Lesson 7: APIs

Application Programming Interfaces

What is an API?

- API stands for Application Programming Interface
- Allows software programs to communicate with each other
- Provides structured way to expose data and functionality from an application to other applications
- Allows other developers to access and integrate with an application without needing to understand complex implementation details

Spotify API:

- Access Spotify music catalog data
- Get info on artists, albums, tracks
- Manage user libraries and playlists
- Analyze audio features and metrics

Reddit API:

- Programmatic access to Reddit
- Get submissions and comments from subreddits
- Get metadata on Reddit content
- Create bot integrations

OpenWeatherMap API:

- Access current and historical weather data
- Get weather forecasts by city location
- Temperature, precipitation, humidity etc
- Weather alerts, wind speed and direction
- Multiple data formats and languages

Google Maps API:

- Integrate Google Maps into web/mobile apps
- Generate maps, routes, location searches
- Add markers, overlays and customizations
- Calculate distances and travel times

Public APIs vs Private APIs

Public vs private refers to access and visibility

Public APIs

- Available openly on the internet for **any developer** to use
- Just need to sign up and get an **API key** in order to access
- Examples: Twitter, Spotify, YouTube

Private APIs

- Access is **restricted** and requires authorization
- Only usable by **internal apps** or **trusted external partners**
- Useful for exposing internal data and functionality safely
- Examples: APIs for internal tooling, bank partnerships

REST vs SOAP

REST vs SOAP differ in underlying protocol, flexibility, and complexity

REST

- Uses simple HTTP requests to GET, POST, PUT, DELETE data
- Generally returns data in easy to parse formats like JSON
- Good for public-facing APIs, mobile applications
- Tend to be faster and more flexible

SOAP

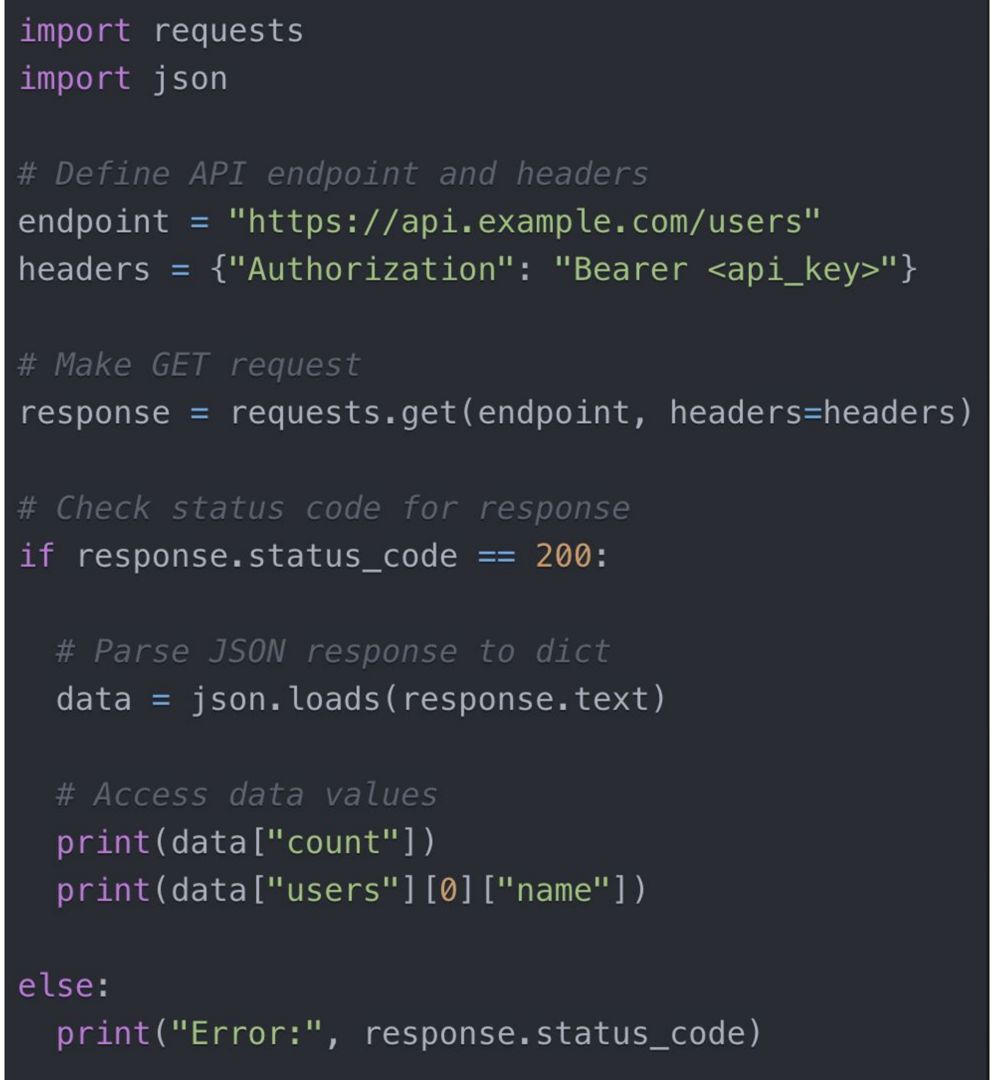
- Uses XML for requests and responses
- Follows stricter set of protocols and standards
- Defines rigid contract for security, transactions etc
- Useful for enterprise applications and internal systems
- Provides more built-in security features
- Can be more complex and verbose to implement

API Keys

- Unique string that acts like a password obtained by registering as a developer
- Lets the API identify the source of requests, enabling usage tracking and access limits
- Prevents abuse and provides API security
- Most APIs require adding key to request headers
    - Example header:
        - X-Api-Key: 237482378428234442
- Keep your key secret - don't share or expose!

Making API Requests

- HTTP requests allow client apps to connect to server APIs over the web
- Common HTTP methods used:
  - GET - Retrieve data
  - POST - Submit data to be processed
  - PUT - Update existing data
  - DELETE - Delete data
- The endpoint URL follows a structure defined in docs, like:
  - https://api.service.com/v1/users
- Can include query parameters to filter data
  - ?type=customer&region=eu
- Most APIs require authorization header with API key



```python
import requests
import json

# Define API endpoint and headers
endpoint = "https://api.example.com/users"
headers = {"Authorization": "Bearer <api_key>"}

# Make GET request
response = requests.get(endpoint, headers=headers)

# Check status code for response
if response.status_code == 200:

  # Parse JSON response to dict
  data = json.loads(response.text)

  # Access data values
  print(data["count"])
  print(data["users"][0]["name"])

else:
  print("Error:", response.status_code)
```

Parsing Responses

JSON format:

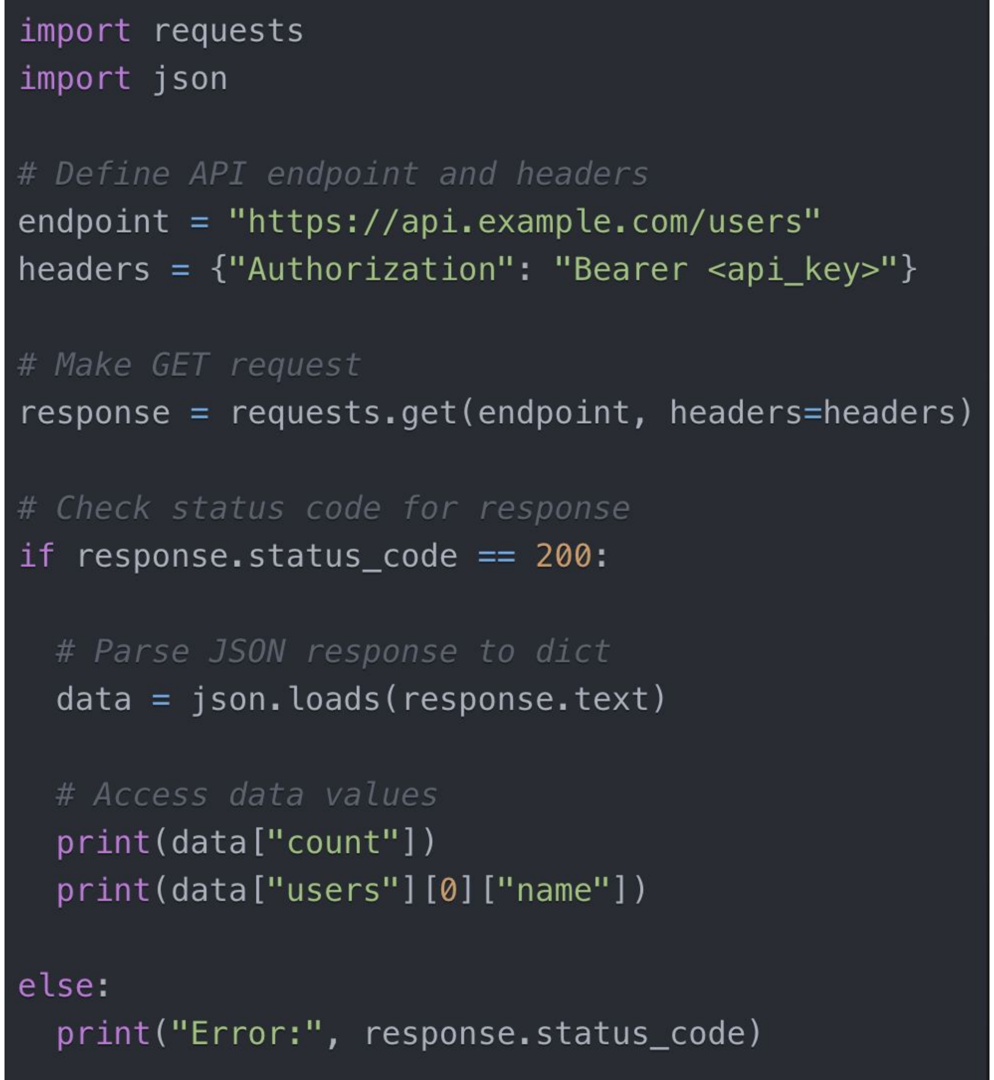
- Human-readable collection of key-value pairs
- Easy to access values programmatically

XML format:

- Nested structured format with opening and closing tags
- Can use libraries like ElementTree to parse
- Query to extract specific elements

Steps to parse:

1. Make request
2. Receive raw string response
3. Parse string into workable structure
4. Extract values needed for app



```python
import requests
import json

# Define API endpoint and headers
endpoint = "https://api.example.com/users"
headers = {"Authorization": "Bearer <api_key>"}

# Make GET request
response = requests.get(endpoint, headers=headers)

# Check status code for response
if response.status_code == 200:

  # Parse JSON response to dict
  data = json.loads(response.text)

  # Access data values
  print(data["count"])
  print(data["users"][0]["name"])

else:
  print("Error:", response.status_code)
```

Class Exercise: