**📌 Python requests Library – What You Need to Know**

**1. What is the requests library?**

**Answer:**  
The requests library is a popular Python package used to send **HTTP requests** and interact with APIs (GET, POST, PUT, DELETE, etc.) in a simple and user-friendly way.

Example:

import requests

response = requests.get("https://api.github.com")

print(response.status\_code)

**2. How do you make a GET request?**

import requests

response = requests.get("https://jsonplaceholder.typicode.com/posts/1")

print(response.json()) # JSON response

**3. How do you pass query parameters in a GET request?**

params = {"q": "python", "sort": "stars"}

response = requests.get("https://api.github.com/search/repositories", params=params)

print(response.url) # shows full URL with query params

**4. How do you send data with a POST request?**

data = {"username": "meghana", "password": "secret"}

response = requests.post("https://httpbin.org/post", data=data)

print(response.json())

**5. How do you send JSON data with a POST request?**

json\_data = {"name": "Meghana", "role": "QA Engineer"}

response = requests.post("https://httpbin.org/post", json=json\_data)

print(response.json())

**6. How do you send headers with a request?**

headers = {"Authorization": "Bearer your\_token\_here"}

response = requests.get("https://api.github.com/user", headers=headers)

print(response.status\_code)

**7. How do you handle response status codes?**

response = requests.get("https://api.github.com")

if response.status\_code == 200:

print("Success")

elif response.status\_code == 404:

print("Not Found")

**8. What are response.text, response.json(), and response.content?**

* response.text → Returns response as a string (decoded text).
* response.json() → Returns parsed JSON (dict).
* response.content → Returns raw bytes.

Example:

res = requests.get("https://jsonplaceholder.typicode.com/posts/1")

print(res.text) # raw text

print(res.json()) # dictionary

**9. How do you handle timeouts?**

try:

res = requests.get("https://httpbin.org/delay/5", timeout=3)

except requests.exceptions.Timeout:

print("Request timed out!")

**10. How do you handle errors (exceptions)?**

try:

res = requests.get("https://api.github.com/unknown", timeout=5)

res.raise\_for\_status() # raises HTTPError if not 200

except requests.exceptions.HTTPError as e:

print("HTTP Error:", e)

**11. How do you send authentication with requests?**

* **Basic Auth**

from requests.auth import HTTPBasicAuth

response = requests.get("https://api.github.com/user", auth=HTTPBasicAuth("user", "pass"))

* **Bearer Token (common in APIs)**

headers = {"Authorization": "Bearer <your\_token>"}

response = requests.get("https://api.github.com/user", headers=headers)

**12. How do you upload files?**

files = {"file": open("test.txt", "rb")}

response = requests.post("https://httpbin.org/post", files=files)

print(response.json())

**13. How do you send cookies?**

cookies = {"session\_id": "12345"}

response = requests.get("https://httpbin.org/cookies", cookies=cookies)

print(response.json())

**14. How do you maintain a session across multiple requests?**

session = requests.Session()

session.get("https://httpbin.org/cookies/set/sessionid/12345")

res = session.get("https://httpbin.org/cookies")

print(res.json())

**15. How do you handle redirects?**

res = requests.get("http://github.com", allow\_redirects=True)

print(res.url) # final redirected URL

**16. How do you send custom SSL certificates or ignore SSL verification?**

# Ignore SSL Verification (not safe for prod)

response = requests.get("https://self-signed.badssl.com/", verify=False)

**17. How do you download a file with requests?**

url = "https://www.example.com/sample.pdf"

res = requests.get(url, stream=True)

with open("sample.pdf", "wb") as f:

for chunk in res.iter\_content(1024):

f.write(chunk)

**18. What are common exceptions in requests?**

* requests.exceptions.Timeout
* requests.exceptions.ConnectionError
* requests.exceptions.HTTPError
* requests.exceptions.TooManyRedirects

**19. How do you retry a failed request?**

from requests.adapters import HTTPAdapter

from requests.packages.urllib3.util.retry import Retry

session = requests.Session()

retry = Retry(total=3, backoff\_factor=1)

adapter = HTTPAdapter(max\_retries=retry)

session.mount("http://", adapter)

session.mount("https://", adapter)

res = session.get("https://httpbin.org/status/500")

**20. What are the HTTP methods supported by requests?**

* GET
* POST
* PUT
* PATCH
* DELETE
* HEAD
* OPTIONS