BEGINNING OF CLASS:
GRAB A PIECE OF CARDSTOCK AND CREATE A NAME TENT WITH THE FIRST NAME YOU'D LIKE TO GO BY - write BIG

Stand name tents on each desk

Join our iClicker class:

https://join.iclicker.com/XSFZ

LECTURE 1

Intro to CSCI 3022

Intro to the data science lifecycle and exploratory data analysis with Pandas

CSCI 3022 @ CU Boulder

Maribeth Oscamou

Content credit: Acknowledgments

WHAT DOES THE DATA TELL US TO DO?
WE ONLY HAVE BAD DATA ON THIS.
DOES THE BAD DATA SUGGEST WE SHOULD DO WHAT WE WANTED TO DO ANYWAY?
WELL, YES.
THAT'S CALLED "GOOD DATA."

Lesson Learning Objectives:

- Meet your classmates
- Name and explain the stages of the data science lifecycle
- Identify 5 key data properties to consider when doing Exploratory Data Analysis and implement using demo data
- Read in data from a .csv file to a Pandas DataFrame

Roadmap

Lecture 01, CSCI 3022

- Intros & Logistics
- What is Data Science?
- Exploratory Data Analysis & Wrangling
- Intro to Pandas
  - Jupyter Notebook Demo: EDA using Pandas

- Supporting Materials:
  - Intro to Pandas Data Structures

Intros

In Groups of 4 INTRODUCE YOURSELF :

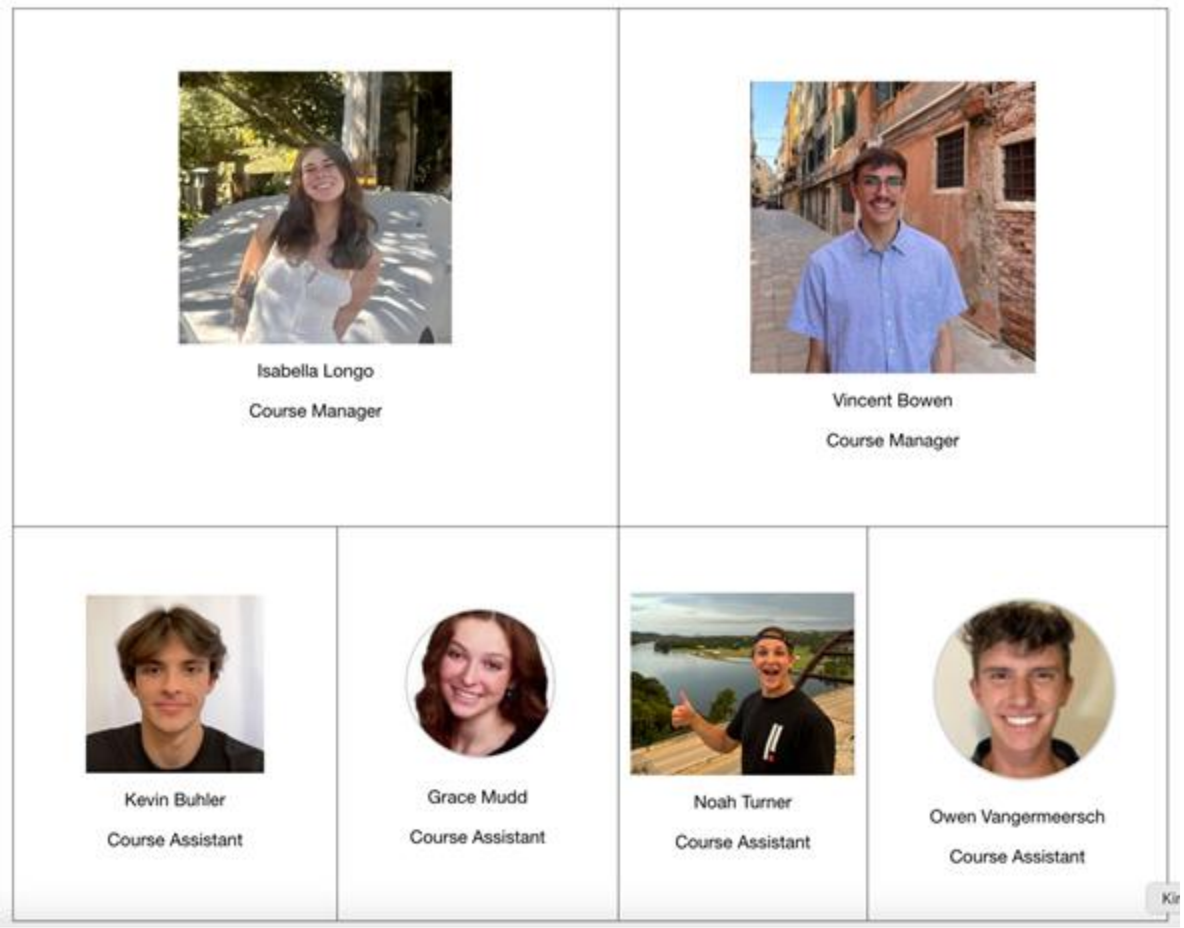
- NAME, YEAR, MAJOR, HOMETOWN
- HOBBIES/INTERESTS
- SOME RANDOM FUN FACT ABOUT YOU

Getting To Know You:

I'd like to get a chance to be introduced to each of you!

1. **Please sign-up for a 15 min. timeslot here: link**
   (this link is also in the Week 1 Info announcements on Canvas and Piazza)
   to meet with me during the first couple weeks to briefly introduce yourself and meet a few other classmates.

Meet The Course Team



Isabella Longo

Course Manager

Vincent Bowen

Course Manager

Kevin Buhler

Course Assistant

Grace Mudd

Course Assistant

Noah Turner

Course Assistant

Owen Vangermeersch

Course Assistant

Ki

Course Online platforms

Canvas (https://canvas.colorado.edu/courses/117881)

- Where all course information including lectures, assignments, announcements and grades are posted

CSCI 3022 JuptyerHub (https://coding.csel.io/)

- Where you will work on all assignments (links on Canvas assignments automatically take you here).

Piazza (linked from Canvas)

- How to contact professor and discuss questions with other students

Gradescope (linked from Canvas)

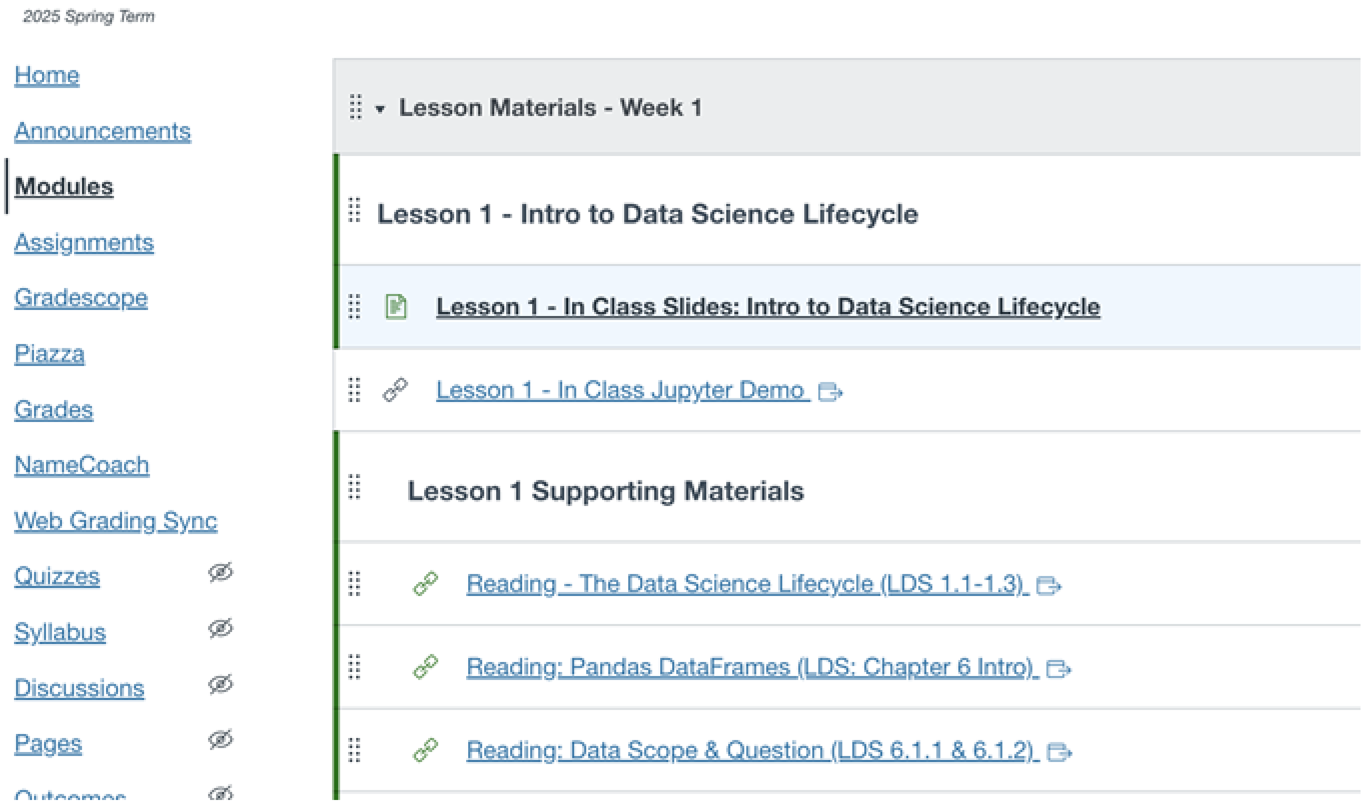
- Where all assignments are submitted

iClicker (https://student.iclicker.com/#/login**)**

- Where you answer polls during class

Accessing Lecture Slides and Jupyter Notebook

Canvas - Modules - Lesson Materials https://canvas.colorado.edu/courses/117881/modules



2025 Spring Term

- Home
- Announcements
- Modules
- Assignments
- Gradescope
- Piazza
- Grades
- NameCoach
- Web Grading Sync
- Quizzes
- Syllabus
- Discussions
- Pages
- Outcomes

▾ Lesson Materials - Week 1

Lesson 1 - Intro to Data Science Lifecycle

Lesson 1 - In Class Slides: Intro to Data Science Lifecycle

Lesson 1 - In Class Jupyter Demo

Lesson 1 Supporting Materials

Reading - The Data Science Lifecycle (LDS 1.1-1.3)

Reading: Pandas DataFrames (LDS: Chapter 6 Intro)

Reading: Data Scope & Question (LDS 6.1.1 & 6.1.2)

Course Logistics: Your *Typical* Week At A Glance

Mon	Tues	Wed	Thurs	Fri
Attend & Participate in Class		Attend & Participate in Class	HW Due 11:59pm via Gradescope	In Class Quiz (beginning of class) Attend & Participate in Class
Previous week quiz grades posted		Previous week HW grades posted		Next week HW released

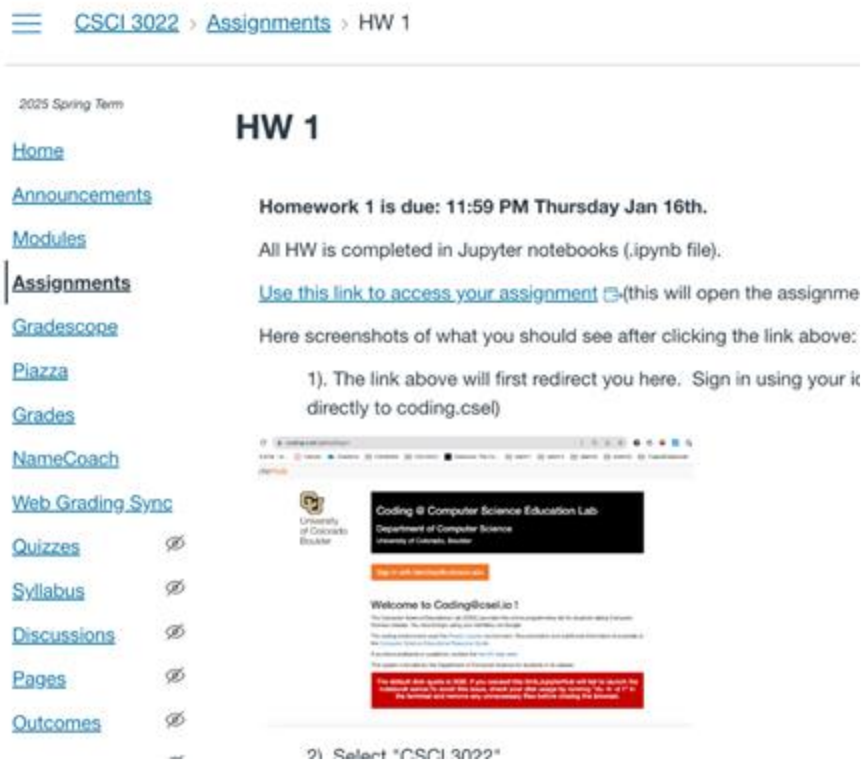
Course Logistics: Your First Week At A Glance

Mon 1/13	Tues 1/14	Wed 1/15	Thurs 1/16	Fri 1/17	
Attend & Participate in Class Office Hours Begin (See Schedule on Canvas)		Attend & Participate in Class	HW 1 Due 11:59pm via Gradescope (Includes Intro to CSCI 3022 Video assignment)	Attend & Participate in Class ~~In Class Quiz (beginning of class)~~	
				HW 2 released	

Accessing HW 1

Canvas -> Assignments

https://canvas.colorado.edu/courses/117881/assignments/2238036



CSCI 3022 › Assignments › HW 1
2025 Spring Term
Home
Announcements
Modules
Assignments
Gradescope
Piazza
Grades
NameCoach
Web Grading Sync
Quizzes
Syllabus
Discussions
Pages
Outcomes
Coding @ Computer Science Education Lab
Department of Computer Science
University of Colorado, Boulder
University of Colorado Boulder

HW 1

Homework 1 is due: 11:59 PM Thursday Jan 16th.

All HW is completed in Jupyter notebooks (.ipynb file).

Use this link to access your assignment (this will open the assignme

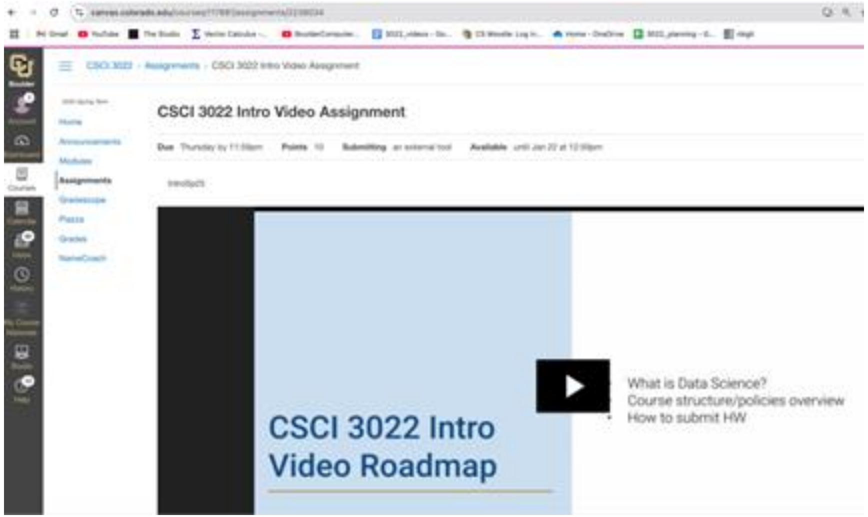
Here screenshots of what you should see after clicking the link above:

1). The link above will first redirect you here. Sign in using your id directly to coding.csel)

Welcome to Coding@csel.io !

2) Select "CSCI 3022"

HW 1 Includes an Intro Video Assignment



CSCI 3022 › Assignments › CSCI 3022 Intro Video Assignment
Home
Announcements
Modules
Assignments
Gradescope
Piazza
Grades
NameCoach
CSCI 3022 Intro Video Roadmap

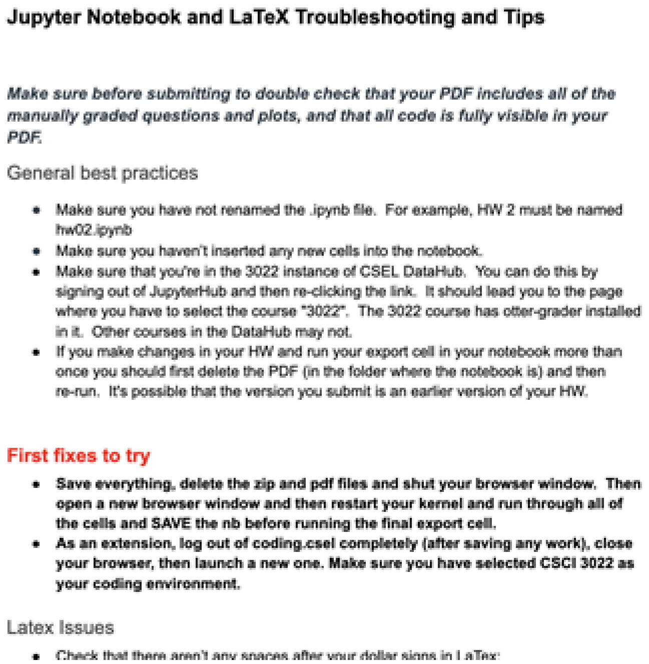
CSCI 3022 Intro Video Assignment

Due Thursday by 11:59pm Points 10 Submitting an external tool Available until Jan 22 at 12:59pm

What is Data Science?
Course structure/policies overview
How to submit HW

Office Hours:

https://canvas.colorado.edu/courses/117881/pages/hw-slash-office-hours



Jupyter Notebook and LaTeX Troubleshooting and Tips

Make sure before submitting to double check that your PDF includes all of the manually graded questions and plots, and that all code is fully visible in your PDF.

General best practices

- Make sure you have not renamed the .ipynb file. For example, HW 2 must be named hw02.ipynb
- Make sure you haven't inserted any new cells into the notebook.
- Make sure that you're in the 3022 instance of CSEL DataHub. You can do this by signing out of JupyterHub and then re-clicking the link. It should lead you to the page where you have to select the course "3022". The 3022 course has otter-grader installed in it. Other courses in the DataHub may not.
- If you make changes in your HW and run your export cell in your notebook more than once you should first delete the PDF (in the folder where the notebook is) and then re-run. It's possible that the version you submit is an earlier version of your HW.

First fixes to try

- **Save everything, delete the zip and pdf files and shut your browser window. Then open a new browser window and then restart your kernel and run through all of the cells and SAVE the nb before running the final export cell.**
- **As an extension, log out of coding.csel completely (after saving any work), close your browser, then launch a new one. Make sure you have selected CSCI 3022 as your coding environment.**

Latex Issues

- Check that there aren't any spaces after your dollar signs in LaTex:

https://docs.google.com/document/d/1ndr3Wj1PSF5qzlLMaBJznwh6QGeEXjd5TAJ6nf9EJvo/edit?usp=sharing

Course Prerequisites (minimum grade C-)

- Data Structures (CSCI 2270 or equivalent)
- Calculus 2 (APPM 1360 or MATH 2300 or equivalent)
- Discrete Math (CSCI 2824 or equivalent)

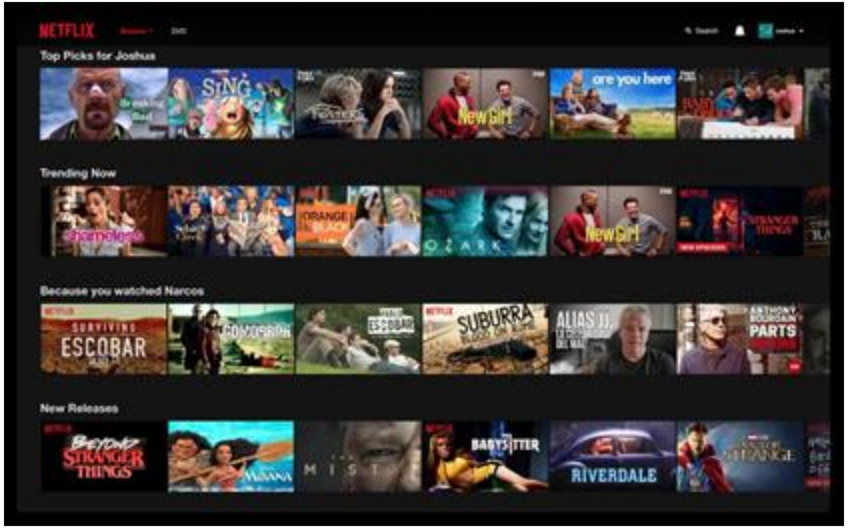
What is Data Science?

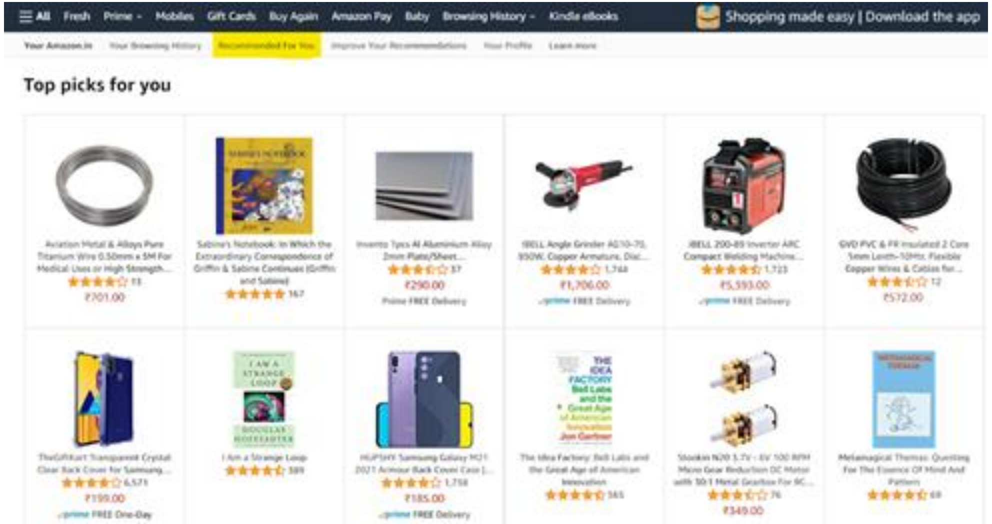
Lecture 01, CSCI 3022

- Intros & Logistics
- **What is Data Science?**
- Exploratory Data Analysis & Wrangling
- Pandas Data Structures
- Jupyter Notebook Demo: EDA using Pandas

Some examples of Data Science?

Recommendation Systems





Top picks for you

Aviation Metal & Alloys Pure Titanium Wire 0.50mm x 5M For Medical Uses or High Strength... 13 ₹701.00

Sabine's Notebook: In Which the Extraordinary Correspondence of Griffin & Sabine Continues (Griffin and Sabine) 167

Inventa Tycs Al Aluminium Alloy 2mm Plate/Sheet... 37 ₹290.00 Prime FREE Delivery

IBELL Angle Grinder AG10-75, 950W, Copper Armature, Disc... 1,744 ₹1,706.00 prime FREE Delivery

IBELL 200-89 Inverter ARC Compact Welding Machine... 1,723 ₹5,593.00 prime FREE Delivery

GVD PVC & FR Insulated 2 Core 1mm Lenth-10Mtr, Flexible Copper Wires & Cables for... 12 ₹572.00

TheGiftKart Transparent Crystal Clear Back Cover for Samsung... 6,571 ₹159.00 prime FREE One-Day

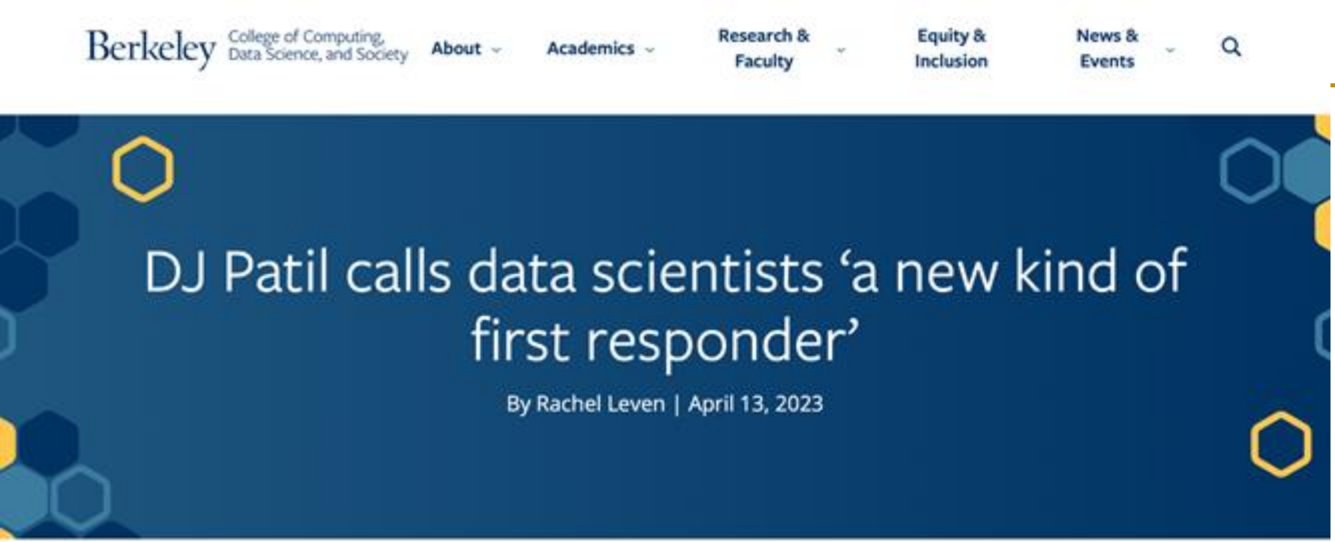
I Am a Strange Loop 389

HUPSHY Samsung Galaxy M21 2021 Armour Back Cover Case |... 1,758 ₹185.00 prime FREE Delivery

The Idea Factory: Bell Labs and the Great Age of American Innovation 365

Shookin N20 3.7V - 6V 100 RPM Micro Gear Reduction DC Motor with 30:1 Metal Gearbox For RC... 76 ₹349.00

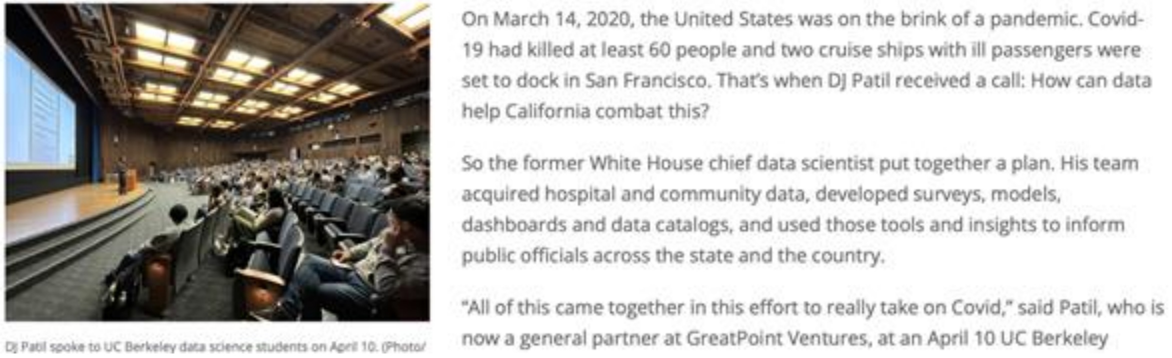
Metamagical Themas: Questing For The Essence Of Mind And Pattern 69



DJ Patil calls data scientists ‘a new kind of first responder’

By Rachel Leven | April 13, 2023

https://www.youtube.com/watch?v=LiHMrn2AHpw



DJ Patil spoke to UC Berkeley data science students on April 10. (Photo/

On March 14, 2020, the United States was on the brink of a pandemic. Covid-19 had killed at least 60 people and two cruise ships with ill passengers were set to dock in San Francisco. That's when DJ Patil received a call: How can data help California combat this?

So the former White House chief data scientist put together a plan. His team acquired hospital and community data, developed surveys, models, dashboards and data catalogs, and used those tools and insights to inform public officials across the state and the country.

"All of this came together in this effort to really take on Covid," said Patil, who is now a general partner at GreatPoint Ventures, at an April 10 UC Berkeley

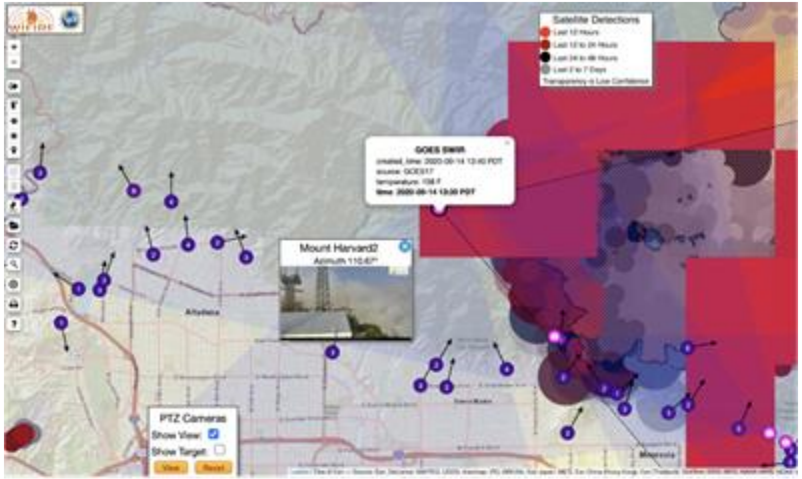
First U.S. Chief Data Scientist
(Obama Adm.)

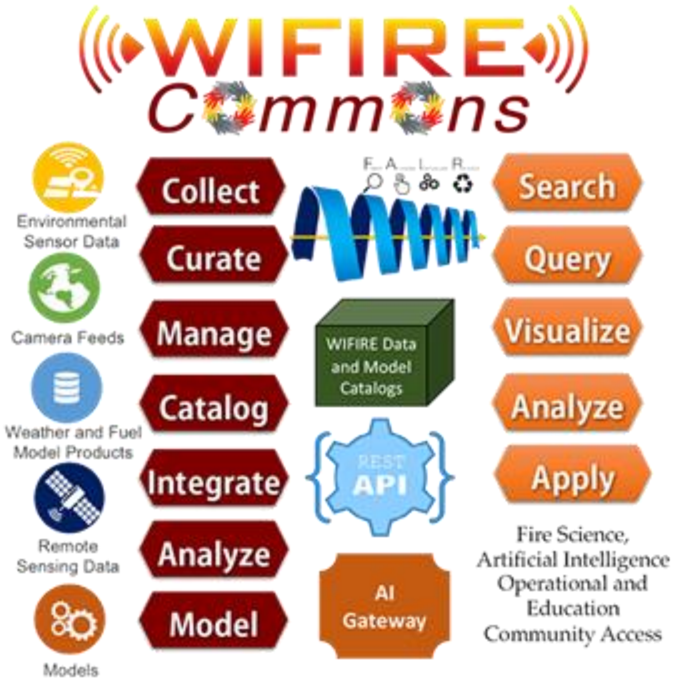
https://www.sciencefriday.com/segments/an-exit-interview-with-u-s-chief-data-scientist-dj-patil/

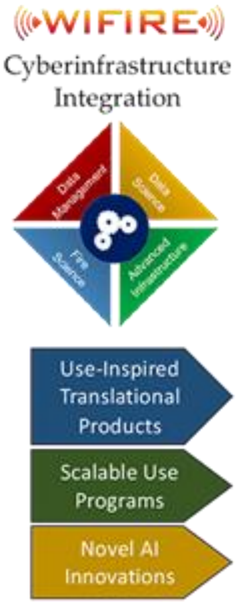
WIFIRE (UCSD) - Wildfire modeling and management

https://wifire.ucsd.edu

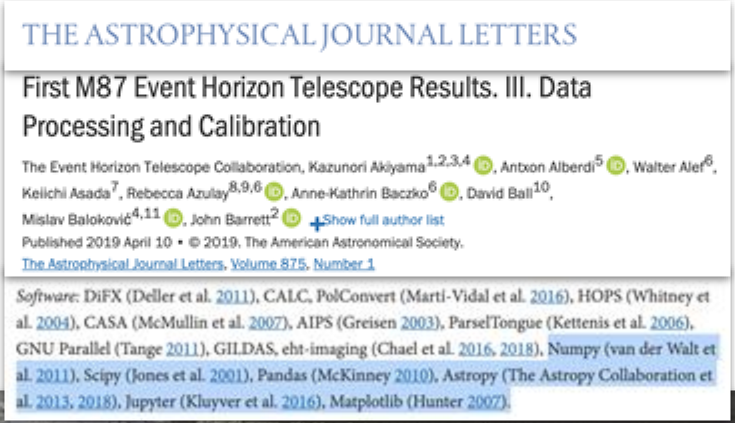
FIRE INTEGRATED REAL-TIME INTELLIGENCE SYSTEM
COVERAGE YOU COUNT ON
FIGHTING FIRE WITH TECHNOLOGY
7




WIFIRE
Commons
Environmental Sensor Data
Camera Feeds
Weather and Fuel Model Products
Remote Sensing Data
Models
Collect
Curate
Manage
Catalog
Integrate
Analyze
Model
F... A... I... R...
WIFIRE Data and Model Catalogs
REST API
AI Gateway
Search
Query
Visualize
Analyze
Apply
Fire Science, Artificial Intelligence Operational and Education Community Access



2019: First Image of a Black Hole



THE ASTROPHYSICAL JOURNAL LETTERS

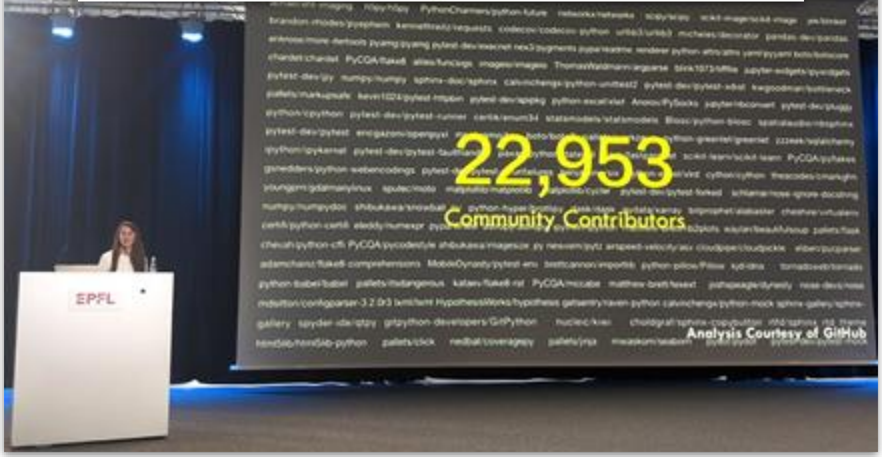
First M87 Event Horizon Telescope Results. III. Data Processing and Calibration

The Event Horizon Telescope Collaboration, Kazunori Akiyama[1,2,3,4], Antxon Alberdi[5], Walter Alef[6], Keiichi Asada[7], Rebecca Azulay[8,9,6], Anne-Kathrin Baczko[6], David Ball[10], Mislav Baloković[4,11], John Barrett[2] + Show full author list

Published 2019 April 10 • © 2019. The American Astronomical Society.

The Astrophysical Journal Letters, Volume 875, Number 1

Software: DiFX (Deller et al. 2011), CALC, PolConvert (Martí-Vidal et al. 2016), HOPS (Whitney et al. 2004), CASA (McMullin et al. 2007), AIPS (Greisen 2003), ParselTongue (Kettenis et al. 2006), GNU Parallel (Tange 2011), GILDAS, eht-imaging (Chael et al. 2016, 2018), Numpy (van der Walt et al. 2011), Scipy (Jones et al. 2001), Pandas (McKinney 2010), Astropy (The Astropy Collaboration et al. 2013, 2018), Jupyter (Kluyver et al. 2016), Matplotlib (Hunter 2007).

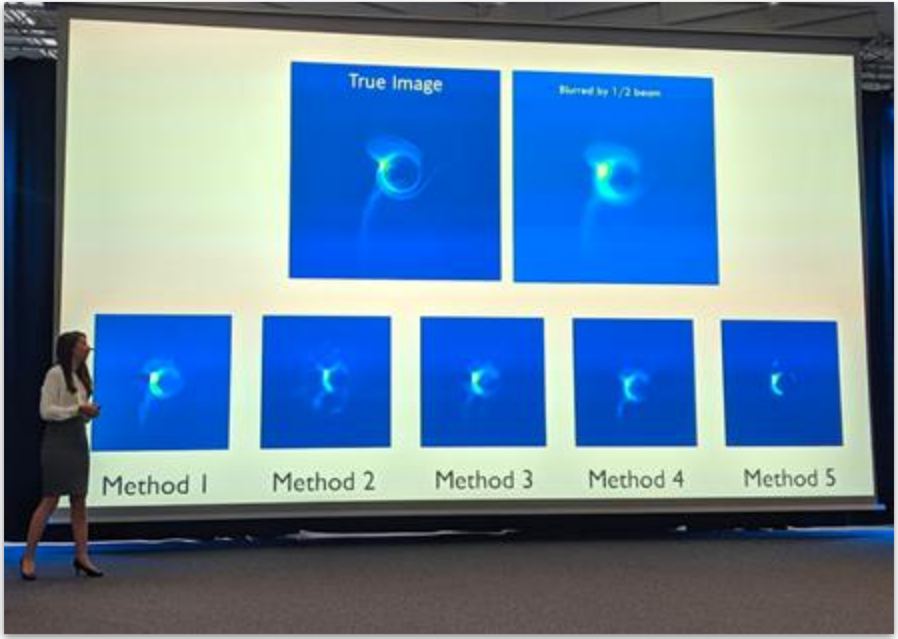


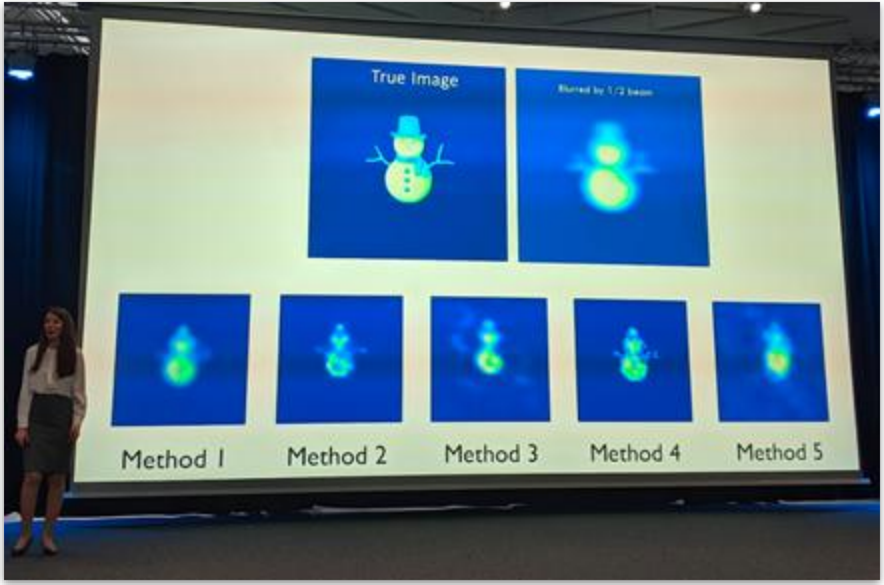
Katie Bouman
MIT/Caltech

Event Horizon Telescope

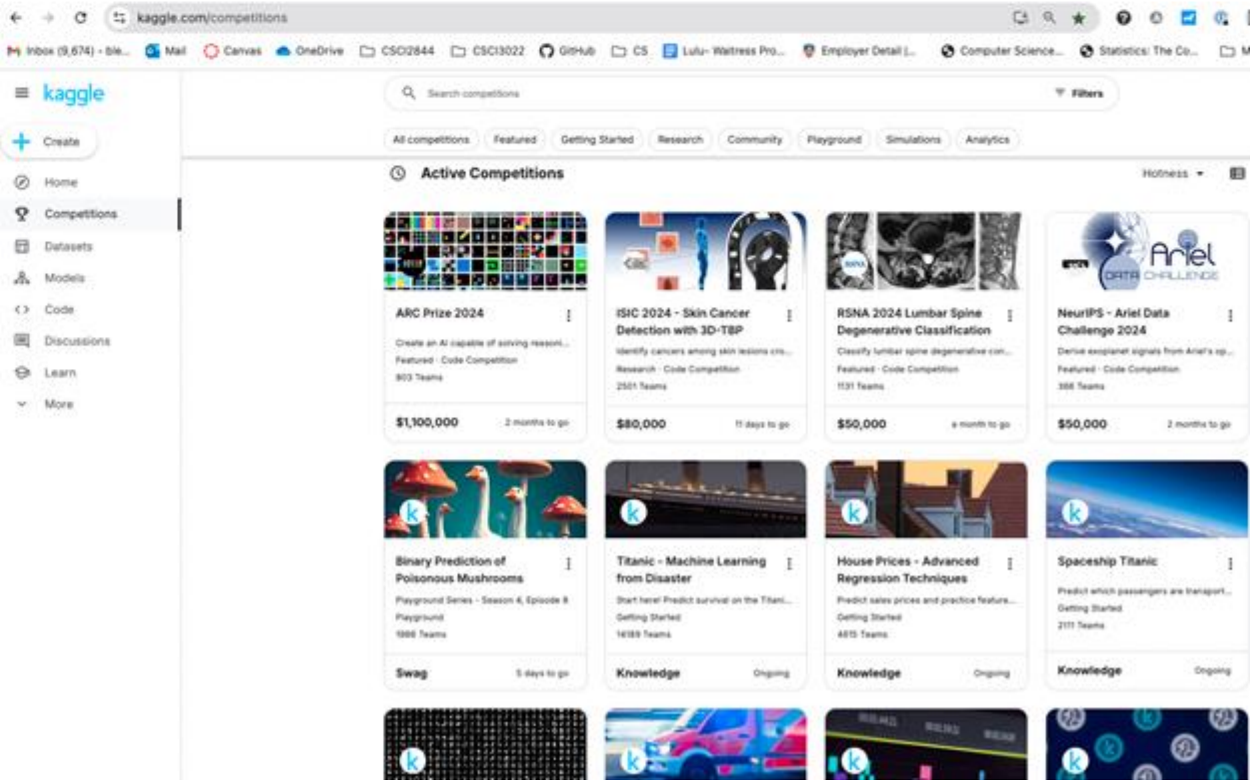
Talk Video: https://youtu.be/TSgpIiktkwc

2019: First Image of a Black Hole





Kaggle: More Data Science Sets and Competitions



Active Competitions

ARC Prize 2024
Create an AI capable of solving reasoni...
Featured · Code Competition
803 Teams
$1,100,000 2 months to go

ISIC 2024 - Skin Cancer Detection with 3D-TBP
Identify cancers among skin lesions cro...
Research · Code Competition
2501 Teams
$80,000 11 days to go

RSNA 2024 Lumbar Spine Degenerative Classification
Classify lumbar spine degenerative con...
Featured · Code Competition
1131 Teams
$50,000 a month to go

NeurIPS - Ariel Data Challenge 2024
Derive exoplanet signals from Ariel's op...
Featured · Code Competition
388 Teams
$50,000 2 months to go

Binary Prediction of Poisonous Mushrooms
Playground Series - Season 4, Episode 8
Playground
1986 Teams
Swag 5 days to go

Titanic - Machine Learning from Disaster
Start here! Predict survival on the Titani...
Getting Started
16189 Teams
Knowledge Ongoing

House Prices - Advanced Regression Techniques
Predict sales prices and practice feature...
Getting Started
4815 Teams
Knowledge Ongoing

Spaceship Titanic
Predict which passengers are transport...
Getting Started
2111 Teams
Knowledge Ongoing

https://www.kaggle.com/competitions

List of Topics to be Covered in CSCI 3022

- **Exploring, Cleaning and Visualizing Data**
  - Intro to Pandas and NumPy
  - Exploratory Data Analysis
  - Wrangling Data
  - Visualizing Data using matplotlib, seaborn & plotly

- **Probability & Statistics for Data Science**
  - Independence
  - Conditioning and Bayes Theorem
  - Discrete and Continuous Random Variables
  - Distributions and Joint Distributions
  - Expectation & Variance
  - Central Limit Theorem
  - Sampling
  - Using Statistical Simulation To Draw Inferences from Data:
    - Hypothesis and A/B Testing
    - Bootstrapping Confidence Intervals

- **Modeling/Intro to Machine Learning**
  - Model design and loss formulation
  - Simple Linear Regression
  - Multiple Linear Regression
  - Logistic Regression
  - Feature Engineering
  - Cross-Validation
  - Regularization

iclicker Poll: https://join.iclicker.com/XSFZ

Which course topic are you most interested to learn?

Lesson Learning Objectives:

- Name and explain the stages of the data science lifecycle

The Data Science Lifecycle

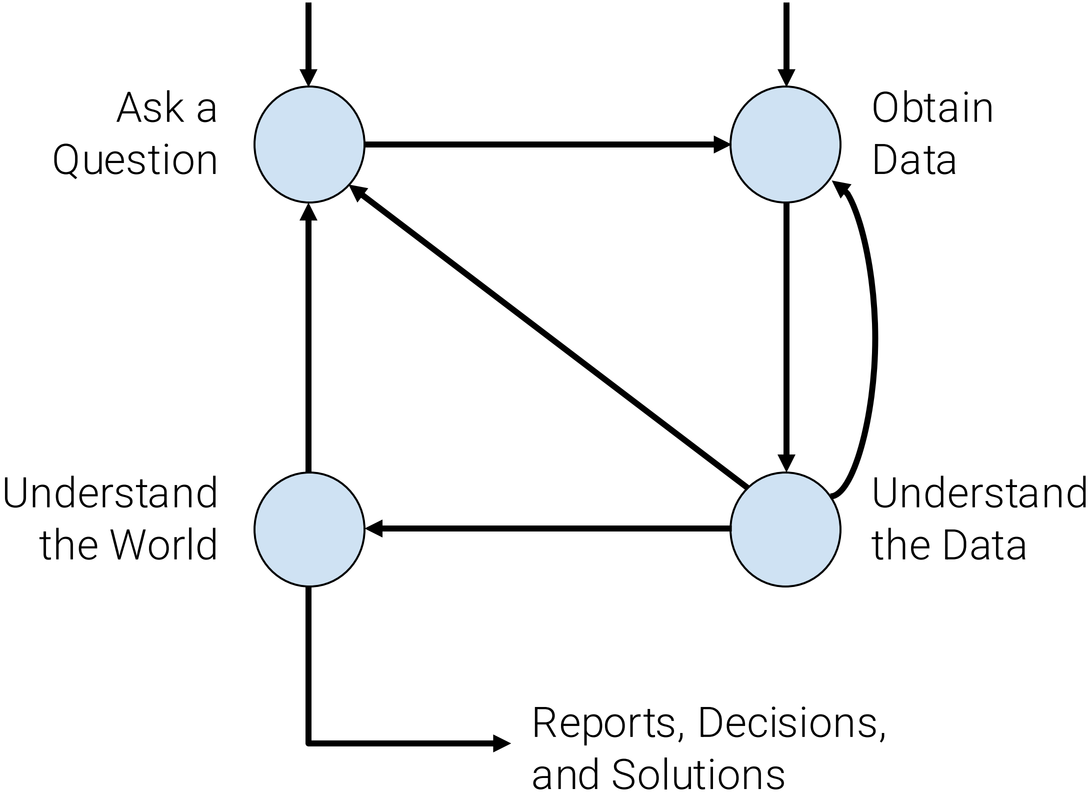
Lecture 01, CSCI 3022

- Intros & Logistics
- What is Data Science?
  - **Data Science Lifecycle**
- Exploratory Data Analysis & Wrangling
- Intro to Pandas
  - Jupyter Notebook Demo: EDA using Pandas

Data science lifecycle

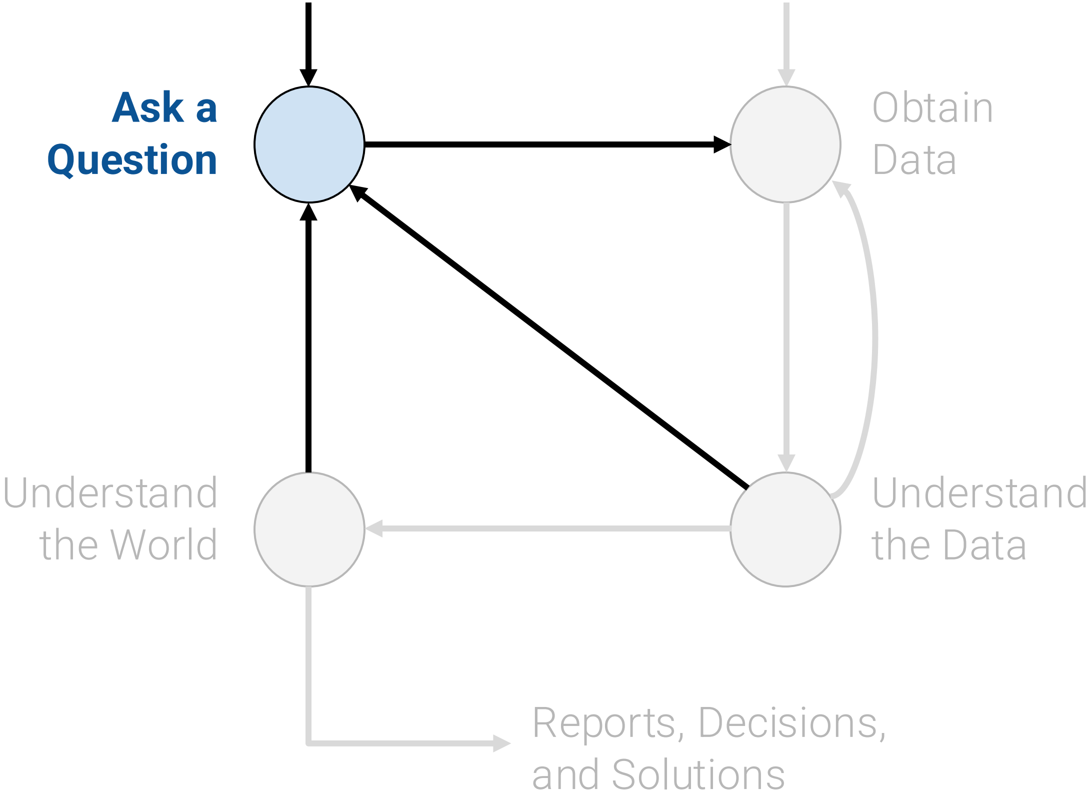
The data science lifecycle is a **high-level description** of the data science workflow.

Note the two distinct entry points!



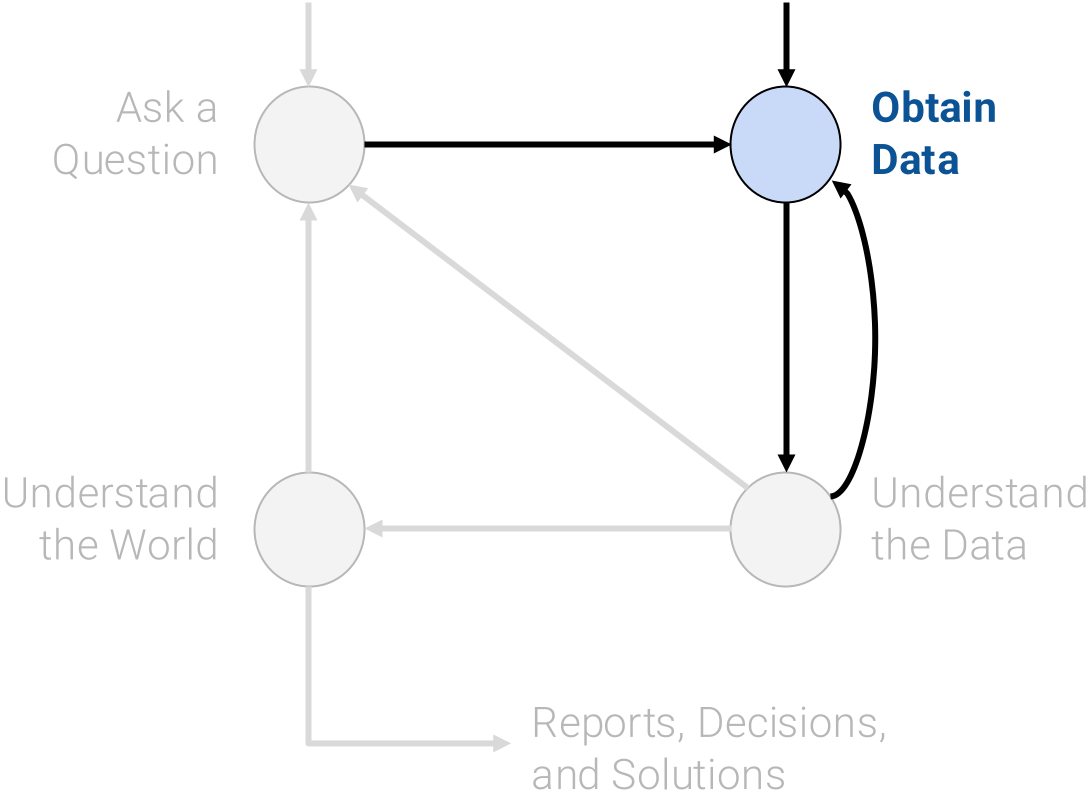
1. Question/Problem Formulation

- What do we want to know?
- What problems are we trying to solve?
- What are the hypotheses we want to test?
- What are our metrics for success?



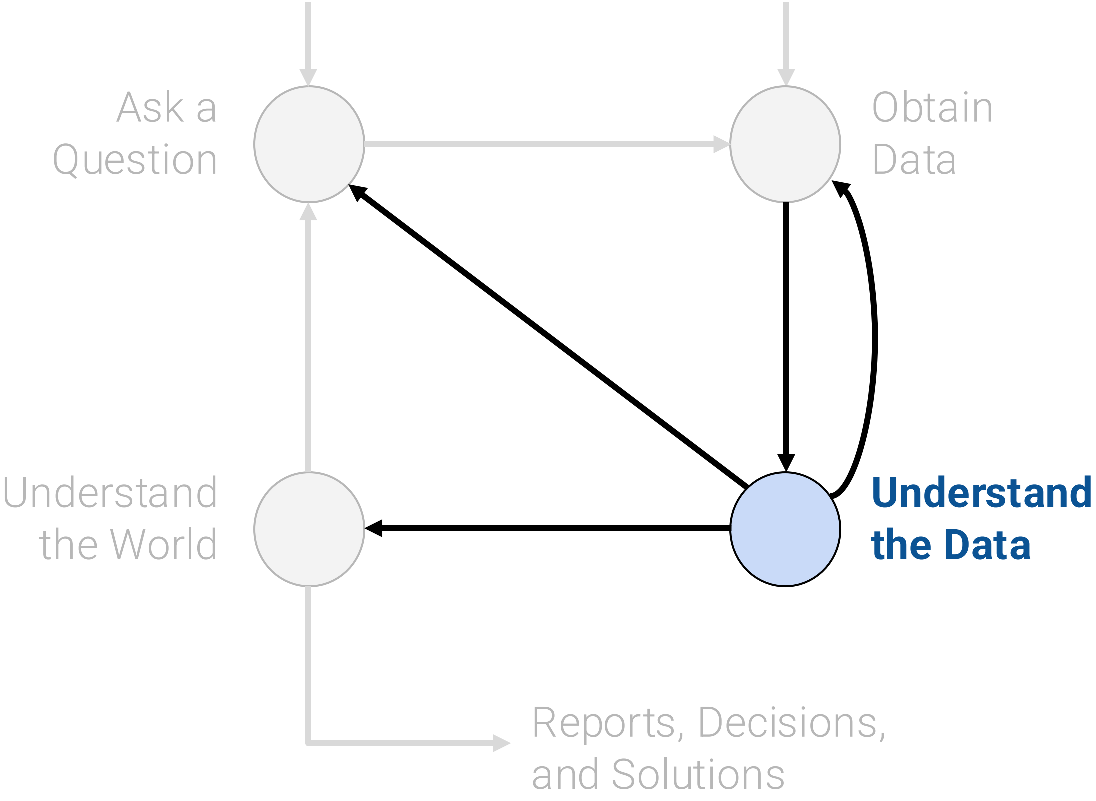
2. Data Acquisition and Cleaning

- What data do we have and what data do we need?
- How will we sample more data?
- Is our data representative of the population we want to study?



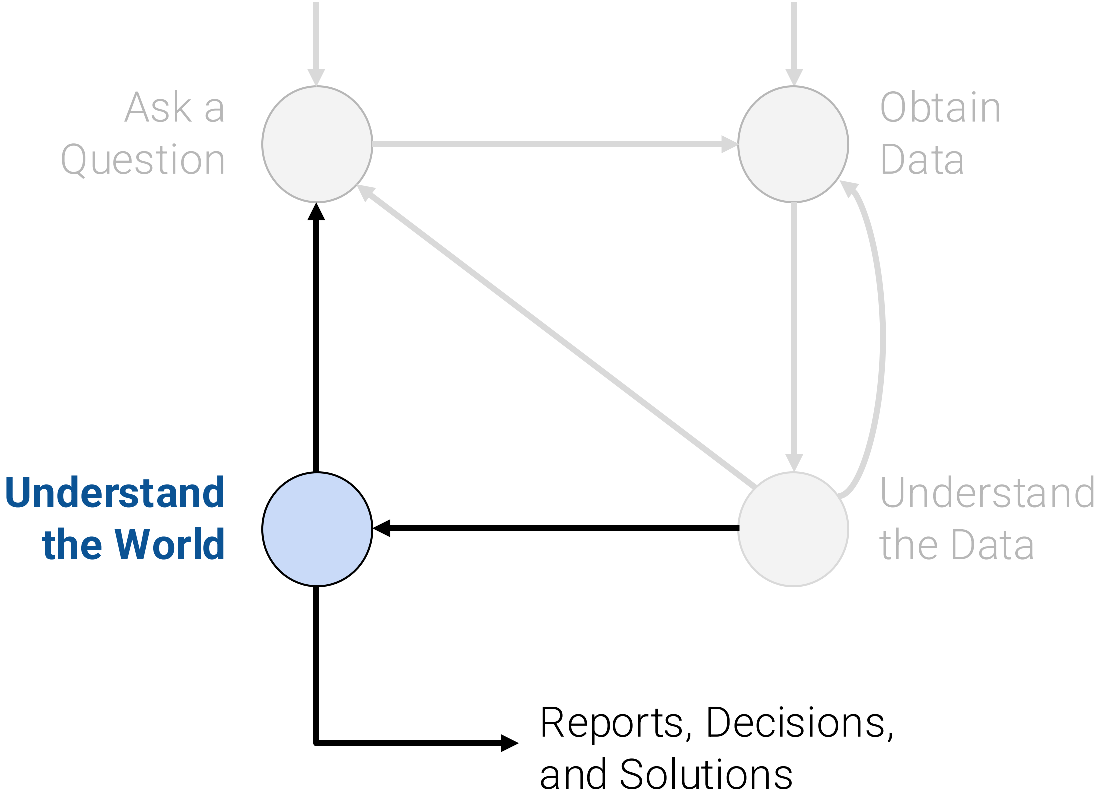
3. Exploratory Data Analysis & Visualization

- How is our data organized and what does it contain?
- Do we already have relevant data?
- What are the biases, anomalies, or other issues with the data?
- How do we transform the data to enable effective analysis?

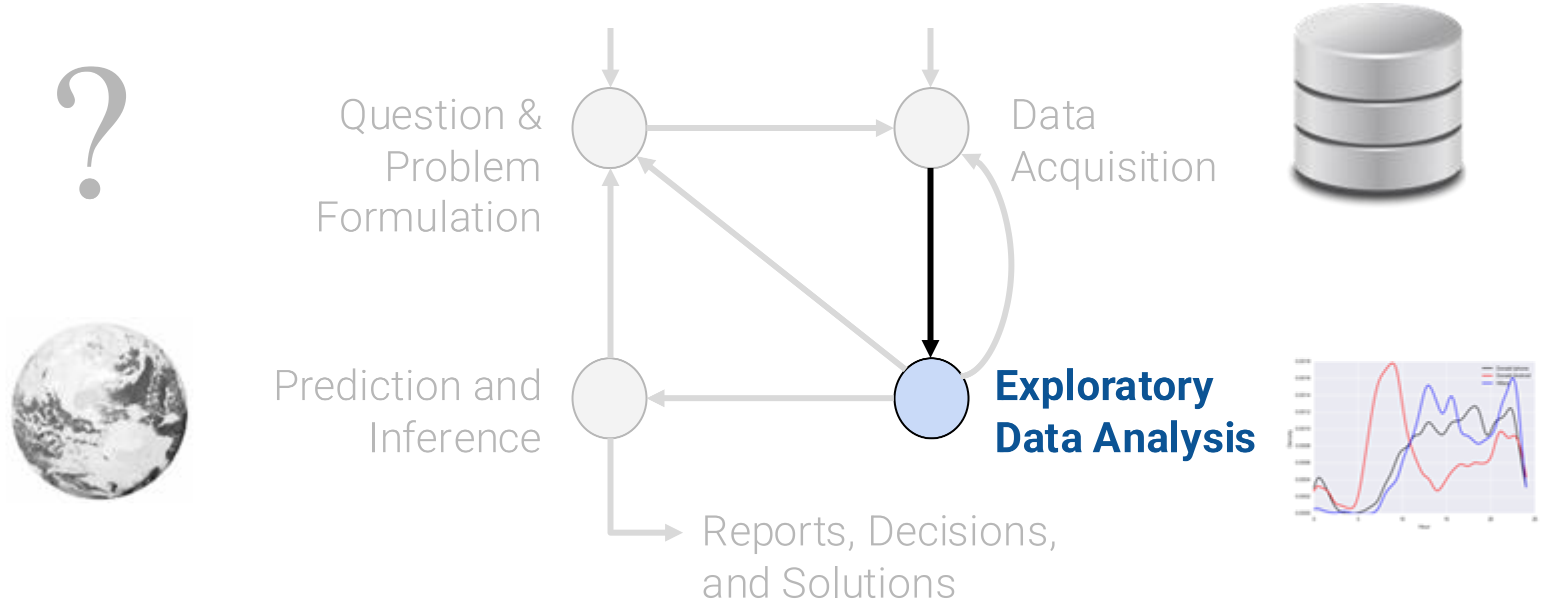


4. Prediction and Inference

- What does the data say about the world?
- Does it answer our questions or accurately solve the problem?
- How robust are our conclusions and can we trust the predictions?



Plan for first few weeks



Question & Problem Formulation

Data Acquisition

Prediction and Inference

Exploratory Data Analysis

Reports, Decisions, and Solutions

(Weeks 1 and 2)

EDA, Wrangling and Data Visualization

Lesson Learning Objectives:

- Meet your classmates
- Name and explain the stages of the data science lifecycle
- **Identify 5 key data properties to consider when doing Exploratory Data Analysis and implement using demo data**

EDA & Wrangling

Lecture 01, CSCI 3022

- Intros & Logistics
- What is Data Science?
- **Exploratory Data Analysis & Wrangling**
- Intro to Pandas:
  - Jupyter Notebook Demo: EDA using Pandas

Congratulations!!!

You **have collected** or **have been given** a box of data.

What do you do next?

One Option: Exploratory Data Analysis (EDA)

> *"Getting to know and understand the data"*

The process of **transforming**, **visualizing**, and **summarizing** data to:

- Build/confirm understanding of the data and its **provenance**
- Identify and address potential issues in the data.
- Inform the subsequent analysis.
- Discover *potential* hypotheses …

Provenance: origin of data; methodology by which data were produced

EDA is an open-ended analysis.

- Informal, no specific idea of what we are looking for.
- Be willing to find something surprising!

Contrast with **confirmatory analysis**:

- Questions are fixed in advance.
- Allows for more rigorous statistical analysis.

Another Option: Data Wrangling (Data Cleaning)

Data Wrangling, or **Data Cleaning**:

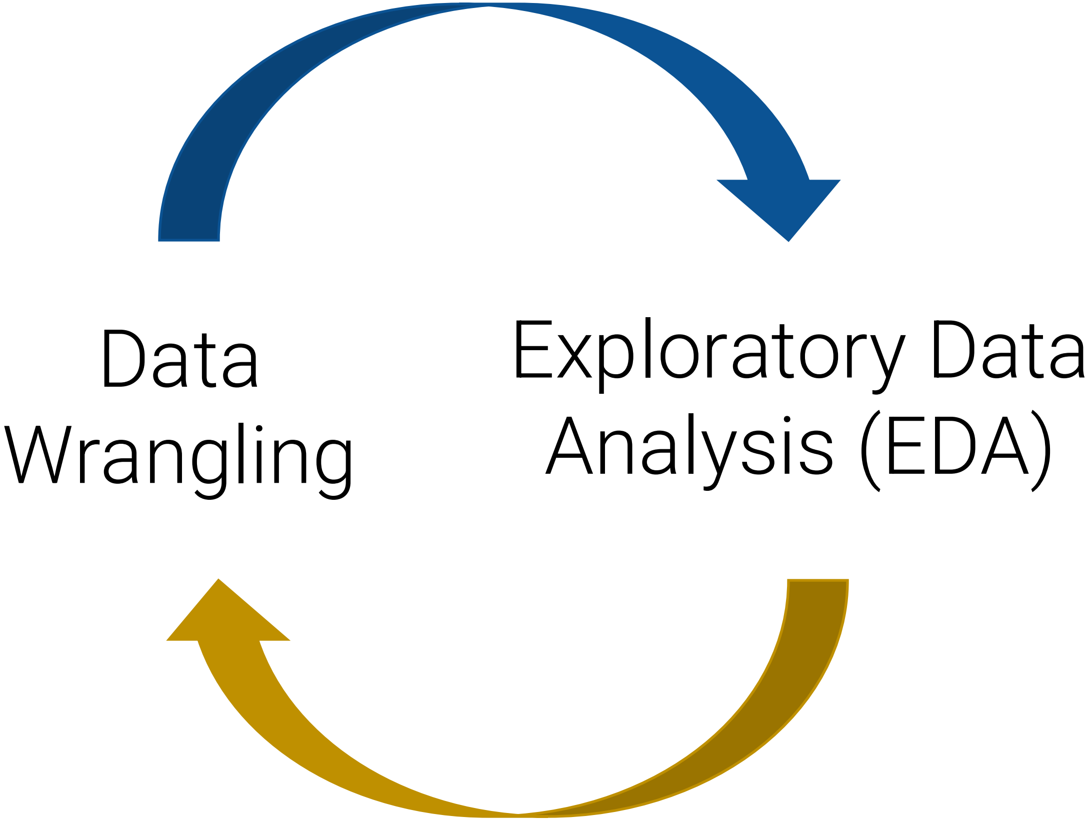
The process of transforming **raw data** to facilitate subsequent analysis.

Often addresses **issues** like…

- structure / formatting
- missing or corrupted values
- unit conversion
- encoding text as numbers
- …

Sadly, data cleaning is a big part of data science…

The Infinite Loop of Data Science



Data Wrangling

Exploratory Data Analysis (EDA)

EDA is unboxing for data!

Exploratory Data Analysis (EDA) Guiding Principles

"Exploratory data analysis is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those that we believe to be there." – John Tukey

Key Data Properties to Consider in EDA

Structure -- the “shape” of a data file

Granularity -- what does each record represent?

Scope -- how (in)complete is the data

Temporality -- how is the data situated in time

Faithfulness -- how well does the data capture “reality”

File Format

Variable Type

Multiple files
(Primary and Foreign Keys)

Structure -- the “shape” of a data file

Granularity -- how fine/coarse is each datum

Scope -- how (in)complete is the data

Temporality -- how is the data situated in time

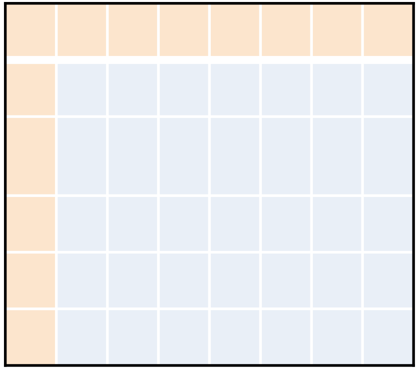
Faithfulness -- how well does the data capture “reality”

Rectangular and Non-rectangular Data

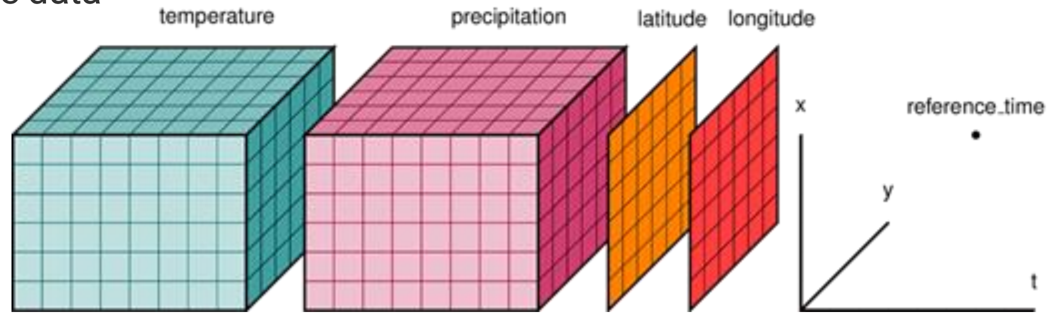
Data come in many different shapes.

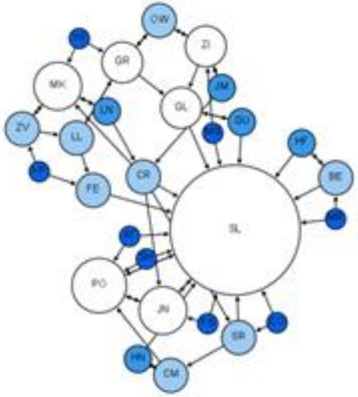
Dataset's *structure:* Mental representation of the data

Tabular (rectangular) data



Non-rectangular data



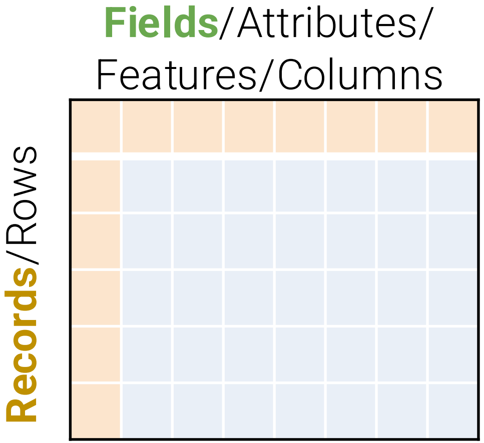


Tabular/Rectangular Data

We often prefer tabular data for data analysis (why?)

- Regular structures are easy manipulate and analyze
- A big part of data cleaning is about transforming data to be more rectangular

Two kinds of tabular data: **Tables** and **Matrices**.



Tables (a.k.a. `DataFrames` in R/Python and relations in SQL)

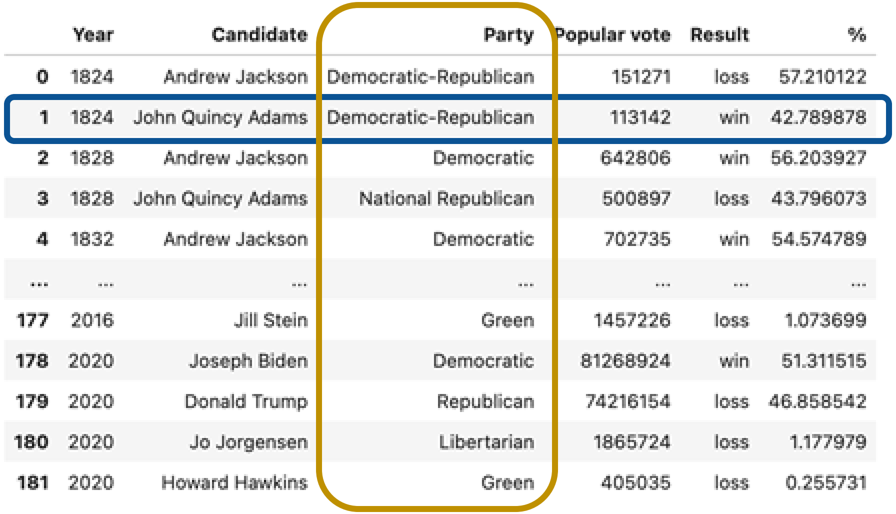
- Named columns with different types
- Manipulated using Pandas data transformation languages (map, filter, group by, join, …)

Matrices

- Numeric data of the same type (float, int, etc.)
- Manipulated using linear algebra

Data Scientists Love Rectangular/Tabular Data

"Tabular= data in a table.



	Year	Candidate	Party	Popular vote	Result	%
0	1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1	1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
2	1828	Andrew Jackson	Democratic	642806	win	56.203927
3	1828	John Quincy Adams	National Republican	500897	loss	43.796073
4	1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...	...
177	2016	Jill Stein	Green	1457226	loss	1.073699
178	2020	Joseph Biden	Democratic	81268924	win	51.311515
179	2020	Donald Trump	Republican	74216154	loss	46.858542
180	2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
181	2020	Howard Hawkins	Green	405035	loss	0.255731

A **row** represents one record (i.e. an observation)

A **column** represents some characteristic, or feature, of that observation (here, the political party of that person).

A tabular **dataset** is **tidy** when each column corresponds to one **variable** in the **dataset**, each row corresponds to one **observation**, and all **variables** in the **dataset** have the same unit of observation.

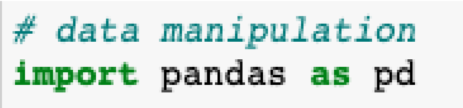
Intro To Pandas

Lecture 01, CSCI 3022

- Intros & Logistics
- What is Data Science?
- Exploratory Data Analysis & Wrangling
- Intro To Pandas
  - Jupyter Notebook Demo: EDA using Pandas

Introducing the Standard Python Data Science Tool: Pandas

The Python Data Analysis Library



```
# data manipulation
import pandas as pd
```

- **Pandas** (derived from Panel Data) is a Data Analysis library to make data cleaning and analysis fast and convenient in Python.
- Pandas adopts many coding idioms from NumPy, the biggest difference is that ***pandas is designed for working with tabular or heterogeneous data***.
  - NumPy by contrast is best suited for working with homogenous numerical data.

Tabular data is one of the most common data formats.

- Will be our primary focus in CSCI 3022

Introducing the Standard Python Data Science Tool: pandas

Using pandas, we can:

- Arrange data in a tidy tabular format.
- Extract useful information filtered by specific conditions.
- Operate on data to gain new insights.
- Apply NumPy functions to our data
- Perform vectorized computations to speed up our analysis.

pandas is the standard tool across research and industry for working with tabular data.

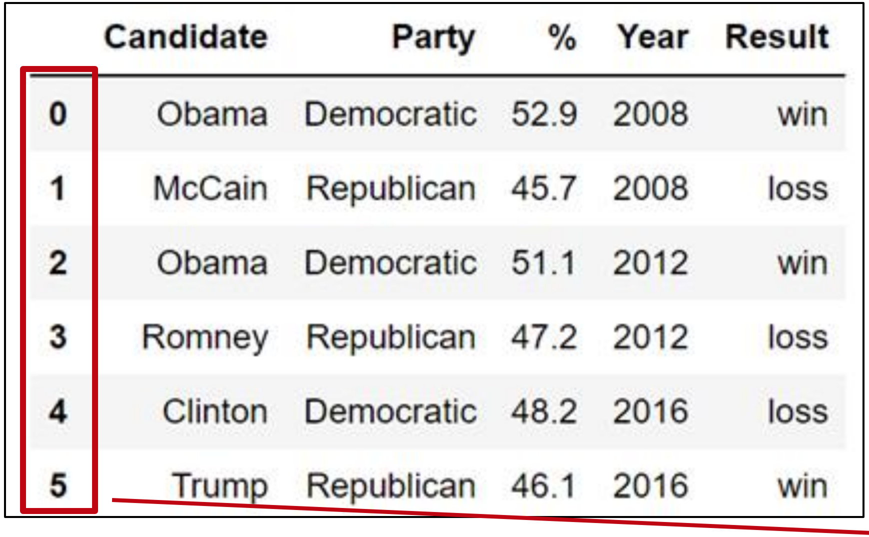
The first week of this course will serve as a "bootcamp" in helping you build familiarity with operating on data with pandas.

Pandas Data Structures

There are three fundamental data structures in pandas:

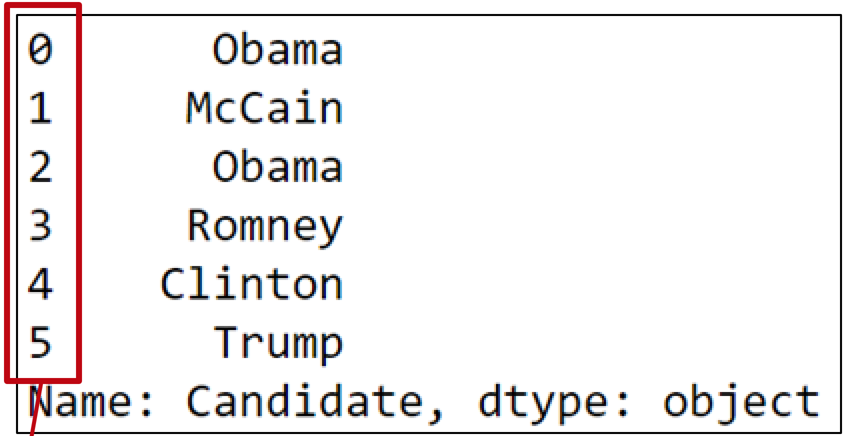
- **Series**: 1D labeled array data. I usually think of it as columnar data.
- **Data Frame**: 2D tabular data with both row and column labels
- **Index**: A sequence of row/column labels.

Data Frame



	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Series named "Candidate"



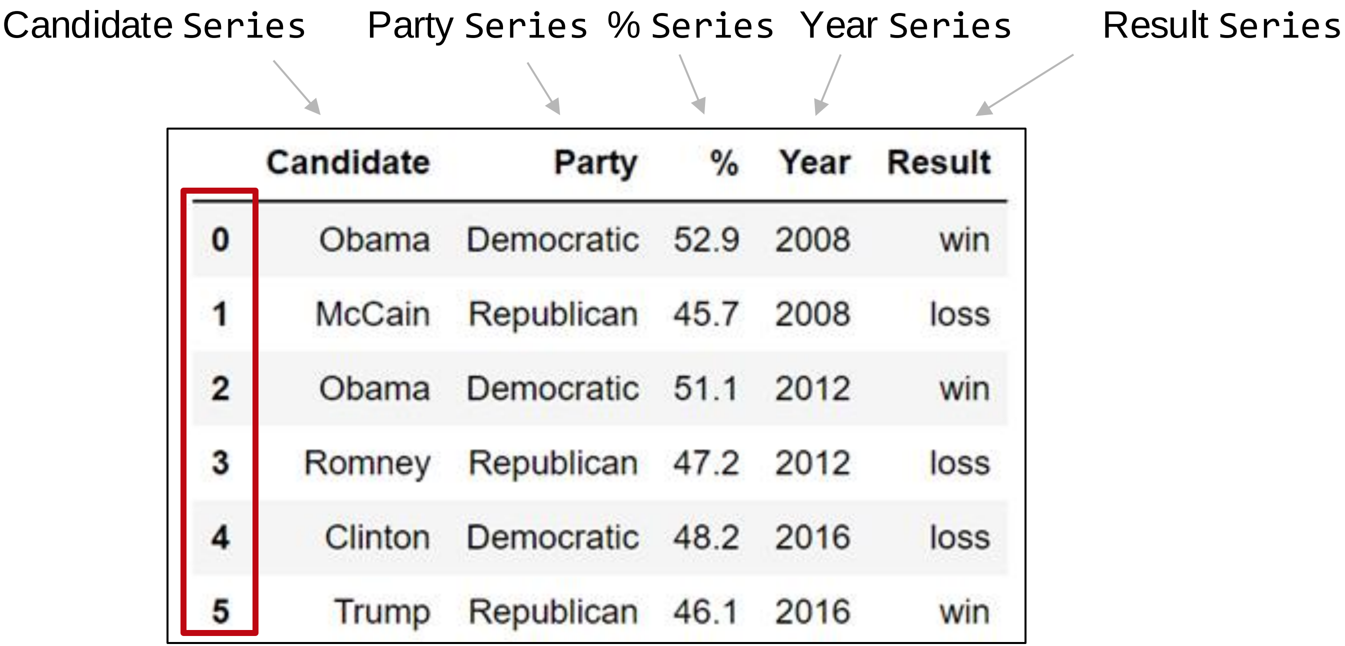
```
0        Obama
1       McCain
2        Obama
3       Romney
4      Clinton
5        Trump
Name: Candidate, dtype: object
```

Index

The Relationship Between DataFrames, Series, and Indices

We can think of a **`DataFrame`** as a collection of **`Series`** that all share the same **`Index`**.

- Candidate, Party, %, Year, and Result **`Series`** all share an **`Index`** from 0 to 5.



Candidate `Series` Party `Series` % `Series` Year `Series` Result `Series`

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Lesson Learning Objectives:

- **Identify 5 key data properties to consider when doing Exploratory Data Analysis and implement using demo data**

Jupyter Demo

Lecture 01, CSCI 3022

- Intros & Logistics
- What is Data Science?
- Exploratory Data Analysis & Wrangling
- Intro to Pandas
  - Jupyter Notebook Demo: EDA using Pandas

Demo Slides

CSV: Comma-Separated Values

Election Data in the US

CSV is a very common **tabular file format**.

- **Records** (rows) are delimited by a newline: `'\n'`, `"\r\n"`
- **Fields** (columns) are delimited by commas: `','`

Pandas: `pd.read_csv(header=...)`

Fields/Attributes/Features/Columns

Records/Rows

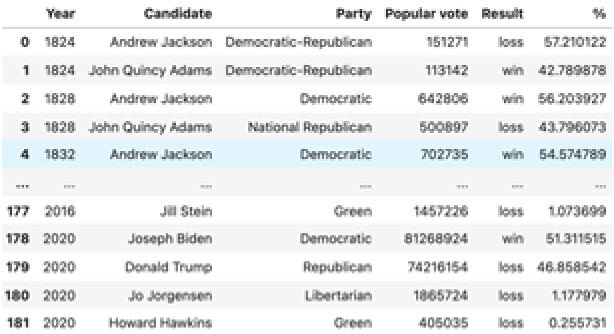
	Year	Candidate	...
0	2024	Kamala Harris	...
1	2024	Donald Trump	...

Creating a `DataFrame`

Many approaches exist for creating a `DataFrame`.

1). From a CSV file.

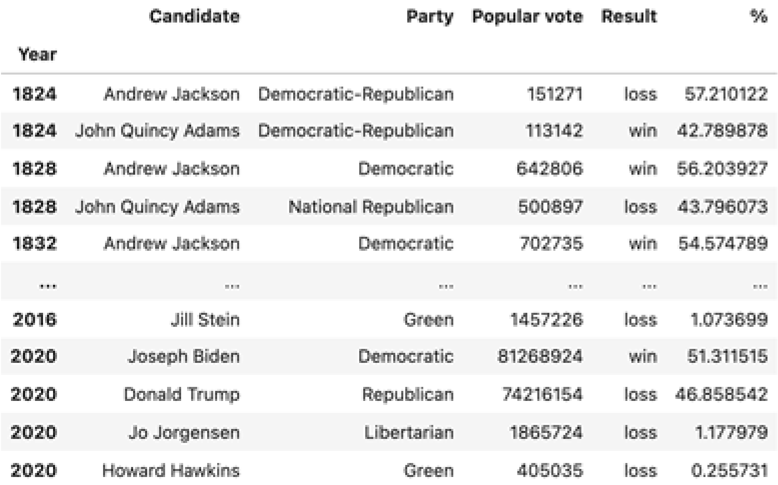
```
elections = pd.read_csv("data/elections.csv")
```



	Year	Candidate	Party	Popular vote	Result	%
0	1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1	1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
2	1828	Andrew Jackson	Democratic	642806	win	56.203927
3	1828	John Quincy Adams	National Republican	500897	loss	43.796073
4	1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...	...
177	2016	Jill Stein	Green	1457226	loss	1.073699
178	2020	Joseph Biden	Democratic	81268924	win	51.311515
179	2020	Donald Trump	Republican	74216154	loss	46.858542
180	2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
181	2020	Howard Hawkins	Green	405035	loss	0.255731

The DataFrame `elections`

```
elections = pd.read_csv("data/elections.csv", index_col="Year")
```



	Candidate	Party	Popular vote	Result	%
Year					
1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
1828	Andrew Jackson	Democratic	642806	win	56.203927
1828	John Quincy Adams	National Republican	500897	loss	43.796073
1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...
2016	Jill Stein	Green	1457226	loss	1.073699
2020	Joseph Biden	Democratic	81268924	win	51.311515
2020	Donald Trump	Republican	74216154	loss	46.858542
2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
2020	Howard Hawkins	Green	405035	loss	0.255731

The DataFrame `elections` with `"Year"` as `Index`

2). See Supporting Materials for other methods to create DataFrame

Supporting Materials

Lecture 01, CSCI 3022

Supporting Materials: Pandas Data Structures:

Series

DataFrames

Indices

Learning Objectives

- Understand the relationship between Series, DataFrames and Indices in Pandas
- Create, filter and perform operations on Series

Pandas Data Structures: Series

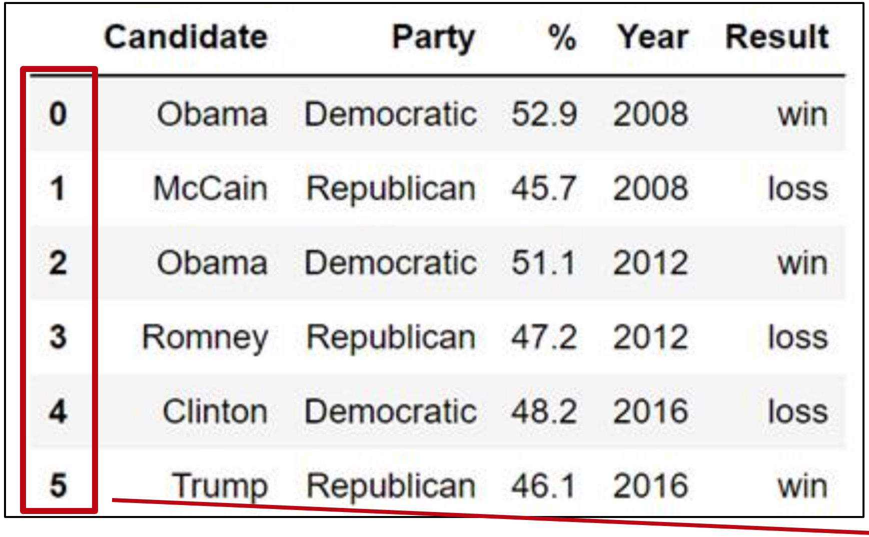
- Introduction to Series

Pandas Data Structures

There are three fundamental data structures in pandas:

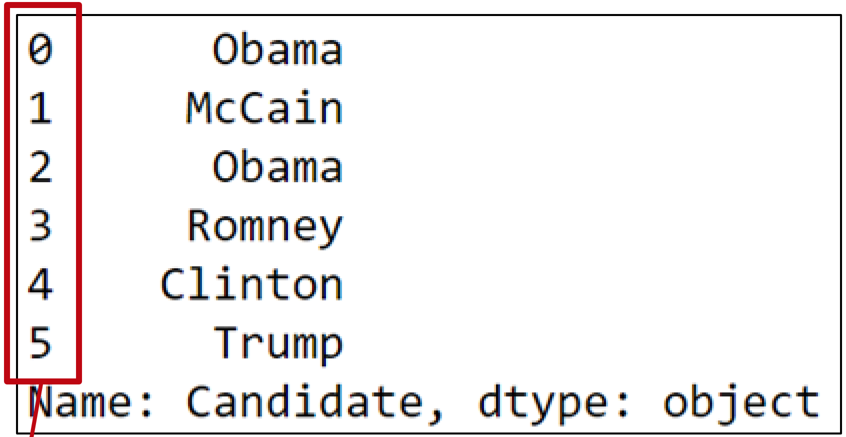
- **Series**: 1D labeled array data. I usually think of it as columnar data.
- **Data Frame**: 2D tabular data with both row and column labels
- **Index**: A sequence of row/column labels.

Data Frame



	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Series named “Candidate”



```
0         Obama
1        McCain
2         Obama
3        Romney
4       Clinton
5         Trump
Name: Candidate, dtype: object
```

Index

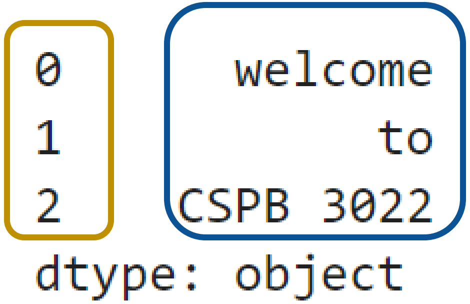
Series

A **Series** is a 1-dimensional array-like object. It contains:

- A sequence of **values** of the same type.
- A sequence of data labels, called the **index**.

pd is the conventional alias for **pandas**

```
import pandas as pd
s = pd.Series(["welcome", "to", "cspb 3022"])
```



```
0      welcome
1           to
2    CSPB 3022
dtype: object
```

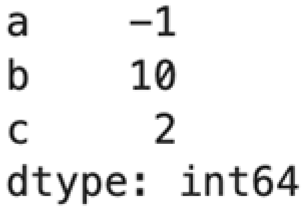
Index, accessed by calling `s.index`

Values, accessed by calling `s.values`

Series - Custom Index

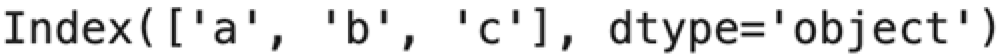
- We can provide index labels for items in a Series by passing an index list.

```
s = pd.Series([-1, 10, 2], index = ["a", "b", "c"])
```



```
a    -1
b    10
c     2
dtype: int64
```

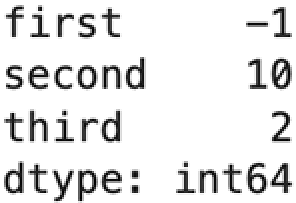
```
s.index
```



```
Index(['a', 'b', 'c'], dtype='object')
```

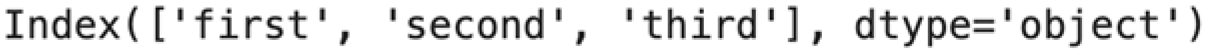
- A Series index can also be changed.

```
s.index = ["first", "second", "third"])
```



```
first     -1
second    10
third      2
dtype: int64
```

```
s.index
```

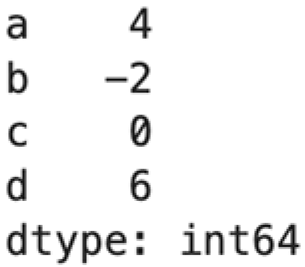


```
Index(['first', 'second', 'third'], dtype='object')
```

Selection in Series

- We can select a single value or a set of values in a Series using:
  - A single label
  - A list of labels
  - A filtering condition

```
s = pd.Series([4, -2, 0, 6], index = ["a", "b", "c", "d"])
```

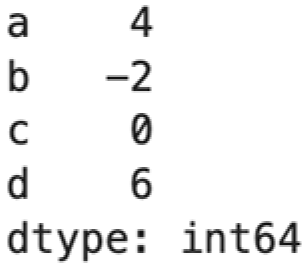


```
a    4
b   -2
c    0
d    6
dtype: int64
```

Selection in Series

- We can select a single value or a set of values in a Series using:
    - **A single label**
    - A list of labels
    - A filtering condition

```
s = pd.Series([4, -2, 0, 6], index = ["a", "b", "c", "d"])
```



```
a     4
b    -2
c     0
d     6
dtype: int64
```

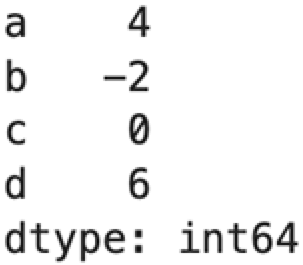
```
s["a"]
```

```
4
```

Selection in Series

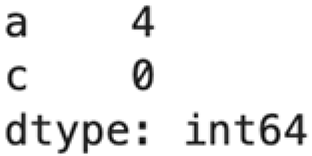
- We can select a single value or a set of values in a Series using:
    - A single label
    - **A list of labels**
    - A filtering condition

```
s = pd.Series([4, -2, 0, 6], index = ["a", "b", "c", "d"])
```



```
a     4
b    -2
c     0
d     6
dtype: int64
```

```
s[["a", "c"]]
```

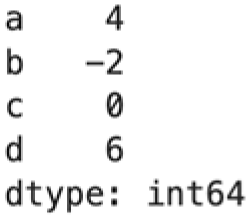


```
a     4
c     0
dtype: int64
```

Selection in `Series`

- We can select a single value or a set of values in a **`Series`** using:
  - A single label
  - A list of labels
  - **A filtering condition**

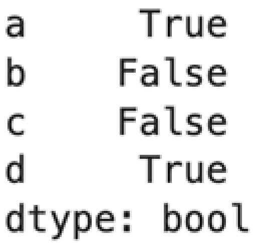
```
s = pd.Series([4, -2, 0, 6], index = ["a", "b", "c", "d"])
```



```
a    4
b   -2
c    0
d    6
dtype: int64
```

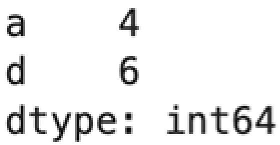
- Say we want to select values in the **`Series`** that satisfy a particular condition:
  1) Apply a boolean condition to the **`Series`**. This creates a **new `Series` of boolean values**.
  2) Index into our **`Series`** using this boolean condition. **`pandas`** will select only the entries in the **`Series`** that satisfy the condition.

```
s > 0
```



```
a     True
b    False
c    False
d     True
dtype: bool
```

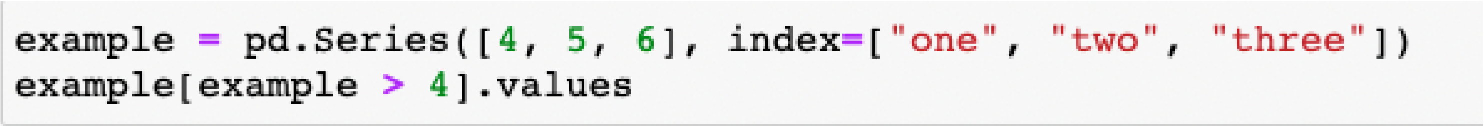
```
s[s > 0]
```



```
a    4
d    6
dtype: int64
```

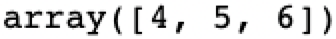
Practice Question

What is the output of the following code?



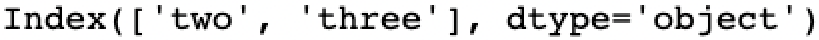
```
example = pd.Series([4, 5, 6], index=["one", "two", "three"])
example[example > 4].values
```

A).



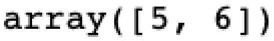
```
array([4, 5, 6])
```

B).



```
Index(['two', 'three'], dtype='object')
```

C).



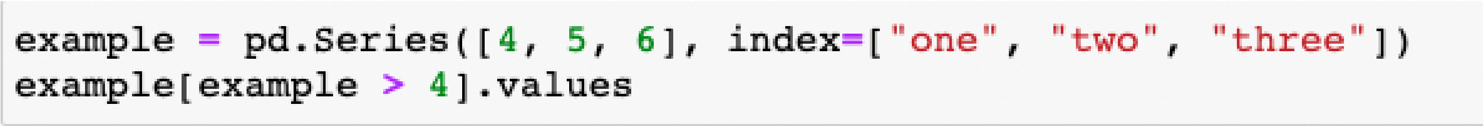
```
array([5, 6])
```

D). [5, 6]

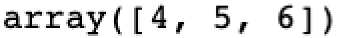
E). None of these

Practice Question

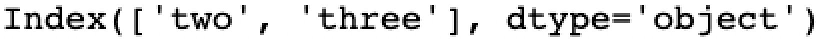
What is the output of the following code?



```
example = pd.Series([4, 5, 6], index=["one", "two", "three"])
example[example > 4].values
```

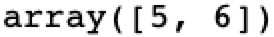
A). 

```
array([4, 5, 6])
```



```
Index(['two', 'three'], dtype='object')
```

B).

[✓ C). 

```
array([5, 6])
```
]

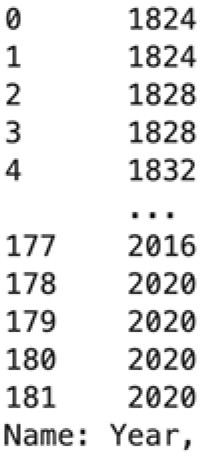
D). [5, 6]

E). None of these

`DataFrame`s of `Series`!

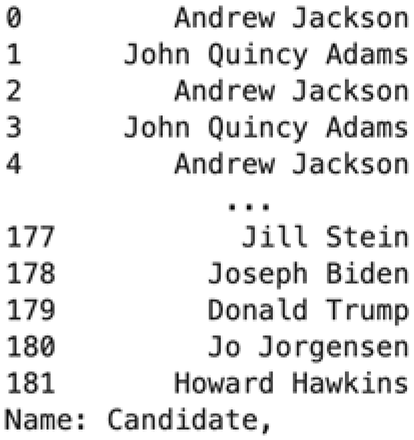
Typically, we will work with **`Series`** using the perspective that they are columns in a **`DataFrame`**.

We can think of a **`DataFrame`** as a collection of **`Series`** that all share the same **`Index`**.



```
0        1824
1        1824
2        1828
3        1828
4        1832
         ...
177      2016
178      2020
179      2020
180      2020
181      2020
Name: Year,
```

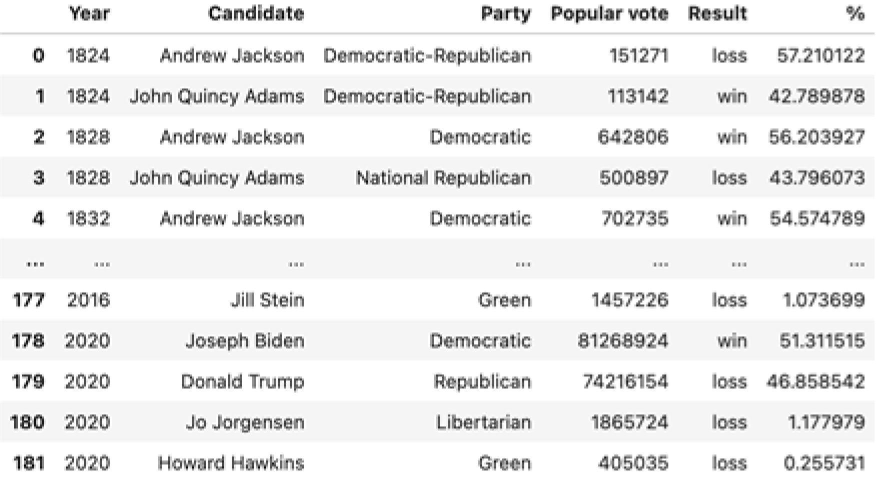
The Series **`"Year"`**



```
0          Andrew Jackson
1       John Quincy Adams
2          Andrew Jackson
3       John Quincy Adams
4          Andrew Jackson
              ...
177            Jill Stein
178          Joseph Biden
179          Donald Trump
180          Jo Jorgensen
181        Howard Hawkins
Name: Candidate,
```

The Series **`"Candidate"`**

[...]



	Year	Candidate	Party	Popular vote	Result	%
0	1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1	1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
2	1828	Andrew Jackson	Democratic	642806	win	56.203927
3	1828	John Quincy Adams	National Republican	500897	loss	43.796073
4	1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...	...
177	2016	Jill Stein	Green	1457226	loss	1.073699
178	2020	Joseph Biden	Democratic	81268924	win	51.311515
179	2020	Donald Trump	Republican	74216154	loss	46.858542
180	2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
181	2020	Howard Hawkins	Green	405035	loss	0.255731

The DataFrame **`elections`**

Non-native English speaker note: The plural of "series" is "series". Sorry.

Learning Objectives

- Understand the relationship between DataFrames and Indices in Pandas
- Create DataFrames
- Manipulate indices

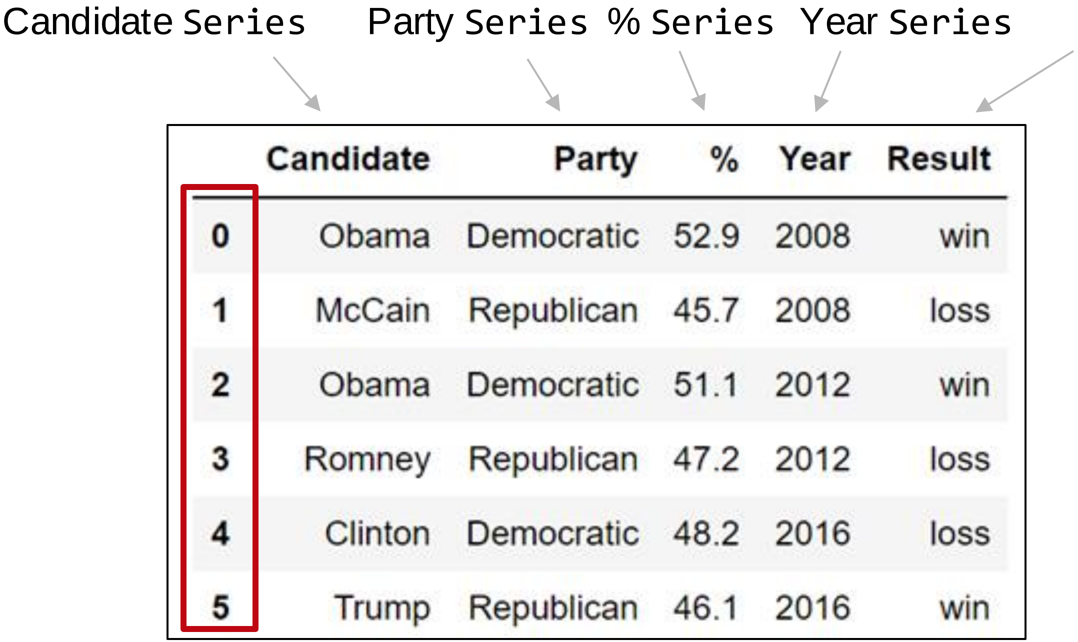
Pandas Data Structures

- Introduction to DataFrames
- Indices

The Relationship Between DataFrames, Series, and Indices

We can think of a **`DataFrame`** as a collection of **`Series`** that all share the same **`Index`**.

- Candidate, Party, %, Year, and Result **`Series`** all share an **`Index`** from 0 to 5.



Candidate `Series`

Party `Series`

% `Series`

Year `Series`

Result `Series`

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

The `DataFrame` API

The API for the **`DataFrame`** class is enormous.

- API: "Application Programming Interface".
- The API is the set of abstractions supported by the class.

Full documentation is at https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

- We will only consider a tiny portion of this API.

We want you to get familiar with the real world programming practice of… Googling!

- Answers to your questions are often found in the **`pandas`** documentation, Stack Overflow, etc.

Creating a `DataFrame`

The syntax of creating **`DataFrame`** is:

```
pandas.DataFrame(data, index, columns)
```

Many approaches exist for creating a **`DataFrame`**. Here, we will go over the most popular ones.

- From a CSV file.
- Using a list and column name(s).
- From a dictionary.
- From a **`Series`**.

Creating a `DataFrame`

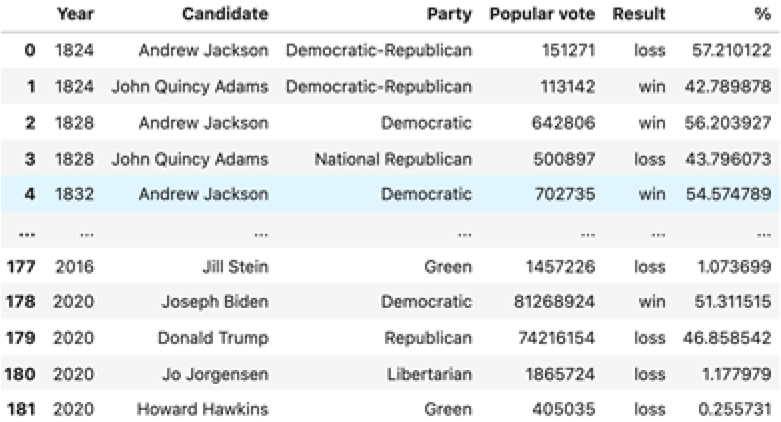
The syntax of creating **`DataFrame`** is:

```
pandas.DataFrame(data, index, columns)
```

Many approaches exist for creating a **`DataFrame`**. Here, we will go over the most popular ones.

- **From a CSV file**.
- Using a list and column name(s).
- From a dictionary.
- From a `Series`.

```
elections = pd.read_csv("data/elections.csv")
```



	Year	Candidate	Party	Popular vote	Result	%
0	1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1	1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
2	1828	Andrew Jackson	Democratic	642806	win	56.203927
3	1828	John Quincy Adams	National Republican	500897	loss	43.796073
4	1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...	...
177	2016	Jill Stein	Green	1457226	loss	1.073699
178	2020	Joseph Biden	Democratic	81268924	win	51.311515
179	2020	Donald Trump	Republican	74216154	loss	46.858542
180	2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
181	2020	Howard Hawkins	Green	405035	loss	0.255731

The DataFrame `elections`

Creating a `DataFrame`

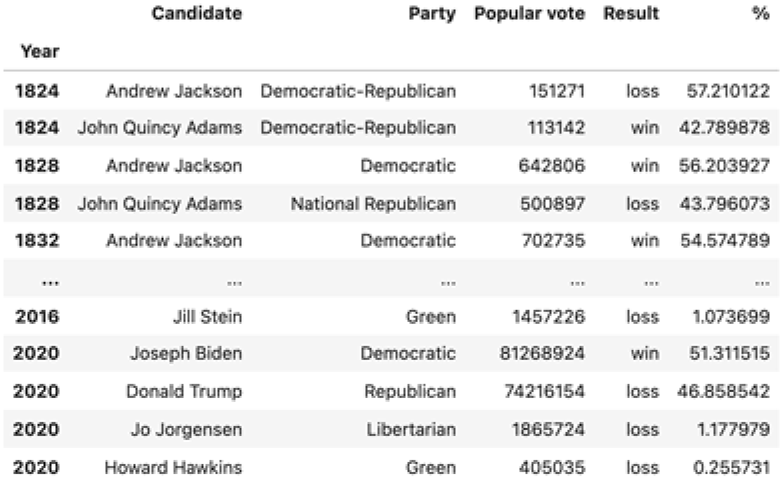
The syntax of creating **`DataFrame`** is:

```
pandas.DataFrame(data, index, columns)
```

Many approaches exist for creating a **`DataFrame`**. Here, we will go over the most popular ones.

- **From a CSV file**.
- Using a list and column name(s).
- From a dictionary.
- From a **`Series`**.

```
elections = pd.read_csv("data/elections.csv", index_col="Year")
```



	Candidate	Party	Popular vote	Result	%
Year					
1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
1828	Andrew Jackson	Democratic	642806	win	56.203927
1828	John Quincy Adams	National Republican	500897	loss	43.796073
1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...
2016	Jill Stein	Green	1457226	loss	1.073699
2020	Joseph Biden	Democratic	81268924	win	51.311515
2020	Donald Trump	Republican	74216154	loss	46.858542
2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
2020	Howard Hawkins	Green	405035	loss	0.255731

The DataFrame **`elections`** with `"Year"` as **`Index`**

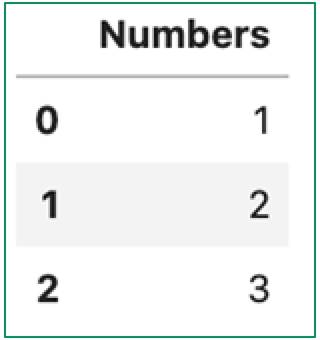
Creating a `DataFrame`

Many approaches exist for creating a `DataFrame`. Here, we will go over the most popular ones.

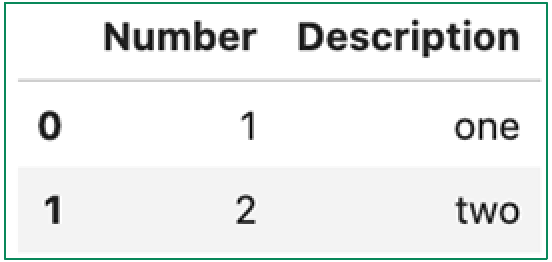
- From a CSV file.
- **Using a list and column name(s)**.
- From a dictionary.
- From a `Series`.

```
pd.DataFrame([1, 2, 3],
             columns=["Numbers"])
```

```
pd.DataFrame([[1, "one"], [2, "two"]],
            columns = ["Number", "Description"])
```



	Numbers
0	1
1	2
2	3



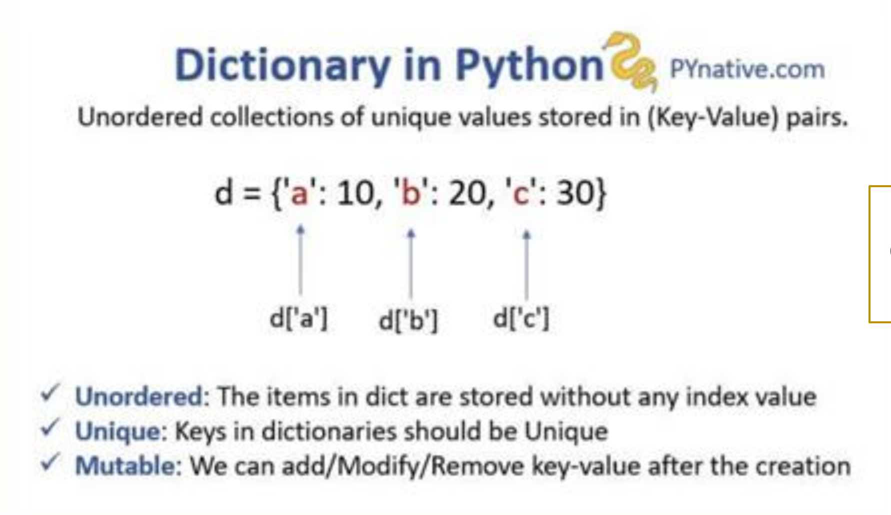
	Number	Description
0	1	one
1	2	two

Creating a `DataFrame`

Many approaches exist for creating a `DataFrame`. Here, we will go over the most popular ones.

- From a CSV file.
- Using a list and column name(s).
- **From a dictionary**.
- From a `Series`.

Wait, what's a dictionary?



Dictionary in Python

PYnative.com

Unordered collections of unique values stored in (Key-Value) pairs.

d = {'a': 10, 'b': 20, 'c': 30}

- ✓ **Unordered**: The items in dict are stored without any index value
- ✓ **Unique**: Keys in dictionaries should be Unique
- ✓ **Mutable**: We can add/Modify/Remove key-value after the creation

```
d2={"Fruit":["Strawberry",
"Orange"],  "Price": [5.49, 3.99]}
```

Creating a `DataFrame`

Many approaches exist for creating a **`DataFrame`**. Here, we will go over the most popular ones.

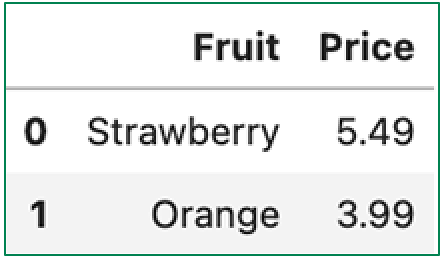
- From a CSV file.
- Using a list and column name(s).
- **From a dictionary**.
- From a **`Series`**.

Specify columns of the `DataFrame`

```
pd.DataFrame({"Fruit":["Strawberry", "Orange"],
               "Price": [5.49, 3.99]})
```

```
pd.DataFrame([{"Fruit":"Strawberry", "Price":5.49},
              {"Fruit":"Orange", "Price":3.99}])
```

Specify rows of the `DataFrame`



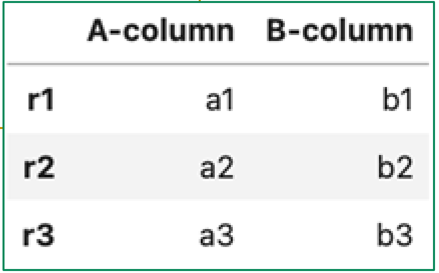
	Fruit	Price
0	Strawberry	5.49
1	Orange	3.99

Creating a `DataFrame`

Many approaches exist for creating a **`DataFrame`**. Here, we will go over the most popular ones.

- From a CSV file.
- Using a list and column name(s).
- From a dictionary.
- **From a `Series`**.

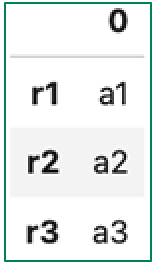
```
s_a = pd.Series(["a1", "a2", "a3"], index = ["r1", "r2", "r3"])
s_b = pd.Series(["b1", "b2", "b3"], index = ["r1", "r2", "r3"])

pd.DataFrame({"A-column":s_a, "B-column":s_b})
```



	A-column	B-column
r1	a1	b1
r2	a2	b2
r3	a3	b3

```
pd.DataFrame(s_a)
```

```
s_a.to_frame()
```



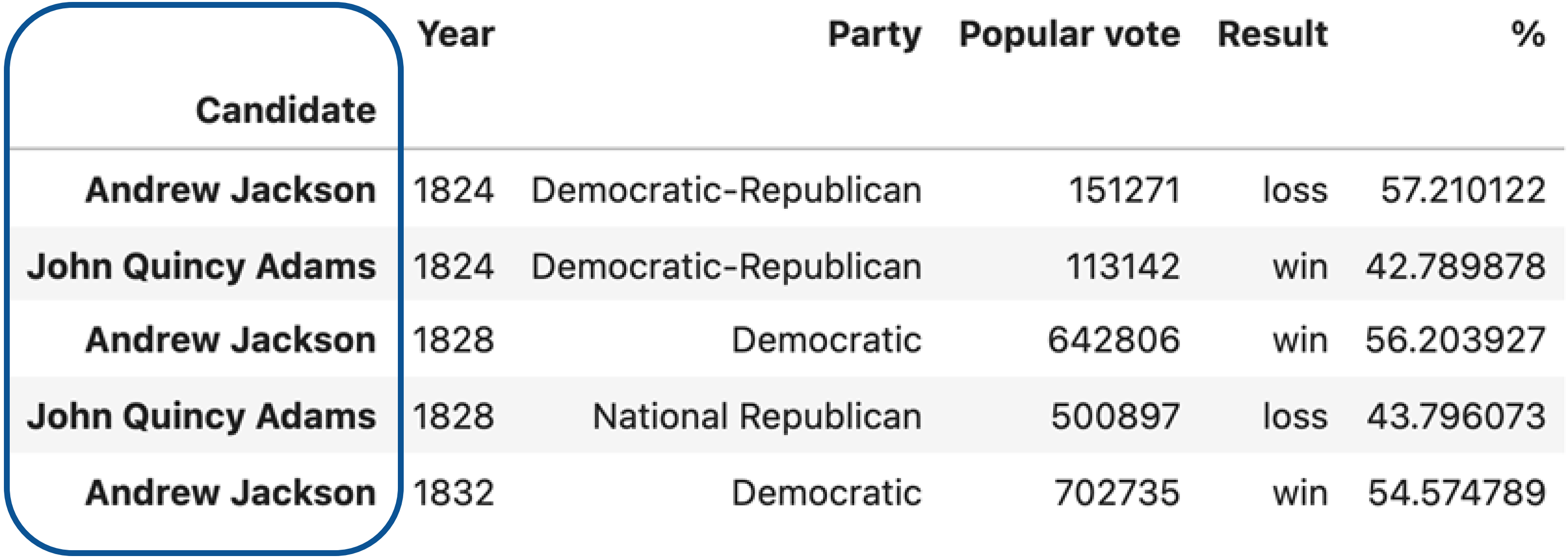
	0
r1	a1
r2	a2
r3	a3

Indices Are Not Necessarily Row Numbers

A **Row Index** (a.k.a. row labels) can also:

- Be non-numeric.
- Have a name, e.g. "Candidate".

```
# Creating a DataFrame from a CSV file and specifying the Index column
elections = pd.read_csv("data/elections.csv", index_col = "Candidate")
```

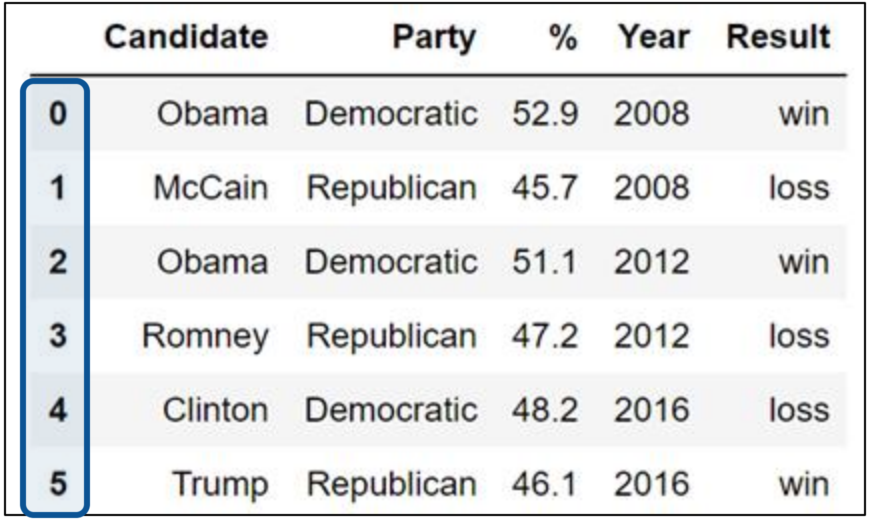


	Year	Party	Popular vote	Result	%
Candidate					
Andrew Jackson	1824	Democratic-Republican	151271	loss	57.210122
John Quincy Adams	1824	Democratic-Republican	113142	win	42.789878
Andrew Jackson	1828	Democratic	642806	win	56.203927
John Quincy Adams	1828	National Republican	500897	loss	43.796073
Andrew Jackson	1832	Democratic	702735	win	54.574789

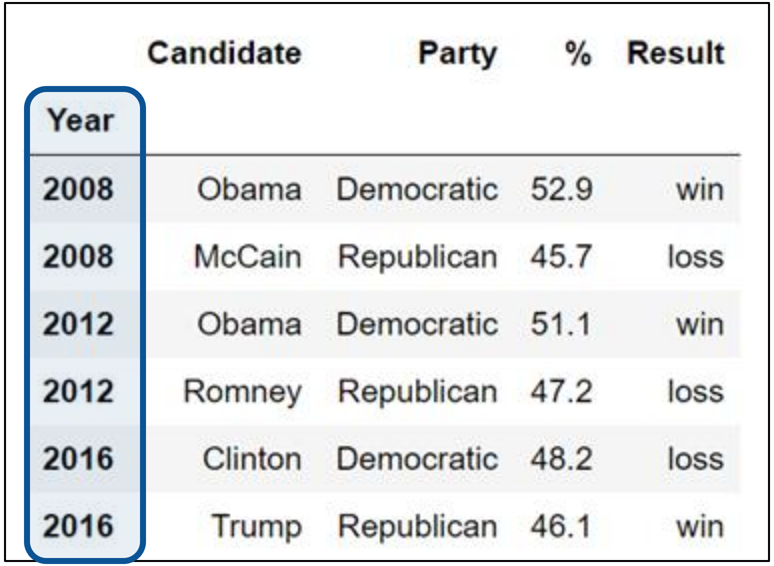
Indices Are Not Necessarily Unique

The row labels that constitute an index do not have to be unique.

- Left: The **index** values are all unique and numeric, acting as a row number.
- Right: The **index** values are named and non-unique.



	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win



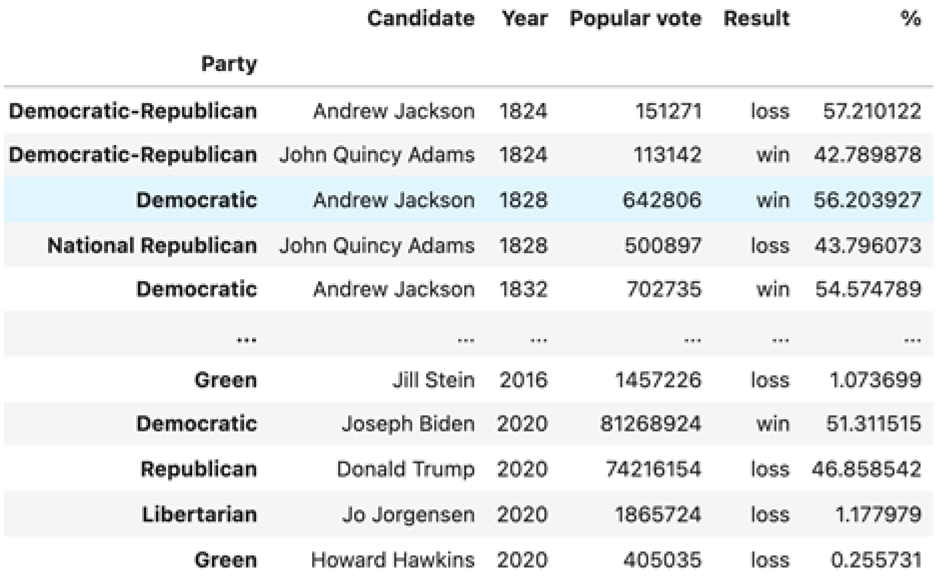
	Candidate	Party	%	Result
Year				
2008	Obama	Democratic	52.9	win
2008	McCain	Republican	45.7	loss
2012	Obama	Democratic	51.1	win
2012	Romney	Republican	47.2	loss
2016	Clinton	Democratic	48.2	loss
2016	Trump	Republican	46.1	win

Modifying Indices

- We can select a new column and set it as the index of the `DataFrame`.

Example: Setting the index to the "Party" column.

```
elections.set_index("Party")
```

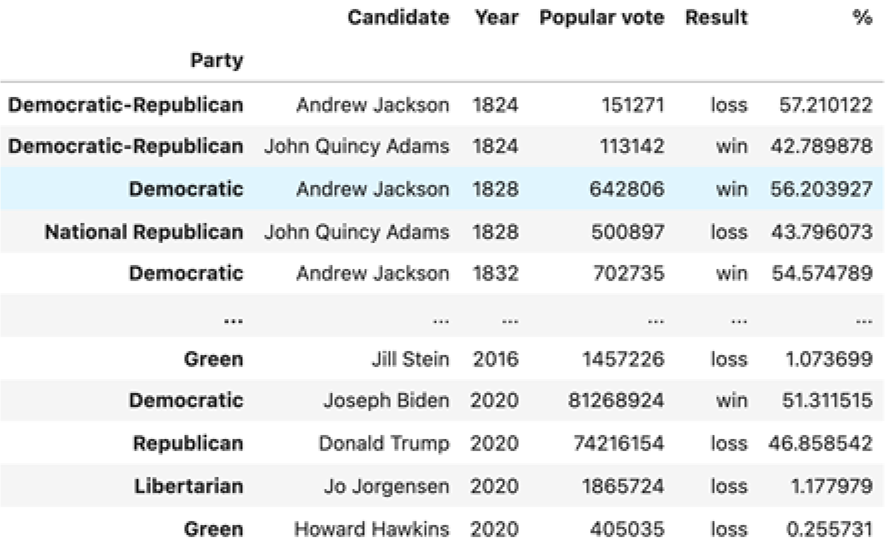


	Candidate	Year	Popular vote	Result	%
Party					
Democratic-Republican	Andrew Jackson	1824	151271	loss	57.210122
Democratic-Republican	John Quincy Adams	1824	113142	win	42.789878
Democratic	Andrew Jackson	1828	642806	win	56.203927
National Republican	John Quincy Adams	1828	500897	loss	43.796073
Democratic	Andrew Jackson	1832	702735	win	54.574789
...	...	...	...	...	...
Green	Jill Stein	2016	1457226	loss	1.073699
Democratic	Joseph Biden	2020	81268924	win	51.311515
Republican	Donald Trump	2020	74216154	loss	46.858542
Libertarian	Jo Jorgensen	2020	1865724	loss	1.177979
Green	Howard Hawkins	2020	405035	loss	0.255731

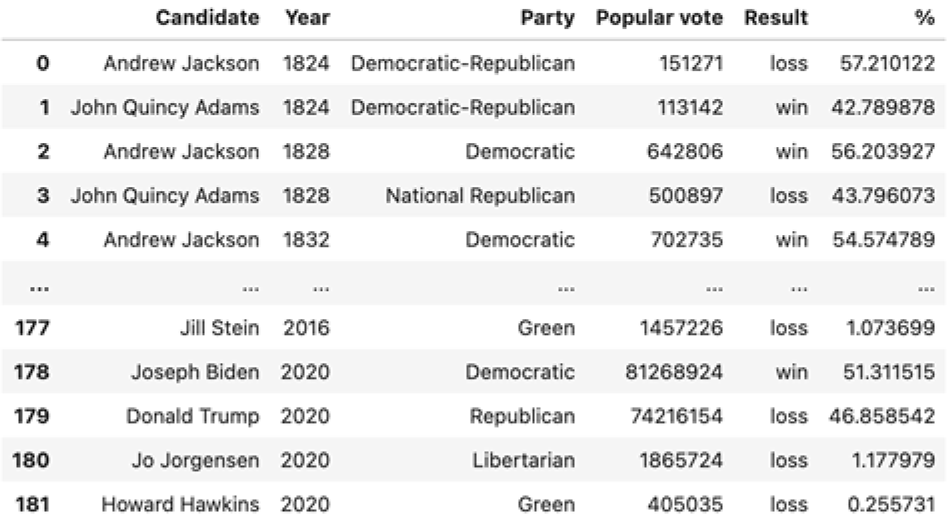
Resetting the `Index`

- We can change our mind and reset the **`Index`** back to the default list of integers.

```
elections.reset_index()
```



	Candidate	Year	Popular vote	Result	%
Party					
Democratic-Republican	Andrew Jackson	1824	151271	loss	57.210122
Democratic-Republican	John Quincy Adams	1824	113142	win	42.789878
Democratic	Andrew Jackson	1828	642806	win	56.203927
National Republican	John Quincy Adams	1828	500897	loss	43.796073
Democratic	Andrew Jackson	1832	702735	win	54.574789
...	...	...	...	...	...
Green	Jill Stein	2016	1457226	loss	1.073699
Democratic	Joseph Biden	2020	81268924	win	51.311515
Republican	Donald Trump	2020	74216154	loss	46.858542
Libertarian	Jo Jorgensen	2020	1865724	loss	1.177979
Green	Howard Hawkins	2020	405035	loss	0.255731

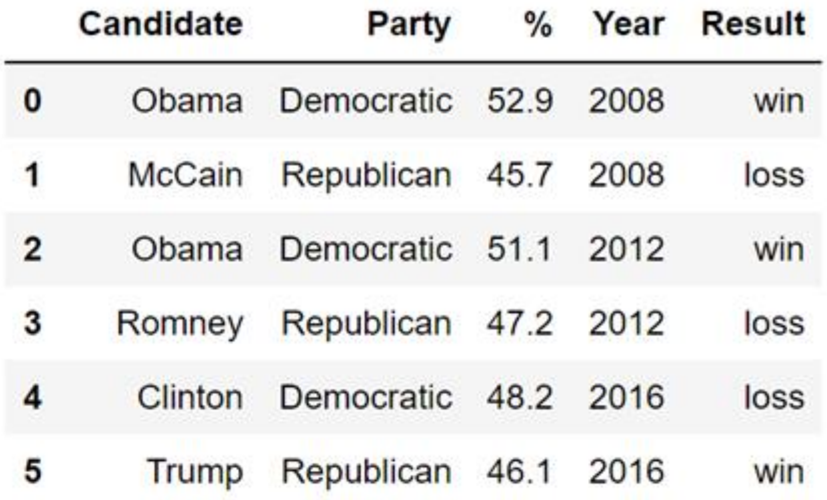


	Candidate	Year	Party	Popular vote	Result	%
0	Andrew Jackson	1824	Democratic-Republican	151271	loss	57.210122
1	John Quincy Adams	1824	Democratic-Republican	113142	win	42.789878
2	Andrew Jackson	1828	Democratic	642806	win	56.203927
3	John Quincy Adams	1828	National Republican	500897	loss	43.796073
4	Andrew Jackson	1832	Democratic	702735	win	54.574789
...	...	...	...	...	...	...
177	Jill Stein	2016	Green	1457226	loss	1.073699
178	Joseph Biden	2020	Democratic	81268924	win	51.311515
179	Donald Trump	2020	Republican	74216154	loss	46.858542
180	Jo Jorgensen	2020	Libertarian	1865724	loss	1.177979
181	Howard Hawkins	2020	Green	405035	loss	0.255731

Column Names Are Usually Unique!

Column names in **pandas** are almost always unique.

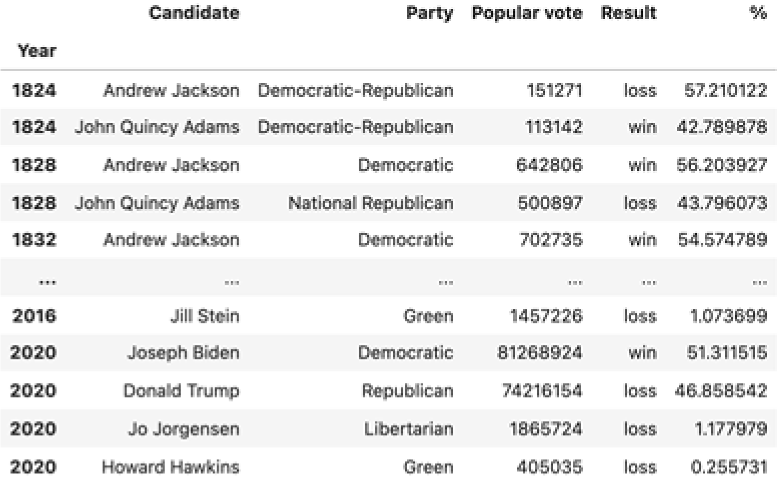
- Example: Really shouldn't have two columns named "Candidate".



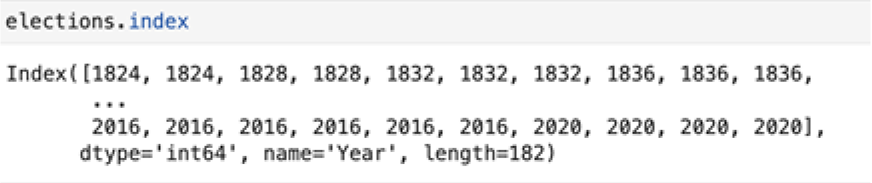
	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Accessing a DataFrame’s columns and row indices: .index and .columns

```
elections = pd.read_csv("data/elections.csv", index_col="Year")
```

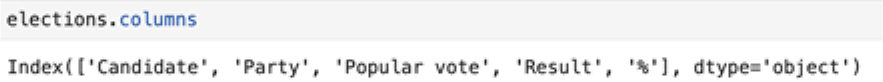


	Candidate	Party	Popular vote	Result	%
Year					
1824	Andrew Jackson	Democratic-Republican	151271	loss	57.210122
1824	John Quincy Adams	Democratic-Republican	113142	win	42.789878
1828	Andrew Jackson	Democratic	642806	win	56.203927
1828	John Quincy Adams	National Republican	500897	loss	43.796073
1832	Andrew Jackson	Democratic	702735	win	54.574789
...	...	...	...	...	...
2016	Jill Stein	Green	1457226	loss	1.073699
2020	Joseph Biden	Democratic	81268924	win	51.311515
2020	Donald Trump	Republican	74216154	loss	46.858542
2020	Jo Jorgensen	Libertarian	1865724	loss	1.177979
2020	Howard Hawkins	Green	405035	loss	0.255731



```
elections.index
```

```
Index([1824, 1824, 1828, 1828, 1832, 1832, 1832, 1836, 1836, 1836,
       ...
       2016, 2016, 2016, 2016, 2016, 2016, 2020, 2020, 2020, 2020],
      dtype='int64', name='Year', length=182)
```



```
elections.columns
```

```
Index(['Candidate', 'Party', 'Popular vote', 'Result', '%'], dtype='object')
```

The DataFrame **`elections`** with "Year" as **`Index`**