PYTHON

requests

FULL COURSE

HTTP • APIs • JSON



LONG VIDEO • 11 MIN

PYTHON • REQUESTS FULL COURSE

Python requests Full Course

Web APIs, JSON, authentication, sessions, retries and downloads

FREE FULL COURSE • MyWebUniversity.com



Wahidullah Lutfy
Founder, MyWebUniversity • CS Teacher

REQUESTS

What You Will Learn

How HTTP works
Requests, responses, methods and status codes.

GET and parameters
Query strings, headers and reading responses.

JSON APIs
Reading and sending JSON data.

Sending data
POST forms, JSON bodies and file uploads.

Robust code
Timeouts, errors, retries, sessions and authentication.

Real projects
Downloading files and building an API client.

Python requests • Full Course

PART 1 · HTTP AND GET REQUESTS

REQUESTS

How HTTP Works

Request
Method + URL + headers (+ body). "Please GET me this page."

Response
Status code + headers + body (HTML, JSON, an image…).

Methods
GET reads, POST creates, PUT/PATCH update, DELETE removes.

Status codes
2xx success • 3xx redirect • 4xx your mistake • 5xx server problem.

REQUESTS

Your First GET Request

Python 3.13

```
$ pip install requests
>>> import requests
>>> r = requests.get("https://api.github.com/users/octocat", timeout=10)


>>> r.status_code, r.ok, r.headers["Content-Type"]
(200, True, 'application/json; charset=utf-8')

>>> data = r.json()
>>> data["name"], data["public_repos"]
('The Octocat', 8)
>>> r.text[:40], r.elapsed
```

The Response object

status_code, ok, headers, text, content, json(), url, elapsed.

text vs content vs json()

text is a string, content is raw bytes, json() is Python data.

Always set timeout

Without it, a slow server can freeze your program forever.

REQUESTS

Query Parameters and Headers

Python 3.13

```
>>> params = {"q": "language:python", "sort": "stars", "per_page": 3}
>>> r = requests.get("https://api.github.com/search/repositories",
params=params, timeout=10)
>>> r.url



>>> [repo["full_name"] for repo in r.json()["items"]]
['public-apis/public-apis', 'donnemartin/system-design-primer',
'TheAlgorithms/Python']

>>> headers = {"User-Agent": "MyWebUniversity-course/1.0", "Accept":
"application/json"}
>>> requests.get(url, headers=headers, timeout=10)
```

params=

A dictionary becomes a correctly encoded query string.

headers=

Extra information: who you are, what format you accept, your API key.

Read the API docs

They list every parameter, header and limit the API supports.

Python requests • Full Course

PART 2 · SENDING DATA AND JSON

REQUESTS

POST: Forms, JSON and Files

Python 3.13

```
>>> r = requests.post("https://httpbin.org/post", data={"name": "Wahid",
"course": "python"}, timeout=10)
>>> r.json()["form"]
{'course': 'python', 'name': 'Wahid'}

>>> r = requests.post("https://httpbin.org/post", json={"title": "Hello",
"tags": ["api", "python"]}, timeout=10)
>>> r.json()["json"]["tags"]
['api', 'python']

>>> with open("report.pdf", "rb") as f:
...     r = requests.post("https://httpbin.org/post", files={"file": f},
timeout=30)
```

data=
Sends form fields, like an HTML form.

json=
Converts a dict to JSON and sets the Content-Type header for you.

files=
Uploads files. Open them in binary mode: "rb".

REQUESTS

Errors, Timeouts and Exceptions

safe_get.py — Python 3.13

```
import requests
def safe_get(url):
   try:
       r = requests.get(url, timeout=(3, 10))   # connect, read
       r.raise_for_status()
       return r.json()

   except requests.Timeout:
       print("The server took too long.")
   except requests.HTTPError as e:
       print("Bad status:", e.response.status_code)
   except requests.RequestException as e:
       print("Network problem:", e)
```

raise_for_status()
Turns 4xx and 5xx responses into exceptions, so errors never pass silently.

timeout=(3, 10)
3 seconds to connect, 10 seconds to receive the data.

Exception family
Timeout, ConnectionError and HTTPError all inherit from RequestException.

Python requests • Full Course

PART 3 · PROFESSIONAL TECHNIQUES

REQUESTS

Sessions and Authentication

client.py — Python 3.13

```
import os, requests
s = requests.Session()
s.headers.update({"Authorization": f"Bearer {os.environ['API_TOKEN']}",
                  "User-Agent": "MyWebUniversity-course/1.0"})



me = s.get("https://api.example.com/me", timeout=10).json()
orders = s.get("https://api.example.com/orders", params={"page": 1},
timeout=10)

r = requests.get("https://httpbin.org/basic-auth/user/pass",
auth=("user", "pass"), timeout=10)
# r.status_code == 200
```

Session()

Reuses connections, cookies and headers across many requests. Faster and cleaner.

Bearer tokens

Most modern APIs use an Authorization: Bearer <token> header.

Keep secrets secret

Read tokens from environment variables. Never commit them to Git.

REQUESTS

Retries and Rate Limits

client.py — Python 3.13

```
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry
retry = Retry(total=3, backoff_factor=1, status_forcelist=[429, 500, 502,
503, 504])
s.mount("https://", HTTPAdapter(max_retries=retry))


r = s.get(url, timeout=10)
if r.status_code == 429:
   wait = int(r.headers.get("Retry-After", 60))
```

Automatic retries

Retry temporary failures 3 times, waiting longer each time (backoff).

429 Too Many Requests

You hit the rate limit. Slow down, and respect Retry-After.

Be a good citizen

Cache results, add small delays, and follow the API terms.

REQUESTS

Mini Project: Download Large Files

download.py — Python 3.13

```
from pathlib import Path
import requests
url = "https://www.python.org/ftp/python/3.13.1/python-3.13.1-amd64.exe"
out = Path("downloads") / url.split("/")[-1]
out.parent.mkdir(exist_ok=True)



with requests.get(url, stream=True, timeout=30) as r:
   r.raise_for_status()
   total = int(r.headers.get("Content-Length", 0))
   with open(out, "wb") as f:
       for chunk in r.iter_content(chunk_size=1024 * 256):
           f.write(chunk)

print(f"Saved {out} ({total / 1e6:.1f} MB)")
```

stream=True
Do not load the whole file into memory; read it piece by piece.

iter_content()
Write each chunk to disk as it arrives. Works for any file size.

Content-Length
The total size, so you can show progress to the user.

REQUESTS

Recap and Practice

Remember
get/post, params, headers, json(), timeout, raise_for_status, Session.

Try it yourself
Build a tiny weather or GitHub client that prints a clean summary.

Watch out
No secrets in code. Respect rate limits and terms of use.