API Connection

Estimated reading time: 10 minutes

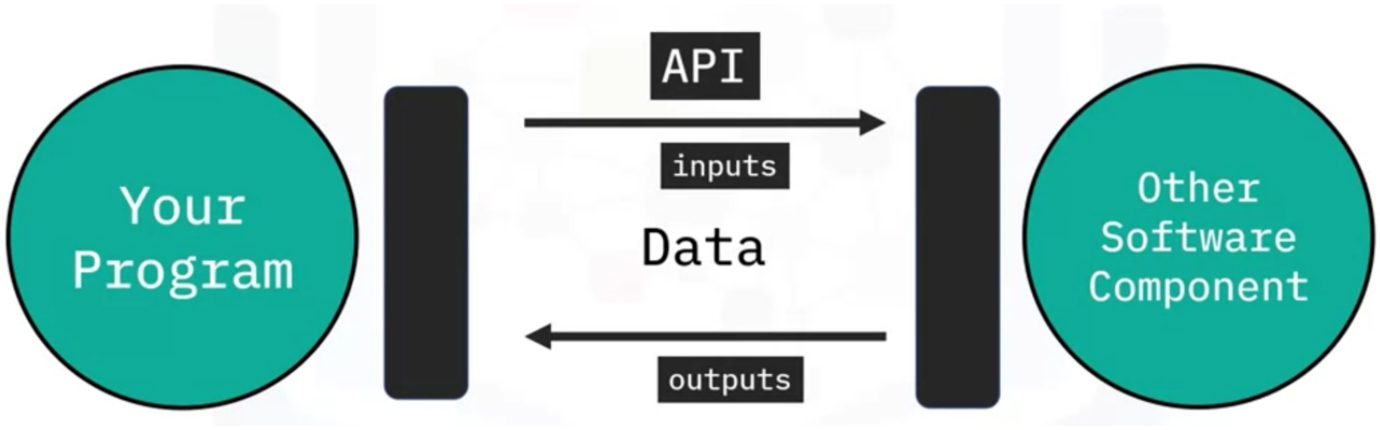
Objectives:

In this reading, you will learn about:

1. Basics of APIs
2. API structure overview
3. Overview of **nba_api** (National Basketball Association API)
4. Comparison between API and REST API

Introduction to APIs

An API acts as a bridge that allows one software application to interact with another, making it possible for them to share data, functionalities, or services without the need for the user to understand the internal workings of each application. APIs can enable integrating different systems, allowing them to work together seamlessly.



How do APIs work?

The operation of an **application programming interface (API)** involves various components that define interactions between software systems or applications. The following words and definitions describe the parts of a typical structure and functionality of an API.

Endpoint

- An endpoint is a specific uniform resource locator (URL) or uniform resource identifier (URI) representing a specific API function or resource. Each endpoint corresponds to a particular operation that the API can perform.

Request methods (HTTP methods)

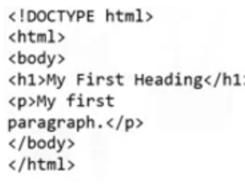
- APIs use HTTP methods to specify the type of action the client wants to perform on a given resource. Common HTTP methods include:
  - **GET**: Retrieve data from the server.
  - **POST**: Send data to the server to create a new resource.
  - **PUT or PATCH**: Update an existing resource on the server.
  - **DELETE**: Remove a resource on the server.

Request headers

Headers contain additional information about the request, such as the content type, authentication tokens, or other metadata.

Request body

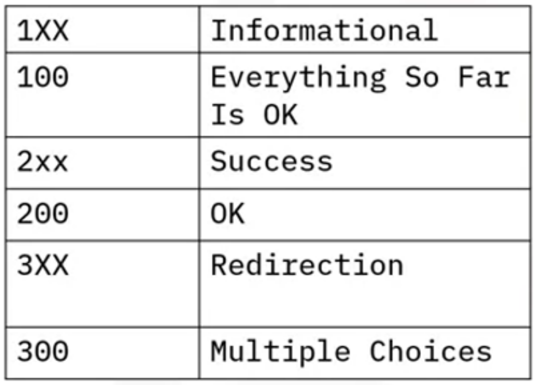
Sometimes, a request may include a body containing data that someone wants sent to the server. This request body is common in POST or PUT requests, where the client sends data to create or update a resource.



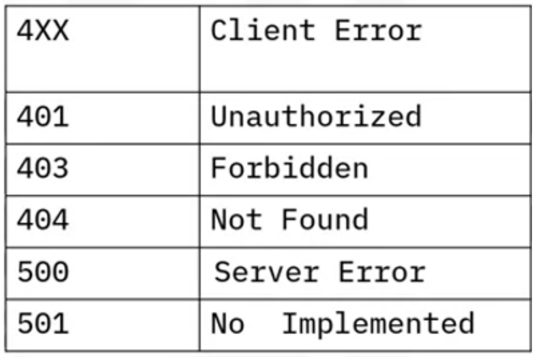
```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first
paragraph.</p>
</body>
</html>
```

Response status code

The server responds to a client request with an HTTP status code, indicating the success or failure of the operation.
These are typical status codes that you may encounter when interacting with a website:



1XX	Informational
100	Everything So Far Is OK
2xx	Success
200	OK
3XX	Redirection
300	Multiple Choices



4XX	Client Error
401	Unauthorized
403	Forbidden
404	Not Found
500	Server Error
501	No Implemented

Authentication

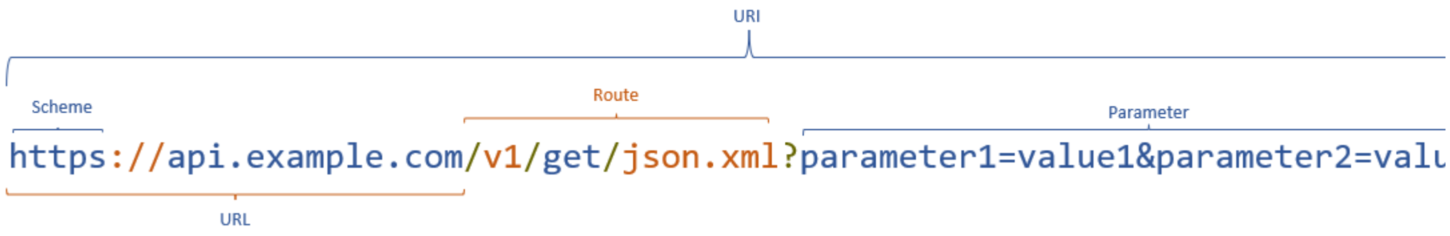
Many APIs require authentication to ensure that only authorized users or applications can access certain resources—some examples of using API keys include OAuth tokens or other authentication mechanisms.

Documentation

Good API design includes comprehensive documentation that explains how to use the API, the available endpoints, request and response formats, and any authentication requirements. This documentation helps you as a developer to understand and integrate with the API effectively.

Structure of API URL

Here is an example of an API URL:


URI
Scheme
Route
Parameter
https://api.example.com/v1/get/json.xml?parameter1=value1&parameter2=valu
URL

Uniform resource identifier (URI)

The entire string https://api.example.com/v1/get/json.xml?parameter1=value1&parameter2=value2 is a URI.
A URI is a string of characters identifying a name or a resource on the internet.

Uniform resource locator (URL)

The URL is a specific type of URI that provides the means to access a resource on the web.
In this case, https://api.example.com is a URL.

Scheme

The protocol is specified at the beginning of the URL as **https://**, indicating that the communication should be secured using the HTTPS protocol.

Route

This is the location on the web server; for example: /v1/get/json.xml

What is NBA_API?

This API provides access to various statistics and data related to the National Basketball Association (NBA). The "nba_api" allows developers to programmatically retrieve player statistics, team information, game results, and more.

API versus REST API

Application programming interface (API) and Representational state transfer API (REST API) are related terms, but there are distinctions between them. Let's look at them:

Feature	API	REST API
Full form	Application programming interface	Representational state transfer API
Definition	A set of protocols and tools for building software applications. It defines how different software components should interact.	A specific type of web API that follows the principles of representational state transfer (REST). It is an architectural style for designing networked applications.
Scope	A broader term encompassing various types of interfaces	Specifically refers to APIs following the principles of REST architecture
Communication	API communication methods can vary (Remote procedure call (RPC), Simple objects access protocol (SOAP), and so on)	Uses standard HTTP methods (GET, POST, PUT, DELETE) for communication
Architectural style	No specific architectural style	Follows the REST architectural style
Data format	Can use different data formats (JSON, XML, and so on)	Typically uses lightweight data formats, commonly JSON, for data exchange
URI	Resource identification methods may vary	Resources identified by URIs, each with multiple representations
Usage	Used in various contexts (web development, libraries, operating systems, and so on)	Primarily used for web services and web-based applications

How to access Free Public APIs

To use a public API in your Python code, you need to know the URL of the API endpoint you want to access. Here's how you can find the URL for a public API from the list provided in the GitHub repository:

- Go to the GitHub repository: https://github.com/public-apis/public-apis
- Scroll down to the table of APIs and find the one that you're interested in using.
- Look for the "API" column in the table, which will give you the base URL for the API. The base is the URL you will use to access the API endpoints.
- If the API requires authentication or has additional instructions, look for the "Auth" or "HTTPS" columns for more information.

Once you have the URL for the API, you can use it in your Python code with the requests library to make HTTP requests and retrieve data from the API.
For example, to retrieve data from the FishWatch API, you can use the following code:

```
# Import the requests and json modules for making HTTP requests and handling JSON data, respectively.
import requests
import json

# Specify the URL of the API endpoint for retrieving information about fish species.
url = "https://www.fishwatch.gov/api/species"

# Make an HTTP GET request to the specified URL and store the response in the data variable.
data = requests.get(url)

# Parse the JSON data received from the API response using json.loads() and store it in the results variable.
results = json.loads(data.text)
```

Copied!

Note that the specific endpoint you want to access may differ for each API, so you should refer to the API documentation for more information on how to use it.

Author

Akansha Yadav