

AI Bootcamp

---

# Sourcing Data for AI Projects

Module 6 Day 1



# Class Objectives

By the end of class, you will be able to:

---

- 1 Read and understand data source terms and conditions and licensing agreements.
- 2 Extract data from HTML tables using the Pandas method `from_html()`.
- 3 Make **GET** requests with the Request library.
- 4 Convert JSON into a Python dictionary.
- 5 Convert JSON into a Pandas DataFrame.



# Instructor **Demonstration**

Dataset Licenses

# Creative Commons (CC) Licenses

The following are the most common Creative Commons licenses that will allow you to freely use datasets for your projects, provided you follow the rules set out by the license:

1

**Creative Commons Public Domain Dedication (CC0):** There are no restrictions on what reusers can do, no attribution required, and no restrictions on how the resulting work should be used or licensed. This license renders a work entirely in the public domain.

2

**Creative Commons Attribution (CC BY):** There are no restrictions on what the reuser's work can be used for, as long as the original creator is fairly credited.

3

**Creative Commons Attribution-ShareAlike (CC BY-SA):** There are no restrictions on what the reuser's work can be used for, as long as the original creator is fairly credited and the new work is licensed under the same exact license as the original.

4

**Creative Commons Attribution Non-Commercial (CC BY-NC):** This license prohibits commercial use. It also requires that the original creator be fairly credited. None of the datasets that we work with in this course fall under this license; however, you should keep a sharp eye out for it in your work outside of the course.

# Open Data (OD) Licenses

The following are the most common Open Data licenses that will allow you to freely use datasets for your projects, provided you follow the rules set out by the license:

1

**Open Data Commons Open Database License (ODbL):** Any works created using the original database must credit the original author, as stipulated in the original database, and new works must also be licensed under the ODbL license. Any new works must be kept open.

2

**Open Data Commons Attribution License (ODC-BY):** Any works created using the original database must credit the original author, as stipulated in the original database.

3

**Open Data Commons Public Domain Dedication and License (PDDL):** There are no restrictions on new work created using these databases, as they are considered to be in the public domain.



# Instructor **Demonstration**

Dataset Sources

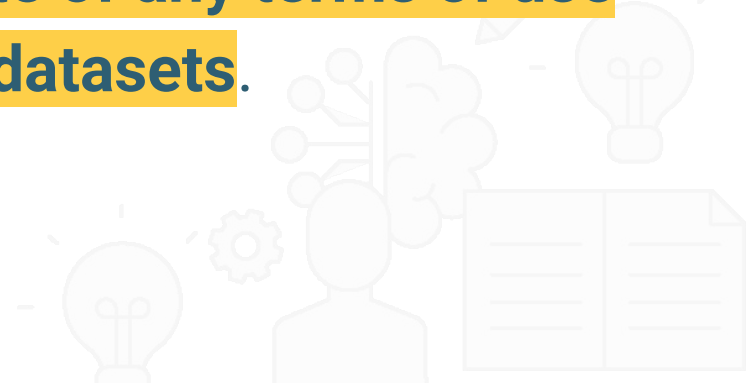
# Dataset Sources Checklist

Datasets are a foundational part of ML applications, so sourcing them is an equally essential skill. Ensure that you do the following when sourcing datasets:

- 1 Check the dataset license.
- 2 Check for terms of use.
- 3 Scrutinize the source. Is it reliable? Is it trustworthy?
- 4 Ensure that you are allowed to use the data for your purpose (e.g., personal or commercial project).



Some institutions, such as government agencies, publish data that is generally of sound integrity and is freely available. You should still, however, **take note of any terms of use before using the datasets.**







## Activity:

Explore Dataset Sources

---

In this activity, you will locate and review terms of use and licenses on various dataset sources.

**Suggested Time:**

5 Minutes





**Time's up!**  
Let's review



**Questions?**



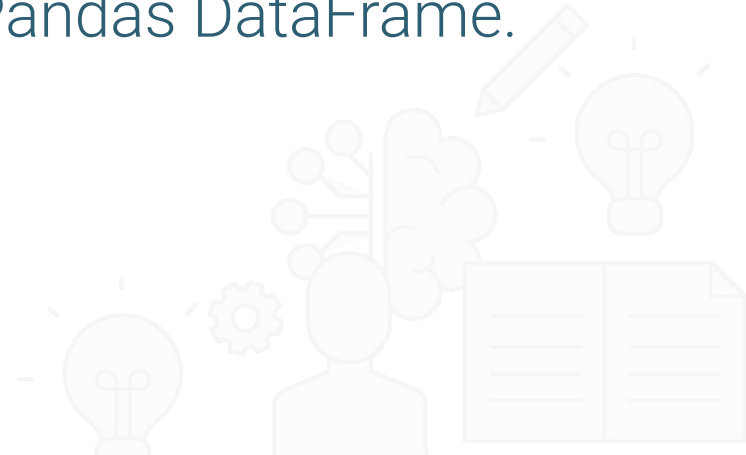


# Instructor **Demonstration**

Sourcing Data from HTML Tables



The **Pandas library** in Python includes a simple web scraper that pulls HTML table data into a Pandas DataFrame.



# Pandas Scraping

Inserting the URL to the method `read_html()` will return the data in a list format.

```
import pandas as pd
```

We can use the `read_html` function in Pandas to automatically scrape any tabular data from a page.

```
url = 'https://en.wikipedia.org/wiki/List_of_Australian_capital_cities'
```

```
tables = pd.read_html(url)
tables
```

```
[
  0
0 WESTERNAUSTRALIA NORTHERNTERRITORY SOUTHAUSTRALIA...
  State/territory Capital City population[2] \
0 Australian Capital Territory Canberra 403468
1 New South Wales Sydney 5029768
2 Northern Territory Darwin 145916
3 Queensland Brisbane 2360241
4 South Australia Adelaide 1324279
5 Tasmania Hobart 224462
6 Victoria Melbourne 4725316
7 Western Australia Perth 2022044

  State/territory population[3] \
0 403468
1 7759274
2 245740
3 4848877
4 1713054
5 517588
6 224462
7 2022044
```

What we get in return is a list of dataframes for any tabular data that Pandas found.

```
type(tables)
```













```
list
```

# Pandas Scrapping

- The Wikipedia page that was scraped contains multiple HTML tables.
- So, the returned list contains each of those tables.
- We can use **list indexing** to specify which table we want to grab.

```
df = tables[1]
df.head()
```

	State/territory	Capital	City population[2]	State/territory population[3]	Percentage of state/territory population in capital city	Established	Capital since	Image
0	Australian Capital Territory	Canberra	403468	403468	100.00%	1913	1913	NaN
1	New South Wales	Sydney	5029768	7759274	64.82%	1788	1788	NaN
2	Northern Territory	Darwin	145916	245740	59.38%	1869	1911	NaN
3	Queensland	Brisbane	2360241	4848877	48.68%	1825	1860	NaN
4	South Australia	Adelaide	1324279	1713054	77.31%	1836	1836	NaN

State and territory capitals of Australia							
State/territory	Capital	City population <sup>[2]</sup>	State/territory population <sup>[3]</sup>	Percentage of state/territory population in capital city	Established	Capital since	Image
 Australian Capital Territory	Canberra	403,468	403,468	100.00%	1913	1913	
 New South Wales	Sydney	5,029,768	7,759,274	64.82%	1788	1788	
 Northern Territory	Darwin	145,916	245,740	59.38%	1869	1911	
				48.68%	1825	1860	
				77.31%	1836	1836	
				43.37%	1804	1826	
				76.47%	1835	1851	
 Western Australia	Perth	2,022,044	2,558,951	79.02%	1829	1829	

# Pandas Scrapping

We often will need to perform a lot of data cleaning on these scraped DataFrames. Examples include:



Dropping unnecessary rows;



Renaming column(s);



Splitting values;



Dropping unnecessary column(s); and



Resetting the index.



# Pandas Scrapping: Rename the Columns

Saves the names of all the columns into a variable.

```
cols = list(df.columns)
```

Renames the 3rd column.

```
cols[2] = "City population"
```

Renames the 4th column.

```
cols[3] = "State/territory population"
```

Saves the new column names back into the DataFrame columns.

```
df.columns = cols
```

```
df.head()
```

# Pandas Scrapping: Rename the Columns

	State/territory	Capital	City population[2]	State/territory population[3]
0	Australian Capital Territory	Canberra	403468	403468

```
cols = list(df.columns)
cols[2] = "City population"
cols[3] = "State/territory population"
df.columns = cols
df.head()
```

	State/territory	Capital	City population	State/territory population
0	Australian Capital Territory	Canberra	403468	403468

# Pandas Scrapping: Dropping Unnecessary Column(s)

```
df = df.drop(['Image'], axis=1)
```

	State/territory	Capital	City population	State/territory population	Percentage of state/territory population in capital city	Established	Capital since	Image
0	Australian Capital Territory	Canberra	403468	403468	100.00%	1913	1913	NaN
1	New South Wales	Sydney	5029768	7759274	64.82%	1788	1788	NaN
2	Northern Territory	Darwin	145916	245740	59.38%	1869	1911	NaN
3	Queensland	Brisbane	2360241	4848877	48.68%	1825	1860	NaN
4	South Australia	Adelaide	1324279	1713054	77.31%	1836	1836	NaN

	State/territory	Capital	City population	State/territory population	Percentage of state/territory population in capital city	Established	Capital since
0	Australian Capital Territory	Canberra	403468	403468	100.00%	1913	1913
1	New South Wales	Sydney	5029768	7759274	64.82%	1788	1788
2	Northern Territory	Darwin	145916	245740	59.38%	1869	1911
3	Queensland	Brisbane	2360241	4848877	48.68%	1825	1860
4	South Australia	Adelaide	1324279	1713054	77.31%	1836	1836



## Activity:

Collect HTML Table Data

---

In this activity, you will extract data from HTML tables with Pandas.

**Suggested Time:**

15 Minutes





**Time's up!**  
Let's review



**Questions?**





# Instructor **Demonstration**

Intro to APIs and the Client-Server Model



**Disclaimer:** The response and content of live data cannot be censored or predicted.






# Introduction to **APIs**





## **Application Programming Interfaces (APIs)**

are a set of functions packaged together that provide developers with a means of communicating with a server and integrating third-party software and technology into new applications.



# Application Programming Interfaces (APIs)

01

APIs are developed by companies looking to offer programmatic services and functions to the development community.

02

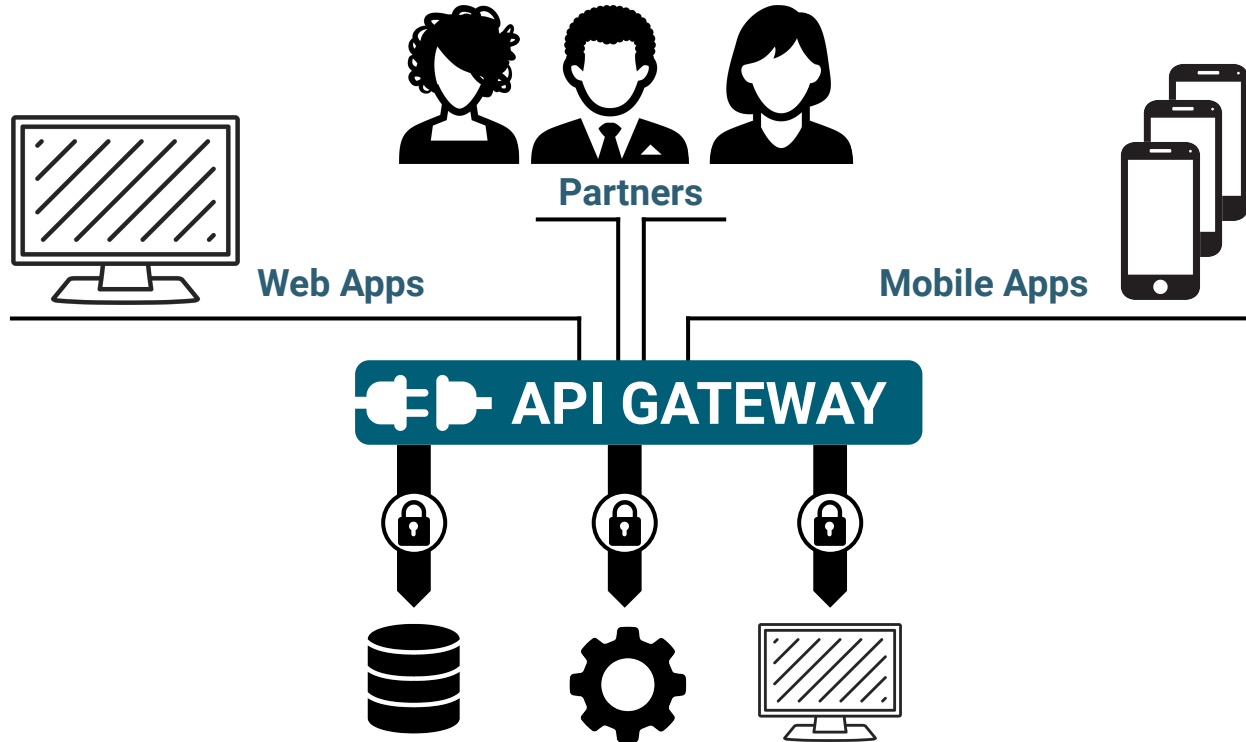
APIs are used to extract data, play games, connect programs to platforms like AWS, and manage personal finances.

03

APIs work like old-school telephone operators. Users submit a request or call to a website or server, and the operator connects them to their party. In this case, the API is the operator.

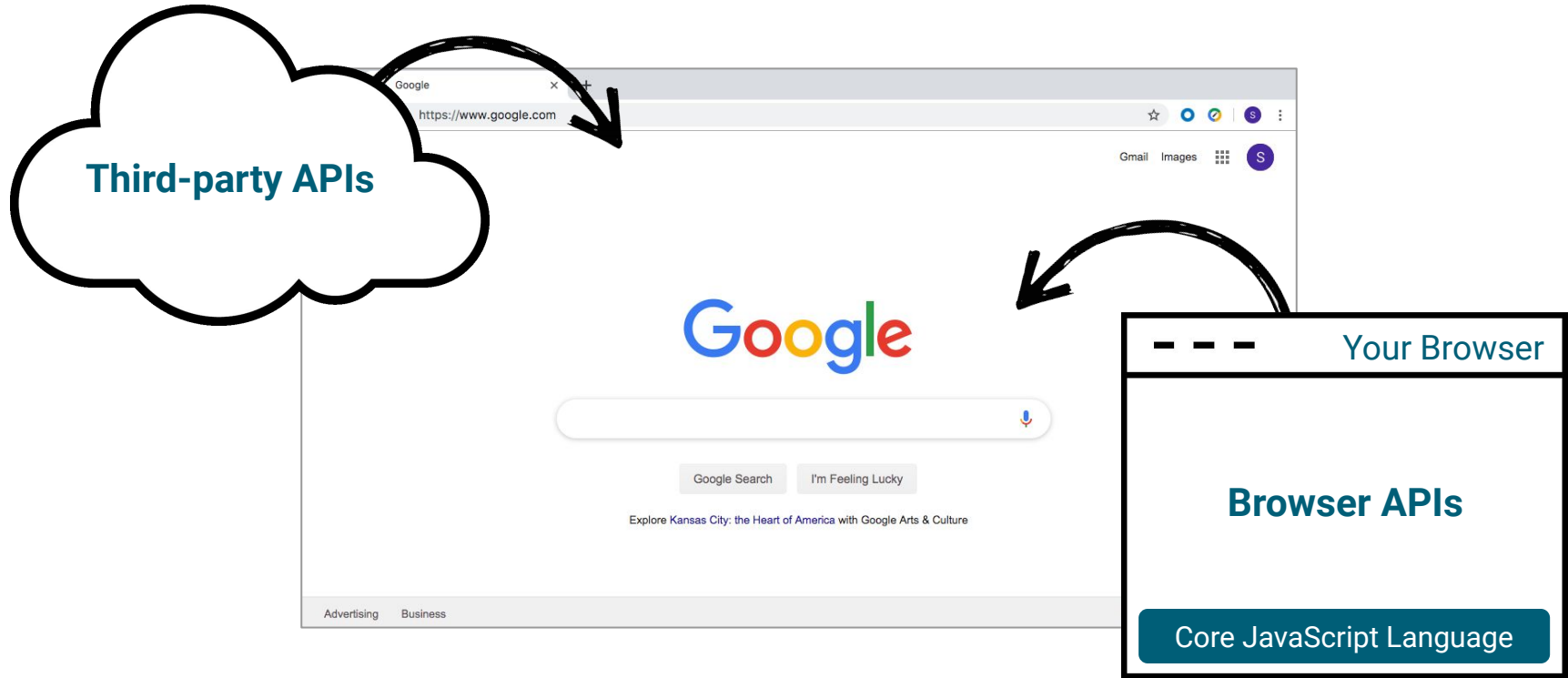
# API Recap

In software development, APIs are often the bridge between different components.



# API Recap

In software development, APIs are often the bridge between different components.

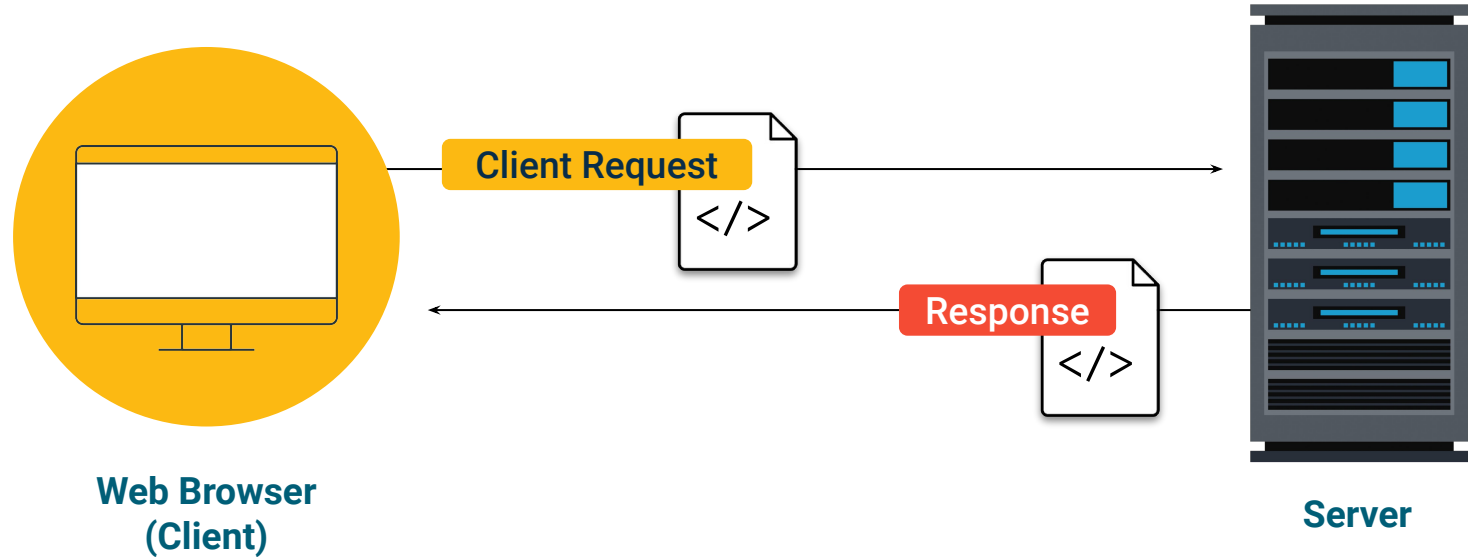


# The Client–Server **Model**



# Client–Server Model

The Client–Server Model is a structure that outlines the relationship and flow of communication between two components: a client and a server.



# The Client–Server Model

A client is any tool or application that is used to connect to or communicate with a server. Example **clients** include:



Web browsers;



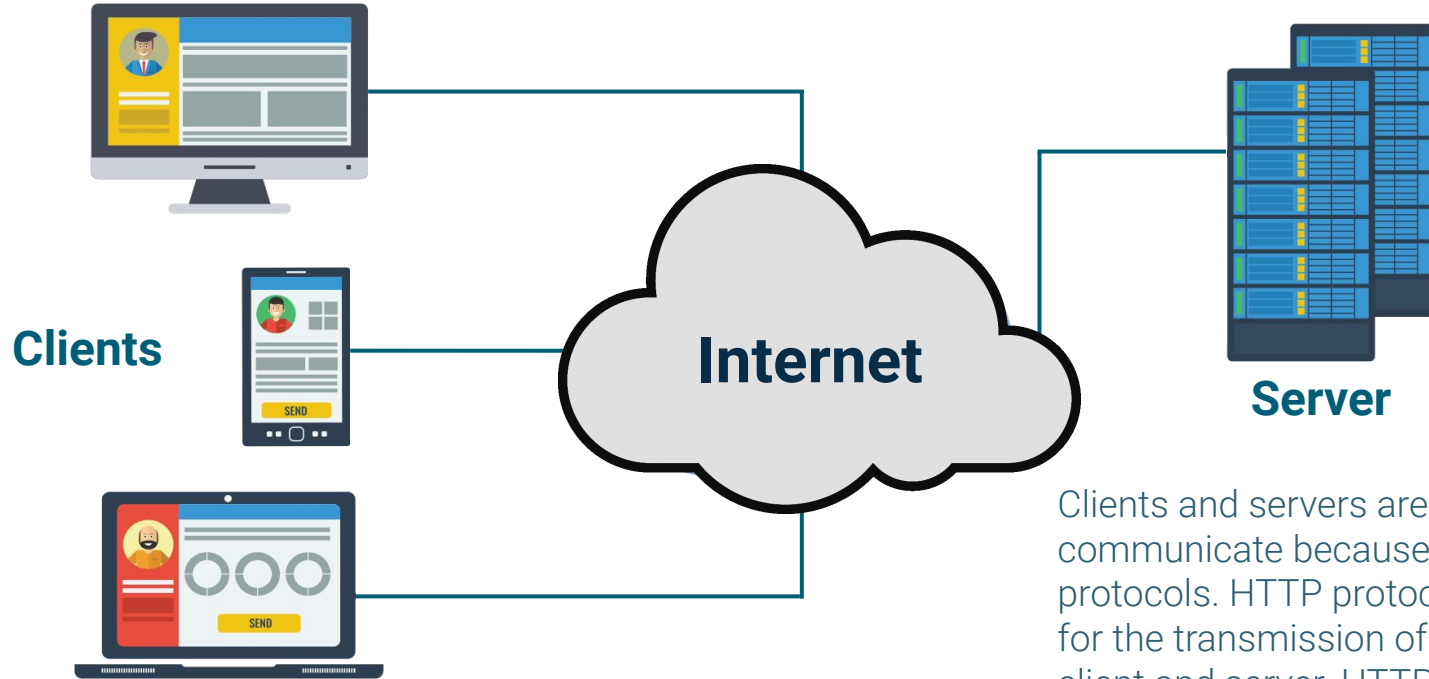
Mobile devices; and



Command-line interfaces.



# The Client–Server Model

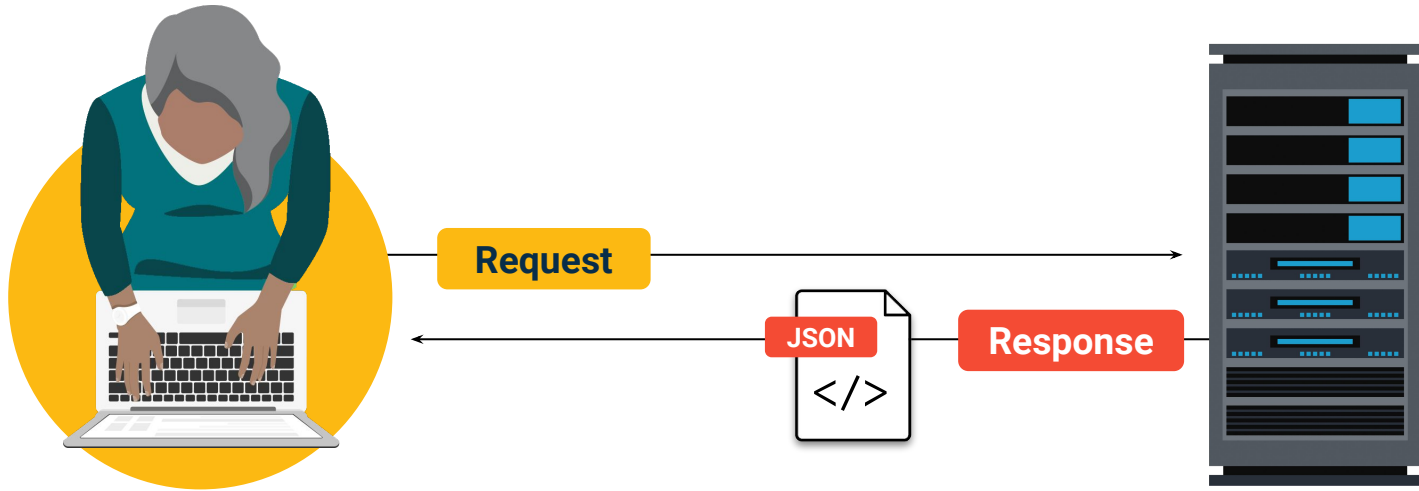


Clients submit requests to servers and then wait for a response from the server.

Clients and servers are able to communicate because of HTTP protocols. HTTP protocols are policies for the transmission of data between client and server. HTTP protocols allow users to execute server functions and clients and servers to exchange data.

# What is a Client versus a Server?

**Analogy:** A patient asks a health question, and a doctor supplies the answer.



A client is an application or device that asks for information.

A server is an application or device that supplies information to the client.



What is an **API**?



# Application Programming Interface (API)



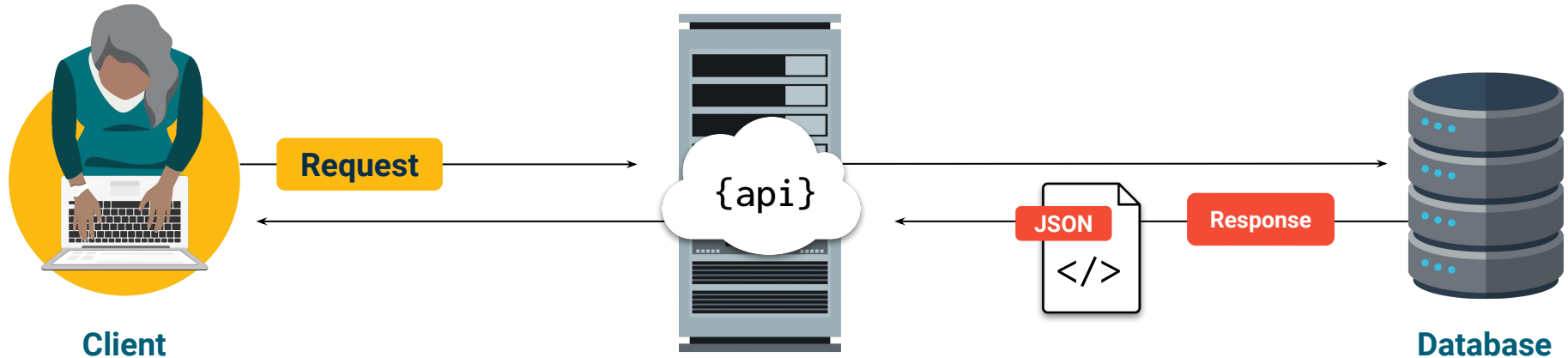
A request is a communication to the API to retrieve data.



API calls are similar to visiting a website in a browser.



They point to a URL and collect some data from the page.



# JavaScript Object Notation (JSON)



A webpage may return a JSON in response to an API call.

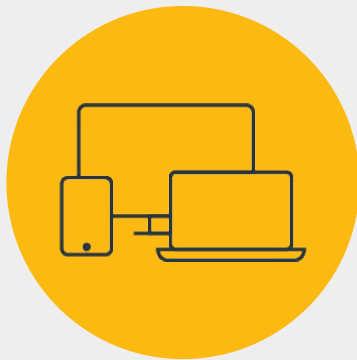


The URLs used to communicate with APIs are called endpoints.



The text in the web browser is identical to what a client script would receive.

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
  },
]
```



# Instructor **Demonstration**

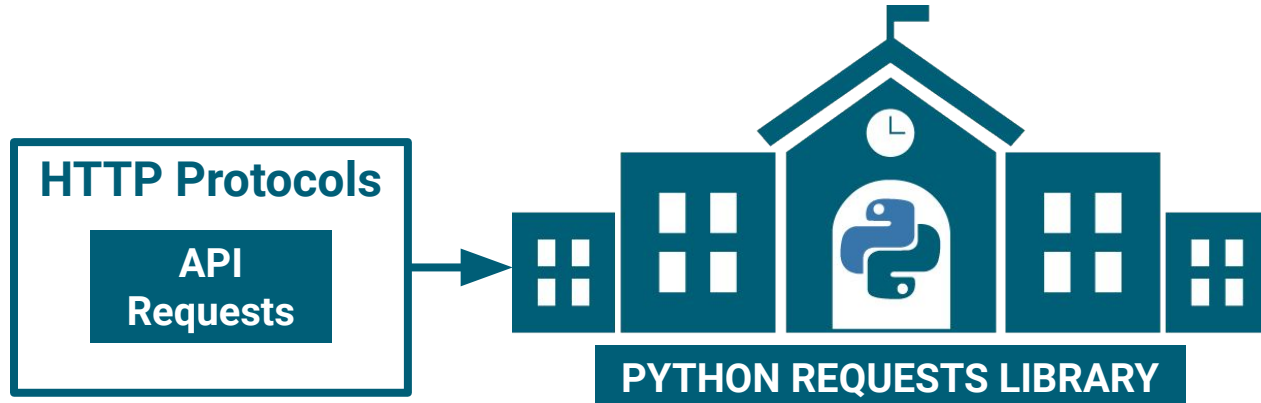
Requests and JSON

# Python **Requests**



# Python Requests

Python has a requests library that can be used to make API calls. The requests library allows developers to submit API requests using HTTP protocols.



The requests library allows developers to use Python like glue, connecting their Python code with multiple third-party APIs. This allows developers to create programs that are an amalgamation of multiple technologies!



# Python Requests

The requests library has its own functions, such as **GET** and **POST**. These can be used to execute API calls programmatically with Python.

With Python's request library, **developers can use Python objects (such as variables and data structures) to make APIs interact with one another** when they normally wouldn't. This allows developers to:

- Pass the output of one API as input to another API;
- Utilize conditionals; and
- Leverage loops.

# Python Requests

Each type of request serves a different purpose:

## GET

GET requests are used to extract/acquire data from a server.

## POST

POST requests are used to push new or updated data to the server.

## PUT

PUT requests are used to overwrite content on the server.

# Common Response Codes

Most APIs incorporate programming that will return code with each server response. These are called response codes. Here are common response codes and their meanings:

<b>100s</b> Informational	<b>200s</b> Success	<b>300s</b> Redirection	<b>400s</b> Client Error	<b>500s</b> Server Error
------------------------------	------------------------	----------------------------	-----------------------------	-----------------------------

1

For example, you've likely encountered a 404 error when trying to access a webpage. This is a client error indicating that the client couldn't find the page you requested.

2

If you are querying APIs an important error code to be aware of is the 401 response code, returned for unauthorized requests. This means that authentication is required to access the content.

3

You might also encounter a 429 response code, which happens when too many requests are being made of a client. Some APIs, like the New York Times APIs, limit the number of requests you're allowed to make in a given timeframe.

# There are two components to our API request

1

`requests.get(url)` Sends a get request to the URL, passed as a parameter.

```
# Dependencies
```

```
import requests
```

```
import json
```

```
# URL for GET requests to retrieve data
```

```
url = "http://api.worldbank.org/v2/country/us/indicator/NY.GDP.MKTP.CD?format=json"
```

```
# Print the response object to the console
```

```
print(requests.get(url))
```

## There are two components to our API request

2 A call to convert the response object into a JSON format, **json.dumps()** is a method used to “pretty print” the response.

```
D": "page 1", "pages": 2, "per_page": 50, "total": 63, "sourceid": "not_established", "2023-07-25"}, [{"indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2022", "value": 25462708000000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2021", "value": 23315080560000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2020", "value": 21060473513000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "Y.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2019", "value": 21380976119000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2018", "value": 20533057312000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2017", "value": 19477336549000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2016", "value": 18695110842000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2015", "value": 18206020741000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2014", "value": 17550680174000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2013", "value": 16643190993000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2012", "value": 1625397223000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "U S", "value": "United States", "countryiso3code": "USA", "date": "2011", "value": 15599728123000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2010", "value": 15048964444000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2009", "value": 14478064934000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "U S", "value": "United States", "countryiso3code": "USA", "date": "2008", "value": 14769857911000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2007", "value": 1447426905000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2006", "value": 13815586948000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States", "countryiso3code": "USA", "date": "2005", "value": 13039199193000, "unit": "", "obs_status": "decimal:0"}, "indicator": {"id": "NY.GDP.KMTP.CD", "value": "GDP (current US$)", "country": {"id": "US", "value": "United States",
```

```
[
  {
    "page": 1,
    "pages": 2,
    "per page": 50,
    "total": 63,
    "sourceid": "2",
    "lastupdated": "2023-07-25"
  },
  [
    {
      "indicator": {
        "id": "NY.GDP.MKTP.CD",
        "value": "GDP (current US$)"
      },
      "country": {
        "id": "US",
        "value": "United States"
      },
      "countryiso3code": "USA",
      "date": "2022",
      "value": 25462700000000,
      "unit": "",
      "obs_status": "",
      "decimal": 0
    },
    {
      "indicator": {
        "id": "NY.GDP.MKTP.CD",
        "value": "GDP (current US$)"
      },
      "country": {
        "id": "US",
        "value": "United States"
      },
      "countryiso3code": "USA",
      "date": "2022",
      "value": 25462700000000,
      "unit": "",
      "obs_status": "",
      "decimal": 0
    }
  ]
]
```



# Activity:

## Introductory API Requests

---

In this activity, you will:

- Use the requests library to make a **GET** request.
- Use `json.dumps` to format JSON output.
- Extract data from a JSON object.

**Suggested Time:**

20 Minutes





**Time's up!**  
Let's review



**Questions?**







**Break**

15 mins



# Instructor **Demonstration**

URL Parameters

# URL Parameters

URL parameters serve as a means of configuring and changing API functionality.

Parameters can be specified in one of two ways. Parameters can follow forward slashes (/) or be specified by parameter name and then by parameter value.

Parameter provided after /

```
http://numbersapi.com/42
```

Parameter provided using parameter name and value

```
http://numbersapi.com/random?min=10?json
```

# URL Parameters

URL



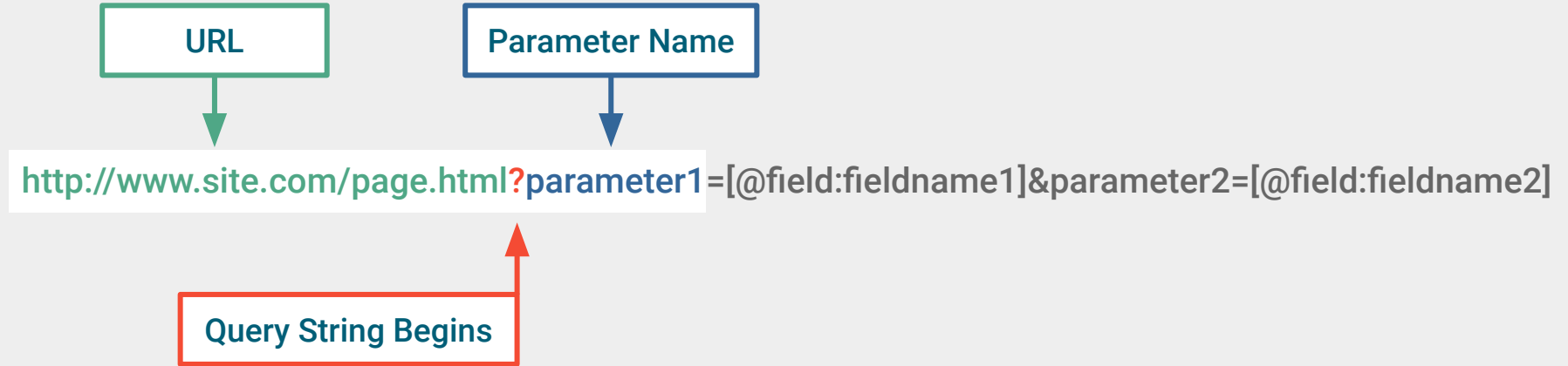
```
graph TD; A[URL] --> B[http://www.site.com/page.html?parameter1=[@field:fieldname1]&parameter2=[@field:fieldname2]]
```

[http://www.site.com/page.html?parameter1=\[@field:fieldname1\]&parameter2=\[@field:fieldname2\]](http://www.site.com/page.html?parameter1=[@field:fieldname1]&parameter2=[@field:fieldname2])

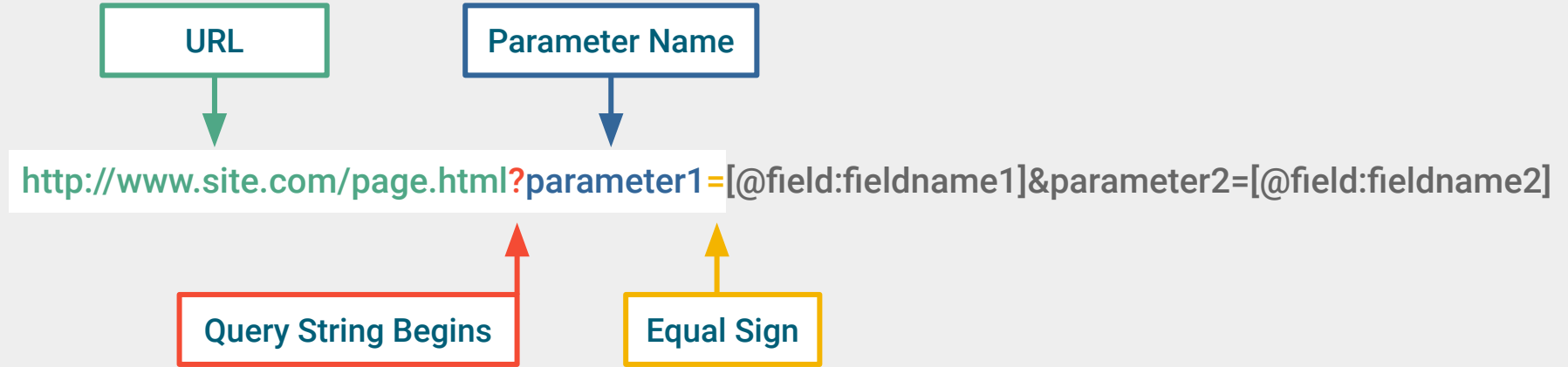
# URL Parameters



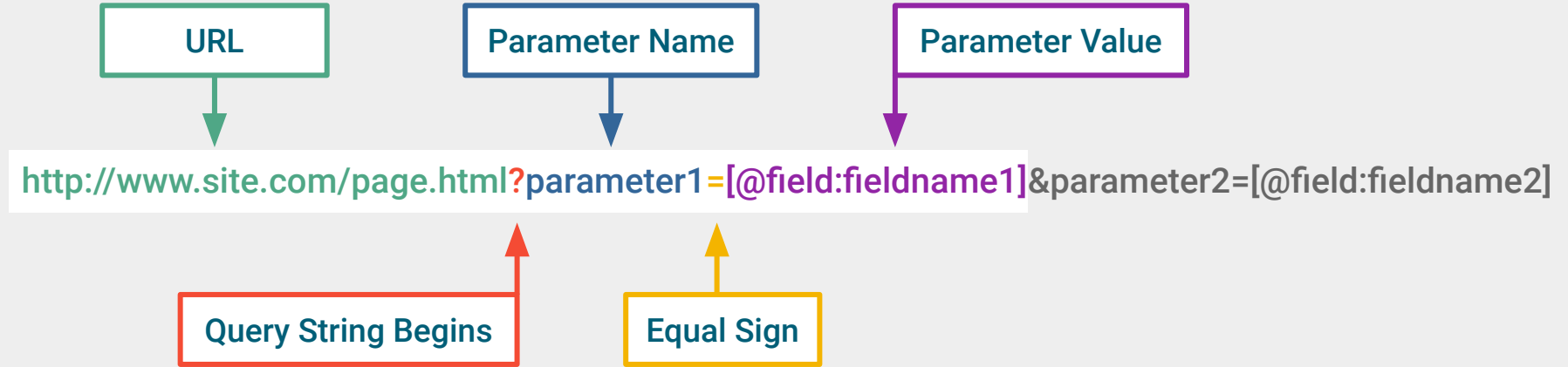
# URL Parameters



# URL Parameters

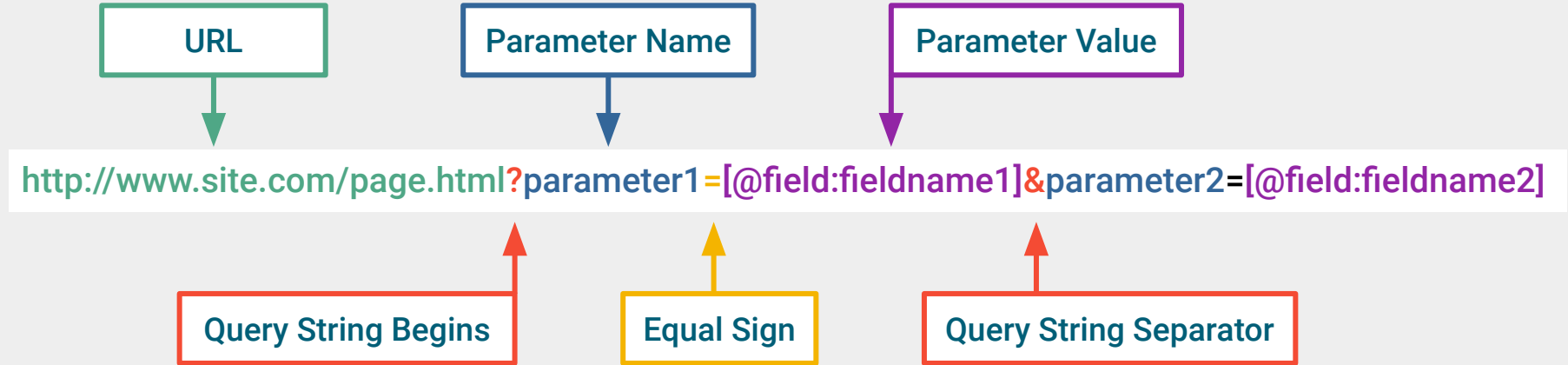


# URL Parameters





# URL Parameters





## Activity:

### House of Requests

---

In this activity, you will use parameters to create API requests that are used to play a virtual game of Blackjack.

**Suggested Time:**

5 Minutes



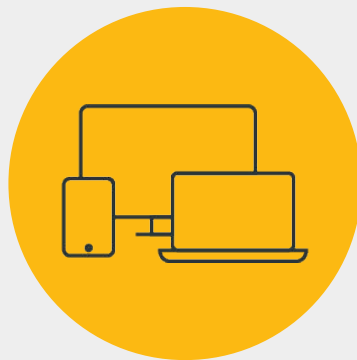


**Time's up!**  
Let's review



**Questions?**





# Instructor **Demonstration**

JSON to Pandas and Iterative Requests

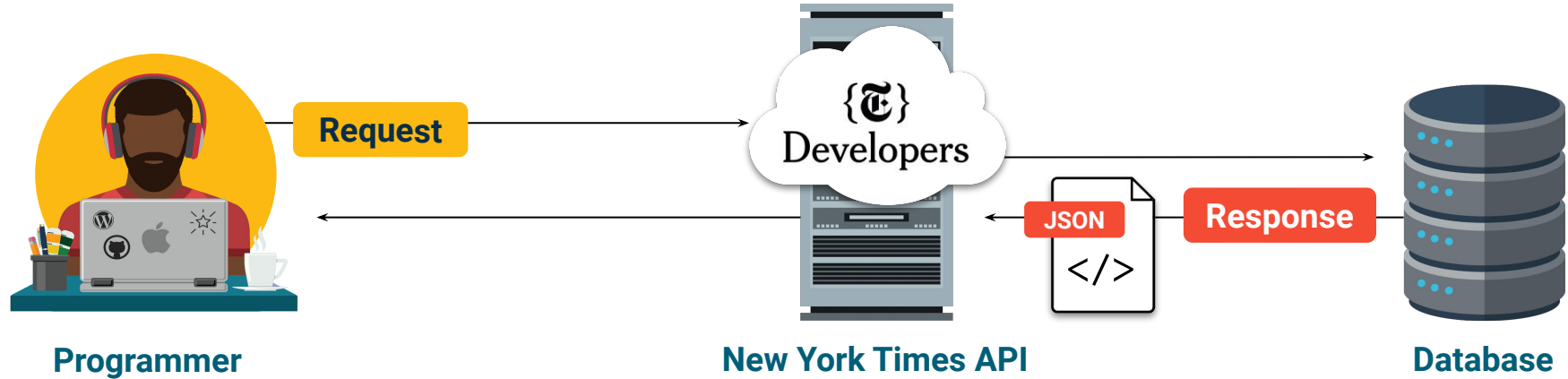


**So far**, we have been able to get all the information we've requested using single requests.



# Some APIs require multiple requests

For example, the New York Times API will return only 10 articles at a time. A programmer would have to make 3 requests to retrieve 30 articles.



# Requests on a loop!

```
# Make a request for 3 pages of results
for x in range(3):
    print(f"Making request number: {x}")

    # Get the results
    post_response = requests.get(url + "?offset=" + str(x * 50))

    # Loop through the "entries" of the results and
    # append them to the response_json list
    for result in post_response["entries"]:
        # Save post's JSON
        response_json.append(result)
```

```
Making request number: 0
Making request number: 1
Making request number: 2
```





## Activity:

TV Ratings DataFrame

---

In this activity, you will make iterative requests and convert JSON to Pandas DataFrames.

**Suggested Time:**

5 Minutes



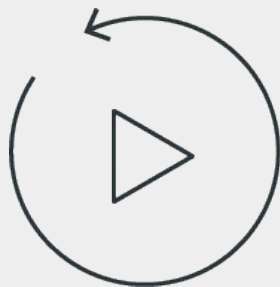


**Time's up!**  
Let's review



**Questions?**





Let's **recap**



# Recap

After today's lesson, you are able to:

---

- 1 Read and understand data source terms and conditions and licensing agreements.
- 2 Extract data from HTML tables using the Pandas method `from_html()`.
- 3 Make `get` requests with the Request library.
- 4 Convert JSON into a Python dictionary.
- 5 Convert JSON into a Pandas DataFrame.



## Next

In the next lesson, you'll learn best practices on keeping API keys secure and then practice submitting requests to the New York Times and the OpenWeather APIs.



**Questions?**





**The End**