**Introduction to API**

**Author:** Apurva Sachin Kulkarni

**Date:** 22nd August, 2025

**Introduction**

An Application Programming Interface (API) has become one of the most essential building blocks of modern software development. APIs allow different applications, platforms, and services to communicate with each other in a structured and standardized manner. They act as a bridge between systems, enabling seamless data exchange and functionality integration without exposing the internal complexities of each system. In this document, we explore the fundamental concepts of APIs, including endpoints, requests, responses, query parameters, and headers. We also discuss different types of APIs, their use cases, and the HTTP methods that make them work. Finally, we implement a practical example by using the CoinGecko API to fetch real-time Bitcoin price data, process it with Python libraries, and visualize it as a line graph. This example demonstrates how APIs can be leveraged to bring external, real-world data into applications for analysis and decision-making.

**What is an API?**

An Application Programming Interface (API) is a set of rules and protocols that allow different software applications to communicate with each other. It acts like a bridge between two systems, enabling them to exchange data or perform actions without needing to understand each other’s internal workings.

In simple words, API is like a waiter in a restaurant:

* You (the client) place an order from the menu.
* The waiter (API) takes your request to the kitchen (server).
* The kitchen prepares the food (data) and gives it to the waiter.
* The waiter brings the food back to your table.

**Key Components of an API**

1. **Endpoint**
   * The specific URL where the API can be accessed.
   * Example: https://api.coingecko.com/api/v3/coins/bitcoin/market\_chart
2. **Request**
   * Sent by the client to the API.
   * Includes the HTTP method (GET, POST, PUT, DELETE) and sometimes additional data.
3. **Response**
   * Data returned by the server, usually in JSON or XML format.
4. **Query Parameters**
   * Extra details added to the endpoint to customize the request.
5. **Headers**
   * Provide extra information with the request (like authentication tokens or content type).

**Types of APIs**

1. **Open APIs (Public APIs)** – Free to use, available to everyone (e.g., CoinGecko API).
2. **Private APIs** – Used internally within an organization.
3. **Partner APIs** – Shared with trusted business partners.
4. **Composite APIs** – Combine multiple APIs into one call.

**HTTP Methods in APIs**

* **GET** → Retrieve data from the server.
* **POST** → Send new data to the server.
* **PUT** → Update existing data.
* **DELETE** → Remove data.

**API Response Format**

Most APIs return data in JSON (JavaScript Object Notation) format. Example (Bitcoin price API response snippet):

{

"prices": [

[1724284800000, 64321.45],

[1724371200000, 64785.10]

]

}

* First element = Timestamp in milliseconds
* Second element = Bitcoin Price in USD

**Why Use APIs?**

* Easy integration between different systems
* Real-time data access (e.g., stock prices, Bitcoin prices, COVID-19 stats)
* Saves development time (no need to build everything from scratch)
* Enables automation

**Example**

**1. Libraries Used**

* **requests** → to send HTTP requests and get data from the API.
* **pandas** → to create and manipulate tabular data (DataFrame).
* **matplotlib.pyplot** → to plot graphs for visualization.



**2. The Class BitCoin\_API**

The code is written using a class structure (BitCoin\_API), which contains three methods:

**a) GetUrl(url, params)**

* Sends a GET request to the API with query parameters (like vs\_currency and days).
* Prints the **status code** (200 = success).
* Returns the API response as a Python dictionary (data).

**b) MakeDataFrame(data)**

* data["prices"] → the JSON has a key "prices", which contains a list of [timestamp, price].
* Creates a **DataFrame** with 2 columns:
  + **Date** → converted from milliseconds (UNIX format) to human-readable dates.
  + **Prices** → the Bitcoin price in USD.

**c) PlotGraph(df)**

* Creates a line chart of Bitcoin prices over time.
* X-axis = Dates, Y-axis = Bitcoin Prices.
* Orange-colored line.
* Labels + Title added.
* X-axis dates are rotated for readability.

A screen shot of a computer program

AI-generated content may be incorrect.

**3. Main Execution Block**

API URL = CoinGecko’s market chart endpoint.

Query Parameters:

* vs\_currency=usd → Bitcoin price in USD.
* days=30 → last 30 days data.

Calls all the methods step by step:

1. Fetch data
2. Convert to DataFrame
3. Plot the graph

A screen shot of a computer program

AI-generated content may be incorrect.

**Output**

**A black background with white text

AI-generated content may be incorrect.**

A graph with orange lines

AI-generated content may be incorrect.

**Conclusion**

APIs are an integral part of the digital ecosystem, powering everything from mobile apps and financial systems to healthcare solutions and e-commerce platforms. They make it possible for developers to build powerful applications by reusing existing services, reducing development time, and enabling real-time access to data. Through the example of the CoinGecko Bitcoin API, we demonstrated how Python can be used to fetch, process, and visualize data from an external API. This exercise highlights the importance of APIs in not just accessing information, but also in integrating, analyzing, and presenting it in meaningful ways. In conclusion, understanding and working with APIs is a crucial skill for developers, data scientists, and IT professionals. Mastering these concepts opens doors to endless opportunities in automation, data analysis, system integration, and innovation.

\*\*\*