API Reference Sheet – Unit 6, Lesson 4

Your Programs Meet the Internet 🌐 📈

---

1 What is an API?

API = **Application Programming Interface**
A way for **programs to talk to other programs** over the internet.

- **Websites** are designed for **people** (you see charts, forms, buttons).
- **APIs** are designed for **code** (your program gets raw data, usually JSON).
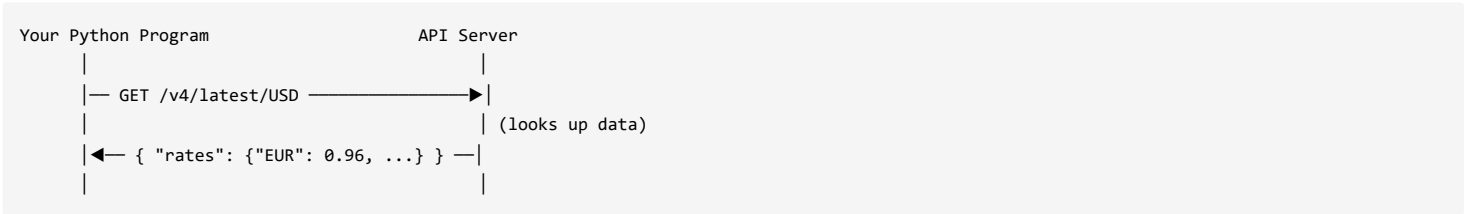
> 💡 **Restaurant Analogy**
>
> - You → **Customer**
> - API → **Waiter**
> - Server → **Kitchen**
>   You never go into the kitchen yourself – the waiter brings your order and the food.

---

2 The Request–Response Cycle

Every API interaction follows the same pattern:

1. **Your program sends a REQUEST** to a specific URL.
2. **The server processes** the request.
3. **The server sends back a RESPONSE** – usually containing JSON data.

```
Your Python Program              API Server
     |                               |
     |— GET /v4/latest/USD ─────────▶|
     |                               | (looks up data)
     |◀── { "rates": {"EUR": 0.96, ...} } —|
     |                               |
```

---

3 Anatomy of an API URL

```
https://api.exchangerate-api.com/v4/latest/USD
```

Part	Example	Meaning
Base URL	`https://api.exchangerate-api.com`	Which server to talk to
Version	`/v4`	Which version of the API
Endpoint	`/latest`	What kind of data (latest rates)
Parameter	`/USD`	Specific item (currency code)

> 🔗 Think of it like a file path: **server / category / specific item**

4️⃣ Query Parameters

Used to **customize** the request – extra key=value pairs after a `?` .

```
https://api.open-meteo.com/v1/forecast?latitude=40.71&longitude=-74.00&current_weather=true
```

Symbol	Meaning	Example
`?`	Starts the parameter list	`?latitude=40.71`
`&`	Separates multiple parameters	`&longitude=-74.00`

> 💡 **Parameters = function arguments!**
> You pass arguments to a Python function – you pass parameters to an API.

5️⃣ HTTP Methods (Verbs)

Method	Purpose	Real-world example
GET	Retrieve data	"Give me the stock price" ✅ *most common*
POST	Send / create data	"Place this trade order"
PUT	Update data	"Change my account settings"
DELETE	Remove data	"Cancel this order"

*For this course we focus on **GET** requests.*

6️⃣ HTTP Status Codes – "Did it work?"

Every response includes a **3-digit status code**.

Code	Category	Meaning
200	✅ Success	OK – here's your data!
201	✅ Success	Created – new record saved
400	❌ Client error	Bad request – your fault
401	❌ Client error	Unauthorized – need an API key
404	❌ Client error	Not found – that resource doesn't exist
429	❌ Client error	Too many requests – slow down!
500	💥 Server error	Internal server error – not your fault

> 🛡️ **Always check the status code before using the data!**

7️⃣ JSON – The Language of APIs

APIs send data in **JSON** – exactly what you learned in Lesson 3!

```
{
  "base": "USD",
  "date": "2025-02-10",
  "rates": {
    "EUR": 0.96,
    "GBP": 0.80,
    "JPY": 151.85
  }
}
```

JSON → Python (Lesson 3 recap):

```
import json
json_string = '{"base": "USD", "rates": {"EUR": 0.96}}'
data = json.loads(json_string)
print(data["rates"]["EUR"])   # 0.96
```

✅ In **Lesson 5** the `requests` library does this automatically with `.json()`.

---

8 Free Public APIs – No Signup, No Key

We'll use these in class. **Paste the URLs into your browser to see JSON!**

API	What it gives	Example URL (try it!)
💱 **ExchangeRate-API**	Live currency rates	`https://api.exchangerate-api.com/v4/latest/USD`
🌧️ **Open-Meteo**	Weather forecasts	`https://api.open-meteo.com/v1/forecast?latitude=40.89&longitude=-74.04&current_weather=true&temperature_uni`
🌍 **REST Countries**	Country data (name, capital, currency, population)	`https://restcountries.com/v3.1/name/japan?fields=name,capital,currencies,population`
📋 **JSONPlaceholder**	Fake business data (posts, todos)	`https://jsonplaceholder.typicode.com/posts/1`
😎 **JokeAPI**	Programming jokes	`https://v2.jokeapi.dev/joke/Programming?type=single`

---

9 Try It Yourself – Browser Exploration URLs

Open each URL in your browser and examine the JSON response.

💱 Exchange Rates

- **Latest rates for USD**
  `https://api.exchangerate-api.com/v4/latest/USD`
- **Latest rates for EUR**
  `https://api.exchangerate-api.com/v4/latest/EUR`

- **Invalid currency (404)**

  `https://api.exchangerate-api.com/v4/latest/FAKEMONEY`

🌍 Countries

- **Japan (filtered fields)**

  `https://restcountries.com/v3.1/name/japan?fields=name,capital,currencies,population`
- **Germany (filtered fields)**

  `https://restcountries.com/v3.1/name/germany?fields=name,capital,currencies,population`
- **All data for Brazil**

  `https://restcountries.com/v3.1/name/brazil`

🌧 Weather (Open-Meteo)

- **Current weather in Hackensack, NJ (Fahrenheit)**

  `https://api.open-meteo.com/v1/forecast?latitude=40.89&longitude=-74.04&current_weather=true&temperature_unit=fahrenheit`
- **Current weather in Tokyo (Celsius – no unit param)**

  `https://api.open-meteo.com/v1/forecast?latitude=35.68&longitude=139.69&current_weather=true`
- **7-day forecast for London**

  `https://api.open-meteo.com/v1/forecast?latitude=51.51&longitude=-0.13&daily=temperature_2m_max,temperature_2m_min&timezone=GMT`

📋 JSONPlaceholder (Fake data)

- **Single post**

  `https://jsonplaceholder.typicode.com/posts/1`
- **All todos for user 1**

  `https://jsonplaceholder.typicode.com/users/1/todos`

## 😂 JokeAPI

- **A random programming joke**

  `https://v2.jokeapi.dev/joke/Programming?type=single`
- **A dark humour joke (if you dare)**

  `https://v2.jokeapi.dev/joke/Dark?type=single`

🔟 API Keys – Some APIs Require a "Password"

🔓 No Key Needed	🔐 Key Required (free tiers exist)
ExchangeRate-API (demo endpoint)	Alpha Vantage (stocks)
Open-Meteo	Google Maps
REST Countries	Stripe (payments)
JSONPlaceholder	Twitter/X
JokeAPI	Most business / financial APIs

> 📇 **API key = company badge** – identifies you and tracks usage.

🎯 Key Takeaways

- ✅ **API** = a program-friendly interface to get data from a server.
- ✅ **Request → Response** – you ask, the server answers with JSON.
- ✅ **URL parts**: base URL + endpoint + parameters.
- ✅ **Status codes**: 200 = success, 4xx = your mistake, 5xx = server problem.

✅ **JSON** is the bridge – APIs send it, Python reads it.

✅ **Browser = client** – typing an API URL in your browser makes a GET request!