1. What is an API?

Definition:  
An API (Application Programming Interface) is a set of rules and protocols that allows one software application to interact with another. It defines how requests should be made, what data formats to use, and how responses are delivered.

Real-world examples:

* Google Maps API: Lets developers embed maps and location services in their own apps or websites.
* Twitter API: Allows applications to access and interact with Twitter data (like posting tweets or reading timelines).

2. What is a REST API?

A REST API (Representational State Transfer API) is a type of web API that follows specific architectural principles to enable communication between systems over HTTP.

Key principles:

* HTTP Methods: Uses standard methods like GET, POST, PUT, DELETE, etc., to perform operations on resources.
* Statelessness: Each request from a client to the server must contain all the information needed to understand and process the request. The server does not store client state.
* Resources: Everything is treated as a resource (e.g., a user, a product) and is accessed via a unique URL.

3. Common HTTP Methods in REST APIs

| Method | Purpose |
| --- | --- |
| GET | Retrieve data from the server (e.g., get a list of users). |
| POST | Create a new resource (e.g., add a new user). |
| PUT | Update an existing resource completely (e.g., replace a user's data). |
| DELETE | Remove a resource (e.g., delete a user). |
| PATCH | Partially update a resource (e.g., change just the user's email). |

4. What is JSON? Why is it commonly used in APIs?

JSON (JavaScript Object Notation) is a lightweight data format used for storing and transporting data, structured in key-value pairs.

Why it's commonly used in APIs:

* Easy to read and write for humans.
* Simple to parse and generate by machines.
* Supported by most programming languages.
* Compact and efficient for data exchange over networks.

2) Part 2: Making API Requests with Python

Task 3: GET Request (Fetching Data)

Use the JSONPlaceholder API (a free fake API for testing):

* Endpoint: https://jsonplaceholder.typicode.com/posts

Instructions:

1. Write a Python script to fetch all posts from the API.
2. Print the response status code.
3. Print the first post in the response (JSON format).

Expected Output:

Status Code: 200

First Post: {'userId': 1, 'id': 1, 'title': '...', 'body': '...'}

import requests # Import the requests library

# 1. Send a GET request to fetch all posts

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

response = requests.get(url)

# 2. Print the response status code

print("Status Code:", response.status\_code)

# 3. Print the first post in the response (in JSON format)

if response.status\_code == 200:

posts = response.json() # Convert response to JSON

print("First Post:")

print(posts[0]) # Print the first post

else:

print("Failed to fetch data.")

output :

Status Code: 200

First Post:

{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'quia et suscipit...'}

**Task 4: POST Request (Sending Data)**

**Instructions:**

1. Use the same API to create a new post.
2. Send a JSON payload with:

{

"title": "New Post",

"body": "This is a test post.",

"userId": 1

}

1. Print the response (should include the new post with an ID).

**Expected Output:**

New Post: {'title': 'New Post', 'body': 'This is a test post.', 'userId': 1, 'id': 101}

import requests

# API endpoint for posts

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

# JSON payload to send

payload = {

"title": "New Post",

"body": "This is a test post.",

"userId": 1

}

# 1. Send POST request with JSON payload

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

# 2. Parse and print the response JSON (new post with ID)

if response.status\_code == 201: # 201 Created

new\_post = response.json()

print("New Post:", new\_post)

else:

print("Failed to create a new post. Status Code:", response.status\_code)

Task 5: Error Handling (Non-existent Endpoint)

import requests

# Try to access a non-existent endpoint

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

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

print("Request successful.")

else:

print(f"Error: Request failed with status code {response.status\_code}. Endpoint may not exist.")

**Task 6: Fetch Weather Data from OpenWeatherMap**

**Note:** You need to sign up on OpenWeatherMap and get your free API key. Replace 'YOUR\_API\_KEY' with your actual API key in the script below.

import requests

# Replace with your actual API key from OpenWeatherMap

API\_KEY = 'YOUR\_API\_KEY'

city = 'London' # Change to your preferred city

# Build the API URL with city, API key, and metric units for Celsius

url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric"

# Send GET request to fetch weather data

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

# Extract required data

temperature = data['main']['temp']

weather\_description = data['weather'][0]['description']

humidity = data['main']['humidity']

# Print the weather information

print(f"Current weather in {city}:")

print(f"Temperature: {temperature}°C")

print(f"Description: {weather\_description}")

print(f"Humidity: {humidity}%")

else:

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