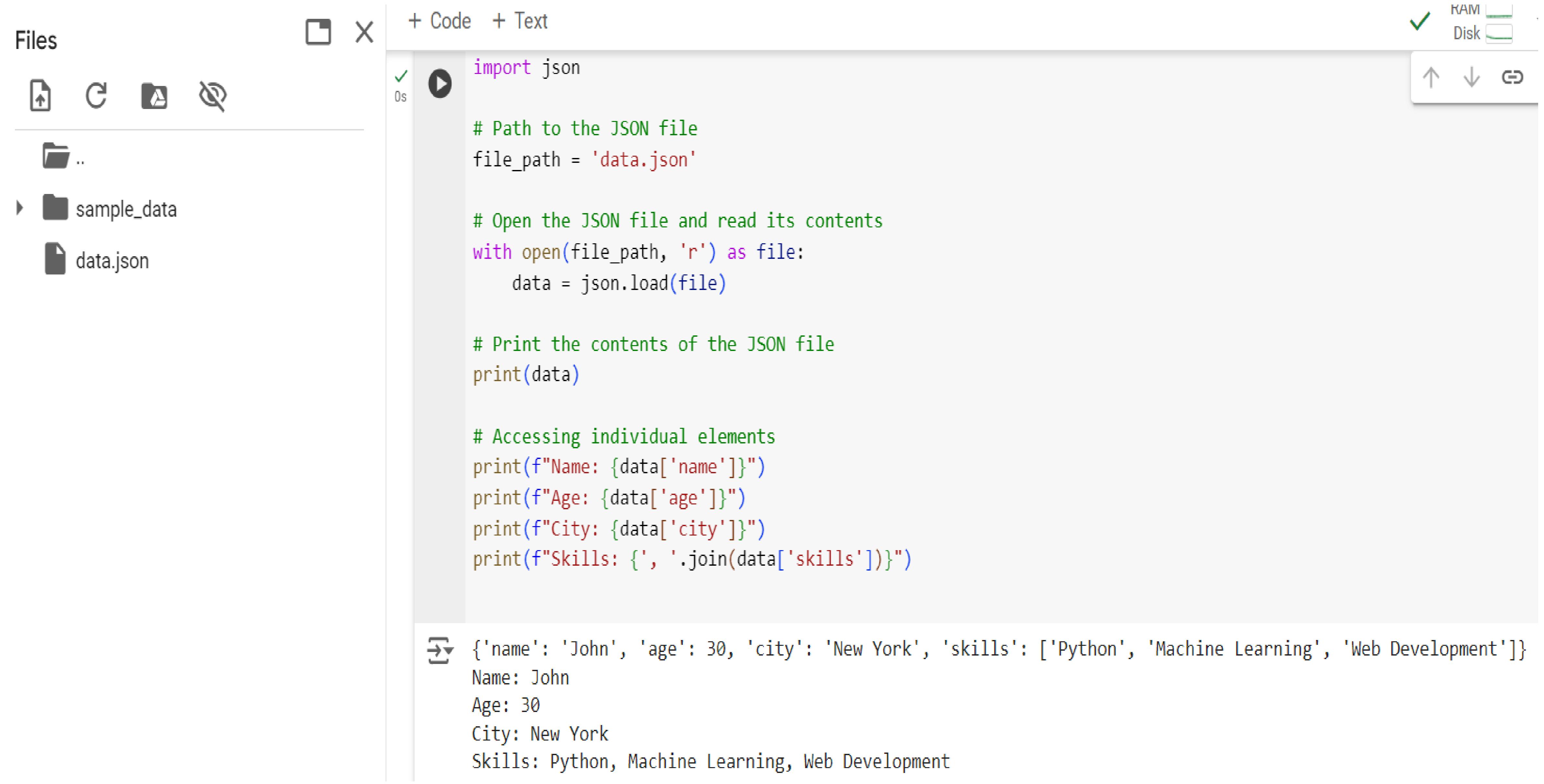
- A JSON array is an ordered collection of values. The values can be strings, numbers, objects, arrays, true, false, or null.

```
[
{
"PizzaName" : "Country Feast",
"Base" : "Cheese burst",
"Toppings" : ["Jalepenos", "Black Olives", ", "Cherry tomatoes"],
"Spicy" : "yes",  },
{
"PizzaName" : "Veggie Paradise",
"Base" : "Thin crust",
"Toppings" : ["Jalepenos", "Black Olives", "Cherry tomatoes"],
"Spicy" : "yes",
}
]
```

Reading JSON File

```
import json

# Sample JSON data in "data.json"
# {"name": "John Doe", "age": 30, "city": "New York"}
file_path = "data.json"

# Reading JSON data from the file
with open(file_path, "r") as json_file:
     data = json.load(json_file)

print(data)
# Output: {'name': 'John Doe', 'age': 30, 'city': 'New
York'}
```

Reading JSON File



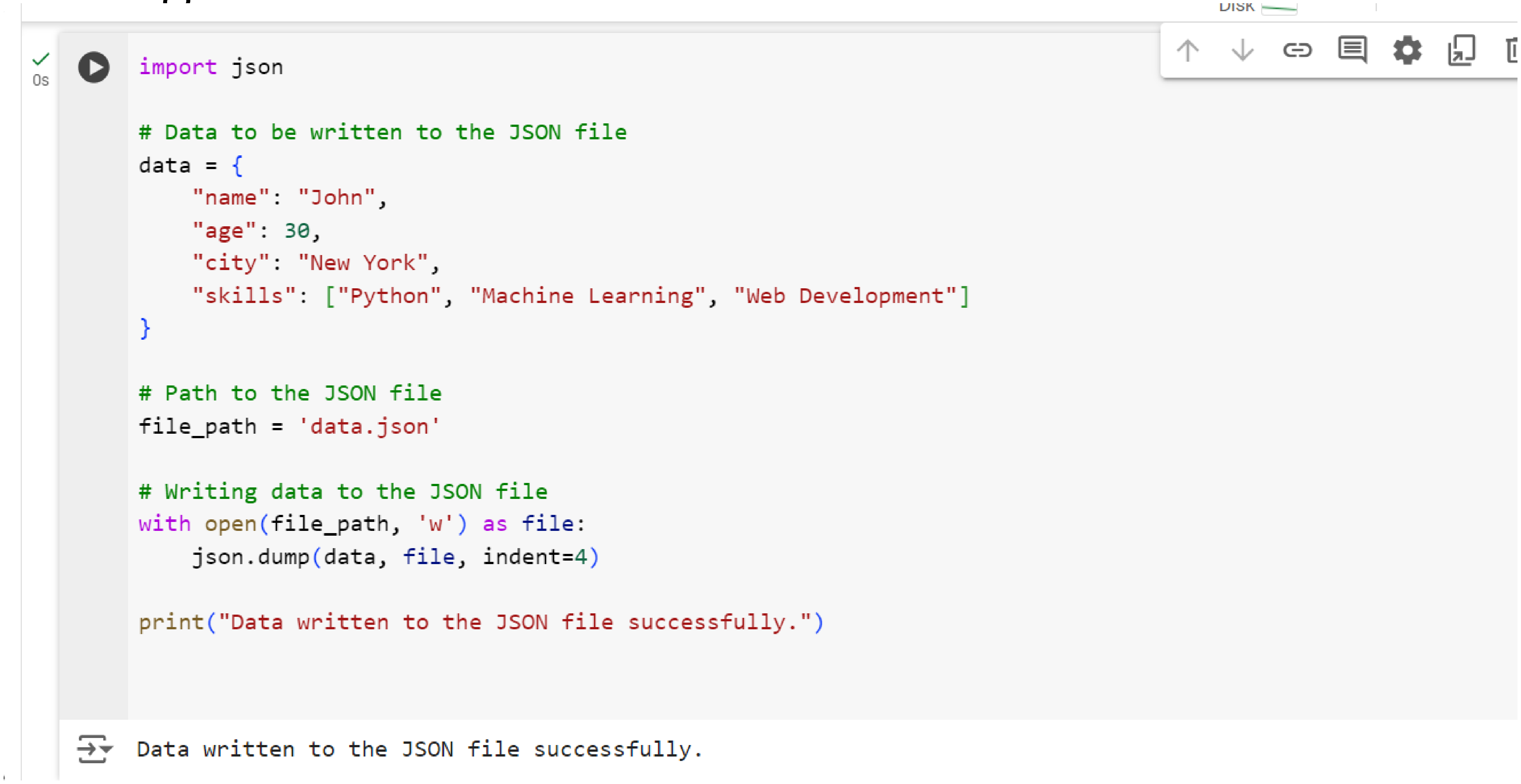
```python
import json

# Path to the JSON file
file_path = 'data.json'

# Open the JSON file and read its contents
with open(file_path, 'r') as file:
    data = json.load(file)

# Print the contents of the JSON file
print(data)

# Accessing individual elements
print(f"Name: {data['name']}")
print(f"Age: {data['age']}")
print(f"City: {data['city']}")
print(f"Skills: {', '.join(data['skills'])}")
```

```
{'name': 'John', 'age': 30, 'city': 'New York', 'skills': ['Python', 'Machine Learning', 'Web Development']}
Name: John
Age: 30
City: New York
Skills: Python, Machine Learning, Web Development
```

Writing JSON file

```
import json
# Sample Python dictionary
data = {
"name": "John Doe",
"age": 30,
"city": "New York"
}

# Writing JSON data to a file
file_path = "output.json"
with open(file_path, "w") as json_file:
json.dump(data, json_file)

# The "output.json" file will contain: {"name": "John
Doe", "age": 30, "city": "New York"}
```

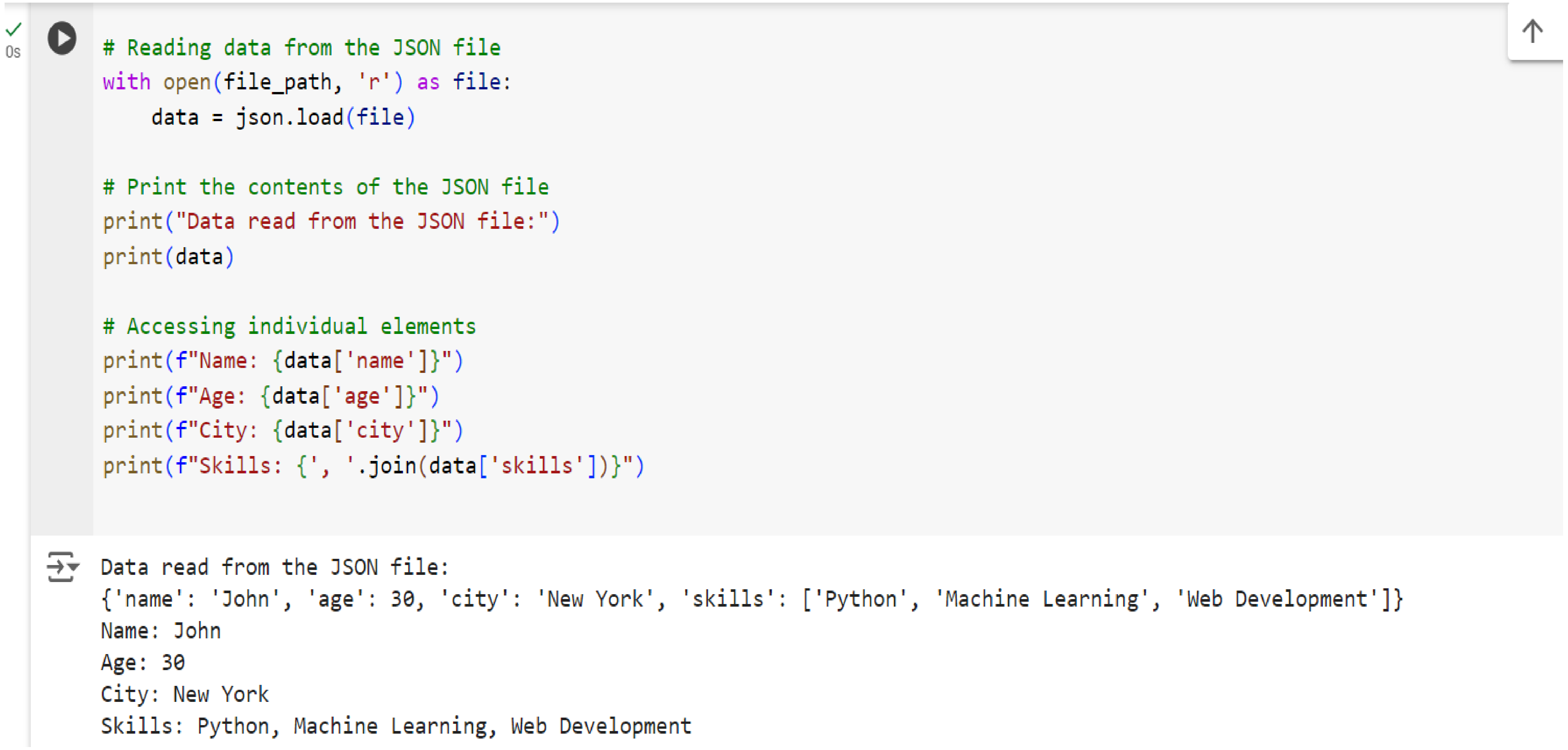
Writing JSON file



```
import json

# Data to be written to the JSON file
data = {
    "name": "John",
    "age": 30,
    "city": "New York",
    "skills": ["Python", "Machine Learning", "Web Development"]
}

# Path to the JSON file
file_path = 'data.json'

# Writing data to the JSON file
with open(file_path, 'w') as file:
    json.dump(data, file, indent=4)

print("Data written to the JSON file successfully.")
```

```
Data written to the JSON file successfully.
```

Writing JSON file



```python
# Reading data from the JSON file
with open(file_path, 'r') as file:
    data = json.load(file)

# Print the contents of the JSON file
print("Data read from the JSON file:")
print(data)

# Accessing individual elements
print(f"Name: {data['name']}")
print(f"Age: {data['age']}")
print(f"City: {data['city']}")
print(f"Skills: {', '.join(data['skills'])}")
```

```
Data read from the JSON file:
{'name': 'John', 'age': 30, 'city': 'New York', 'skills': ['Python', 'Machine Learning', 'Web Development']}
Name: John
Age: 30
City: New York
Skills: Python, Machine Learning, Web Development
```

Loading JSON Data as Python Objects:

```
import json
# Sample JSON data in "nested_data.json"
# {"person": {"name": "John Doe", "age": 30}, "city": "New 
York"}
file_path = "nested_data.json"
# Reading JSON data from the file
with open(file_path, "r") as json_file:
     data = json.load(json_file)

print(data)
# Output: {'person': {'name': 'John Doe', 'age': 30}, 'city': 
'New York'}
print(data['person']['name'])
# Output: 'John Doe'
```

Problem Statement

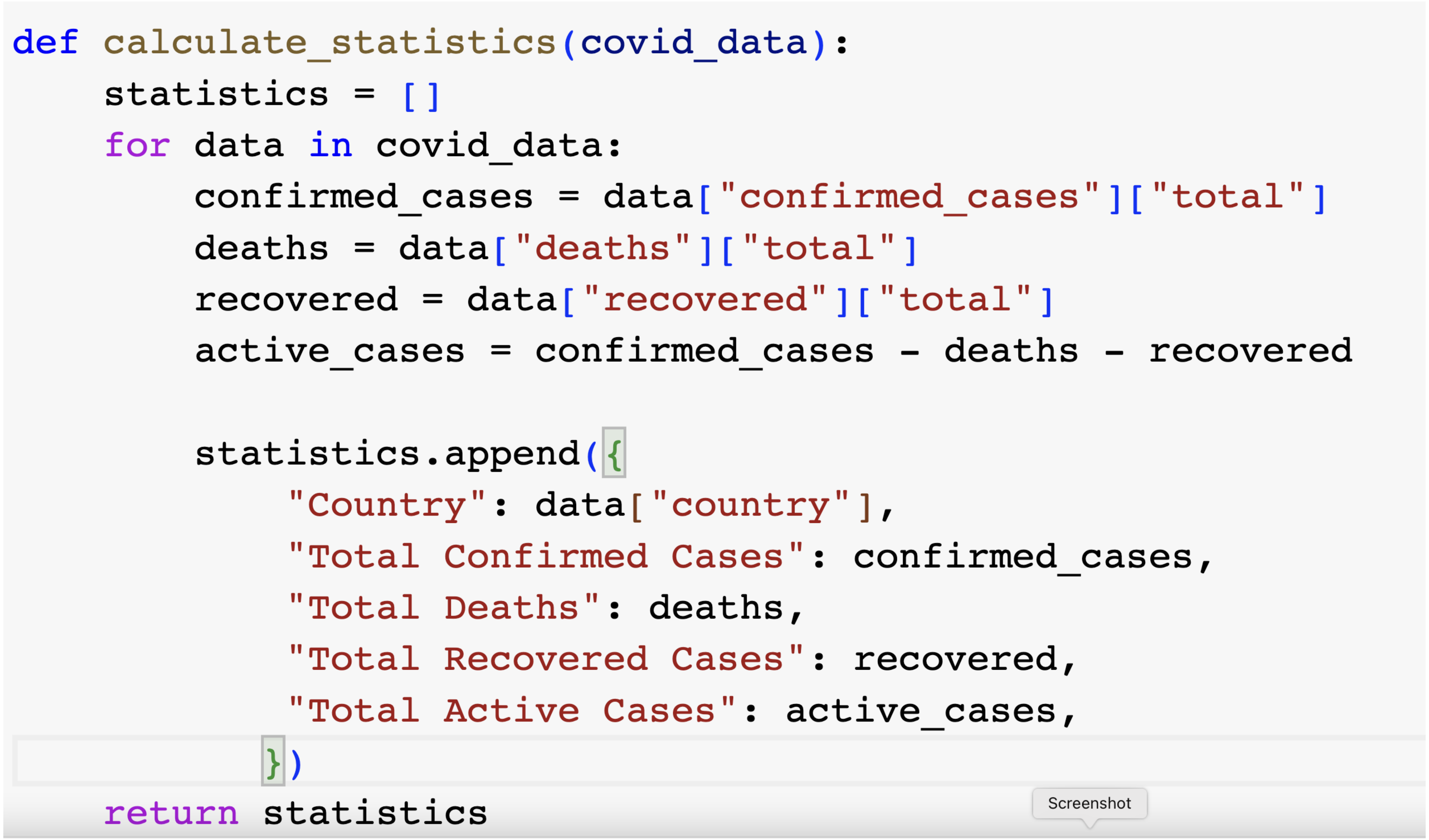
You are working as a data scientist for a healthcare organization, and your team has been tasked with analysing COVID-19 data from multiple countries. The data is stored in JSON files, with each file representing the daily COVID-19 statistics for a specific country. Each JSON file has the following structure:

```
{ "country": "Country Name",
"date": "YYYY-MM-DD",
"confirmed_cases": { "total": 1000, "new": 50 },
"deaths": { "total": 20, "new": 2 },
"recovered": { "total": 800, "new": 30 }
}
```

Your task is to write a Python program that performs the following operations:

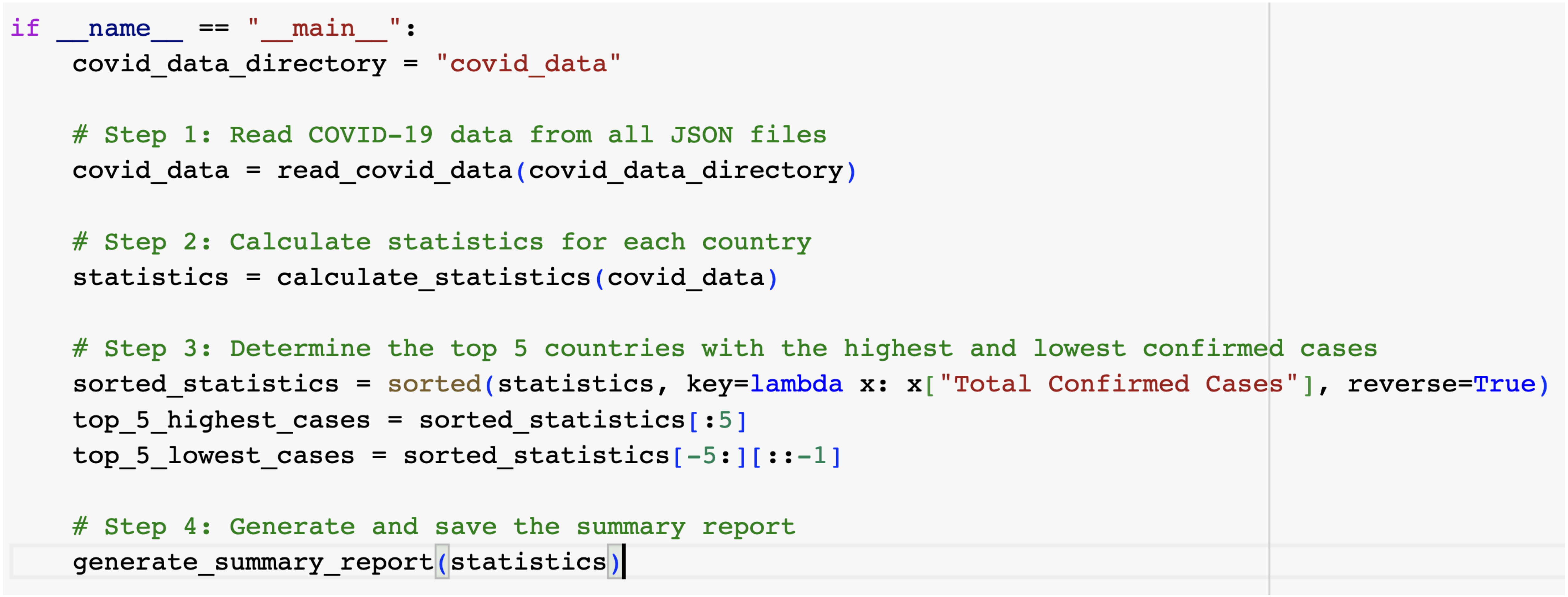
1. Read COVID-19 data from all JSON files in a given directory and its subdirectories.
2. Calculate and display the following statistics for each country:
    1. Total confirmed cases.
    2. Total deaths.
    3. Total recovered cases.
    4. Total active cases (total confirmed cases minus total deaths and total recovered).
3. Determine the top 5 countries with the highest number of confirmed cases and the lowest number of confirmed cases.
4. Generate a summary report in JSON format that includes the statistics for all countries and save it to a file named "covid19_summary.json".

```python
import json
import pandas as pd
import matplotlib.pyplot as plt

def read_covid_data(directory):
    all_covid_data = []
    for root, _, files in os.walk(directory):
        for file in files:
            if file.endswith(".json"):
                file_path = os.path.join(root, file)
                with open(file_path, "r") as json_file:
                    data = json.load(json_file)
                    all_covid_data.append(data)
    return all_covid_data
```



```python
def calculate_statistics(covid_data):
    statistics = []
    for data in covid_data:
        confirmed_cases = data["confirmed_cases"]["total"]
        deaths = data["deaths"]["total"]
        recovered = data["recovered"]["total"]
        active_cases = confirmed_cases - deaths - recovered

        statistics.append({
            "Country": data["country"],
            "Total Confirmed Cases": confirmed_cases,
            "Total Deaths": deaths,
            "Total Recovered Cases": recovered,
            "Total Active Cases": active_cases,
        })
    return statistics
```

```python
def generate_summary_report(statistics):
    with open("covid19_summary.json", "w") as json_file:
        json.dump(statistics, json_file, indent=2)
```



```python
if __name__ == "__main__":
    covid_data_directory = "covid_data"

    # Step 1: Read COVID-19 data from all JSON files
    covid_data = read_covid_data(covid_data_directory)

    # Step 2: Calculate statistics for each country
    statistics = calculate_statistics(covid_data)

    # Step 3: Determine the top 5 countries with the highest and lowest confirmed cases
    sorted_statistics = sorted(statistics, key=lambda x: x["Total Confirmed Cases"], reverse=True)
    top_5_highest_cases = sorted_statistics[:5]
    top_5_lowest_cases = sorted_statistics[-5:][::-1]

    # Step 4: Generate and save the summary report
    generate_summary_report(statistics)
```