Data I/O and Preprocessing with SQL and Python

Module 1: Web scraping & text preprocessing

DeepLearning.AI

Web scraping & text preprocessing

Welcome to this course!

DeepLearning.AI

Web scraping & text preprocessing

Generative AI in this course

DeepLearning.AI

Generative AI in this course

In this course, you'll learn how to use LLMs to:

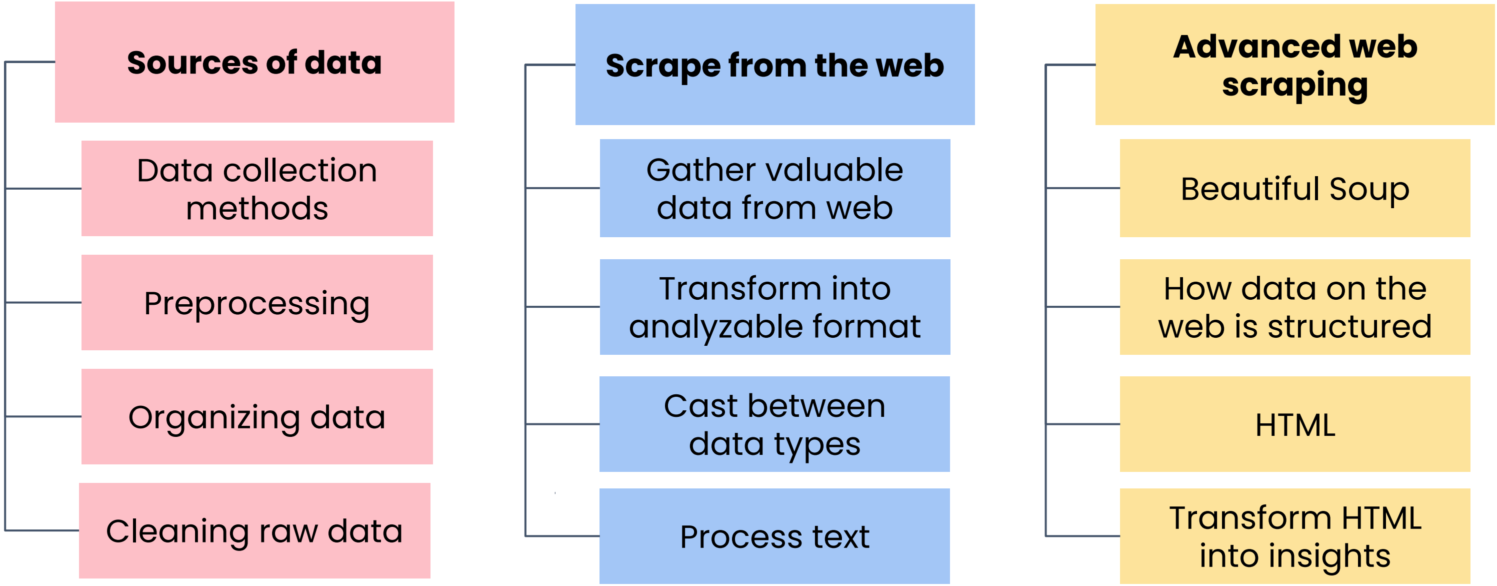
- Find and fix code and database query errors
- Write code based on comments
- Explore new preprocessing methods
- Interpret the results of an analysis
- And more!

Web scraping & text preprocessing

Module 1 introduction

DeepLearning.AI

Module 1 outline



Sources of data

- Data collection methods
- Preprocessing
- Organizing data
- Cleaning raw data

Scrape from the web

- Gather valuable data from web
- Transform into analyzable format
- Cast between data types
- Process text

Advanced web scraping

- Beautiful Soup
- How data on the web is structured
- HTML
- Transform HTML into insights

Web scraping & text preprocessing

The many sources of data

DeepLearning.AI

Data sources

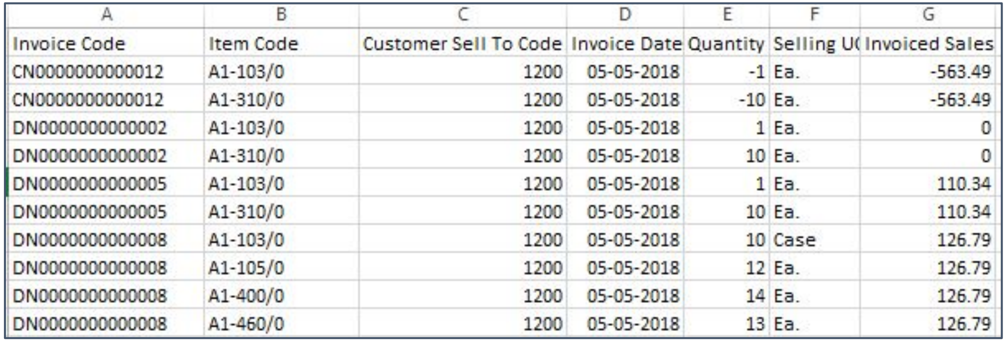
- **Scenario**: Working on analysis
  - ⬜ Load Jupyter notebook
  - ⬜ Load data into DataFrame
  - ⬜ Use .csv file to read in data
- In practice, you will work with wide variety of data sources
  - Flat file
  - Databases
  - APIs
  - Web scraping

Types of information

- **Generated**
  - Example: Sales records
  - Data source: Database
- **Collected**
  - Examples: Competitor prices, weather
  - Data source: Web scraping, APIs
- **Historical**
  - Example: Investment transactions
  - Data source: Flat file like .csv
    - File containing single table

Flat files (like .csv)

- 🎯 **Task**: Analyzing purchasing trends
- 📊 **Dataset:** Sales data
- ✅ For **smaller, static dataset**:
  - .csv might be suited for task
  - Save it on your laptop
  - Make quick changes
  - Load it into Python notebook with a few lines of code



A	B	C	D	E	F	G
Invoice Code	Item Code	Customer Sell To Code	Invoice Date	Quantity	Selling U(	Invoiced Sales
CN000000000012	A1-103/0	1200	05-05-2018	-1	Ea.	-563.49
CN000000000012	A1-310/0	1200	05-05-2018	-10	Ea.	-563.49
DN000000000002	A1-103/0	1200	05-05-2018	1	Ea.	0
DN000000000002	A1-310/0	1200	05-05-2018	10	Ea.	0
DN000000000005	A1-103/0	1200	05-05-2018	1	Ea.	110.34
DN000000000005	A1-310/0	1200	05-05-2018	10	Ea.	110.34
DN000000000008	A1-103/0	1200	05-05-2018	10	Case	126.79
DN000000000008	A1-105/0	1200	05-05-2018	12	Ea.	126.79
DN000000000008	A1-400/0	1200	05-05-2018	14	Ea.	126.79
DN000000000008	A1-460/0	1200	05-05-2018	13	Ea.	126.79

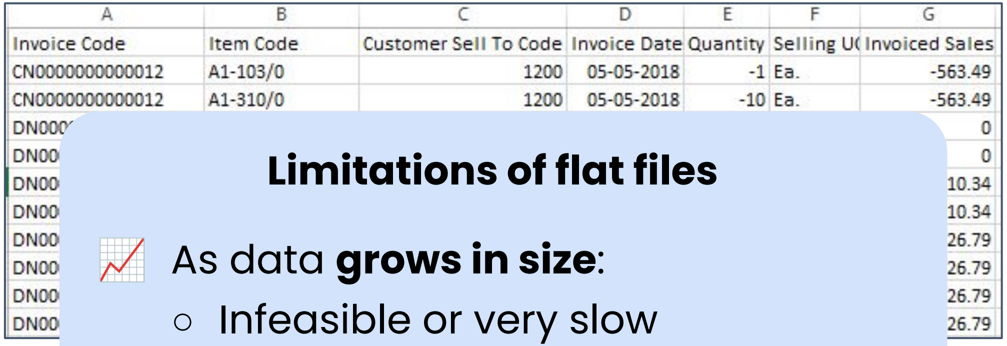
Flat files (like .csv)

🎯 **Task**: Analyzing purchasing trends

📊 **Dataset:** Sales data

✅ For **smaller, static dataset**:

- .csv might be suited for task
- Save it on your laptop
- Make quick changes
- Load it into Python notebook with a few lines of code



A	B	C	D	E	F	G
Invoice Code	Item Code	Customer Sell To Code	Invoice Date	Quantity	Selling U(	Invoiced Sales
CN0000000000012	A1-103/0	1200	05-05-2018	-1	Ea.	-563.49
CN0000000000012	A1-310/0	1200	05-05-2018	-10	Ea.	-563.49
DN000						0
DN00						0
DN00						10.34
DN00						10.34
DN00						26.79
DN00						26.79
DN00						26.79
DN00						26.79

Limitations of flat files

📈 As data **grows in size**:

- Infeasible or very slow

🧩 As it **grows in complexity**:

- Need data in multiple tables with *relationships*
- Need to store and process *unstructured* data

👥 As **more need to use** the file:

- Need to manage access

DeepLearning.AI

Sean Barnes

Databases

- Think of a database as:
    - Many flat files connected by relationships
    - Optimized for speed and efficient storage
- Example: Analyzing purchasing trends
    - 📄 A flat file for customers
    - 📄 A flat file for purchases
    - 🛢 Database can store:
        - Customers and purchases
        - Relationships between each
- Large companies likely have **internal** database
    - Access data directly and load into notebook

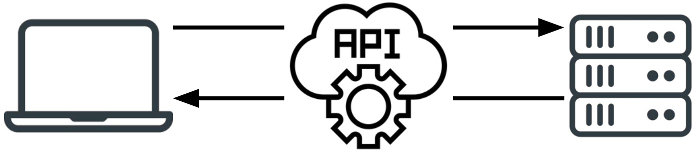
APIs

🎯 **Task**: Understand customer sentiment towards your brand

📊 **Dataset:** Online reviews

- **Structured:** 📍 **New York**
- **Unstructured:** ▤

Application Programming Interface (API)



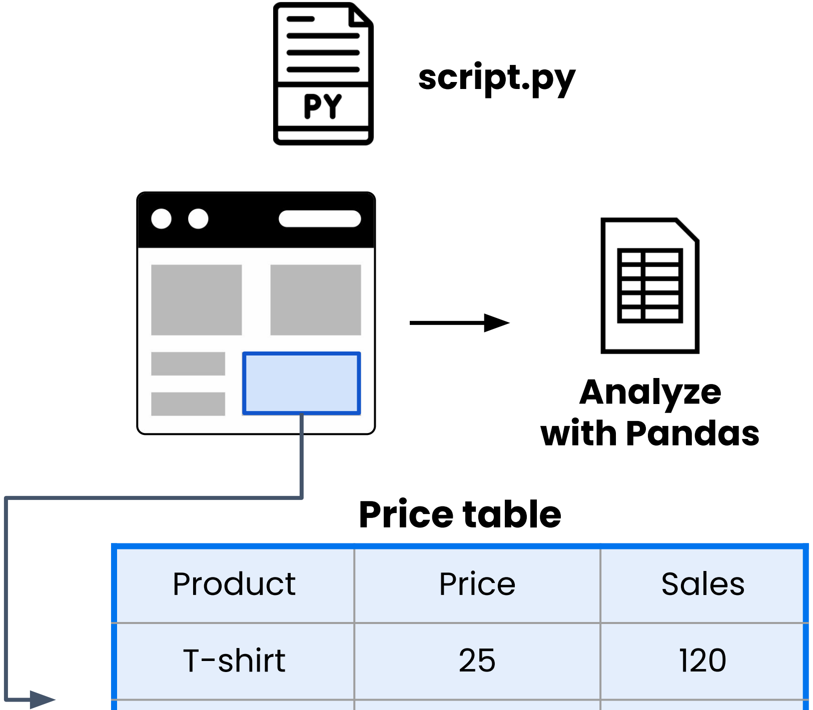
- Request data from a company's server
- Real-time access to high quality, up-to-date information

Web scraping

🎯 **Task**: Track competitor prices

📊 **Dataset:**

- ⛔ Isn't made available through a flat file, database, or an API
- ✅ Try scraping it from their website



Price table

Product	Price	Sales
T-shirt	25	120
Mug	15	250
Book	30	80

Web scraping & text preprocessing

Data cleaning and processing

DeepLearning.AI

Data in the real world

- You explored the **“Ask a manager”** salary dataset

- Includes responses about:
  - Job role
  - Industry
  - Salary

⛔ Inconsistencies!

Preprocessing

1. **Start with raw data:**
   - ○ Unprocessed information in its original form
2. **Make data suitable for analysis:**

⬜ Removing duplicates

⬜ Handling missing values

⬜ Handling outliers

⬜ Fixing inconsistent formatting ⬅

⬜ Selecting a subset of features ⬅

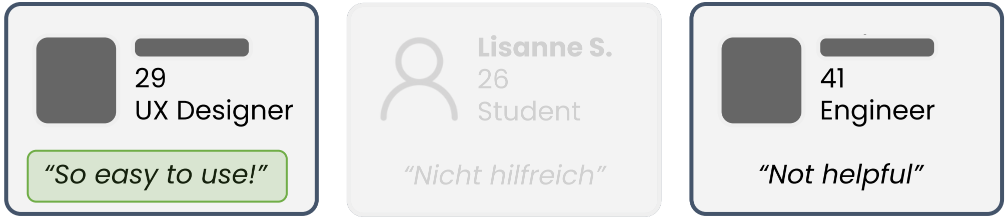
⬜ Scaling values to a common range

⬜ Encoding categorical variables

Preprocessing for the business problem

Goal #1: Identify testimonials for website

- □ Filter posts to include specific language
- □ Remove personally identifying information
- □ Select messages with positive sentiment



29
UX Designer
“So easy to use!”

Lisanne S.
26
Student
“Nicht hilfreich”

41
Engineer
“Not helpful”

Goal #2: Rank hashtags associated with company to help promote products online

- □ Remove all images and videos
- □ Extract all the hashtags
- □ Create list of top 5 unique hashtags and their frequency

Got the cutest new toys for my pup!
#PetLovers #DogToys #HappyPup

1. **#DogToys** – 31
2. **#PetLovers** – 28
3. **#HappyPup** – 24
4. **#FurryFriend** – 20
5. **#PetStoreFinds** – 18

Data preprocessing vs. data cleaning

⭐ Often used interchangeably!

Data Preprocessing

- Steps to prepare raw data for analysis
- Includes:
  - Filtering
  - Transforming
  - Organizing
  - Aligning with analysis goals
  - Data cleaning

Data Cleaning

- Subset of preprocessing
- Focuses on **fixing problems** like:
  - Correcting errors
  - Fixing inconsistencies
  - Removing duplicates
- **Examples**: Fixing typos

Validating clean data

- You'll validate dataset to ensure:
  - Data matches expectations
  - Preprocessing didn't introduce any new problems
- **Examples**:

  ⬜ Are all product names standardized?

  ⬜ Have all missing values been addressed?

  ⬜ Does the number of sales make sense across all locations?

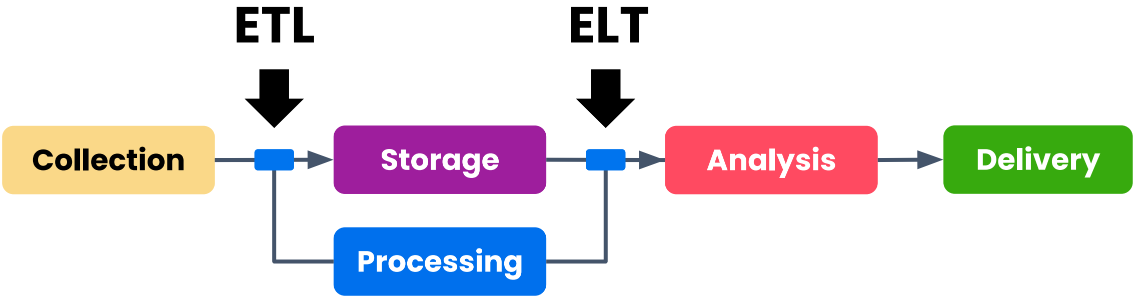
Web scraping & text preprocessing

ETL and ELT

DeepLearning.AI

ETL vs. ELT

- In this course, you've learned:
  - **Storage**: Flat file or database
  - **Analysis**: Python notebook
  - **Data collection**: Done for you, or via API or web scraping
- May want to do processing before **or** after storage



Extract, Transform, Load:

- Extraction → Collection
- Transform → Preprocessing
- Load → Save data into analysis-ready formats
- Preprocessing that everyone needs

Extract, Load, Transform:

- Extraction → Data collected
- Load & transform → Python notebook for preprocessing and analysis
- Data team will leave data as is, or reserve steps for later
- Done if people downstream have different data needs

What is ETL?

Extract

- Data is pulled from various sources (CSVs, APIs, websites)
- Retrieve efficiently & ensure integrity
- Extracting from CSV file: make sure file is not overwritten
- Data is consistent & reliable as it moves

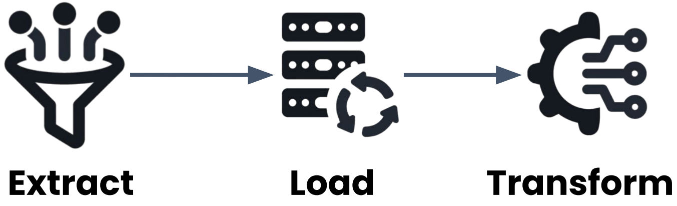
Transform

- Convert formats to a consistent structure
- Example: YYYY-MM-DD
- Remove duplicate rows
- Ensures data is ready for analysis

Load

- Data can be loaded into an analysis-ready format
- Flat file or a database
- Use tool to read data in data for analysis
- Start analysis without having to perform a lot of preprocessing

What is ELT?

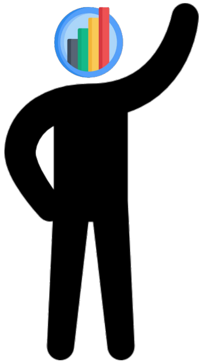


- Raw data is loaded into analysis-ready format, like a CSV or database
- Transformations occur later
- Often used when data is unstructured or rapidly evolving
  - Example: Social media content
- Data might require extra transformation

ETL vs. ELT Continuum

- Preprocessing steps may occur before loading data, or you may perform them
- Transformations before loading will:
  - Ensure data integrity
  - Enable consistent analysis
  - Shouldn't block later analysis
- **Example**: Standardize date formats
  - Unlikely to remove milliseconds
  - Data is preserved in case its useful

Your role



You
Data Analyst

- 🛠️ ETL and ELT are essential for moving and preparing data
- 🤝 Understanding allows you to collaborate with data engineers
- 💡 Knowing whether data has been through ETL or ELT:
    - ○ Better understand its limitations
    - ○ Be equipped to make the most of it in your analysis

Web scraping & text preprocessing

Introduction to web scraping

DeepLearning.AI

What is web scraping?

- Process of extracting data from websites
- Useful when no other data source can provide the information you need
- **The idea**:
  - Using code to go to different websites and collect the data you need
  - Skimming content off surface to process and analyze
    - Hence the term “scrape”

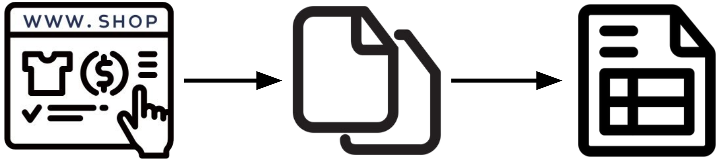
Example: Competitor prices for online pet store

❗ Scattered across several sites

❗ No easy way to download it

❗ Flat file of neat data isn’t an option

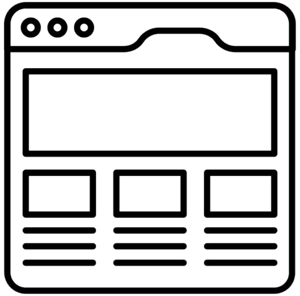
❗ Time consuming and error prone



How web scraping works

- Transforms information designed for human viewing into data for analysis
- Websites are often formatted to provide visually appealing experience
- **Process:**
  - ⬜ Gather document of unstructured text & code that makes up websites
  - ⬜ Use different code techniques to:
    - ○ Extract only data in structured format like rows and columns

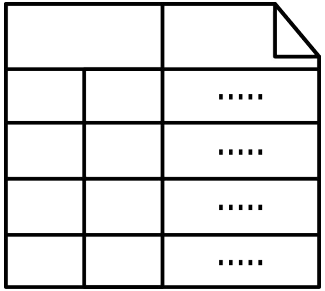
Website



✅ Great for users

⛔ Far from ideal for analysts

Dataframe



✅ Rows and columns

✅ No extra information

Web scraping challenges

⛔ Not all websites are well organized

- Hard for scraper to locate data
- **Example**: Pet supply product prices
  - Sale price might be located in a different part of the page
  - Missing values for any product that’s discounted

⛔ Dynamic content poses problems

- Might update layout or content
- Web scrapers need maintenance to keep up with changes

✅ If **structure** stays consistent while the **content** is dynamic:

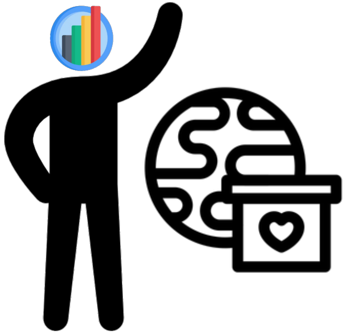
- Web scraper will truly shine
- **Example**: Product pages
  - Write one scraper to find the price of every product on a website

Web scraping & text preprocessing

Scraping tables with Pandas

DeepLearning.AI

Scenario



You
Data Analyst

🎯 **Task**: Researching human and pet populations

📊 **Data**: Webpage containing world population data

⬜ Scraping data

⬜ Preprocessing it for analysis

🏆 **Goal**: Understand how many medical and veterinary aid workers need to be assigned to each country

Recap: Scraping tables

1. Extracts tables to list of DataFrames

   ✅ Quick way to grab data in tables

   ❗ Only works if table is properly structured

2. Extract relevant dataframe from list returned
3. Perform an inspection on the data

URL

```
tables = pd.read_html("http://www.domain.com/")
```

```
df = tables[0]
```

```
df.head()
```

```
df.info()
```

Web scraping & text preprocessing

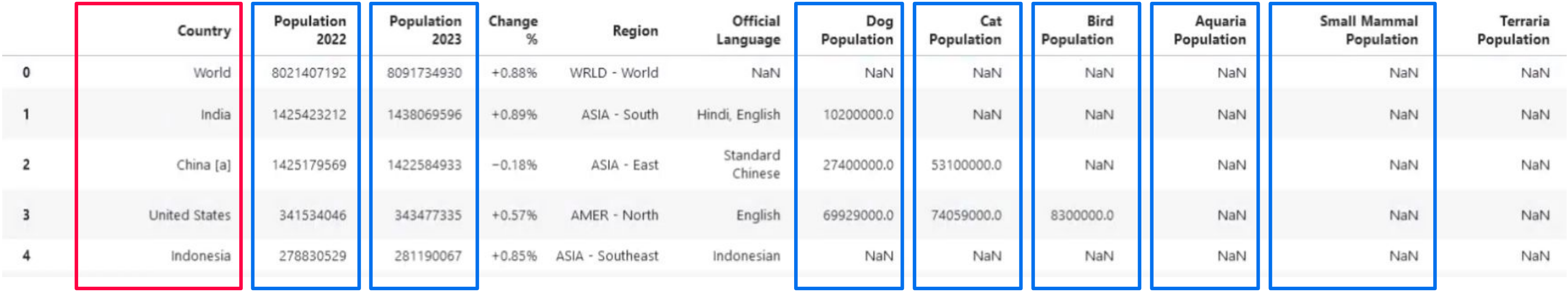
String methods: replace

DeepLearning.AI

Scenario

🎯 **Task**: Research human and animal populations to better allocate aid workers globally

📊 **Data**: Dataframe of all **countries** and their **population sizes**



	Country	Population 2022	Population 2023	Change %	Region	Official Language	Dog Population	Cat Population	Bird Population	Aquaria Population	Small Mammal Population	Terraria Population
0	World	8021407192	8091734930	+0.88%	WRLD - World	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	India	1425423212	1438069596	+0.89%	ASIA - South	Hindi, English	10200000.0	NaN	NaN	NaN	NaN	NaN
2	China [a]	1425179569	1422584933	−0.18%	ASIA - East	Standard Chinese	27400000.0	53100000.0	NaN	NaN	NaN	NaN
3	United States	341534046	343477335	+0.57%	AMER - North	English	69929000.0	74059000.0	8300000.0	NaN	NaN	NaN
4	Indonesia	278830529	281190067	+0.85%	ASIA - Southeast	Indonesian	NaN	NaN	NaN	NaN	NaN	NaN

Sean Barnes

Recap

To replace one part of a string:

Accessor

```
df["Change %"].str.replace("%","")
```

- Including **empty string** to remove text
- **.str** can apply string operations to each value in column

Accessor for DateTimes:

Accessor

```
df["date"].dt.year
```

```
df["date"].dt.month
```

Vectorized accessors

- Apply changes to every value in a column all at once
- Without needing a loop

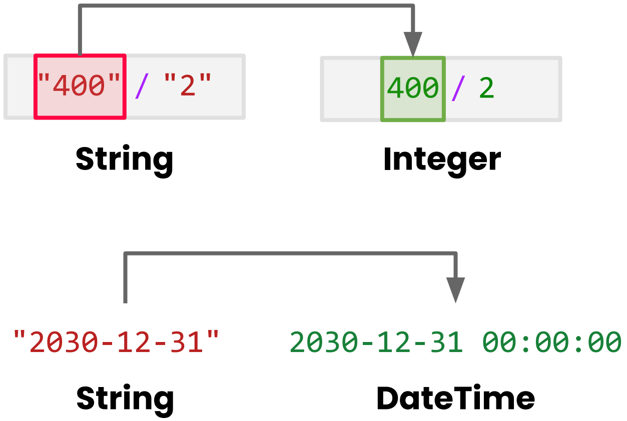
Web scraping & text preprocessing

Casting

DeepLearning.AI

Casting

- The **type** dictates what operations can be performed on it
- Change data types by **casting**
- Common when working with:
    - Data from web scraping
    - Text-heavy datasets

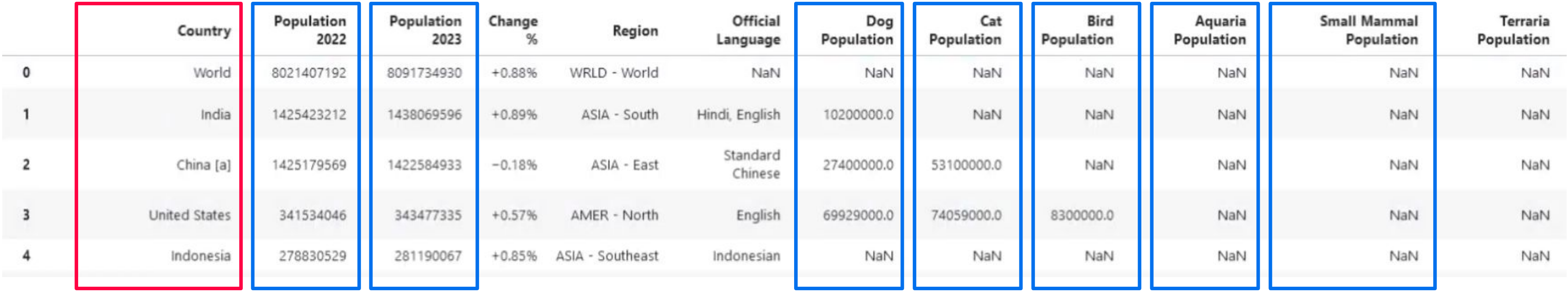


```
date = pd.to_datetime("2030-12-31")

day_of_week = date.weekday()

print(day_of_week)  # Tuesday
```

Scenario

🎯 **Task**: Research human and animal populations to better allocate aid workers globally

📊 **Data**: Dataframe of all **countries** and their **population sizes**



	Country	Population 2022	Population 2023	Change %	Region	Official Language	Dog Population	Cat Population	Bird Population	Aquaria Population	Small Mammal Population	Terraria Population
0	World	8021407192	8091734930	+0.88%	WRLD - World	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	India	1425423212	1438069596	+0.89%	ASIA - South	Hindi, English	10200000.0	NaN	NaN	NaN	NaN	NaN
2	China [a]	1425179569	1422584933	−0.18%	ASIA - East	Standard Chinese	27400000.0	53100000.0	NaN	NaN	NaN	NaN
3	United States	341534046	343477335	+0.57%	AMER - North	English	69929000.0	74059000.0	8300000.0	NaN	NaN	NaN
4	Indonesia	278830529	281190067	+0.85%	ASIA - Southeast	Indonesian	NaN	NaN	NaN	NaN	NaN	NaN

Sean Barnes

Recap: Casting

Cast a column in a Pandas dataframe to another type:

```
df["Change %"].astype("float")
```

String of **type** to cast to

Cast a column to “Float64”:

```
df["Change %"].astype("Float64")
```

- Special Pandas type
- Appropriately handles missing values

Web scraping & text preprocessing

Handling missing values

DeepLearning.AI

Missing data

- Values that are absent from a dataset where they're expected
- Due to:
  - Human error
  - Incomplete data collection
  - Technical issues
- Impact depends on the nature and extent of missing data

Impact of missing data

⛔ Large proportion missing

- **Example**: Only 5% report income
- **Impact**: Might not have enough data to generalize for entire city

⛔ Missing in systematic way

- **Example**: High income individuals less forthcoming about income
- **Impact**: Bias analysis to conclude average income is lower than it is

Dealing with missing data

1. **Removing ("dropping") rows or columns**
   - Consider dropping **rows** if:
     - Small fraction is missing
     - Example: 1% of residents did not share their age
     - Dropping too many rows can reduce representativeness
   - Consider dropping **columns** if:
     - Large fraction is missing
     - Example: 95% missing incomes
     - Remove feature from analysis

2. **Fill in missing values**
   - If assumptions make sense
     - Example:
       - Employment status: `"Employed full time"`
       - Weekly hours: ~~Declined to share~~ `40`

   ❌ Invalid approach: Filling values in with `0`

3. **Fill with descriptive statistics**
   - Mean or mode
   - Use regression or machine learning to fill in best value

Recap: Handling missing data

- **Drop rows with missing values**:

```
df.dropna(subset=["Dog Population"])
```

  - Ensures calculations are based on non-missing data
  - Used **`subset`** to drop rows with missing value in any of those columns

- **Fill in with statistical measures:**
  - Mean, median, or mode
  - Helped when missing data is widespread

- **Fill with values you specify:**

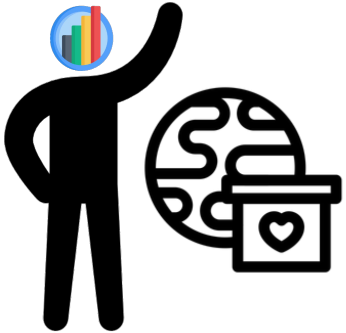
```
df["Dog Population"].fillna()
```

Web scraping & text preprocessing

String methods: contains

DeepLearning.AI

Scenario



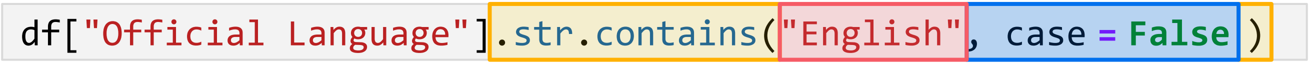
You
Data Analyst

🏆 **Goal**: Identify countries where you can most easily deploy aid workers

💬 Largest pools of aid workers speak English or German

🎯 **Task**: Filter countries where English or German is in the "Official Language" column

Recap: contains



```
df["Official Language"].str.contains("English", case = False)
```

- Great tool for:
  - ✅ Creating columns based on text data
  - ✅ Identify rows that contain specific strings

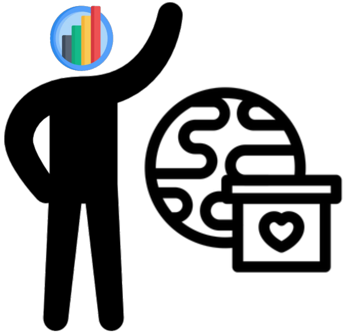
- Argument is often called a **substring**
  - A part of string you're looking for
  - Doesn't have to make up the entire string, though it can
- `.str.contains()` is case sensitive
  - Set **`case=False`** to find all instances

Web scraping & text preprocessing

String methods: split and strip

DeepLearning.AI

Scenario



You
Data Analyst

- 🏆 **Goal**: Identify the number of veterinary and medical aid workers needed in different countries
- 🎯 **Task**: Almost ready, but have a few finishing touches

Recap: String processing

```
df["Country"].str.split("[").str[0].str.strip()
```

- Use `.split()` to break a string into a list of substrings
  - Used open bracket character to split a string into two parts
- Selected the **first string** from that list
- Use `.str.strip()` to strip leading and trailing whitespace
  - Leading - beginning of the string
  - Trailing - at the end
  - Will remove: whitespace, tabs, newline characters

Web scraping & text preprocessing

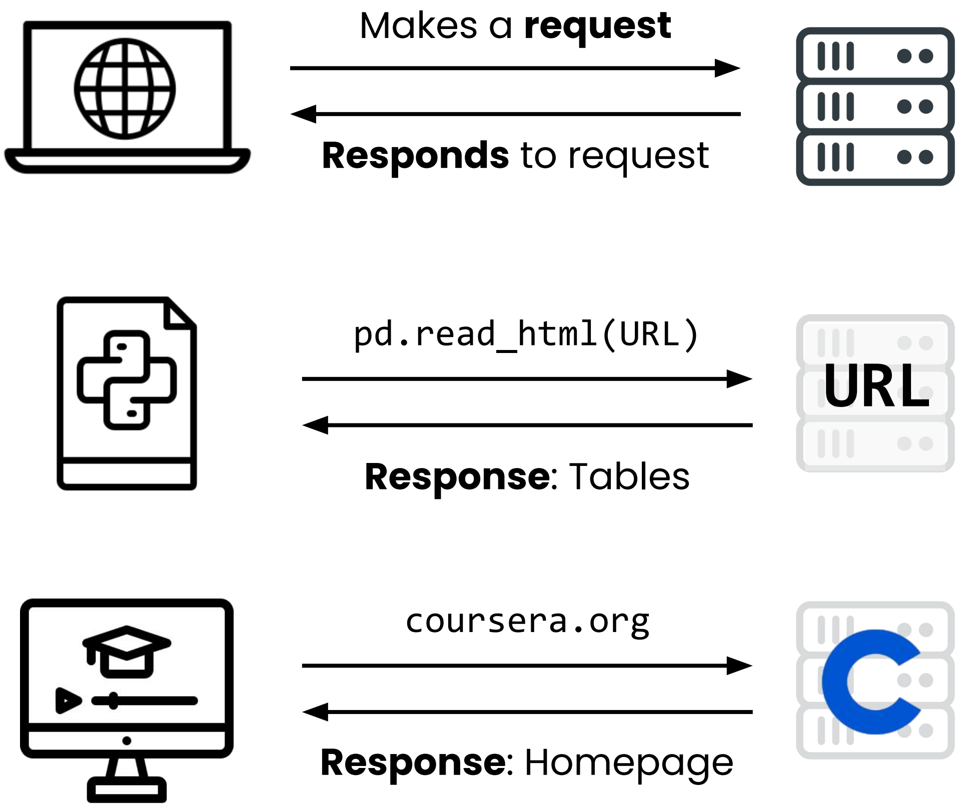
Networking

DeepLearning.AI

Networking

- Conversation between:
  - **Client**: Your computer
  - **Server**: Computer that delivers web content

- Conversation is called the “request-response cycle”



Makes a **request**

Responds to request

`pd.read_html(URL)`

URL

Response: Tables

`coursera.org`

Response: Homepage

HTTP and protocols

- Networking relies on protocols that ensure clients and servers can communicate

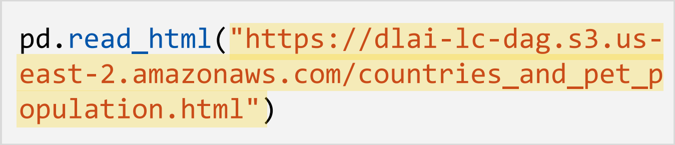
- Most important protocol for web scraping is **HTTP**
  - HyperText Transfer Protocol
  - Tells **clients** and **servers** how to format requests and responses

Parts of HTTP Request

1 **Verb** - action you're taking

- ○ GET - to fetch data
- ○ Options: POST, PATCH, DELETE

2 **Path** - where request is going



```
pd.read_html("https://dlai-lc-dag.s3.us-east-2.amazonaws.com/countries_and_pet_population.html")
```

URL

Networking challenges

- Web scraping involves networking
- Success of scraper depends on factors outside of your control
- In Python, errors can be traced back to your own code
- Like texting a friend to borrow a bike:
  - ⚡ Respond right away - yes!
  - ⏳ Take a while to get back
  - 🤐 Not respond at all
  - 🚫 Respond to say no

⛔ Websites can be slow

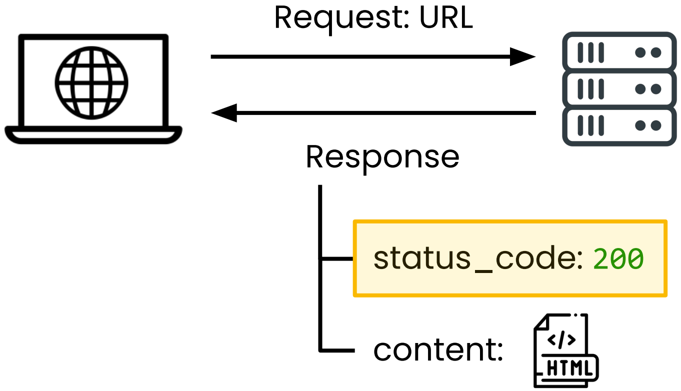
⛔ Web scraper can experience delays

⛔ Request does not guarantee you'll get a response

⛔ Response might just let you know page doesn't exist

Response status codes

- Help you understand whether request was successful or if something went wrong
- A status code is 3 digits
- Different classes:
  - `200` - request worked!
  - Codes starting with **4** (`400`, `404`):
    - Something went wrong on **your** side
    - Used path the server didn't recognize
  - Codes starting with **5** (`500`, `503`):
    - Issue with the server
    - Example: Server down for maintenance



DeepLearning.AI

Sean Barnes

Web scraping & text preprocessing

Scraping webpages with requests

DeepLearning.AI

Scenario

You
Data Analyst

🏆 **Goal**: Align product promotions with celestial events to boost sales

🎯 **Task**: Identifying upcoming events, like meteor showers

Recap: Requests

- Use requests module to retrieve webpage content

```
import requests
```

- Sends a GET request to website you specify:

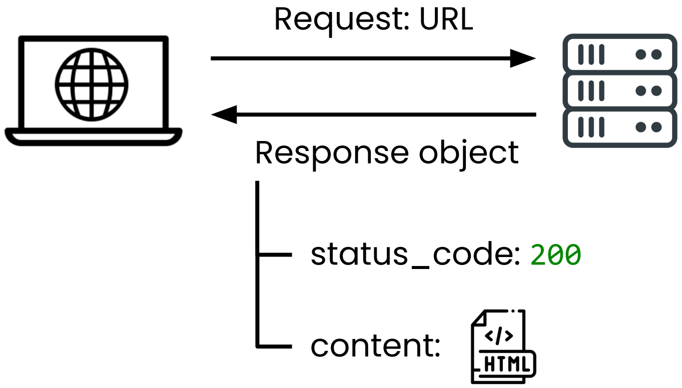
```
url = "http://www.seasky.org/astronomy/astronomy-calendar-2030.html"

response = requests.get(url)
```

- To see whether the request worked:

```
response.status_code
```

- Gives you raw HTML of the webpage

```
response.content
```



Request: URL

Response object

status_code: 200

content:

Web scraping & text preprocessing

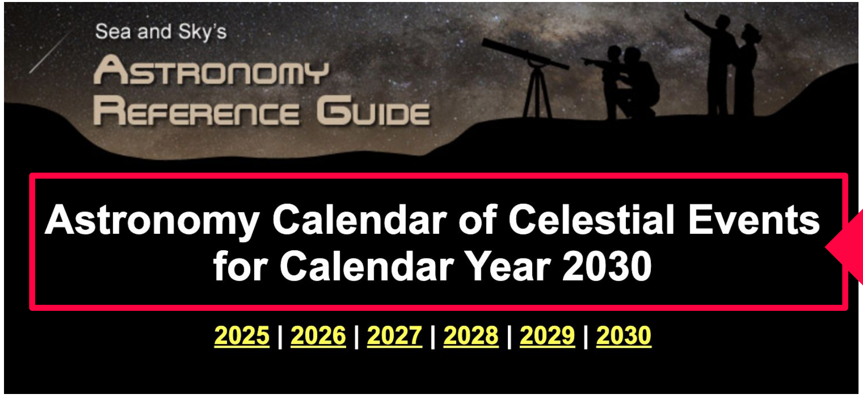
HTML

DeepLearning.AI

HTML: hypertext markup language

Text enhanced with tags

- Provide additional information
- Typically **open** and **close**



Sea and Sky's

Astronomy Reference Guide

Astronomy Calendar of Celestial Events for Calendar Year 2030

2025 | 2026 | 2027 | 2028 | 2029 | 2030

content

HTML element

```
<h1>Astronomy Calendar of Celestial
Events for Calendar Year 2030</h1>
```

`<h1></h1>` → Level 1 heading

`<h2></h2>` `<h6></h6>`

`<p></p>` `<a></a>`

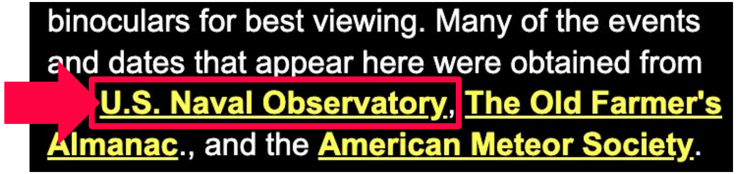
Self-closing tags `<img>`

HTML attributes

Tags can have attributes

Additional info: size, color, position, source

Inside angle brackets of opening tag



binoculars for best viewing. Many of the events and dates that appear here were obtained from U.S. Naval Observatory, The Old Farmer's Almanac., and the American Meteor Society.

```
<a href="http://www.usno.navy.mil/USNO">U.S. Naval Observatory</a>
```

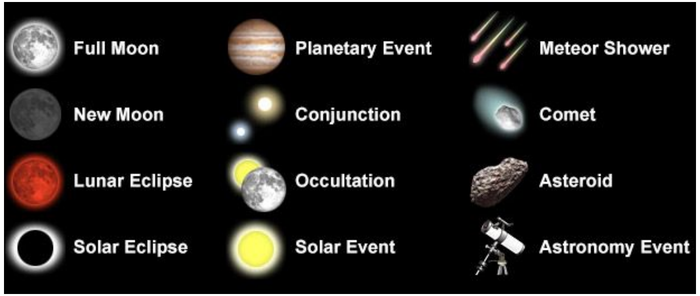
“Hypertext reference” → where the link takes you

HTML attributes

Tags can have attributes

Additional info: size, color, position, source

Inside angle brackets of opening tag



```html
<img
src="../astronomy/assets/images/calendar_legend.jpg"
alt="Legend for astronomy calendar icons"
width="618"
height="260">
```

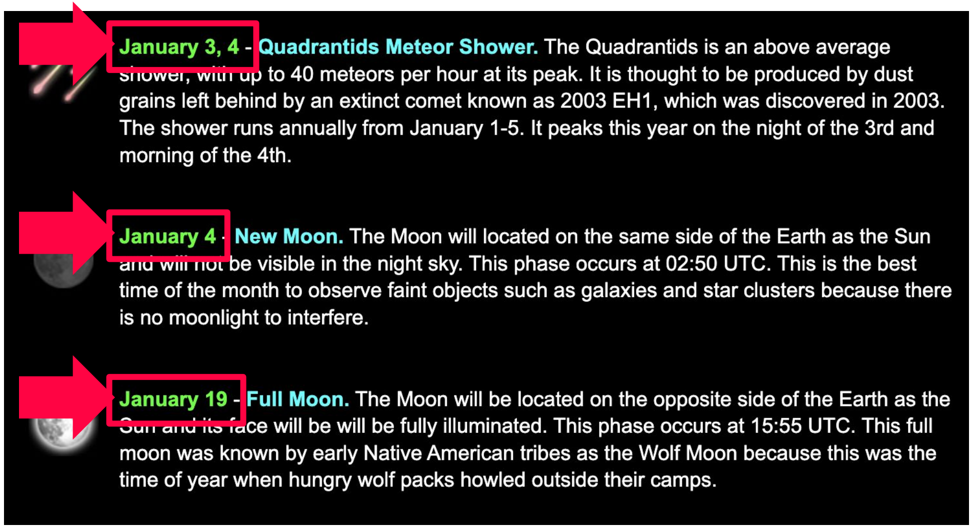
Each HTML element takes up space on the page

`src` (“source”) → URL or file path of the image
`alt` (“alternative text”) → shows up if fails to load
`width` → tells browser how large to display
`height` → tells browser how large to display

HTML classes

Special type of attribute

- Group elements together
- Apply consistent styling or functionality



January 3, 4 - Quadrantids Meteor Shower. The Quadrantids is an above average shower, with up to 40 meteors per hour at its peak. It is thought to be produced by dust grains left behind by an extinct comet known as 2003 EH1, which was discovered in 2003. The shower runs annually from January 1-5. It peaks this year on the night of the 3rd and morning of the 4th.

January 4 - New Moon. The Moon will located on the same side of the Earth as the Sun and will not be visible in the night sky. This phase occurs at 02:50 UTC. This is the best time of the month to observe faint objects such as galaxies and star clusters because there is no moonlight to interfere.

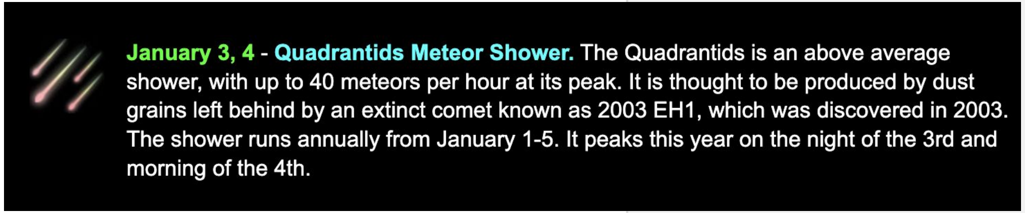
January 19 - Full Moon. The Moon will be located on the opposite side of the Earth as the Sun and its face will be will be fully illuminated. This phase occurs at 15:55 UTC. This full moon was known by early Native American tribes as the Wolf Moon because this was the time of year when hungry wolf packs howled outside their camps.

```
<p class="date-text">January 3, 4</p>

<p class="date-text">January 4   </p>

<p class="date-text">January 19  </p>
```

HTML tree structure

- HTML document structures elements in a hierarchy
- Elements can contain other elements

```
<html>
     <p>Many of the events and dates that appear here
        were obtained from the
          <a href="http://www.usno.navy.mil/USNO">U.S.
            Naval Observatory</a>,
          <a href="http://www.almanac.com/">The Old
            Farmer's Almanac</a>, and the
          <a href="http://www.amsmeteors.org">
            American Meteor Society</a>.
     </p>
</html>
```



January 3, 4 - **Quadrantids Meteor Shower.** The Quadrantids is an above average shower, with up to 40 meteors per hour at its peak. It is thought to be produced by dust grains left behind by an extinct comet known as 2003 EH1, which was discovered in 2003. The shower runs annually from January 1-5. It peaks this year on the night of the 3rd and morning of the 4th.

Container elements

Organize content logically by grouping other elements

```
<div class="container">
  <p>One paragraph</p>
  <p>Another paragraph</p>
</div>
```

- Acts as an empty box
- Block starts on new line
- Takes up full width

```
<ul class="bullets">
  <li>One item</li>
  <li>Another item</li>
</ul>
```

- Bulleted list

```
<span class="inline">
     Text here
</span>
```

- Don’t start on new line
- Don’t take up full width

```
<ol class="numbered">
  <li>One item</li>
  <li>Another item</li>
</ol>
```

- Numbered list

Using `pd.read_html()`

- Used to scrape tables from website
- Only works for:
  - Properly organized table elements
  - Type of container
- Throws error with SeaSky website because there aren’t any HTML table elements
- Don’t feel the need to memorize:
  - The more you work with them, the better you’ll understand them.
  - You can always chat with LLM to help remember the details.

```
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
  </tr>
  <tr>
    <td>The Hip Cafe</td>
    <td>Maria</td>
  </tr>
  <tr>
    <td>Ned’s Supply</td>
    <td>Ned</td>
  </tr>
</table>
```

Web scraping & text preprocessing

Planning HTML parsing

DeepLearning.AI

Scenario

You
Data Analyst

🏆 **Goal**: Align product promotions with celestial events to boost sales

🎯 **Task**: identifying upcoming celestial events

Planning actions for scraping

```
# Create a list to store all your events
# Find all <li> elements (with any class)
# Create a loop to look at each element
    # Date: <span> with class="date-text"
    # Title: <span> with class="title-text"
    # Description: <p>
    # Extract only text from elements
    # Create list containing date, title, description
    # Save list into your list of events
```

Web scraping & text preprocessing

Parsing HTML with Beautiful Soup

DeepLearning.AI

- To get a list of HTML elements you needed:

  ```
  soup.find_all("li", class_ = True)
  ```

- Used a loop to process each event one at a time

  ```
  for event in soup.find_all("li", class_ = True):
  ```

  - To locate first occurrence of specific element with specific class

    ```
    date = soup.find("span", class_ = "date-text")
    ```

  - To extract the inner text of the tags

    ```
    date = soup.find("span", class_ = "date-text").text
    ```

  - Assembled event into a list and appended to a list

    ```
    event_list.append([date, title, description])
    ```

- Constructed a DataFrame from those lists

  ```
  df = pd.DataFrame(event_list)
  ```

Web scraping & text preprocessing

DataFrame setup

DeepLearning.AI

Scenario

You
Data Analyst

- 🏆 **Goal**: Align product promotions with celestial events to boost sales
- 🎯 **Task**: Identifying upcoming celestial events
  - ⬜ Create calendar of events to time promotions and social media posts

Recap: DataFrame setup

1. To give DataFrame column names:

```
df.columns = ["date", "title", "description"]
```

2. To clean up values in DataFrame

```
df["title"] = df["title"].str.replace(".", "", regex=False)
```

```
df["date"] = df["date"].str.split(",").str[0]
```

```
df["description"] = df["description"].str.split(".", n=1 ).str[1]
```

Number of splits

3. Use helper function to cast dates from strings to datetimes

```
from helper_functions import convert_datetime_column

df = convert_datetime_column(df, "date", 2030)
```

✅ Treat events as a time series

Web scraping & text preprocessing

Regular expressions

DeepLearning.AI

Scenario

You
Data Analyst

🎯 **Task**: Identifying upcoming celestial events

✅ Dataframe containing date, name, description of various astronomical events

⬜ Search this calendar to pinpoint key events

Regular expressions

- **Goal**: Search description of each event to add the time it occurs as a new column

`11:40` `1:27` `00:29`

- **Solution:** Pattern match
  - Typically in **HH:MM** or **H:MM** format
  - Numbers are different
  - Pattern is consistent

- Use **regular expressions** (regex):
  - Sequences of characters that define a search pattern
  - Searching for structured text formats

- Dates: `YYYY-MM-DD`
- Phone numbers: `(###) ###-####`

Regular expressions (regex)

Text pattern to match

- ⬜ Specific character or set of characters
    - ○ Examples: `"12"` , `"happy"`
- ⬜ Character classes
    - ○ `[0-9]` - any single digit
    - ○ `[A-Z]` - any capital letter
    - ○ `[a-z]` - any lowercase letter
    - ○ `[A-Za-z]` - any letter at all
    - ○ `[0-9A-Za-z]` - any number or letter
- ⬜ Special sequences
    - ○ `\d` - any digit
    - ○ `\s` - any whitespace character

Frequency

- ⬜ Specify the frequency
    - ○ Example: Time formatted as HH:MM
        - ■ Any 2 digits, colon, any 2 digits
- ⬜ Use curly braces with number of repeating characters
    - ○ Pattern: `"\d{2}:\d{2}"`

Example:

- ● Any digit followed by any letter:
    - ○ Examples: `"9a"`, `"7Q"`
    - ○ Pattern: `"\d[A-Za-z]"`

Special characters

- Help match more flexible patterns
  - `+` - matches one or more characters
  - `*` - matches zero or more characters

- **Example**: Match any sequence of digits
  - Pattern: `"\d+"`
  - Match 1 or more digits

- **Example**: Phone number ### ####
  - 3 digits, any space (including 0), four digits
  - Pattern: `"\d{3}\s*\d{4}"`

Regex and string methods

- Use regular expressions with string methods:
  - `str.contains()`
  - `str.replace()`
  - `str.extract()`
    - **Returns** matched text, rather than just telling you whether pattern is present

- **Example:** Messages about customers
  - Extract phone numbers:

    ### - ### - ####

Raw strings

```
pattern = r"\d{3}-\d{3}-\d{4}"

df["message"].str.extract(pattern)
```

- Returns first match it finds

Web scraping & text preprocessing

Writing regular expressions with LLMs

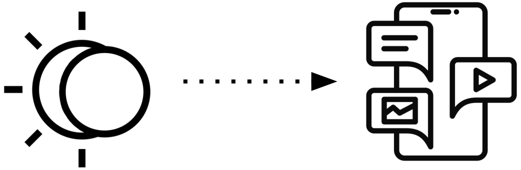
DeepLearning.AI

Recap

📊 **Data**: DataFrame with information about astronomical events taking place in 2030

- Columns include:
  - date
  - name (e.g. Full Moon)
  - description

🎯 **Task:** Focus on **days and times** events occur to optimize the timing of social media posts



□ Write a match pattern to extract times from the event descriptions

Web scraping & text preprocessing

The ethics of web scraping

DeepLearning.AI

Copyright and access issues

- Just because data is online doesn't mean you're allowed to use it
- Some websites restrict the reuse of content, particularly for commercial purposes

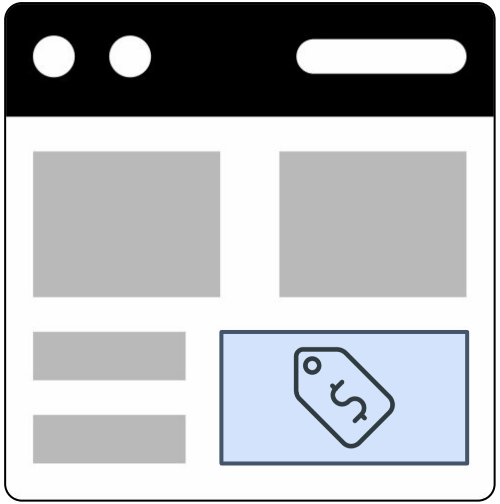
⛔ Could get your scraper blocked

✅ Always check:

  ○ Terms of Service

  ○ Licensing rules

Example: Competitor's e-commerce website



Server load and resource use

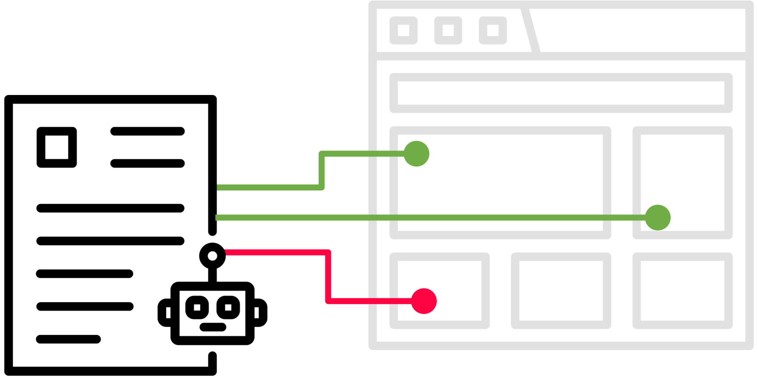
- Every time you scrape a website:
  - 🔄 Server has to process request
  - 💰 Costs time and energy on their end
- Too many requests too quickly:
  - ⛔ Could overwhelm the server
  - ⛔ Slower response times
  - ⛔ Outages
- Scrape **once**, then process on your own computer or server
- Separate request code from processing code
- Excessive requests can trigger CAPTCHAs or bans
- Limit your requests

```
import time
for page in pages_to_scrape:
    scrape_page(page)
    time.sleep(1)
```

- Pause code for one second between requests

Robots.txt

- Websites are aware of web scrapers
- Use robots.txt file to set boundaries about:
  - ✅ What you can collect
  - ⛔ What you shouldn't touch
- To access, add robots.txt to root domain
  - `google.com`
  - `deeplearning.ai`
  - `wikipedia.org`



`robots.txt`

DeepLearning.AI

Sean Barnes