📘 Lecture 18: APIs (Application Programming Interfaces) in Python

---

🔷 What is an API?

- API = **Bridge** between two systems
- Allows communication between **client** and **server**
- Analogy:
  You = Client
  Waiter = API
  Kitchen = Server

🔷 Key Components

Component	Role
Client	Sends request (e.g., browser, app)
API	Middleman between client & server
Server	Stores data, processes request

🛠️ How APIs Work

1. **Request** → Client asks for data/service
2. **Processing** → Server handles request
3. **Response** → Data returned via API

🐍 Using APIs in Python

✅ Install requests:

```
nginx
CopyEdit
pip install requests
```

✅ HTTP Methods

Method	Purpose
GET	Retrieve data
POST	Send data / Create item
PUT	Update existing data
DELETE	Remove data

🔷 Example: GET Request

```
python
CopyEdit
import requests

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

print(response.status_code)      # 200 = OK
print(response.json())           # JSON data
```

🔷 Example: POST Request

```
python
CopyEdit
import requests

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

data = {
    "title": "New Post",
    "body": "This is a new post.",
    "userId": 1
}

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

print(response.status_code)      # 201 = Created
print(response.json())
```

🔷 API Endpoint = Address of API

Example:
`https://api.weather.com/forecast`

- Different endpoints = Different services
- Think of them as "house addresses" on the internet

◈ Types of APIs

Type	Description
REST	🌍 Common, uses JSON over HTTP
SOAP	□ Old, uses XML, more complex
GraphQL	📊 Flexible, query-specific data

◈ This lecture focused on REST APIs

⚠ Error Handling in APIs

```
python
CopyEdit
import requests

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

if response.status_code == 200:
    print("Success!")
else:
    print(f"Error {response.status_code}")
```

Common Status Codes:

Code	Meaning
200	OK
201	Created
404	Not Found

Code	Meaning
500	Server Error

□ Exercises

- ✅ Make a **GET** request using `requests`
- ✅ Practice a **POST** request
- ✅ Implement **error handling**
- ✅ Explore real-world APIs (weather, maps, etc.)

□ Summary Table

Step	Description
Request	Client sends request using `requests`
Processing	Server processes the request
Response	Server sends data back via API

✅ Final Thoughts

- APIs connect modern software systems
- `requests` is key for calling APIs in Python
- Error handling = 💪 Stability of your programs
- Practice with real APIs to master the skill

🏁 **End of Lecture**
🗄️ Next: Dive deeper into **API authentication & working with headers**
Till then, Allah Hafiz!
