Perfect 🚀 Here's your **colorful study notes** version of the `` **module in Python** — clean, structured, and beginner-friendly 🚃

🌐 Python `requests` Module – Student Notes

🔹 1. What is it?

- A **third-party Python library** for making **HTTP requests** (GET, POST, PUT, DELETE, etc.).
- Used for working with **APIs, websites, and web services**.
- Much simpler than `urllib`.

First install it:

```
pip install requests
```

🔹 2. Importing

```
import requests
```

🔹 3. Basic Usage

✅ GET Request

```
import requests

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

print(response.status_code)   # 200 = OK
print(response.text)          # Raw response text
print(response.json())        # JSON → Python dict
```

✅POST Request (sending data)

```
data = {"title": "foo", "body": "bar", "userId": 1}
response = requests.post("https://jsonplaceholder.typicode.com/posts",
json=data)

print(response.json())
```

✅Sending Query Parameters

```
payload = {"userId": 1}
response = requests.get("https://jsonplaceholder.typicode.com/posts",
params=payload)

print(response.url)   # Final URL with ?userId=1
print(response.json())
```

✅Sending Headers

```
headers = {"User-Agent": "MyApp/1.0"}
response = requests.get("https://httpbin.org/headers", headers=headers)

print(response.json())
```

✅Timeout Example

```
response = requests.get("https://httpbin.org/delay/3", timeout=2)
```

Raises error if request takes longer than 2 seconds.

🔹 4. Common Methods in `requests`

Method	Use
`get()`	Retrieve data
`post()`	Send data

Method	Use
`put()`	Update data
`delete()`	Remove data
`head()`	Get headers only
`options()`	Get allowed methods

🔹 5. Checking Responses

```
print(response.status_code)  # 200, 404, 500
print(response.ok)           # True if < 400
print(response.headers)      # Response headers
print(response.content)      # Raw bytes (good for images/files)
```

🔹 6. Downloading a File

```
url = "https://www.example.com/image.png"
response = requests.get(url)

with open("image.png", "wb") as file:
    file.write(response.content)
```

🔹 7. Authentication Example

```
from requests.auth import HTTPBasicAuth

response = requests.get("https://api.github.com/user",
auth=HTTPBasicAuth("username", "token"))
print(response.json())
```

🔝 Summary

- `requests` = makes web/API calls simple.
- Supports **GET, POST, PUT, DELETE**, authentication, headers, params.

- Great for **web scraping, APIs, downloading files, and automation**.
