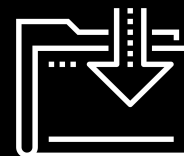




# Unit 6 Day 1 - Python APIs

Web Development  
Lesson 6.1



# Class Objectives

---

By the end of today's class you will be able to:



Make GET requests with Python requests library



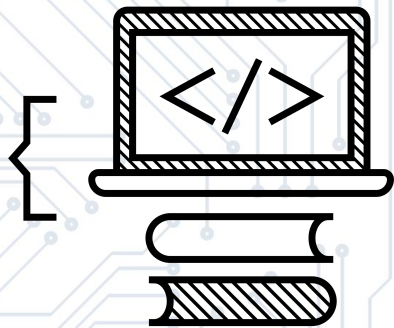
Manipulate JSON response to retrieve necessary values



Store JSON responses in Python lists and dictionaries

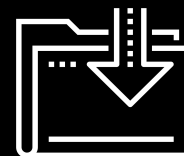


Identify and generate the type of request needed to request movies by leveraging the OMDb API documentation



## Unit 6 Day 1 - Python APIs

Web Development  
Lesson 6.1



# What is a client versus a server?

---



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



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



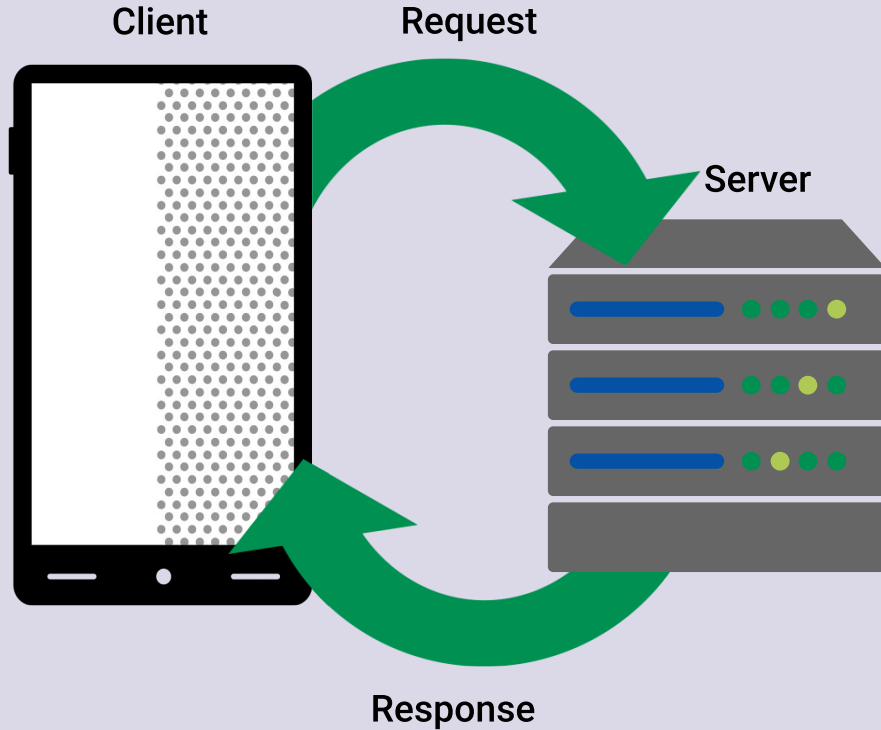
Analogy: A doctor and patient



What is an API?

# Application Programming Interface

---



- A request is a communication to the API to retrieve data.
- API calls are similar to visiting a website manually. They point to a URL and collect some data from the page.

# JSON

---

```
[
  {
    "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"
  },
]
```

- A web page may return a JSON in response to an API call.
- A URL used to communicate with APIs are called endpoints.
- The text in the web browser is identical to what a client script would receive.



# Instructor Demonstration

## Intro to Requests



# There are two components to our API request

01

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 vehicle data
url = "https://api.spacexdata.com/v2/launchpads"
```

```
# Print the response object to the console
print(requests.get(url))
```

02

.json()

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

```
# Pretty Print the output of the JSON
response = requests.get(url).json()
print(json.dumps(response, indent=4, sort_keys=True))
```

```
[
  {
    "details": "SpaceX primary Falcon 9 launch pad, where all east coast Falcon 9s launch ed prior to the AMOS-6 anomaly. Initially used to launch Titan rockets for Lockheed Martin. H eavily damaged by the AMOS-6 anomaly with repairs expected to be complete by late summer 201 7.",
    "full_name": "Cape Canaveral Air Force Station Space Launch Complex 40",
    "id": "ccafs_slc_40",
    "location": {
      "latitude": 28.5618571,
      "longitude": -80.577366,
      "name": "Cape Canaveral",
      "region": "Florida"
    },
    "status": "under construction",
    "vehicles_launched": "falcon 9"
  },
  {
    "details": "SpaceX new launch site currently under construction to help keep up with the Falcon 9 and Heavy manifests. Expected to be completed in late 2018. Initially will be li
```

# <Time to Code>





## Activity: Requesting SpaceX

In this activity you will dig into a rather simple and well-documented API - The SpaceX API - and make calls to the API using the Requests library.

(Instructions sent via Slack.)

**Suggested Time:**  
10 Minutes



# Requesting SpaceX Instructions

---

- Take a few minutes to explore the SpaceX V3 API:
  - [GitHub](<https://github.com/r-spacex/SpaceX-API>)
  - [API Documentation](<http://bit.ly/SpaceXAPI>)
- Once you understand the structure of the API and its endpoint, choose one of the endpoints and do the following:
  - Retrieve and print the JSON for *all* of the records from your chosen endpoint.
  - Retrieve and print the JSON for the a *specific* record from your chosen endpoint.

# Example SpaceX Response

---

```
{
  "details": "SpaceX west coast landing pad, has not yet been used. Expected to first be used during the Formosat-5 launch.",
  "full_name": "Vandenberg Air Force Base Space Launch Complex 4W",
  "id": "vafb_slc_4w",
  "location": {
    "latitude": 34.6332043,
    "longitude": -120.6156234,
    "name": "Vandenberg Air Force Base",
    "region": "California"
  },
  "status": "active",
  "vehicles_launched": "falcon 9"
}
```



**Time's Up!** Let's Review.



# Instructor Demonstration

## Manipulating Responses

# Working with JSON Responses

---

01

## Simple Method

Use `requests.get()`, store the output, and print the JSON response

- Must interpret the full JSON object each time
- More difficult to import into pandas
- Less scalable

02

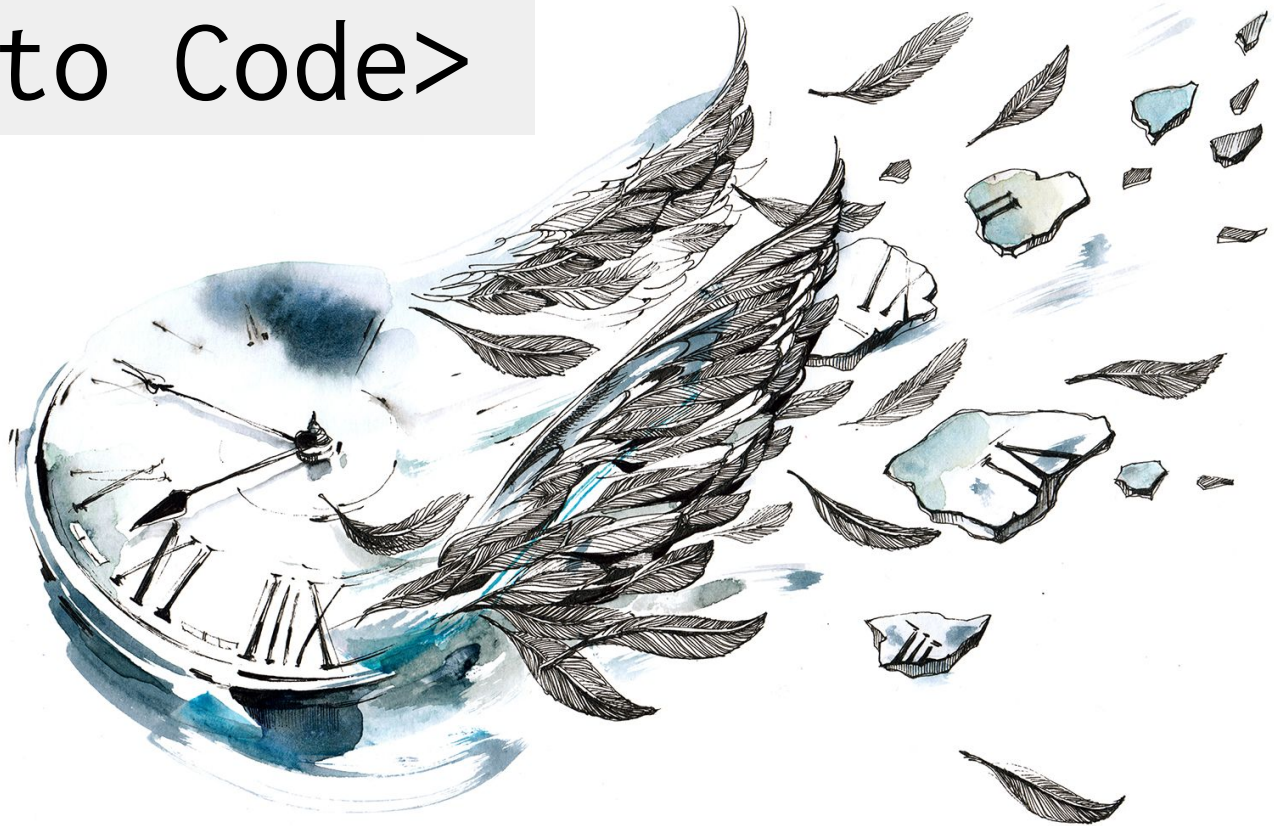
## Advanced Method

Store the `requests.get()` object, store the `response.json()` and access the JSON object directly

- Navigate the JSON object like a dictionary
- Easy to import into pandas
- More Scalable



# <Time to Code>





## Activity: Requesting a Galaxy Far Far Away

In this activity you will create an application that accesses data from the Star Wars API and prints out values from within it.

(Instructions sent via Slack.)

**Suggested Time:**  
15 Minutes



# Requesting a Galaxy Far Far Away Instructions

---

- Using the starter file provided, collect the following pieces of information from the Star Wars API.
  - The name of the character
  - The number of films they were in
  - The name of their first starship
- Once the data has been collected, print it out to the console.
- **Hints:**
  - It would be in your best interest to print out the JSON from the initial request before anything else. This will let you know what keys you should reference.
  - The "starship" values are links to another API call. This means that you will need to create a request based off of the values of a previous request.
- **Bonus:** Collect and print out all of the films a character appeared in.

# Requesting a Galaxy Far Far Away

---

```
# Print character name and how many films they were in  
print(f"{character_name} was in {film_number} films")
```

Darth Vader was in 4 films

```
# Print what their first ship was  
print(f"Their first ship: {first_ship}")
```

Their first ship: TIE Advanced x1



**Time's Up!** Let's Review.



## Activity: Number Facts

In this activity you and a partner will join forces in creating an interactive application that uses the "numbers" API. The application will take in a number and then return a random fact about that number.

(Instructions sent via Slack.)

**Suggested Time:**  
20 Minutes



# Number Facts Instructions

---

- Using the [Numbers API](<http://numbersapi.com>), create an application that takes in a user's inputs and returns a number fact based upon it.
- **Hints:**
  - The URL to make your request to must have `?json`` at its end so that the data format returned is JSON. The default response is pure text.
  - Make sure to read through the documentation when creating your application. Some types require more or less data than others.



**Time's Up!** Let's Review.



# Take a Break!

---

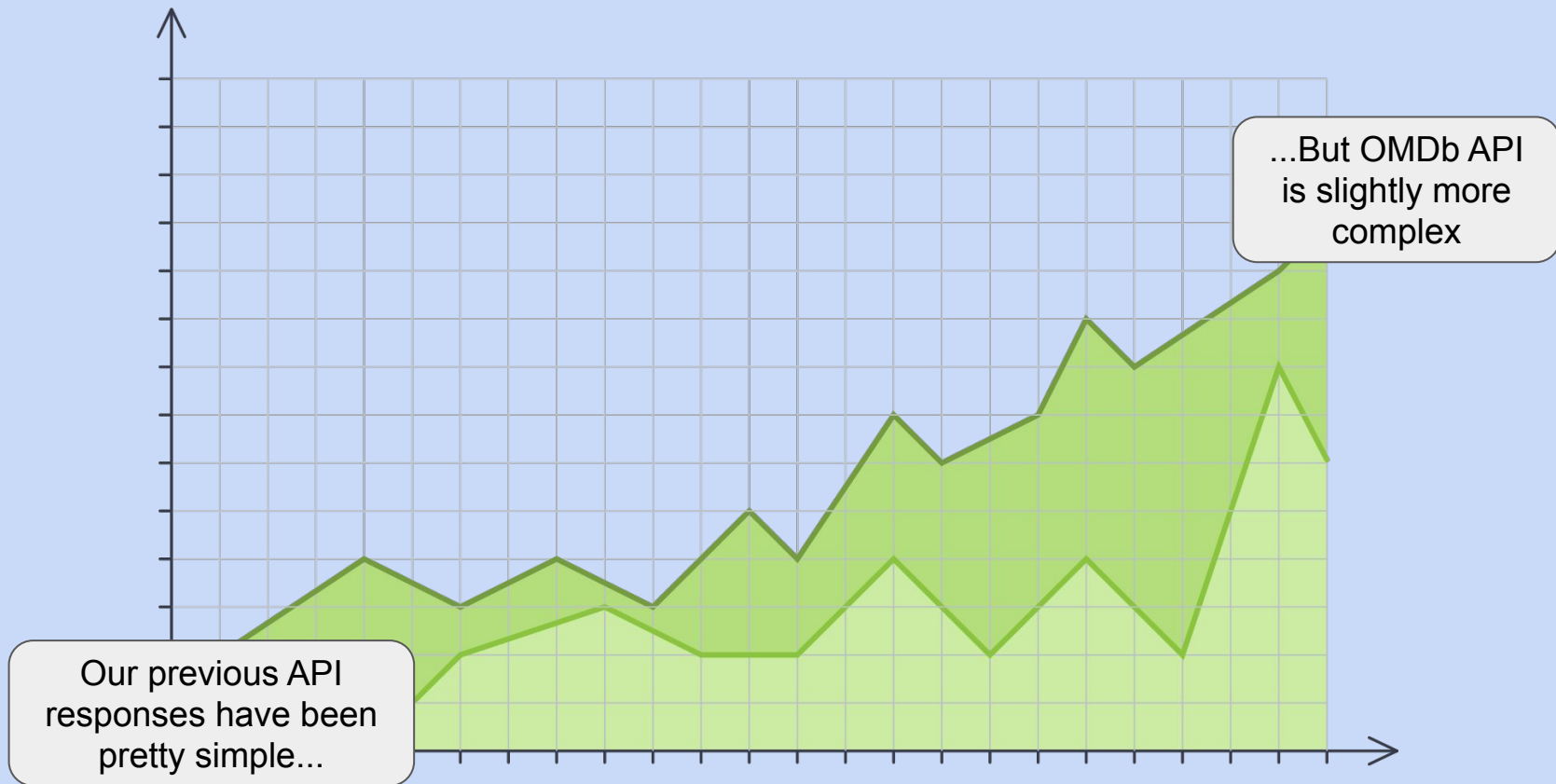




# Instructor Demonstration

## OMDb API

# JSON responses so far



# URL Parameters

01

?t=

- Stands for title.
- Example:

<http://www.omdbapi.com/?t=Aliens>

- This is asking the API to return all information on movies with the title "Aliens."

02

api\_key

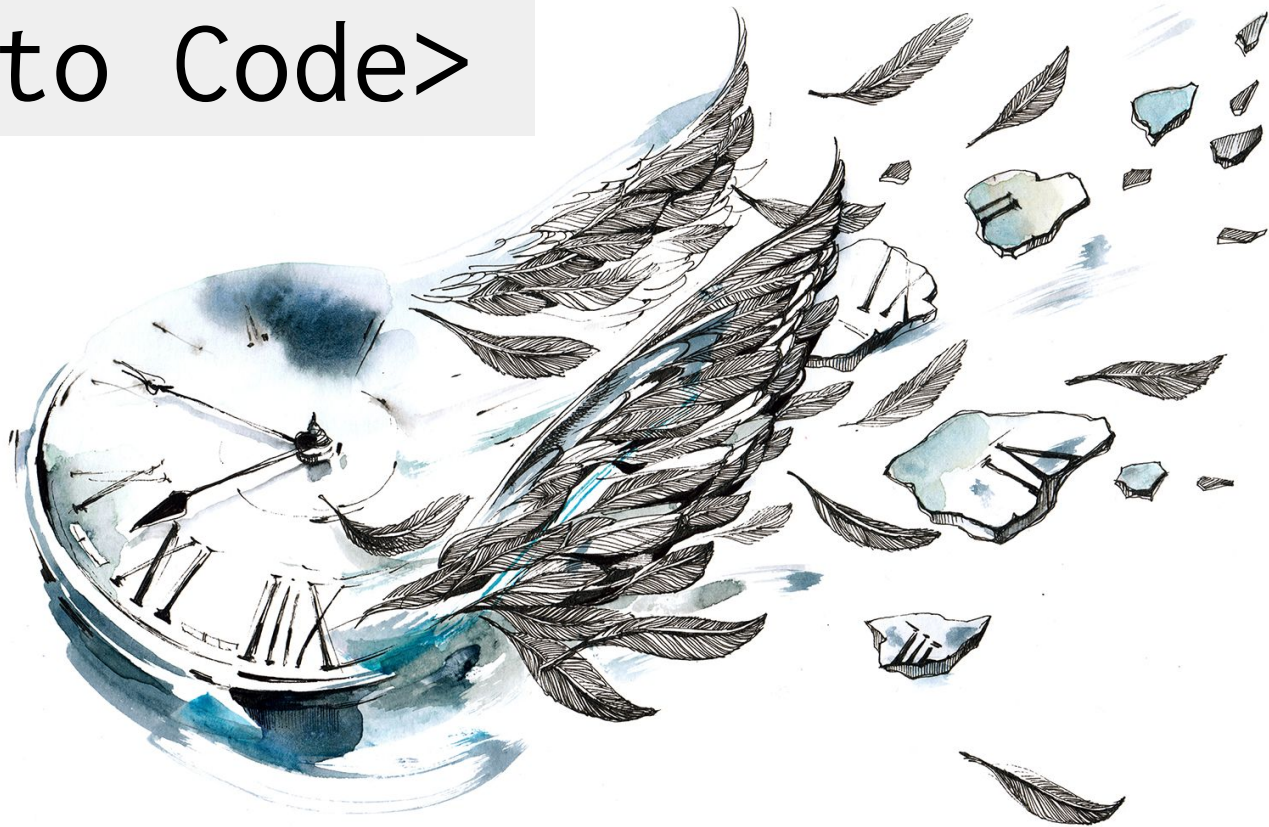
- API Keys restrict API access to specific users
- Without an API key no data would be returned.

```
# Note that the ?t= is a query param for the t-itle of the
# movie we want to search for.
url = "http://www.omdbapi.com/?t="
api_key = "&apikey=trilogy"
```

```
# Performing a GET request similar to the one we executed
# earlier
response = requests.get(url + "Aliens" + api_key)
print(response.url)
```

<http://www.omdbapi.com/?t=Aliens&apikey=trilogy>

# <Time to Code>





## **Activity: Study the OMDb API**

In this activity, you will take some time reviewing the documentation for OMDb API and testing it out!

(Instructions sent via Slack.)

**Suggested Time:**  
5 Minutes



# Study the OMDb API Instructions

---

- Read the OMDb documentation, and make a few API calls to get some information about your favorite movie:
  - <http://www.omdbapi.com/>





## Activity: Movie Questions

In this activity, you will now test your skills with the OMDB API and collect some data from the API in order to answer a series of questions.

(Instructions sent via Slack.)

**Suggested Time:**  
20 Minutes





# Move Questions Instructions

---

- Use the OMDb API to retrieve and print the following information.
  - Who was the director of the movie **Aliens**?
  - What was the movie **Gladiator** rated?
  - What year was **50 First Dates** released?
  - Who wrote **Moana**?
  - What was the plot of the movie **Sing**?



**Time's Up!** Let's Review.



# Instructor Demonstration

## Iterative Request

# Our Requests so far

---

- All information being requested so far has worked from a single request.
- Sometimes, APIs will only respond with some of the information.
  - Some APIs limit the amount of data in response to a single API call.
  - 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 each of the indices
for x in range(len(indices)):
    print(f"Making request number: {x} for ID: {indices[x]}")

    # Get one of the posts
    post_response = requests.get(url + str(indices[x]))

    # Save post's JSON
    response_json.append(post_response.json())
```

```
Making request number: 0 for ID: 30
Making request number: 1 for ID: 54
Making request number: 2 for ID: 44
Making request number: 3 for ID: 76
Making request number: 4 for ID: 46
Making request number: 5 for ID: 75
Making request number: 6 for ID: 69
Making request number: 7 for ID: 40
Making request number: 8 for ID: 48
Making request number: 9 for ID: 5
```

# <Time to Code>





## Activity: Iterative Requests

In this activity, you will test your knowledge of iterative requests by looping through a list of movies and collecting data from the OMDB API on each movie.

(Instructions sent via Slack.)

**Suggested Time:**  
10 Minutes



# Movie Questions

---

The director of Aliens was James Cameron.

The rating of Gladiator was R.

The movie 50 First Dates was released in 2004.

Moana was written by Jared Bush (screenplay by), Ron Clements (story by), John Musker (story by), Chris Williams (story by), Don Hall (story by), Pamela Ribon (story by), Aaron Kandell (story by), Jordan Kandell (story by).

The plot of Sing was: In a city of humanoid animals, a hustling theater impresario's attempt to save his theater with a singing competition becomes grander than he anticipates even as its finalists' find that their lives will never be the same..



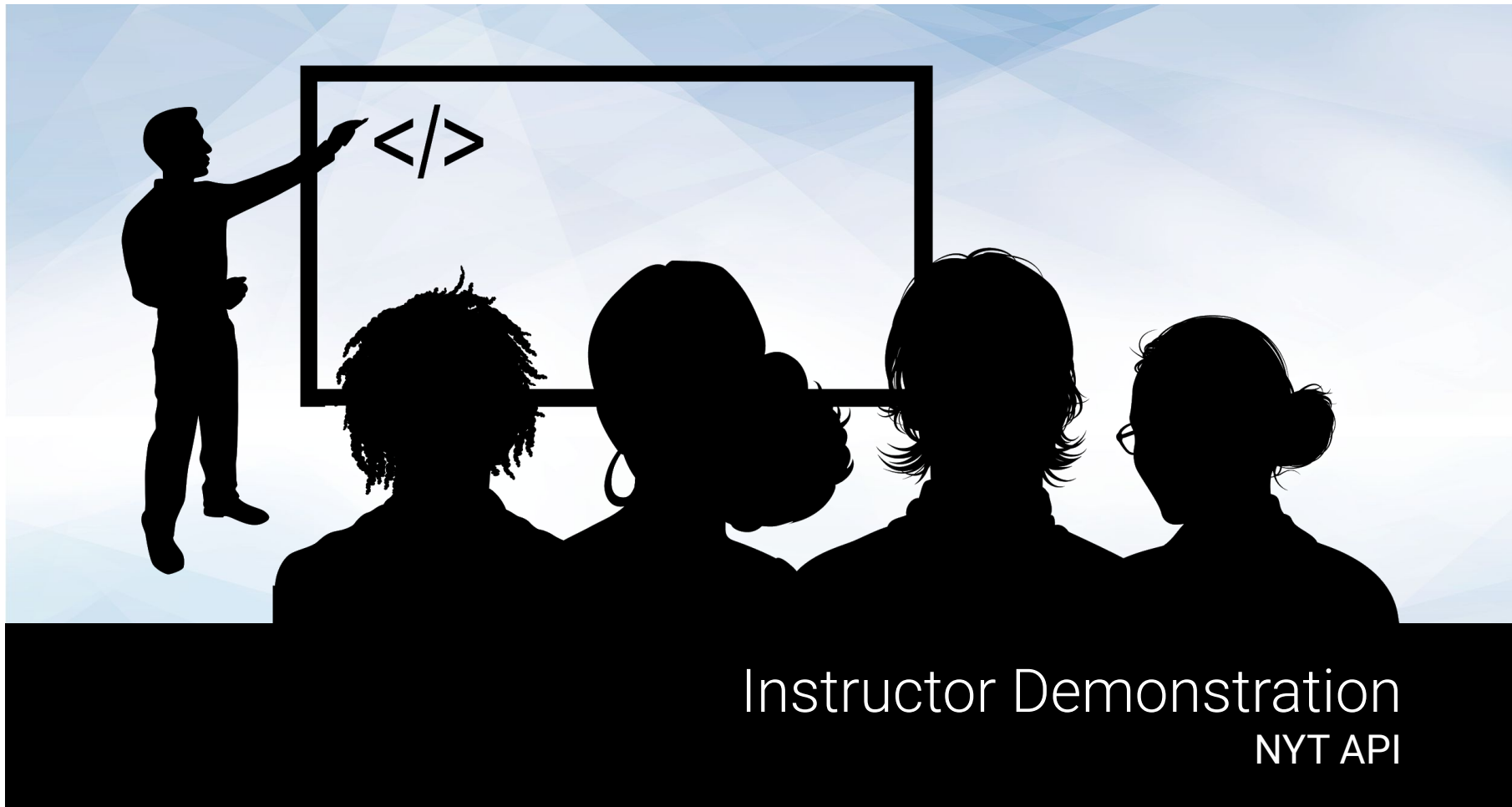
# Movie Questions Instructions

---

- Consider the following list of movie titles:
  - `movies = ["Aliens", "Sing", "Moana"]`
- Make a request to the OMDb API for each movie in the list. Then:
  - Print the director of each movie
  - Save the responses in another list



**Time's Up!** Let's Review.



# Instructor Demonstration

NYT API

# NYT API Signup

---

- This API requires a sign up.
- Fill out the form on <https://developer.nytimes.com/signup>

## Create a New API Key

Create a new API key or retrieve an existing one.

Name

Email

Web Site


API

☐ I'm not a robot   
reCAPTCHA  
Privacy - Terms

*By clicking "Create API Key" you're accepting our [Terms of Use](#).*

**Create API Key**

# NYT API Documentation

 **Article Search API** Source: [Swagger 2.0] [README](#) [Documentation](#) [Console](#)

q

Stories

GET /articlesearch.json

## Stories

### GET /articlesearch.json

Article Search

Article Search requests use the following URI structure:

Hide details ↑ Try it out →

**Parameters**

**q** string  
Location: query ?q=xyz  
Search query term. Search is performed on the article body, headline and byline.

**fq** string  
Location: query ?fq=xyz  
"Filtered search query using standard Lucene syntax.  
The filter query can be specified with or without a limiting field: label.  
See Filtering Your Search for more information about filtering."

**begin date** string

**Responses**

**200**

The docs requested by the article search.

[Schema](#) [Example](#)

```
▼ {
  response: ▼ {
    docs: ▼ [
      ► {}
    ]
    meta: ▼ {
      hits: integer
      time: integer
      offset: integer
    }
  }
}
```

# NYT API Signup

- When using API keys be sure to store in a config.py file.
- Add the config file to a .gitignore file so keys don't get added to a public repo.



The image shows a Jupyter Notebook interface. At the top, a file explorer shows a file named 'config.py' with a checkmark and the date '11/28/2017'. Below this, the notebook contains a Python script. Two blue arrows point from the 'config.py' file to the script: one points to the line 'from config import api\_key' in the dependencies section, and the other points to the line 'api\_key = "164b73c522a8420c8e05343ef1da0a7e"' in the script body.

```
# Dependencies
import requests
from pprint import pprint
from config import api_key

url = "https://api.nytimes.com/svc/search/v2/articlesearch.json?"

# Search for articles that mention granola
query = "granola"

# Build query URL
query_url = url + "api-key=" + api_key + "&q=" + query

# Request articles
articles = requests.get(query_url).json()

# The "response" property in articles contains the actual articles
# list comprehension.
articles_list = [article for article in articles["response"]["docs"]]
pprint(articles_list)

# Print the web_url of each stored article
print("Your Reading List")
for article in articles_list:
    print(article["web_url"])
```

# <Time to Code>





## Activity: Retrieving Articles

In this activity, you will create an application that grabs articles from the NYT API, stores them within a list, and prints snippets of the articles to the screen.

(Instructions sent via Slack.)

**Suggested Time:**  
20 Minutes





# Retrieving Articles

---

A snippet from the article: When the president goes to the Illinois Capitol next month to speak, he will call for a less divisive politics, aides say.

A snippet from the article: The first time the president met with House Democrats, in 2009, there were 257 of them. Now there are 188. Some point a finger at him.

A snippet from the article: Tucked into a piece of legislation are a few words that will change how American officials treat Israeli settlements in the West Bank.

A snippet from the article: "We must confront the reality that around the world anti-Semitism is on the rise," the president urged at an event to posthumously honor individuals who protected Jews during the Holocaust.

A snippet from the article: Katie Beirne Fallon, the president's legislative director, has been credited with improving his relations with lawmakers on Capitol Hill.

A snippet from the article: Mrs. Clinton also said that Mr. Sanders's proposal for a single-payer health care system would thrust the nation into "a terrible, terrible national debate."

A snippet from the article: A federal order will deprive local authorities of critical tools in an age of heightened fears about terrorism and mass shootings, some law enforcement leaders say.

A snippet from the article: The new rules, announced by the Obama administration, allow banks to provide direct financing for products other than agricultural commodities.

A snippet from the article: A compromise to reaffirm concealed-carry reciprocity with 25 states looks like a sellout to gun-control advocates.

A snippet from the article: Nader Modanlo, released by the United States this month in a prisoner exchange with Iran, was reluctant to accept the deal because it required him to abandon an appeal of his conviction.



# Retrieving Articles Instructions

---

- Save the following to variables in your script:
  - The NYT API endpoint - make sure you include the right query parameter to retrieve JSON data!
  - Your selected search term
- Build your query URL, and save it to a variable.
- Retrieve your list of articles with a GET request.
- Store each article in the response inside of a list.
- Print a ``snippet`` from each article.





**Time's Up!** Let's Review.