Python Days of Our Dogs API Tutorial

Contents

Time required: 60 minutes

- Comment each line of code as show in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

What is an API?

An **API** stands for **Application Programming Interface**. It's a way for different software programs to talk to each other.

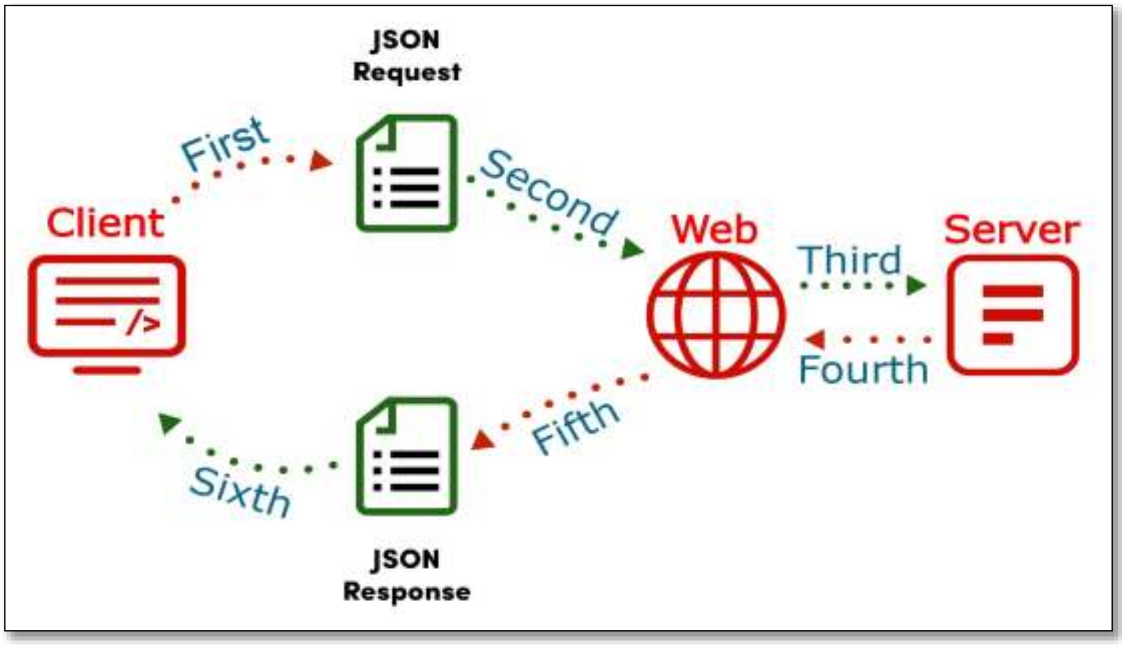
Think of an API like a **waiter at a restaurant**. You (the user) tell the waiter what you want. The waiter takes your order to the kitchen (the system that prepares the data). Then the waiter brings the food (the response) back to your table. You don't need to know how the kitchen works—just how to place your order.

APIs define:

- **What you can ask for**
- **How to ask for it**
- **What the response will look like**

For example, when you check the weather on your phone, an API connects your app to a weather service like the National Weather Service. The app asks for today's forecast, and the API brings that data back.

APIs help different apps, websites, or services work together—without you needing to know what's going on behind the scenes.



Intro to APIs in Python

When we visit websites, our browser asks a web server for a **webpage**.

When we use an **API**, our **Python program asks for data** instead.

That data usually comes back in **JSON format**, which looks like a Python dictionary.
It's easy to read and work with.

How It Works

1. Python sends a **request** to the API (web server).
2. The API sends back **data** and a **status code**.

3. Your program uses the data in any way you need.

Some APIs require an **API key**—a kind of password that identifies your program.

Using the requests Library

Python doesn't handle web requests by default, so we use the requests library.
You may need to install it first.

pip install requests

Here's a simple example:

```
import requests

# Send a GET request to a test API
response = requests.get('https://dummyjson.com/test')

print(response.status_code)  # Shows if the request worked (200 = success)
print(response.json())       # Converts JSON response into a Python dictionary
```

response is just a variable name. You can name it anything you want.

What Can APIs Do?

Once you're connected, you can use real-world data in your programs—such as:

- Weather reports from around the world
- Number of astronauts in space
- Image recognition and more

If the data is public and the API allows access, Python can grab it!

Understanding Status Codes in API Requests

When your Python program sends a request to an API, the server always replies with a **status code**. This is a 3-digit number that tells you what happened with your request.

Think of it like a **traffic signal** for data requests—did it go through? Did it fail? Should you try again?

You can check the status code in your Python code by using:

```
print(response.status_code)
```

The word response is whatever variable name you used when calling the API.

Common Status Codes You'll See

- **200 — OK**
  This means everything worked! The data was found and returned correctly.
  You'll probably get the data in **JSON format**, which Python can easily work with.

- **301 — Moved Permanently**
  The server is saying, "This has moved."
  Your request was redirected to another location. This usually happens when a website changes its URL or API endpoint.

- **400 — Bad Request**
  This means something was wrong with your request.
  Maybe you forgot a parameter or sent the data in the wrong format.
  Double-check what the API expects.

- **401 — Unauthorized**
  You didn't include the correct login information or **API key**.
  The server is saying, "I don't know who you are."

- **403 — Forbidden**
  You are **not allowed** to access this data—even if you're logged in.
  Some APIs block certain types of requests unless special permissions are given.

- **404 — Not Found**
  The resource you're asking for **doesn't exist** at that location.
  Maybe the URL is wrong, the data was moved, or it was deleted.

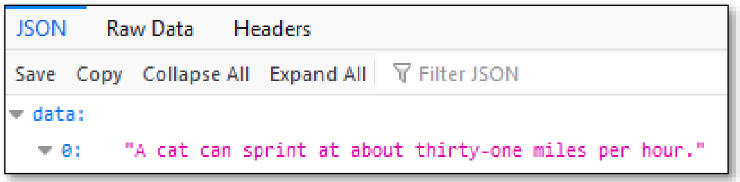
Status codes are extremely helpful for **debugging your code**. If your API call isn't working, always check the status code first. Most of the time, it will give you a good clue about what went wrong.

What is JSON?

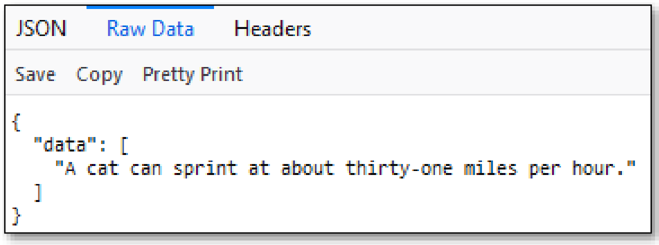
Most API's provide a JSON response.

JavaScript Object Notation (JSON) is an open standard file and data interchange format. It uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types.

Go to https://meowfacts.herokuapp.com/ to see some raw JSON data in two different formats.



```
data:
  0:    "A cat can sprint at about thirty-one miles per hour."
```



```
{
  "data": [
    "A cat can sprint at about thirty-one miles per hour."
  ]
}
```

What is a Dictionary?

A dictionary is a list of items. Dictionaries are mutable, they can be changed.

Here is a Python list that contains the number of days in the months of the year.

```
days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

If we want the number of days in January, use `days[0]`. December is `days[11]`.

Here is a dictionary of the days in the months of the year:

```
days = {'January':31, 'February':28, 'March':31, 'April':30, 'May':31,
        'June':30, 'July':31, 'August':31, 'September':30, 'October':31,
        'November':30, 'December':31}
```

To get the number of days in January, we use the following code.

```
days.get('January')
>> 31
```

Parameter	Details
`Key`	The desired key to lookup
`value`	The value to set or return

API Response Dictionaries and Lists

Meow Facts is a simple API with only one item returned → a random Meow Fact.

Most API responses are combinations of **dictionaries** and **lists**. **Meow Facts** is an example of a **dictionary** with a **key** and a **list** for a **value**.

To determine how to retrieve data from an API response. Work from the outside in.

1. What is the first data structure: **dictionary**
    a. **Key = data**
    b. **Value = List**
2. To retrieve the first value from the **list**: [0]
3. Combine them together: **dog_facts.get("data")[0]**
4. Dictionary:
    a. Key = attributes
    b. Value = body
5. Dictionary:
    a. Key = body
    b. Value = text to be displayed

With the outside in method, you can figure out any API JSON response.

```
# Dictionary with the key "data"
# The value is a list [0]
dog_facts = {
    # Inside the key value is a single item list, [0]
    "data": [
        "attributes" = {
            "body": {response}
        }
    ]
}
# Access the API dictionary response
# with list inside the dictionary
dog_fact = dog_facts.get("data")[0].get("attributes").get("body")

print(dog_fact)
```

Public API’s

There are hundreds of public API’s available. This website contains a list of some of them.

https://github.com/public-apis/public-apis

Tutorial: Dog Facts

Install the Python Requests Library

To retrieve information from API’s, we expand the functionality of default Python. Requests is a Python Library that lets you send HTTP/1.1 requests, add headers, form data, multipart files, and parameters with Python dictionaries. It lets you access the response data in the same way.

Requests are synchronous. Only one HTTP call can be made at a time.

1. In Windows → Click the **Start** button → type **cmd** → Click **Command Prompt (Terminal)**
2. Type the following command.

```
# Windows
pip install requests
```

Linux

```
# Linux
sudo apt update
sudo apt install python3-requests
```

Mac OS

```
# Mac
python3 -m ensurepip
```

Further info for a Mac: https://www.groovypost.com/howto/install-pip-on-a-mac/

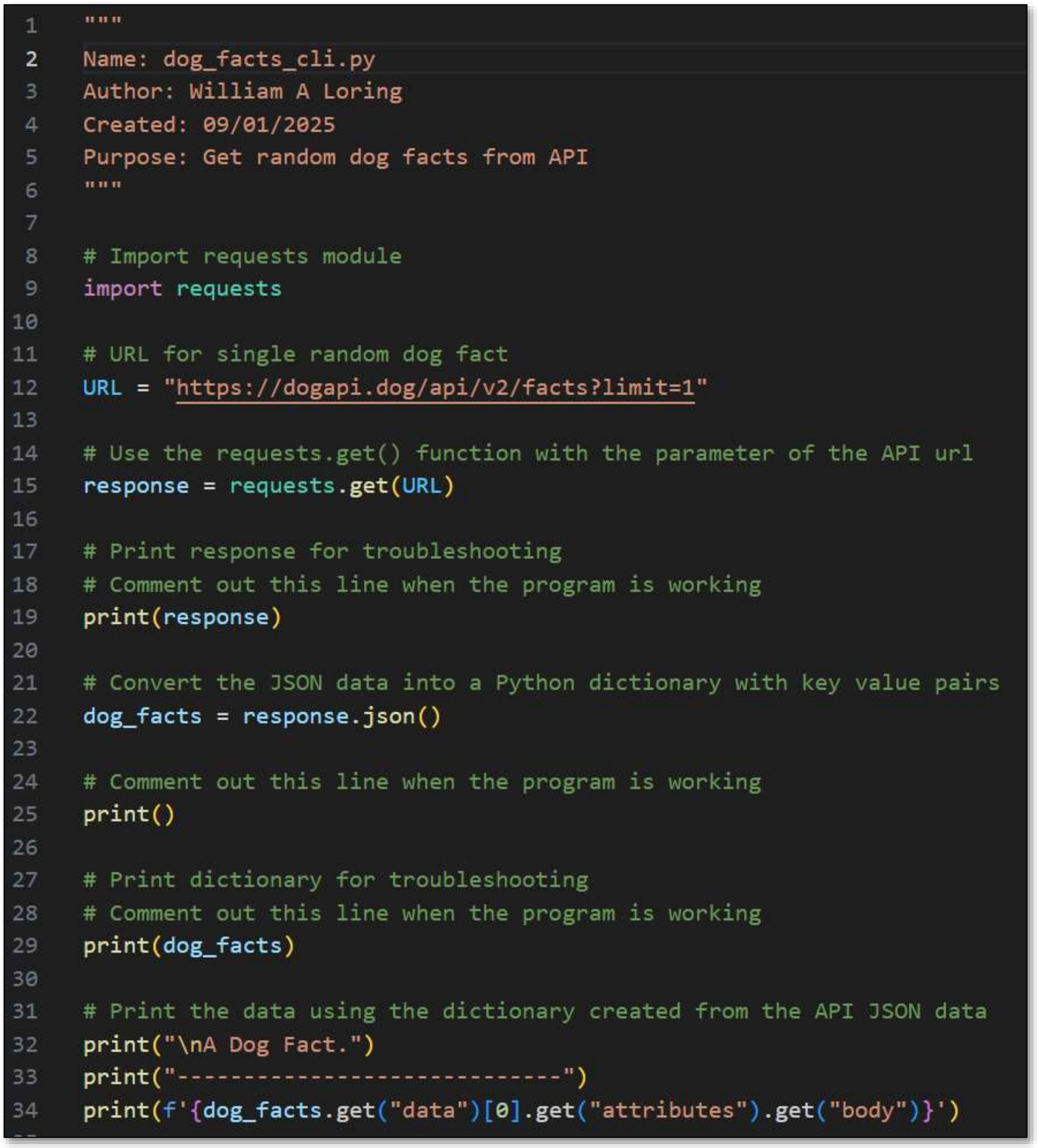
The ensure pip command should either install requests or confirm that it is already installed.

Create the Program

Create a Python file named: **dog_facts.py**

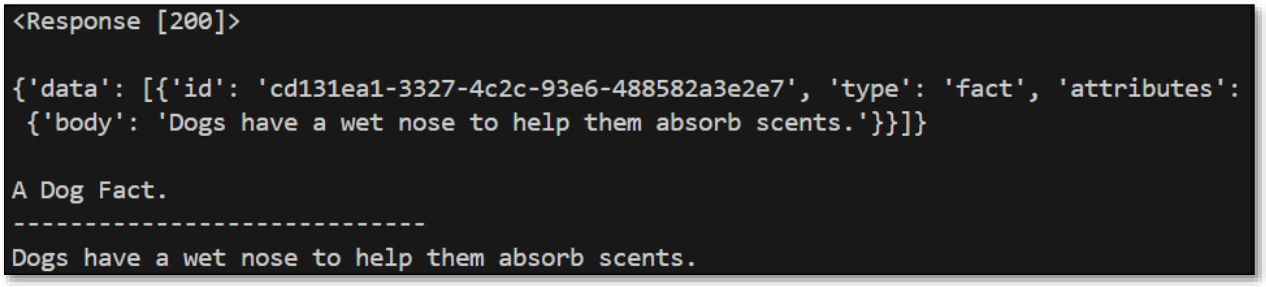
Right Click the URL below → Click **Copy Link location**. Paste this into your program to avoid typo’s.

https://dogapi.dog/api/v2/facts?limit=1



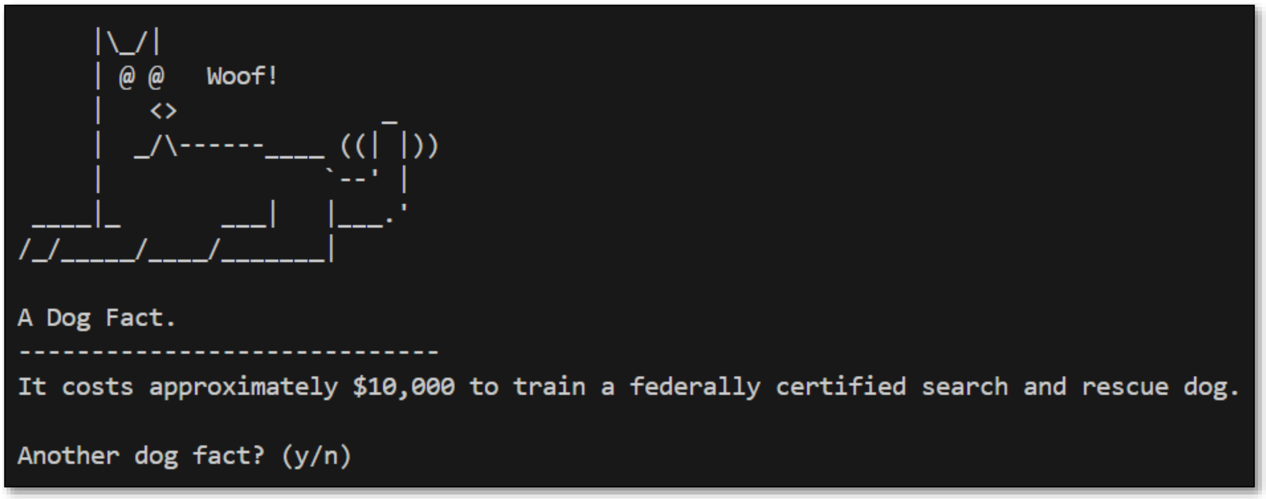
```python
"""
Name: dog_facts_cli.py
Author: William A Loring
Created: 09/01/2025
Purpose: Get random dog facts from API
"""

# Import requests module
import requests

# URL for single random dog fact
URL = "https://dogapi.dog/api/v2/facts?limit=1"

# Use the requests.get() function with the parameter of the API url
response = requests.get(URL)

# Print response for troubleshooting
# Comment out this line when the program is working
print(response)

# Convert the JSON data into a Python dictionary with key value pairs
dog_facts = response.json()

# Comment out this line when the program is working
print()

# Print dictionary for troubleshooting
# Comment out this line when the program is working
print(dog_facts)

# Print the data using the dictionary created from the API JSON data
print("\nA Dog Fact.")
print("----------------------------")
print(f'{dog_facts.get("data")[0].get("attributes").get("body")}')
```

Example run:



```
<Response [200]>

{'data': [{'id': 'cd131ea1-3327-4c2c-93e6-488582a3e2e7', 'type': 'fact', 'attributes':
 {'body': 'Dogs have a wet nose to help them absorb scents.'}}]}

A Dog Fact.
-----------------------------
Dogs have a wet nose to help them absorb scents.
```

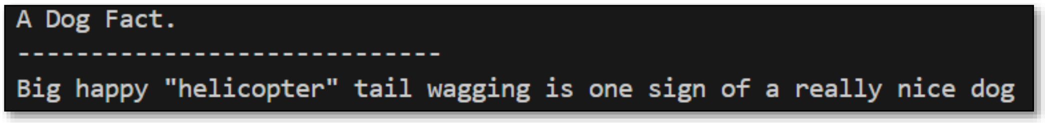
Example run from a student project:



```
     |\_/|
     | @ @   Woof!
     |   <>              _
     |  _/\------____ ((| |))
     |               `--' |
 ____|_       ___|   |___.'
/_/_____/____/_______|

A Dog Fact.
-----------------------------
It costs approximately $10,000 to train a federally certified search and rescue dog.

Another dog fact? (y/n)
```

Comment out the lines mentioned when the program is working.

Final example run:



```
A Dog Fact.
-----------------------------
Big happy "helicopter" tail wagging is one sign of a really nice dog
```

Challenge

Add a menu loop to the program. Allow the user to choose to quit or see another Dog Fact.

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.