

API : Understanding & Building APIs

Presented by: Elena Petrova

Agenda

1

What is an API?

2

Exploring APIs

3

All Together Now

What is an API?

Understanding APIs – The Basics

API = Application Programming Interface

Acts as a messenger that allows two applications to communicate.

APIs define rules for how software components should interact.

Real-World Analogy:

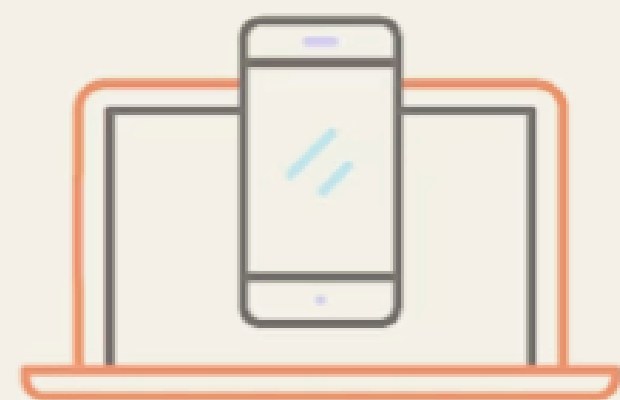
APIs are like a Waiter in a Restaurant

You (Client) order food.

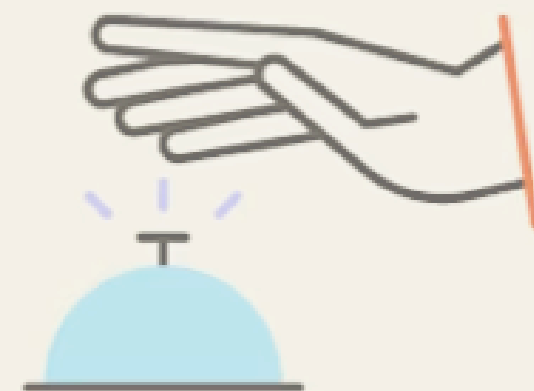
The waiter (API) takes your request to the kitchen (Server).

The kitchen prepares your dish (Processes the request).

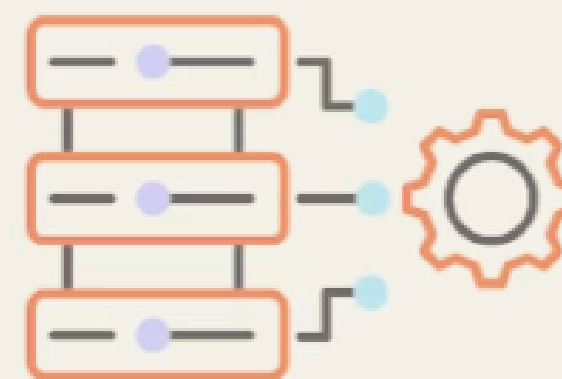
The waiter returns your meal (API Response).



Application



Request



Data source



API response



Why Are APIs Important?

Automation – APIs allow applications to talk to each other without human intervention.

Data Exchange – APIs enable apps to send & receive data efficiently.

Enhanced Functionality – Developers can integrate third-party services (e.g., Google Maps, Stripe).

Scalability – APIs allow businesses to expand digital services without reinventing the wheel.

Real-World Examples of APIs in Action

Application	APIs It Uses
Uber	Google Maps API + Payment API
Instagram	Facebook Graph API (fetching user data)
Weather Apps	OpenWeather API for real-time weather updates
YouTube	YouTube Data API for video analytics
Finance Apps	Stripe API for online payments

How APIs Work (Behind the Scenes)

Understanding the API Request-Response Cycle

✓ **Step 1:** The Client (User) Sends a Request

Example: A weather app requests today's temperature.

✓ **Step 2:** The API Receives the Request

The API acts as a middleman between the client and the server.

✓ **Step 3:** The Server Processes the Request

The server retrieves the required data (e.g., weather information).

✓ **Step 4:** The API Sends Back a Response

The API delivers the response in a format like JSON or XML.

✓ **Step 5:** The Client Displays the Data

The app updates with the latest weather information.

Example API Request in Python (Fetching Weather Data)

```
import requests

api_key = "YOUR_API_KEY"
city = "New York"
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric"

response = requests.get(url)
data = response.json()

print(f"Temperature in {city}: {data['main']['temp']}°C")
```

API Components & Key Terminology

Endpoint

The URL where the API service is hosted.

```
bash
```

```
https://api.openweathermap.org/data/2.5/weather
```

HTTP Methods


- **GET:** Retrieve data.
- **POST:** Send new data.
- **PUT:** Update existing data.
- **DELETE:** Remove data.

Parameters

Data sent along with the request to customize the response.

```
params = {  
    "q": "London",  
    "appid": "your_api_key",  
    "units": "metric"  
}
```

Response Codes



200 OK: Successful request.

401 Unauthorized: Invalid API key or permissions.

404 Not Found: Incorrect endpoint.

500 Internal Server Error: Problem on the server side.

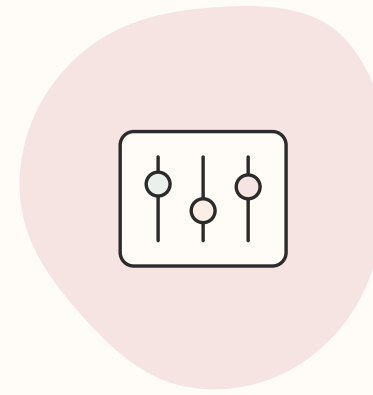
JSON (JavaScript Object Notation)

```
json

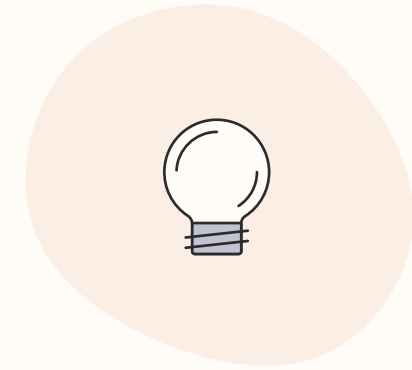
{
  "main": {
    "temp": 10.5,
    "humidity": 80
  },
  "weather": [
    {
      "description": "light rain"
    }
  ]
}
```

Exploring APIs Online

How to Explore APIs Online



Find API
Documentation



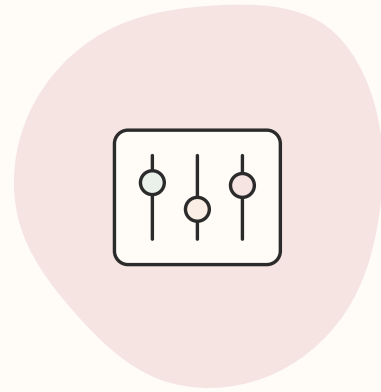
Examples of API
Documentation



Practice Exploring
APIs



Interactive API
Platforms



Visit the API provider's website to **understand**:

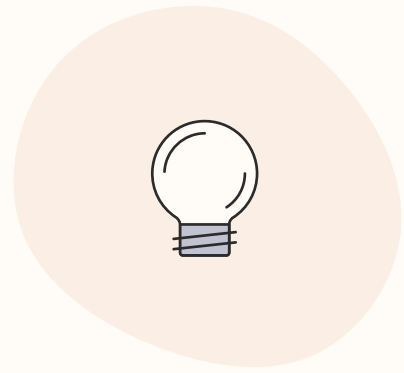
Find API
Documentation

Endpoints

Parameters

Authentication requirements

Rate limits



Examples of API
Documentation

Examples of API Documentation

OpenWeather API Docs:

<https://openweathermap.org/api>

YouTube Data API Docs:

<https://developers.google.com/youtube/v3>

Twitter API Docs:

<https://developer.twitter.com>



Practice Exploring
APIs

Start with free, open APIs like:

Public APIs List: <https://public-apis.io/>

Cat Facts API: Provides random cat facts.

Advice Slip API: Fetches life advice!




Interactive API
Platforms

Interactive API Platforms

RapidAPI: Find and test APIs from various providers.

Postman: Visualize and test API calls without writing code.

Swagger UI: View live documentation for RESTful APIs.



Using Tools to Explore APIs

Exploring APIs with User-Friendly
Tools

Why Use API Tools?

Simplifies testing and exploring API endpoints.

No need to write code for initial tests.

Visualize request-response cycles clearly.

Top API Tools:

Postman

- Allows you to send API requests easily.
- Add headers, parameters, and authentication tokens.
- Activity: Test OpenWeather API with Postman and analyze the JSON response.

Insomnia

- Similar to Postman but lighter and easier to use.
- Activity: Test any free public API (e.g., Cat Facts API).

Swagger UI

- Interactive API documentation tool.
- Explore endpoints in a browser with live testing capabilities.

RapidAPI Hub

- A marketplace for finding and testing APIs.
- Explore trending APIs, like JokeAPI or Dog Facts API.



Using Helper Libraries in Python

Simplify API Requests with Helper Libraries



Why Use Helper Libraries?

- Abstract away the complexity of constructing HTTP requests.
- Handle parameters, headers, and authentication seamlessly.
- Provide additional functionality like retries and error handling.

Don't publish your API key

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

# Set API credentials directly
SPOTIPY_CLIENT_ID = "e869ca961fc947158692bfd3db8d1a6b6"
SPOTIPY_CLIENT_SECRET = "807d294eb1da4760ab3dbf46fa0e0e0e"

client_credentials_manager = SpotifyClientCredentials(client_id=SPOTIPY_CLIENT_ID,
                                                       client_secret=SPOTIPY_CLIENT_SECRET)

# Authenticate Spotipy
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
```

Introducing the Project

Building a Real-World API-Powered Application



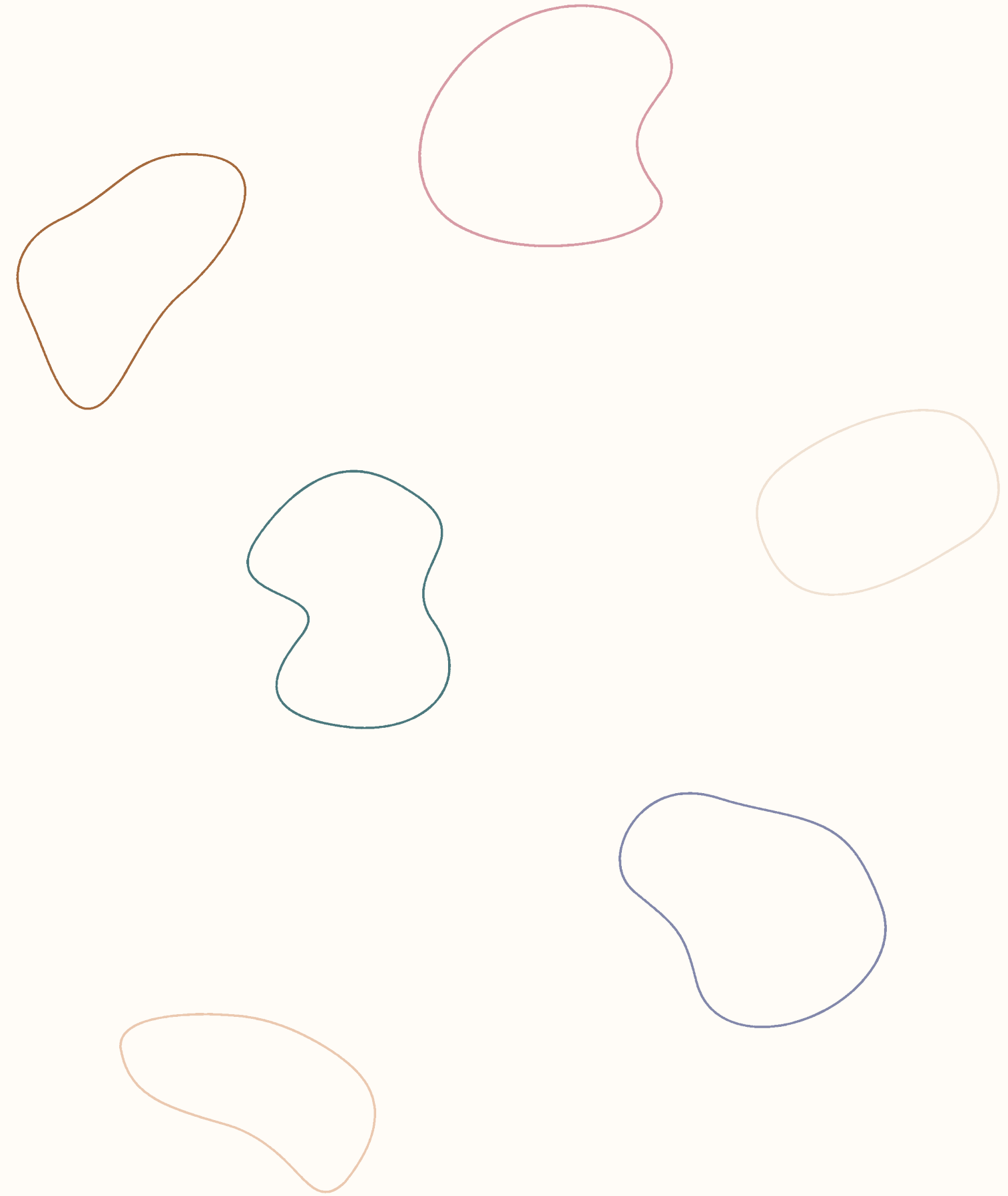
Apply everything we've learned to create a fully functional project using APIs.

Project Options:

- Weather Dashboard 🌤️ → Fetch real-time weather data using OpenWeather API.
- Spotify Playlist Analyzer 🎵 → Retrieve and analyze playlist data using the Spotify API.
- YouTube Comment Sentiment Analysis 💬 → Scrape and analyze YouTube comments.
- Custom API Mashup 🔗 → Combine multiple APIs to create something unique (e.g., weather + Spotify mood-based playlist).

Project Structure:

- Step 1: Fetch data from an API.
- Step 2: Process and clean the data.
- Step 3: Display the data in a user-friendly way (console, web app, or visualisation).
- Step 4: Handle errors and optimize API calls.





Wrap-Up & Takeaways

Final Thoughts & What We
Learned

1 APIs Allow Applications to Communicate

APIs act as a bridge between different applications.
Examples: OpenWeather, Spotify, YouTube, Twitter APIs.

2 How to Interact with APIs

Use cURL, Postman, or Python (requests) to make API requests.
API responses are typically in JSON format.

3 Building a Real-World API-Powered App

Fetching weather data from OpenWeather API.
Analyzing Spotify playlists with Spotipy.
Writing a custom Flask API and consuming it in JavaScript.

4 Error Handling & Debugging API Calls

Check HTTP response codes (200 OK, 401 Unauthorized, 404 Not Found).
Use try-except blocks in Python to handle API failures.



Q&A Session – Ask Me Anything About APIs!