**Part 1: Understanding APIs and REST APIs**

**Task 1: Research & Short Answers**

1. **What is an API?**
   * Provide a definition and give two real-world examples.

**Ans:- Definition:**  
An **API (Application Programming Interface)** is a set of rules and protocols that allow different software applications to communicate with each other. It defines how requests and responses should be formatted and processed.

**Real-World Examples:**

* **Google Maps API:**
* **Twitter API:**

1. **What is a REST API?**
   * Explain the key principles (HTTP methods, statelessness, resources).

Ans:- A **REST API (Representational State Transfer API)** is a type of web API that uses standard HTTP methods and follows REST principles for building scalable and maintainable web services.

**Key Principles:**

* **HTTP Methods:** REST APIs use HTTP methods like GET, POST, PUT, DELETE to perform operations on resources.
* **Statelessness:** Each request from a client must contain all the information the server needs to understand and respond. The server does not store any state about the client session.
* **Resources:** Data is represented as resources, typically accessed via URLs .

1. **List the common HTTP methods used in REST APIs and their purposes.**
   * (GET, POST, PUT, DELETE, PATCH)

Ans:-

| **Method** | **Purpose** |
| --- | --- |
| **GET** | Retrieve data from the server (read). |
| **POST** | Submit data to the server to create a new resource. |
| **PUT** | Update an existing resource or create if it doesn’t exist (replaces entire resource). |
| **DELETE** | Remove a resource from the server. |
| **PATCH** | Partially update an existing resource (only the changed data). |

1. **What is JSON? Why is it commonly used in APIs?**

Ans:- **JSON (JavaScript Object Notation)** is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate.

**Why it’s commonly used in APIs:**

* It’s **language-independent**, but widely supported.
* It’s **lightweight and efficient** for transmitting data.

**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
* response = requests.get("https://jsonplaceholder.typicode.com/posts")
* print("Status Code:", response.status\_code)
* posts = response.json()
* print("First Post:", posts[0])

**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

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

payload = {

    "title": "New Post",

    "body": "This is a test post.",

    "userId": 1

}

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

print("New Post: ",response.json())

**Task 5: Error Handling & Authentication**

**Instructions:**

1. Try accessing a non-existent endpoint (e.g., https://jsonplaceholder.typicode.com/nonexistent).
2. Handle the error (check status code, print an error message if request fails).

**Example Code:**

response = requests.get("https://jsonplaceholder.typicode.com/nonexistent")

import requests

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

response = requests.get(url)

if response.status\_code == 200:

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

else:

    print(f"Error: Received status code {response.status\_code} - Endpoint not found.")

**Part 4: Challenge Task (Real-World API Usage)**

**Task 6: Fetch Weather Data (OpenWeatherMap API)**

1. Sign up for a free API key at [OpenWeatherMap](https://openweathermap.org/api).
2. Fetch the current weather for a city of your choice.
3. Print temperature, weather description, and humidity.

**Endpoint:**

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

**Submission:** Submit your Python script.

import requests

# Step 1: Your API Key (replace this with yours if needed)

api\_key = "1e9a435ccaf4b8003e56cb3d97308b41"

# Step 2: Input city name

city = input("Enter the name of city to check its current weather: ")

# Step 3: Build the URL

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

# Step 4: Make GET request to the API

response = requests.get(url)

data = response.json()

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

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

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

print(f"\nWeather in {city}:")

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

print(f"Description: {description.title()}")

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