# Project Title: Understanding REST APIs – A Beginner's Guide with Examples

This is a complete beginner's guide on how to get started and coding with REST APIs.

## By the end of this user guide, you’ll have

* A detailed understanding of REST APIs
* Flask Installation
* Applying REST to build and run REST APIs with examples

## Prerequisites

* Python for Python Examples – (<https://www.python.org/downloads/>)
* Python (requests library)
* ([Node.js](http://node.js)) for Javascript/([Node.js](http://node.js)) examples.
* Code Editor or IDE (preferably Visual Studio Code) (https://code.visualstudio.com/)
* Any web browser like Chrome or Firefox
* API Testing Tool like Postman (optional) – (<https://www.postman.com/downloads/>)

---

## Overview

### What is an API?

An API, also known as an ***Application Programming Interface***, is a set of rules that allows different programs to communicate with each other. It enables one application to request services or data from another.

It is currently used in applications like mobile apps, web apps, and cloud services where the principle of the API can be the request-response exchange between a client and a server.

---

### How APIs Work – Simple Analogy

An application programming interface (API) is a software programming code that enables two software programs to communicate. With an API, there is an exchange of data and other functionalities between software applications.

---

For beginners, it is easier to understand when you consider the scenario in a restaurant.

Here, the waiter is the equivalent of an API as he handles the communication between the kitchen and the customer.

\*\*You(the client)\*\* asks for the food(data).

The customer does not directly interact with \*\*the kitchen(server).\*\*

\*\*The waiter\*\* acts as a middleman (API), facilitating communication between them.

---

APIs can be classified based on different criteria, such as by access, by protocols, and so on.

With time, APIs have evolved with changing technology, thereby adding to the services they provide.

Based on protocols, the different types of APIs include REST, RPC, and SOAP.

In this article, we will learn about REST APIs and how they work with examples for a thorough understanding.

## What is a REST API?

REST APIs are used for building web services. REST stands for Representational State Transfer, and a majority of APIs are powered by it, especially in web applications.

REST APIs are versatile interfaces used to send and receive data through HTTP requests in different formats, like XML and JSON, among other formats.

REST APIs use the principles of the REST architectural design when communicating between clients and servers.

### REST principles (GET, POST, PUT, DELETE)

A REST API defines a set of standard operations (GET, POST, PUT, DELETE) that use HTTP for communication.

There are five popular API request methods: GET, POST, PUT, PATCH, and DELETE.

1. GET: retrieves information or data from a specified resource
2. POST: submits data to be processed to a specified resource
3. PUT: updates a specified resource with new data
4. DELETE: deletes a specified resource

## JSON format

### JSON stands for JavaScript Object Notation.

JSON is used to organize and share data using text. Using JSON, data is written in a structured list, using key-value pairs. This language-independent data format serializes and transmits structured data between two programs.

When a REST API makes a call, the request and the response are formatted as JSON.

``` Example of a JSON string

{"name":"David",

  "age":25,

  "job":"barber"}

This code defines an object with 3 properties, name, age, and job, each of which has a value.

If you parse the JSON string with a JavaScript program, you can access the data as an object:

let personName = obj.name;

let personAge = obj.age;

## How REST APIs Work – A Simple Example

Consider a free public API

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

The above URL is an end point. A REST API endpoint is a URL that points to a resource(s) on a server. A well-designed REST URL makes it easy to guess what it returns or does. Thus, REST is very simple and easy to read by the users.

Paste it in your browser or Postman.

*(In Postman, click on the + tab or click the “New” button in the top left.*

*You will see a dropdown on the left that says GET and a long empty bar next to it.*

*Paste the above URL with GET in the dropdown.*

*Click the SEND button.)*

You will get the following:

---

{

  "id": 1,

  "name": "Leanne Graham",

  "username": "Bret",

  "email": "Sincere@april.biz"

  ...

}

You have got the API response. That’s how you call a REST API successfully.

---

## Walkthrough with a public API (OpenWeatherMap)

In this section, we will look at how to retrieve data from a public URL, OpenWeatherMap. This URL can be used to get the temperature of a city and display it. Let us look at the steps.

1. First, obtain an API Key:
2. Go to OpenWeatherMap's website and create a free account (https://home.openweathermap.org/users/sign\_up).
3. After signing up, your API key will be found on your account page.
4. Copy your API key.
5. The API key is used to make requests to the OpenWeatherMap API.

* fetch() is a built-in JavaScript function used to make HTTP requests to APIs.
* For this request, you get a response object from the server.

---

Create an HTML file called weather.html and paste the following code:

---

<!DOCTYPE html>

<html>

<head>

  <title>Weather App</title> <!-- Sets the title of the browser tab -->

</head>

<body>

 <!-- Page heading -->

  <h2>Weather Info</h2>

  <!-- Input field for entering the city name -->

  <input type="text" id="cityInput" placeholder="Enter city name" />

  <!-- Button that triggers the weather fetch function when clicked -->

  <button onclick="getWeather()">Get Weather</button>

  <p id="weather">Weather data will appear here.</p>

  <script>

    // This function gets called when the user clicks the "Get Weather" button

    function getWeather() {

      const apiKey = "YOUR\_API\_KEY\_HERE"; // Replace with your real API key

      const city = document.getElementById("cityInput").value;

           // Construct the API URL with the city name and your API key

 const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

      fetch(apiUrl)

        .then(response => {

          if (!response.ok) {

            throw new Error("City not found");

          }

          return response.json();

        })

        .then(data => {

          const temp = data.main.temp;

          const desc = data.weather[0].description;

          document.getElementById("weather").textContent =

            `Temperature in ${city}: ${temp}°C, ${desc}`;

        })

        .catch(error => {

          document.getElementById("weather").textContent = "Error: " + error.message;

        });

    }

  </script>

</body>

</html>

---

1. Save the file as weather.html.
2. Double-click it to open in any browser.
3. Type a city name like London and click "Get Weather."
4. You’ll see the current temperature and weather conditions.

## Making API Requests Using Python

API requests in Python are made using the requests library, which can send requests and handle responses.

However, it is not a part of Python's standard library and is installed using pip.

```pip install requests

## GET and POST request example

---

### Using GET requests to retrieve data from an API.

import requests     # imports the requests module used to send HTTP requests

api\_url = "https://jsonplaceholder.typicode.com/users/1"    # stores the URL endpoint of the API

response = requests.get(api\_url)    # Sends an HTTP GET request to the URL

if response.status\_code == 200:    # if HTTP response status code is 200, it means the request was successful

    data = response.json()  # Parse JSON response

    print("Data received:", data)    # displays the parsed data

else:

    print(f"Error: {response.status\_code} - {response.text}") # if code is not 200, error is printed with a message

---

### Using POST Requests:

We can use a POST request in a similar manner.

POST requests send data to an API to create or update resources.

import requests

api\_url = "https://api.example.com/new\_entry"

payload = {"name": "Roll\_number", "value": 123}

headers = {"Content-Type": "application/json"}

response = requests.post(api\_url, json=payload, headers=headers)

if response.status\_code == 201:  # 201 Created for successful creation

    print("Entry created successfully.")

else:

    print(f"Error: {response.status\_code} - {response.text}")

## Basic Error Handling in Python

When writing code, we may encounter errors such as divide by zero or accessing a file that doesn’t exist. We can handle these errors using Python. Let us look at a simple example.

---

try:

    number = int(input("Enter a number: "))

    result = 10 / number

    print(f"Result: {result}")

except ZeroDivisionError:

    print("You can't divide by zero!")

except ValueError:

    print("Please enter a valid number.")

---

Here, we enter the code that may have an error under try

* try: The code that might cause an error goes here.
* except ZeroDivisionError: This block runs if the user enters 0.
* except ValueError: This runs if the user types a non-numeric value.

## Testing APIs Using Postman

Postman is used widely for building and testing APIs. It offers both manual and automated testing for different API types such as SOAP and REST.

Let us look at a step-by-step approach on how to test APIs with Postman.

First, install Postman in your system from the [official Postman](https://www.postman.com/downloads/) site.

## Create a Request

1. We can create a new request in the Postman application by choosing the required HTTP request method from the drop-down and supplying the URL for the request.

A screenshot of a computer

AI-generated content may be incorrect.

1. Click on Send to execute the request. Postman will send the request to the specified URL and receive a response.
2. One can add  assertions to ensure that your API responses contain the expected data or meet specific requirements.

## Status Codes in Rest APIs

Given below are some status codes and their meanings which we should interpret.

### 3xx Status Codes (Redirection)

**\*\*300 -\*\*** The request has more than one possible response from which the user should choose one.

**\*\*301 -\*\*** The URL of the requested resource has been changed permanently.

### 2xx Status Codes (Success)

**\*\*200\*\* -** This indicates that the request has succeeded.

**\*\*201 -** \*\* This indicates that the request has succeeded and thus, a new resource has been created.

Some of the most common errors in REST APIs and their handling methods are as follows:

### Client Errors - 4xx codes

**\*\*400 Bad Request:\*\***

This error occurs when the client sends invalid data, invalid requests, or missing required parameters. The error can be addressed by validating the input data thoroughly on the server side.

\*\***401 Unauthorized:\*\***

This error indicates that the client lacks the required valid authentication credentials. It can be handled by applying proper authentication mechanisms, such as API keys ot OAuth tokens.

\*\***403 Forbidden:\*\***

The client is authenticated but does not have the required permissions to access the necessary resource.

Always check the user roles or permissions before sending a request.

\*\*404 Not Found:\*\*

The requested resource does not exist at the specified URL. To handle this, verify the accuracy of the requested URL and ensure the resource exists.

### Server Errors - 5xx codes

**\*\* 500 Internal Server Error:\*\***

A general error indicating an unexpected condition on the server that prevented it from fulfilling the request. A way to handle this is by using comprehensive error logging to identify the root cause of the error.

\*\* **503 Service Unavailable: \*\***

The server is temporarily unable to handle the request due to maintenance or overload. Mechanisms to handle server overload should be implemented.

*\*RESTful APIs are an important standard for modern web development,Their design principles and the use of HTTP methods make them the best choice in terms of scalability, flexibility, and ease of integration. \**