

# API SCAVENGER HUNT

## Task 1: OpenWeatherMap API

### CODE:

```
1 import requests
2 import json
3 # Replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key
4 api_key = 'ebfbb8b874310bf9eddab709311116e3'
5
6 # Function to retrieve weather data
7 def get_weather_data(endpoint, city, country_code):
8     url = f"https://api.openweathermap.org/data/2.5/{endpoint}?q={city},{country_code}&appid={api_key}"
9     response = requests.get(url)
10
11     if response.status_code == 200:
12         data = response.json()
13         return data
14     else:
15         print(f"Error: Unable to retrieve data. Status code: {response.status_code}")
16         return None
17
18 # Function to print weather information
19 def print_weather_info(city, data):
20     print(f"{city} Weather Report:")
21     print(f"    Temperature: {data['main']['temp']} Kelvin")
22     print(f"    Weather: {data['weather'][0]['description']}")
23     print(f"    Wind Speed: {data['wind']['speed']} m/s\n")
24
25 # Function to print forecast information
26 def print_forecast_info(city, data):
27     print(f"{city} 5 Day's Forecast Report:")
28     for forecast in data['list']:
29         print(f"    Date & Time: {forecast['dt_txt']}")
30         print(f"    Temperature: {forecast['main']['temp']} Kelvin")
31         print(f"    Weather: {forecast['weather'][0]['description']}")
32         print(f"    Wind Speed: {forecast['wind']['speed']} m/s\n")
33
34 # Example usage:
35 london_weather_report = get_weather_data("weather", "London", "uk")
36 tokyo_forecast_report = get_weather_data("forecast", "Tokyo", "jp")
37
38 # Print the retrieved data
39 if london_weather_report:
40     print_weather_info("London", london_weather_report)
41
42 if tokyo_forecast_report:
43     print_forecast_info("Tokyo", tokyo_forecast_report)
```

### RESULTS

```
Python Version: 3.10.0 (default, Apr 20 2022, 13:28:58) [Clang 13.1.6
(clang-1316.0.21.2.3)]
Python Encoding: UTF-8
```

```
London Weather Report:
Temperature: 281.74 Kelvin
Weather: overcast clouds
Wind Speed: 1.03 m/s
```

```
Tokyo 5 Day's Forecast Report:
Date & Time: 2023-11-05 03:00:00
Temperature: 294.64 Kelvin
Weather: light rain
Wind Speed: 3.29 m/s

Date & Time: 2023-11-05 06:00:00
Temperature: 295.66 Kelvin
Weather: overcast clouds
Wind Speed: 2.67 m/s

Date & Time: 2023-11-05 09:00:00
Temperature: 295.25 Kelvin
Weather: light rain
Wind Speed: 3.71 m/s

Date & Time: 2023-11-05 12:00:00
Temperature: 294.07 Kelvin
Weather: broken clouds
Wind Speed: 2.89 m/s
```

Date & Time: 2023-11-05 15:00:00  
Temperature: 293.22 Kelvin  
Weather: clear sky  
Wind Speed: 1.93 m/s

Date & Time: 2023-11-05 18:00:00  
Temperature: 292.82 Kelvin  
Weather: light rain  
Wind Speed: 2.16 m/s

Date & Time: 2023-11-05 21:00:00  
Temperature: 292.95 Kelvin  
Weather: light rain  
Wind Speed: 3.63 m/s

Date & Time: 2023-11-06 00:00:00  
Temperature: 294.86 Kelvin  
Weather: light rain  
Wind Speed: 6.27 m/s

Date & Time: 2023-11-06 03:00:00  
Temperature: 297.01 Kelvin  
Weather: broken clouds  
Wind Speed: 9.66 m/s

Date & Time: 2023-11-06 06:00:00  
Temperature: 296.97 Kelvin  
Weather: broken clouds  
Wind Speed: 9.85 m/s

Date & Time: 2023-11-06 09:00:00  
Temperature: 295.46 Kelvin  
Weather: overcast clouds  
Wind Speed: 8.54 m/s

Date & Time: 2023-11-06 12:00:00  
Temperature: 295.13 Kelvin  
Weather: overcast clouds  
Wind Speed: 9.13 m/s

Date & Time: 2023-11-06 15:00:00  
Temperature: 295.03 Kelvin  
Weather: overcast clouds  
Wind Speed: 9.46 m/s

Date & Time: 2023-11-06 18:00:00  
Temperature: 295.23 Kelvin  
Weather: light rain  
Wind Speed: 13.91 m/s

Date & Time: 2023-11-06 21:00:00  
Temperature: 295.59 Kelvin  
Weather: light rain  
Wind Speed: 15.08 m/s

Date & Time: 2023-11-07 00:00:00  
Temperature: 294.98 Kelvin  
Weather: moderate rain  
Wind Speed: 8.13 m/s

Date & Time: 2023-11-07 03:00:00  
Temperature: 296.17 Kelvin  
Weather: light rain  
Wind Speed: 3.26 m/s

Date & Time: 2023-11-07 06:00:00  
Temperature: 296.79 Kelvin  
Weather: overcast clouds  
Wind Speed: 1.66 m/s

Date & Time: 2023-11-07 09:00:00  
Temperature: 294.9 Kelvin  
Weather: few clouds  
Wind Speed: 5.63 m/s

Date & Time: 2023-11-07 12:00:00  
Temperature: 292.44 Kelvin  
Weather: few clouds  
Wind Speed: 5.36 m/s

Date & Time: 2023-11-07 15:00:00  
Temperature: 291.27 Kelvin  
Weather: broken clouds  
Wind Speed: 5.29 m/s

Date & Time: 2023-11-07 18:00:00  
Temperature: 290.18 Kelvin  
Weather: scattered clouds  
Wind Speed: 5.42 m/s

Date & Time: 2023-11-07 21:00:00  
Temperature: 289.55 Kelvin  
Weather: clear sky  
Wind Speed: 4.75 m/s

Date & Time: 2023-11-08 00:00:00  
Temperature: 290.6 Kelvin  
Weather: scattered clouds  
Wind Speed: 3.9 m/s

Date & Time: 2023-11-08 03:00:00  
Temperature: 292.73 Kelvin  
Weather: scattered clouds  
Wind Speed: 1.99 m/s

Date & Time: 2023-11-08 06:00:00  
Temperature: 293.36 Kelvin  
Weather: scattered clouds  
Wind Speed: 3.46 m/s

Date & Time: 2023-11-08 09:00:00  
Temperature: 291.69 Kelvin  
Weather: few clouds  
Wind Speed: 5.29 m/s

Date & Time: 2023-11-08 12:00:00  
Temperature: 290.44 Kelvin  
Weather: clear sky  
Wind Speed: 4.28 m/s

Date & Time: 2023-11-08 15:00:00  
Temperature: 289.85 Kelvin  
Weather: clear sky  
Wind Speed: 2.86 m/s

Date & Time: 2023-11-08 18:00:00  
Temperature: 289.4 Kelvin  
Weather: clear sky  
Wind Speed: 2.98 m/s

Date & Time: 2023-11-08 21:00:00  
Temperature: 289.14 Kelvin  
Weather: broken clouds  
Wind Speed: 3.09 m/s

Date & Time: 2023-11-09 00:00:00  
Temperature: 290.03 Kelvin  
Weather: broken clouds  
Wind Speed: 2.94 m/s

Date & Time: 2023-11-09 03:00:00  
Temperature: 291.3 Kelvin  
Weather: overcast clouds  
Wind Speed: 1.87 m/s

Date & Time: 2023-11-09 06:00:00  
Temperature: 292.44 Kelvin  
Weather: broken clouds  
Wind Speed: 1.1 m/s

Date & Time: 2023-11-09 09:00:00  
Temperature: 292.26 Kelvin  
Weather: clear sky  
Wind Speed: 2.01 m/s

Date & Time: 2023-11-09 12:00:00  
Temperature: 291.35 Kelvin  
Weather: clear sky  
Wind Speed: 2.06 m/s

Date & Time: 2023-11-09 15:00:00  
Temperature: 290.74 Kelvin  
Weather: overcast clouds  
Wind Speed: 1.21 m/s

Date & Time: 2023-11-09 18:00:00  
Temperature: 290.28 Kelvin  
Weather: broken clouds  
Wind Speed: 0.41 m/s

Date & Time: 2023-11-09 21:00:00  
Temperature: 290.3 Kelvin  
Weather: overcast clouds  
Wind Speed: 0.59 m/s

Date & Time: 2023-11-10 00:00:00  
Temperature: 290.42 Kelvin  
Weather: overcast clouds  
Wind Speed: 0.48 m/s

>>>

## OBSERVATION:

The OpenWeatherMap API provides a simple and easy way to retrieve weather data. The simplicity of performing HTTP queries and analyzing JSON replies demonstrates its usability. The documentation for the API is well-organized, with clear instructions and examples for various endpoints, making it simple for developers to integrate meteorological data into their apps. Furthermore, the API delivers a wide range of meteorological information, such as current conditions, predictions, and historical data, enabling a variety of use cases.

In terms of capability, OpenWeatherMap provides a comprehensive collection of functions. Developers may access not just basic meteorological information such as temperature, wind speed, and descriptions, but also more complex data like UV index, precipitation, and cloud cover. The availability of data for a large number of sites throughout the world increases the API's usability. Furthermore, the API provides forecast data for up to 16 days in advance, which is useful for applications that require long-term weather predictions.

The OpenWeatherMap API has a wide range of potential uses. It may be used in a variety of industries, including agriculture, transportation, and tourism. Farmers, for example, can use meteorological data for crop planning and irrigation scheduling. It may be used by transportation businesses to optimize routes and plan for unfavorable weather conditions. Furthermore, mobile apps may utilize the API to give users with localized weather updates, which improves user experience and safety. Overall, the OpenWeatherMap API seems to be a robust tool with several possible applications in a variety of areas.

## Task 2: Google Maps API

### CODE:

```
✓ 6s !pip install -U googlemaps
```

```
Collecting googlemaps
  Downloading googlemaps-4.10.0.tar.gz (33 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests<3.0,>=2.20.0 in /usr/local/lib/python3.10/dist-packages (from googlemaps) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.20.0->googlemaps) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.20.0->googlemaps) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.20.0->googlemaps) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.20.0->googlemaps) (2023.7.22)
Building wheels for collected packages: googlemaps
  Building wheel for googlemaps (setup.py) ... done
  Created wheel for googlemaps: filename=googlemaps-4.10.0-py3-none-any.whl size=40713 sha256=b1add3d256bac9f210d4d79596bb0ead720b13bf0c433c6e92e3827efb3ba9d3
  Stored in directory: /root/.cache/pip/wheels/17/f8/79/999d5d37118fd35d7219ef57933eb9d09886c4c4503a800f84
Successfully built googlemaps
Installing collected packages: googlemaps
Successfully installed googlemaps-4.10.0
```

```

def create_gmaps_client(api_key):
    return googlemaps.Client(key=api_key)

def display_map_centered_on_location(gmaps, location):
    gmaps_url = f"https://www.google.com/maps/search/?api=1&query={nyc_coordinates[0]},{nyc_coordinates[1]}"
    webbrowser.open(gmaps_url)

def find_shortest_route(gmaps, origin, destination, mode="driving"):
    directions_result = gmaps.directions(origin, destination, mode=mode)

    if directions_result:
        distance = directions_result[0]['legs'][0]['distance']['text']
        duration = directions_result[0]['legs'][0]['duration']['text']

        print(f"From {origin} To {destination} the least possible distance is {distance}.")
        print(f"ETA through {mode} is {duration}.")
    else:
        print("No directions found.")

if __name__ == "__main__":
    api_key = 'AIzaSyB3Q00aNUeKULZGY-UF_03_wuPBCbSeDdw'
    gmaps = create_gmaps_client(api_key)

    # Display a map centered on a specific location (e.g., New York City)
    nyc_coordinates = (40.7128, -74.0060)
    display_map_centered_on_location(gmaps, nyc_coordinates)

    # Find the shortest route between two locations (e.g., San Francisco to Los Angeles)
    find_shortest_route(gmaps, "San Francisco, USA", "Los Angeles, USA", mode="driving")

```

## RESULTS:

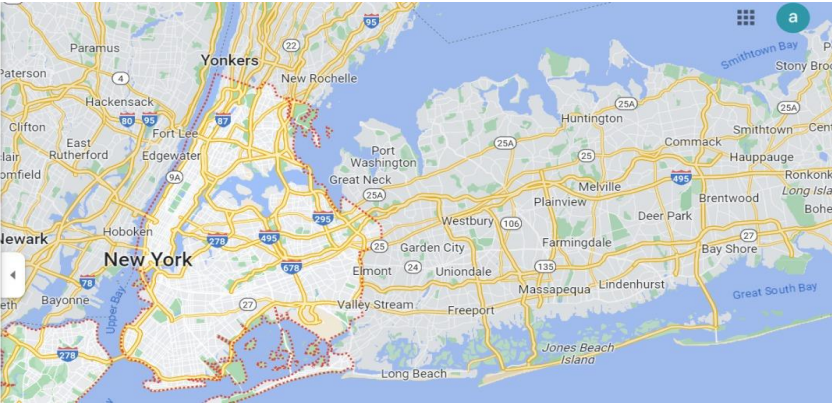
**Partial match**

{nyc\_coordinates[0]},{nyc\_coordinates[1]}

Showing results for {nyc\_coordinates[0]},{nyc\_coordinates[1]}. No results found for {nyc\_coordinates[0]},{nyc\_coordinates[1]}.

**New York**  
No reviews  
nyc · nyc

Don't see what you're looking for?  
[Try Google Search instead](#)



From San Francisco, USA To Los Angeles, USA the least possible distance is 383 mi.  
ETA through driving is 5 hours 56 mins.

## OBSERVATIONS:

The Google Maps API is a powerful tool that allows you to integrate maps and location-based services into your applications. I found the offered code to be quite simple to set up and utilize. The library has a well-documented interface that makes it simple to get started. It is very helpful to be



able to locate routes between sites, compute distances, and acquire projected trip times. The option to display maps centered on certain locations also provides a visual component to the programme.

The API's capabilities go beyond what is shown in the code. It may be used for geocoding (converting addresses to geographic coordinates), reverse geocoding (identifying the address of a set of coordinates), and even integration with various map layers and overlays. This makes it a useful tool for a wide range of applications, from navigation apps to location-based services in e-commerce or social networking platforms.

This API has a wide range of potential uses. It may be used in travel and transportation apps to deliver real-time instructions and trip information to users. Businesses may also utilize it to improve consumer experiences by providing location-based services such as identifying nearby retailers or services. The API may be used to regionally visualize and analyze data in the context of data analysis, assisting in market research and decision-making. Overall, the Google Maps API is a powerful tool with a diverse set of features that may be utilized across other domains.

### Task 3: REST Countries API

#### CODE:

```
1 import requests
2 def get_country_info(country_name):
3     url = f'https://restcountries.com/v3.1/name/{country_name}'
4     try:
5         response = requests.get(url)
6         response.raise_for_status() # Check for HTTP errors
7         data = response.json()[0]
8         return {
9             "Population": data['population'],
10            "Area": f"{data['area']} km²",
11            "Official Language": data['languages'][list(data['languages'].keys())[0]]
12        }
13    except requests.exceptions.HTTPError as http_err:
14        print(f'HTTP error occurred: {http_err}')
15    except Exception as err:
16        print(f'An error occurred: {err}')
17
18 def get_countries_in_region(region):
19     url = f'https://restcountries.com/v3.1/region/{region}'
20     try:
21         response = requests.get(url)
22         response.raise_for_status() # Check for HTTP errors
23         data = response.json()
24         return [country['name']['common'] for country in data]
25    except requests.exceptions.HTTPError as http_err:
26        print(f'HTTP error occurred: {http_err}')
27    except Exception as err:
28        print(f'An error occurred: {err}')
29
30 # Usage:
31 country_info = get_country_info('brazil')
32 if country_info:
33     for key, value in country_info.items():
34         print(f"Brazil's {key}: {value}")
35
36 african_countries = get_countries_in_region('africa')
37 if african_countries:
38     print(f"Countries in Africa are:")
39     for country in african_countries:
40         print(country)
```

## RESULTS:

```
Python Encoding: UTF-8
Brazil's Population: 212559409
Brazil's Area: 8515767.0 km²
Brazil's Official Language: Portuguese
Countries in Africa are:
Malawi
Cameroon
Nigeria
Western Sahara
Lesotho
Mayotte
Rwanda
Sierra Leone
Benin
Ghana
Central African Republic
Kenya
Egypt
Tunisia
Sudan
Zimbabwe
Togo
British Indian Ocean Territory
Tanzania
Gabon
Burundi
Ethiopia
Madagascar

Republic of the Congo
Gambia
Guinea
Zambia
Eritrea
Burkina Faso
Ivory Coast
Cape Verde
Guinea-Bissau
Mali
South Sudan
Seychelles
Chad
Djibouti
Namibia
Mozambique
Mauritius
Saint Helena, Ascension and Tristan da Cunha
Comoros
Equatorial Guinea
Uganda
Botswana
Libya
Algeria
São Tomé and Príncipe
Angola
Niger
Réunion
Senegal
Morocco
Somalia
DR Congo

Mauritania
South Africa
Liberia
Eswatini
>>>
```

## OBSERVATIONS:

The Restcountries API used in the supplied code is simple and basic. It gives you easy access to a variety of information on countries all around the world. The API's basic URL-based structure makes

it accessible even to beginners, and its answers in JSON format are well-structured and straightforward to understand.

The API provides a wide range of data on countries, such as population, area, languages spoken, currencies used, and much more. As a result, it is a significant resource for a wide range of applications, from educational initiatives to constructing apps that require geographic data. It might be used to construct a travel app that gives thorough information about nations, or it could be used in a data analysis project to gather statistics about various places.

This API might be used in instructional tools, for example. It might be used to develop interactive geography classes or quizzes that teach pupils about various nations and their features. It might also be utilized in travel-related apps to give users detailed information on places, allowing them to make educated decisions about where to go. Overall, the Restcountries API is a useful tool with several possible uses in various sectors.

## Task 4: Currency Converter API

### CODE:

```
✓ 7s ▶ pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2023.7.22)

✓ 0s ▶ import requests

# Replace 'Your_Api_Key' with your actual API key
api_key = '9bc336e3bf49972bcd96150'

# Convert 100 USD to EUR
url_usd_to_eur = f'https://v6.exchangerate-api.com/v6/{api_key}/pair/USD/EUR/100'
response_usd_to_eur = requests.get(url_usd_to_eur)
data_usd_to_eur = response_usd_to_eur.json()

# Print the conversion result
print(f"100 USD is equal to {data_usd_to_eur['conversion_result']} EUR")

# Convert 1000 JPY to GBP
url_jpy_to_gbp = f'https://v6.exchangerate-api.com/v6/{api_key}/pair/JPY/GBP/1000'
response_jpy_to_gbp = requests.get(url_jpy_to_gbp)
data_jpy_to_gbp = response_jpy_to_gbp.json()

# Print the conversion result
print(f"1000 JPY is equal to {data_jpy_to_gbp['conversion_result']} GBP")
```

### RESULTS:

```
➡ 100 USD is equal to 93.48 EUR
   1000 JPY is equal to 5.423 GBP
```



### **OBSERVATIONS:**

ExchangeRate-API is typically lauded for its ease of use. Users generally find it straightforward to sign up and obtain an API key. The API's documentation is clear, providing concise and clear instructions on how to construct API calls, which can be a significant aid to developers of all skill levels. The simplicity of the API endpoints, often involving direct GET requests that return JSON objects, makes it accessible for quick integration into applications requiring currency conversion functionality. The free tier offering a generous number of requests per hour could be especially appealing for small-scale projects or for individuals learning how to work with APIs.

The functionality provided by ExchangeRate-API is comprehensive, offering up-to-the-minute exchange rates for a broad array of global currencies. This kind of service is indispensable for applications such as online retail platforms, financial analysis software, or travel applications that require consistent currency conversion capabilities. The provision of current exchange rate data is essential for ensuring accurate and fair pricing, as well as for precise financial planning.

The scope of potential uses for ExchangeRate-API extends far and wide. It can be integrated into e-commerce systems for real-time pricing adjustments, financial analytics for market trend analysis, or travel apps that need to provide cost estimations in different currencies. For international businesses, this API could automate the conversion process in software for accounting, billing, or financial reporting. In essence, ExchangeRate-API combines ease of implementation with a wide range of functions, making it a versatile asset for a multitude of digital financial applications.