1. **What Are APIs?**
   * **API (Application Programming Interface)**:
     + Allows interaction between different software applications.
     + In DevOps, APIs are commonly used for monitoring tools, cloud services, CI/CD pipelines, etc.
   * APIs use HTTP methods:
     + GET: Retrieve data.
     + POST: Send data to the server.
     + PUT: Update existing data.
     + DELETE: Remove data.
2. **Python HTTP Libraries**:
   * **requests** library: Simplifies HTTP operations.
   * Install it using:

bash

Copy code

pip install requests

1. **JSON Handling**:
   * Most APIs return data in JSON format.
   * Use the json module to parse and handle JSON data.

**Practical Exercises**

**1. Make a GET Request**

* Objective: Fetch data from a public API.
* Script:

python

Copy code

import requests

url = "https://jsonplaceholder.typicode.com/posts/1"

response = requests.get(url)

if response.status\_code == 200:

print("Response Data:")

print(response.json())

else:

print(f"Failed to fetch data. Status Code: {response.status\_code}")

* Output:

css

Copy code

Response Data:

{'userId': 1, 'id': 1, 'title': 'Sample Title', 'body': 'Sample Body'}

**2. Parse JSON Data**

* Objective: Extract specific information from the API response.
* Script:

python

Copy code

import requests

url = "https://jsonplaceholder.typicode.com/users/1"

response = requests.get(url)

if response.status\_code == 200:

user\_data = response.json()

print(f"Name: {user\_data['name']}")

print(f"Email: {user\_data['email']}")

print(f"Address: {user\_data['address']['city']}, {user\_data['address']['street']}")

* Output:

makefile

Copy code

Name: Leanne Graham

Email: Sincere@april.biz

Address: Gwenborough, Kulas Light

**3. Make a POST Request**

* Objective: Send data to an API.
* Script:

python

Copy code

import requests

url = "https://jsonplaceholder.typicode.com/posts"

data = {

"title": "DevOps with Python",

"body": "Learning APIs in Python for DevOps.",

"userId": 1

}

response = requests.post(url, json=data)

if response.status\_code == 201:

print("Data posted successfully!")

print("Response:", response.json())

else:

print(f"Failed to post data. Status Code: {response.status\_code}")

* Output:

css

Copy code

Data posted successfully!

Response: {'id': 101, 'title': 'DevOps with Python', 'body': 'Learning APIs in Python for DevOps.', 'userId': 1}

**4. Handle Query Parameters**

* Objective: Send query parameters with a GET request.
* Script:

python

Copy code

import requests

url = "https://jsonplaceholder.typicode.com/posts"

params = {"userId": 1}

response = requests.get(url, params=params)

if response.status\_code == 200:

posts = response.json()

print(f"User 1 has {len(posts)} posts:")

for post in posts[:5]:

print(f"- {post['title']}")

* Output:

diff

Copy code

User 1 has 10 posts:

- Title 1

- Title 2

**5. Error Handling for API Calls**

* Objective: Manage errors like timeouts or invalid responses.
* Script:

python

Copy code

import requests

url = "https://jsonplaceholder.typicode.com/invalid\_endpoint"

try:

response = requests.get(url, timeout=5)

response.raise\_for\_status()

print(response.json())

except requests.exceptions.HTTPError as http\_err:

print(f"HTTP error occurred: {http\_err}")

except requests.exceptions.RequestException as err:

print(f"Other error occurred: {err}")

except requests.exceptions.Timeout:

print("The request timed out.")

**6. Integrate APIs into DevOps**

* **Scenario**: Fetch weather data for server location.
* Use OpenWeatherMap API (requires a free API key from [OpenWeatherMap](https://openweathermap.org/)).
* Script:

python

Copy code

import requests

api\_key = "your\_api\_key\_here"

location = "London"

url = f"http://api.openweathermap.org/data/2.5/weather?q={location}&appid={api\_key}&units=metric"

response = requests.get(url)

if response.status\_code == 200:

weather\_data = response.json()

print(f"Weather in {location}: {weather\_data['weather'][0]['description']}")

print(f"Temperature: {weather\_data['main']['temp']}°C")

else:

print(f"Failed to fetch weather data. Status Code: {response.status\_code}")

**7. Challenge: Automate Server Monitoring**

* **Scenario**: Check server health using a dummy API.
* Script:

python

Copy code

import requests

servers = ["https://jsonplaceholder.typicode.com/posts/1",

"https://jsonplaceholder.typicode.com/posts/1000"]

def check\_server(url):

try:

response = requests.get(url, timeout=5)

response.raise\_for\_status()

return f"Server {url} is healthy. Status: {response.status\_code}"

except requests.exceptions.RequestException as e:

return f"Server {url} is down. Error: {e}"

for server in servers:

print(check\_server(server))

**8. Bonus: Use API Tokens for Authentication**

* **Scenario**: Access a protected API endpoint using tokens.
* Script:

python

Copy code

import requests

url = "https://api.example.com/secure-data"

token = "your\_api\_token\_here"

headers = {"Authorization": f"Bearer {token}"}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

print("Secure data:", response.json())

else:

print(f"Failed to fetch data. Status Code: {response.status\_code}")