ECON 220 Lab
(Week 2)

Introduction to Python (Part I)

Justin S. Eloriaga

Outline

- Data
- Pandas, Numpy, and Matplotlib
- Loading Data
- Some Simple Analysis

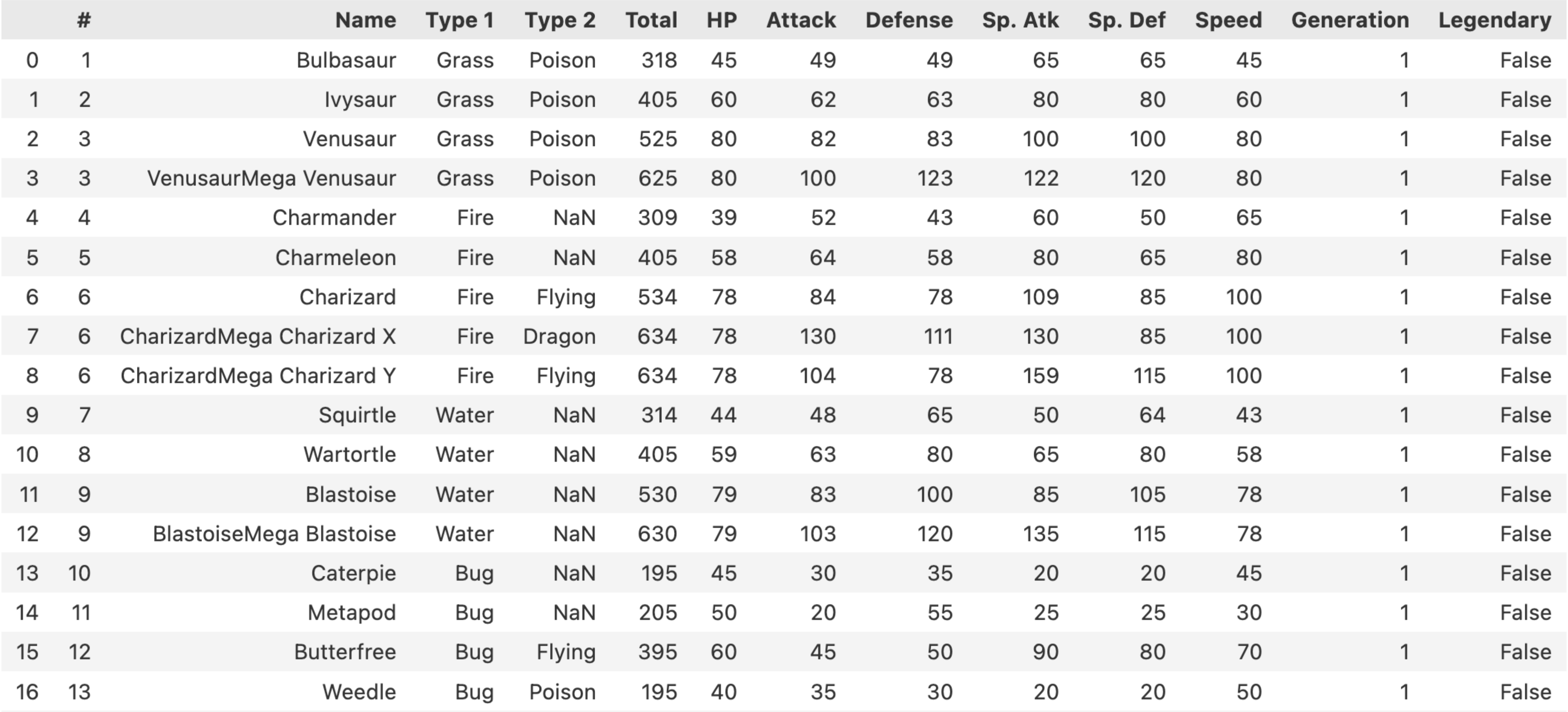
Pokemon

- I started playing **Pokemon** in 2004 (Yes, I am aware, that is older than some of you)
- I even competed in the South East Asia Pokemon Championships.
- I assume most people here have watched the series and probably a greater number have played the games
- Total Hours Justin Spent on Pokemon: ***1823 hours (and counting)***

Oak: Here, take one of these rare Pokémon.

Pokemon images obtained from Bulbapedia

Today's Dataset = The (National complete) Pokedex

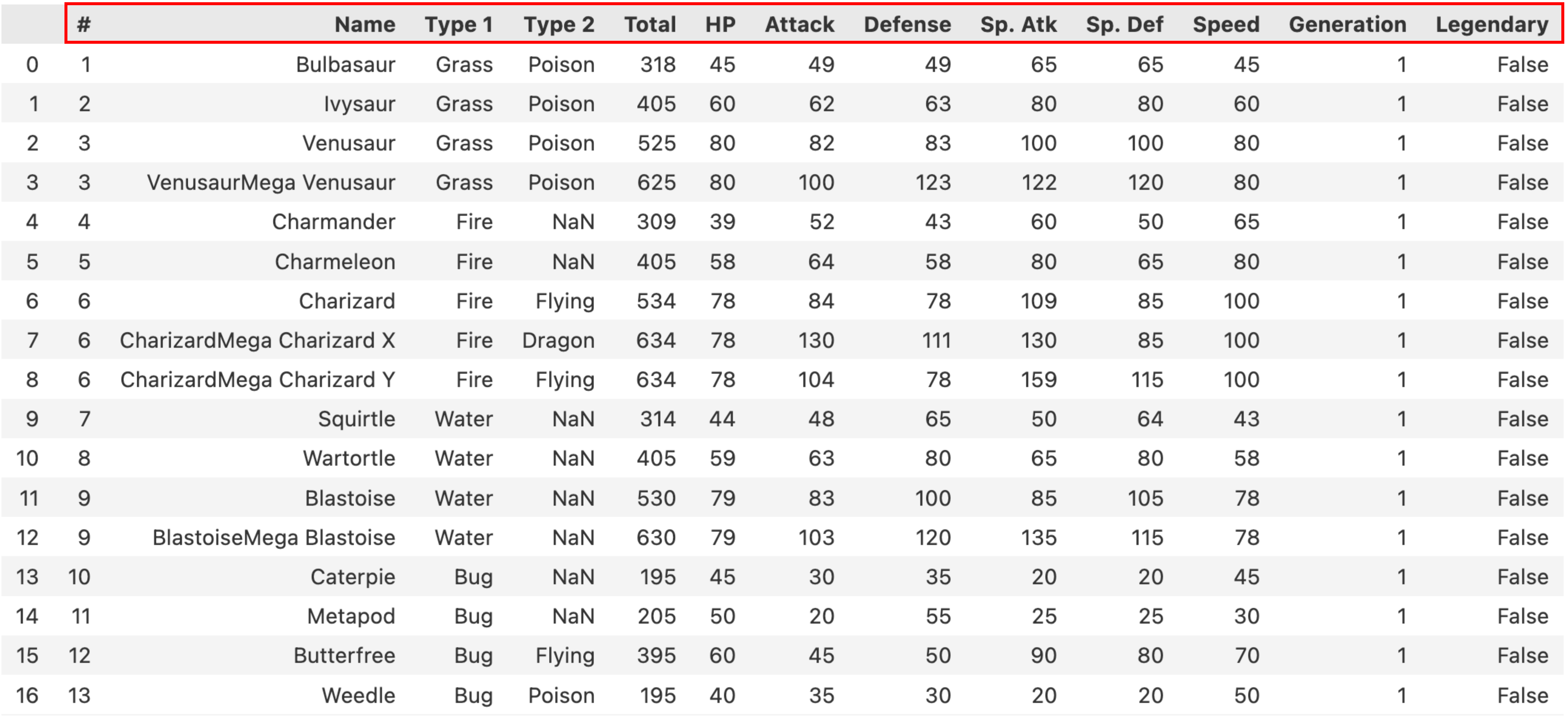


	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False
9	7	Squirtle	Water	NaN	314	44	48	65	50	64	43	1	False
10	8	Wartortle	Water	NaN	405	59	63	80	65	80	58	1	False
11	9	Blastoise	Water	NaN	530	79	83	100	85	105	78	1	False
12	9	BlastoiseMega Blastoise	Water	NaN	630	79	103	120	135	115	78	1	False
13	10	Caterpie	Bug	NaN	195	45	30	35	20	20	45	1	False
14	11	Metapod	Bug	NaN	205	50	20	55	25	25	30	1	False
15	12	Butterfree	Bug	Flying	395	60	45	50	90	80	70	1	False
16	13	Weedle	Bug	Poison	195	40	35	30	20	20	50	1	False

Pokemon images obtained from Bulbapedia

Today's Dataset

13 Columns = 13 Variables



	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False
9	7	Squirtle	Water	NaN	314	44	48	65	50	64	43	1	False
10	8	Wartortle	Water	NaN	405	59	63	80	65	80	58	1	False
11	9	Blastoise	Water	NaN	530	79	83	100	85	105	78	1	False
12	9	BlastoiseMega Blastoise	Water	NaN	630	79	103	120	135	115	78	1	False
13	10	Caterpie	Bug	NaN	195	45	30	35	20	20	45	1	False
14	11	Metapod	Bug	NaN	205	50	20	55	25	25	30	1	False
15	12	Butterfree	Bug	Flying	395	60	45	50	90	80	70	1	False
16	13	Weedle	Bug	Poison	195	40	35	30	20	20	50	1	False

Pokemon images obtained from Bulbapedia

Scales of Measurement

- **Nominal** – just a label
- **Ordinal** – a label with a scale
- **Interval** – a number with fixed increments
- **Ratio** – a number with variable increments

- What type of scale doe the following variables use?
  - Type 1?
  - Attack?
  - Defense?
  - Legendary?

Scales of Measurement

- **Nominal** – just a label
- **Ordinal** – a label with a scale
- **Interval** – a number with fixed increments
- **Ratio** – a number with variable increments

- What type of scale doe the following variables use?
  - Type 1? NOMINAL
  - Attack? INTERVAL
  - Defense? INTERVAL
  - Legendary? NOMINAL

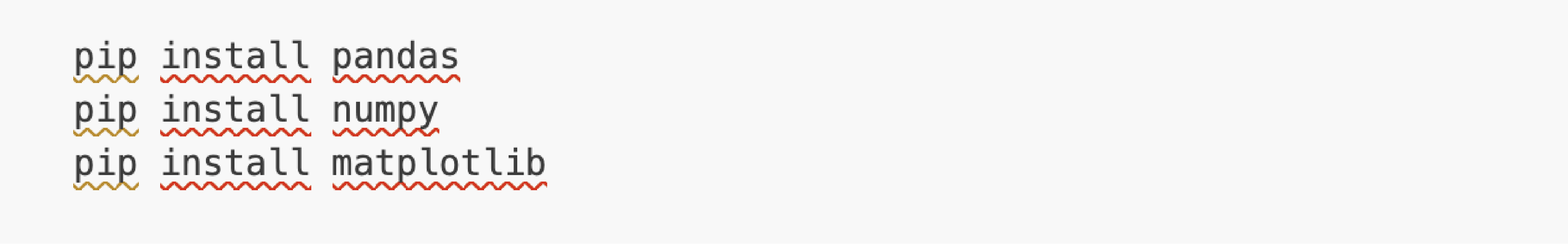
Note: The increment for base stats in Pokemon is always 1. There will be no decimals (so it cannot be a ratio)

Libraries

- *Pandas*
  - Data Management (Loading, Cleaning, Exporting, etc.)
- *Matplotlib*
  - Data Visualization (Plots, Charts, etc.)
- *Numpy*
  - Computation (Average, Sum, Median, etc.)

Installing Libraries

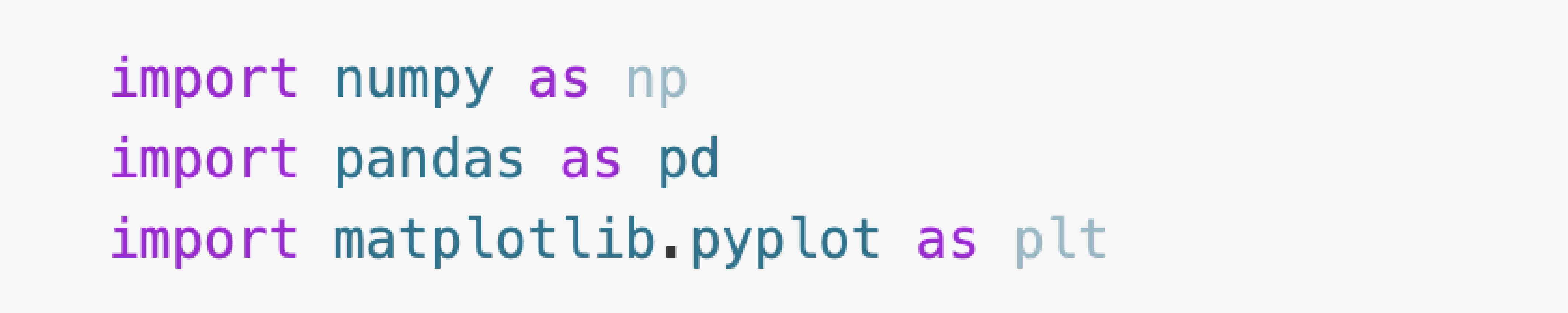
- We need to first install the libraries. This can be done using the pip command



```
pip install pandas
pip install numpy
pip install matplotlib
```

- Notes:
  - Download **Jupyter Notebook** in the VSCode extensions.
  - VSCode might require you to install Ipykernel. Please do so!
  - Make sure to select the most recent version of Python as the kernel of installation.

Step 1: Load the Libraries



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

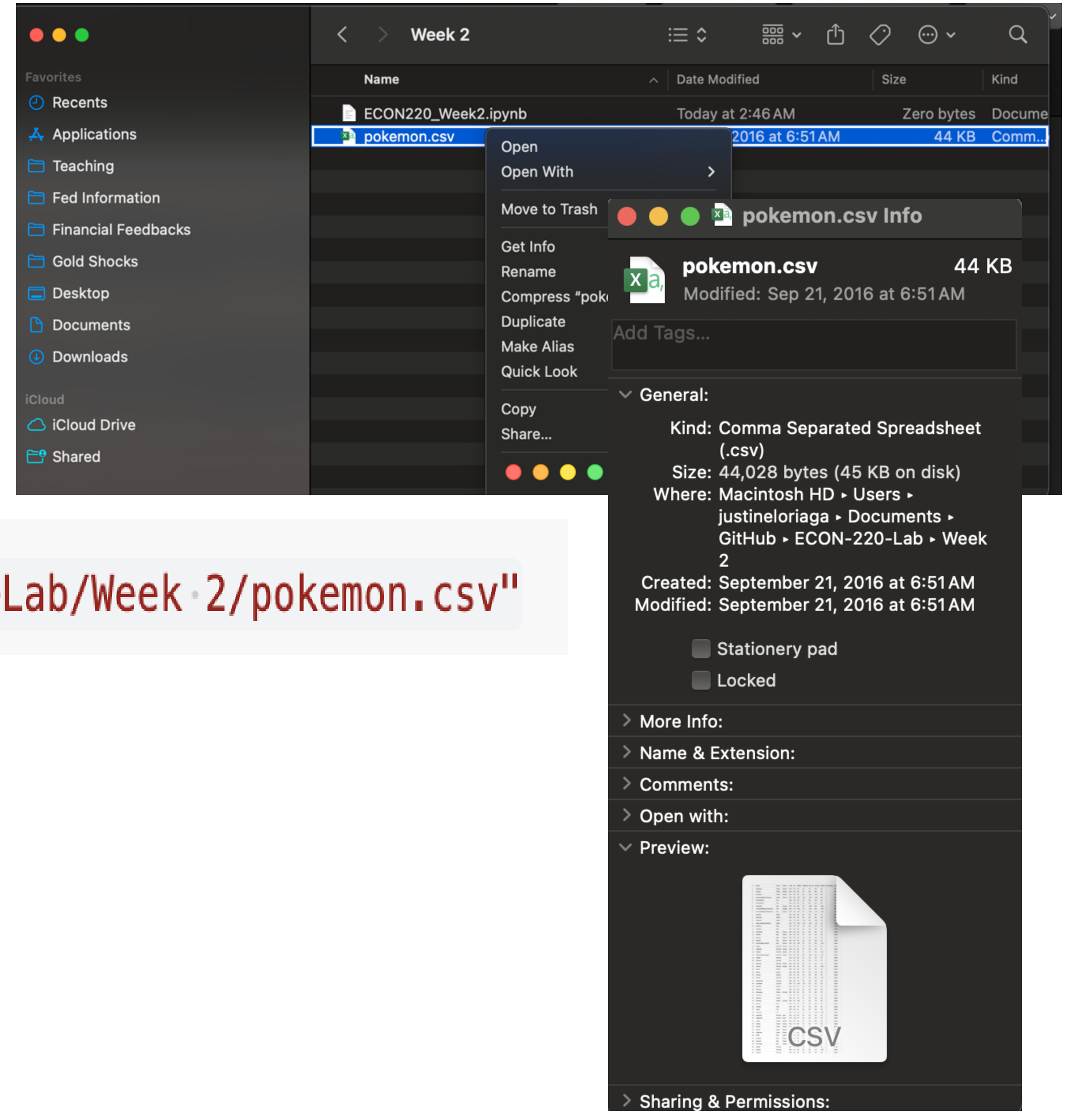
- The function **import** tells Python to load the library
- The option **as** allows us to define short-hand notation for calling functions inside of libraries.
- *Numpy* = np; *Pandas* = pd; *Matplotlib* = plt. Very standard!

Step 2: Loading a Dataset

- It is common to load datasets into Python through the use of a file path. To do so, you can generate a variable that contains the path.

```
path = "/Users/justineloriaga/Documents/GitHub/ECON-220-Lab/Week 2/pokemon.csv"
```

- Finding the file path is easy! Use the Get Info on Mac!



Name	Date Modified	Size	Kind
ECON220_Week2.ipynb	Today at 2:46 AM	Zero bytes	Docume
pokemon.csv	2016 at 6:51 AM	44 KB	Comm...

pokemon.csv Info

pokemon.csv 44 KB
Modified: Sep 21, 2016 at 6:51 AM

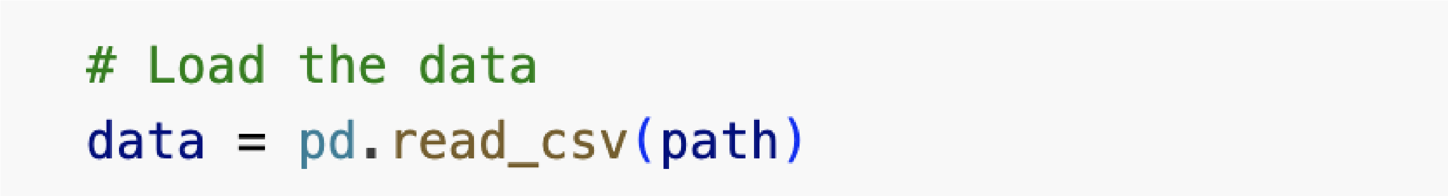
General:

Kind: Comma Separated Spreadsheet (.csv)
Size: 44,028 bytes (45 KB on disk)
Where: Macintosh HD ▸ Users ▸ justineloriaga ▸ Documents ▸ GitHub ▸ ECON-220-Lab ▸ Week 2
Created: September 21, 2016 at 6:51 AM
Modified: September 21, 2016 at 6:51 AM

□ Stationery pad
□ Locked

Step 2: Loading a Dataset

- Then, use the read_csv() function of the pandas (pd) library to load the dataset (which we will call data)

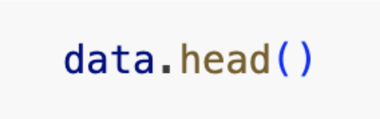


```
# Load the data
data = pd.read_csv(path)
```

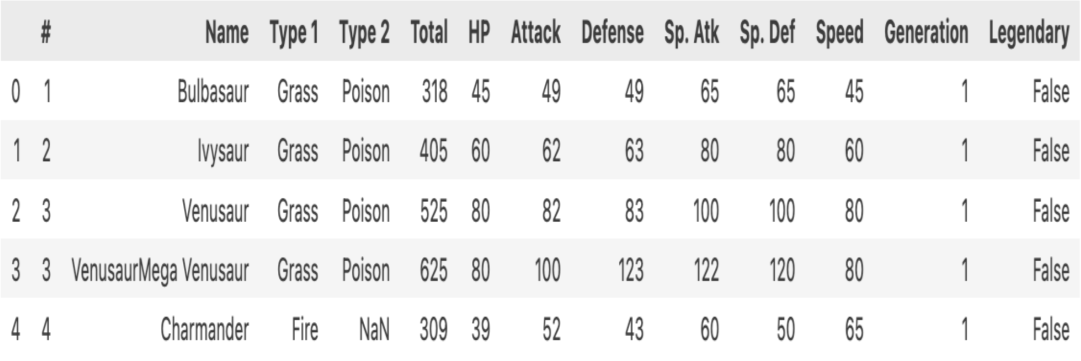
- Comments are ***important***
  - Use the # key to place comments on your code! Super important!
  - It is good to guide whoever reads your code on what you are doing!
  - It also makes me aware that you know what you are coding.

Using head.() and tail.()

head.() shows the first few rows of the dataset

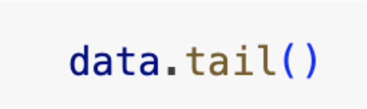


```python
data.head()
```

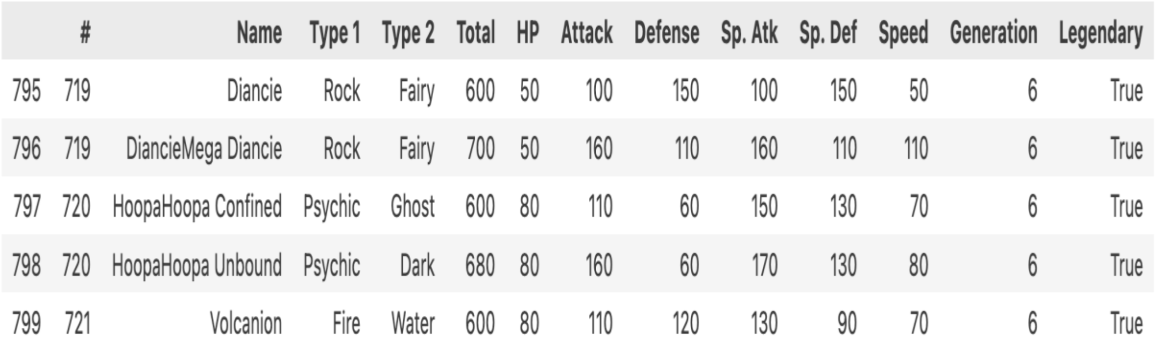


	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

tail.() shows the last few rows of the dataset



```python
data.tail()
```

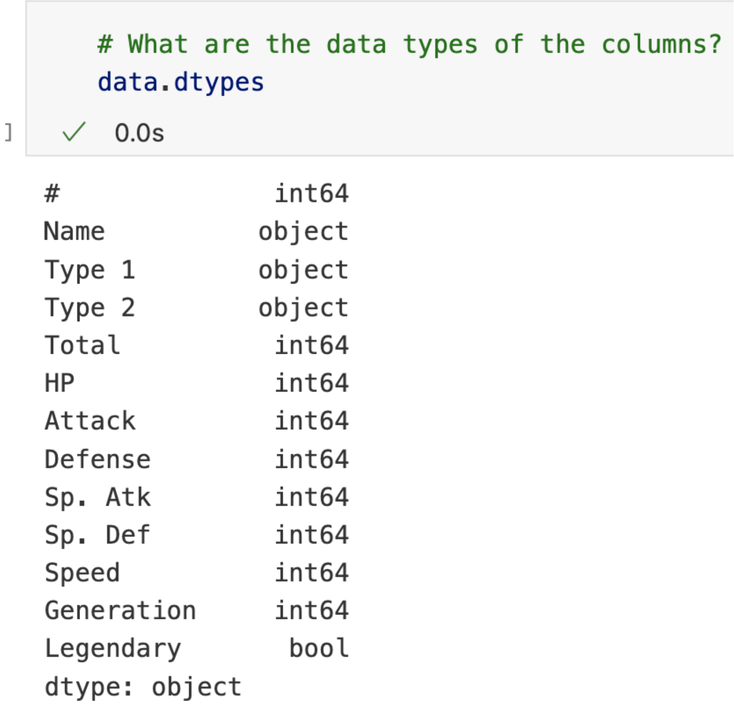


	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

If you put a number inside the parenthesis, it will display a specific number of rows. By default, it is 5. data.head(10) for example displays the first 10 rows.

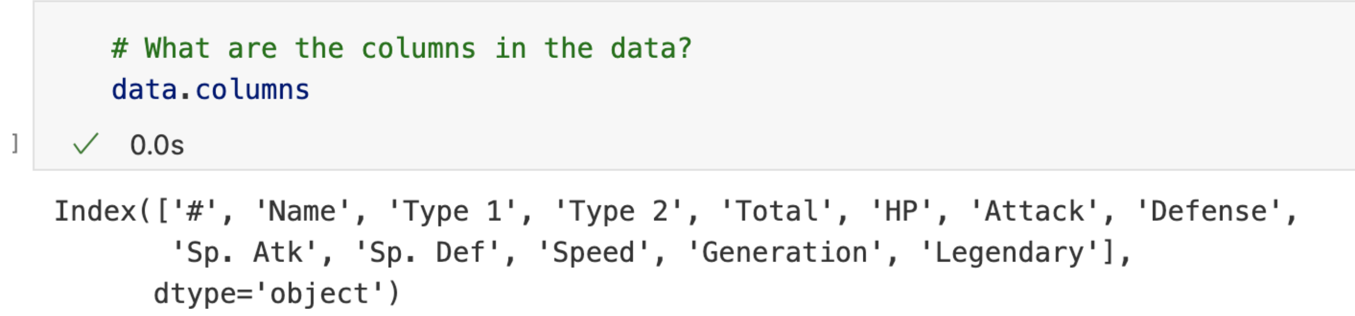
Some Basic Things about the Dataset

- It is good to have a first glance at the dataset using the following functions
  - **.columns** – tells you the column names (in case you want to manipulate some variables later on)
  - **.dtypes** – tells you the characteristic of each column (i.e. is it ordinal, an integer, a ratio, a character, etc)
  - **.shape** – tells you the number of rows and columns



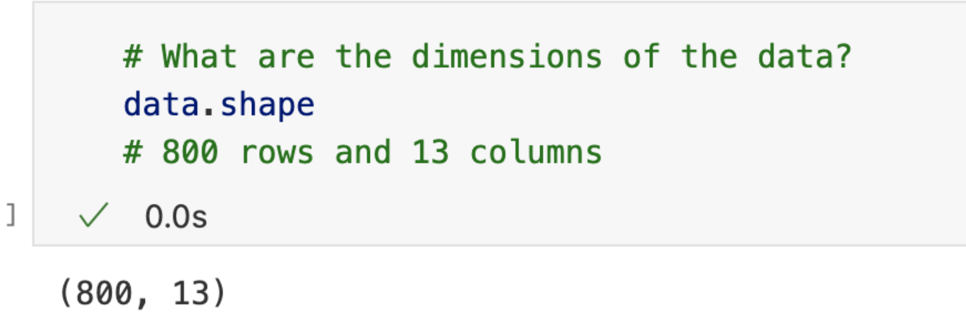
```
# What are the data types of the columns?
data.dtypes
```

```
#               int64
Name           object
Type 1         object
Type 2         object
Total           int64
HP              int64
Attack          int64
Defense         int64
Sp. Atk         int64
Sp. Def         int64
Speed           int64
Generation      int64
Legendary        bool
dtype: object
```



```
# What are the columns in the data?
data.columns
```

```
Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defense',
       'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

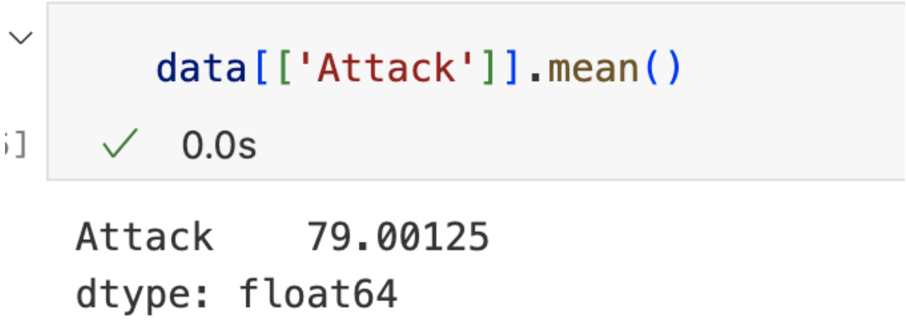


```
# What are the dimensions of the data?
data.shape
# 800 rows and 13 columns
```

```
(800, 13)
```

Selecting Columns and Means

- Suppose you want to find the mean of the variable ‘Attack’



```
data[['Attack']].mean()
```

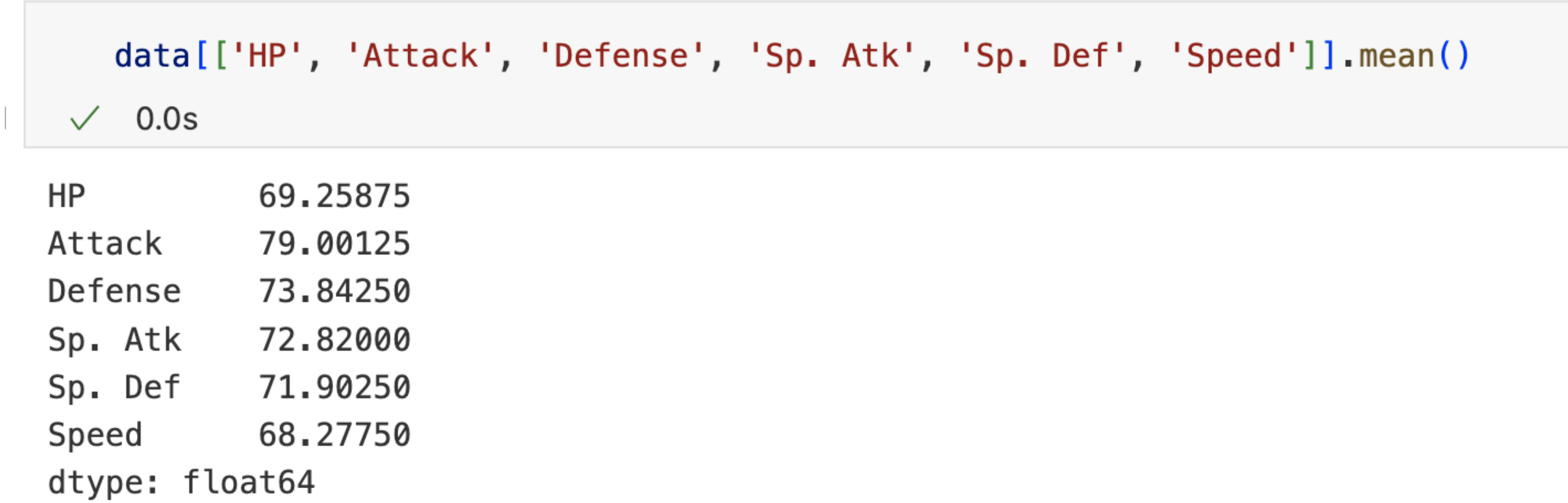
✓ 0.0s

```
Attack    79.00125
dtype: float64
```

- The command data[[]] tells Python to look inside the object data for a column named ‘Attack’ -> Focuses on a single column
- Then, the function/attribute .mean() from numpy is meant to calculate the mean.

Selecting Columns and Means

- It is straightforward to apply this to more than one column/variable



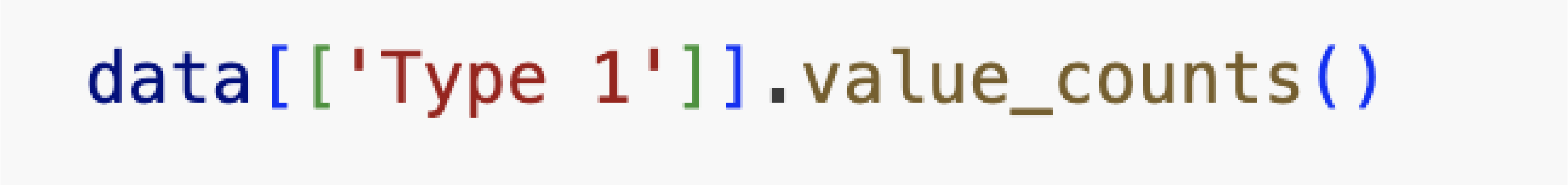
```
data[['HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed']].mean()
```

✓ 0.0s

```
HP          69.25875
Attack      79.00125
Defense     73.84250
Sp. Atk     72.82000
Sp. Def     71.90250
Speed       68.27750
dtype: float64
```

Making a Bar Chart

- To make a bar chart, we first need data to plot
  - Look at the variable Type 1. A Pokemon can have at most two types. Type 1 is their main type and Type 2 is the secondary type.
  - For example, consider my favorite Pokemon
    - *Dragonite* is Type 1 (Dragon) and Type 2 (Flying)



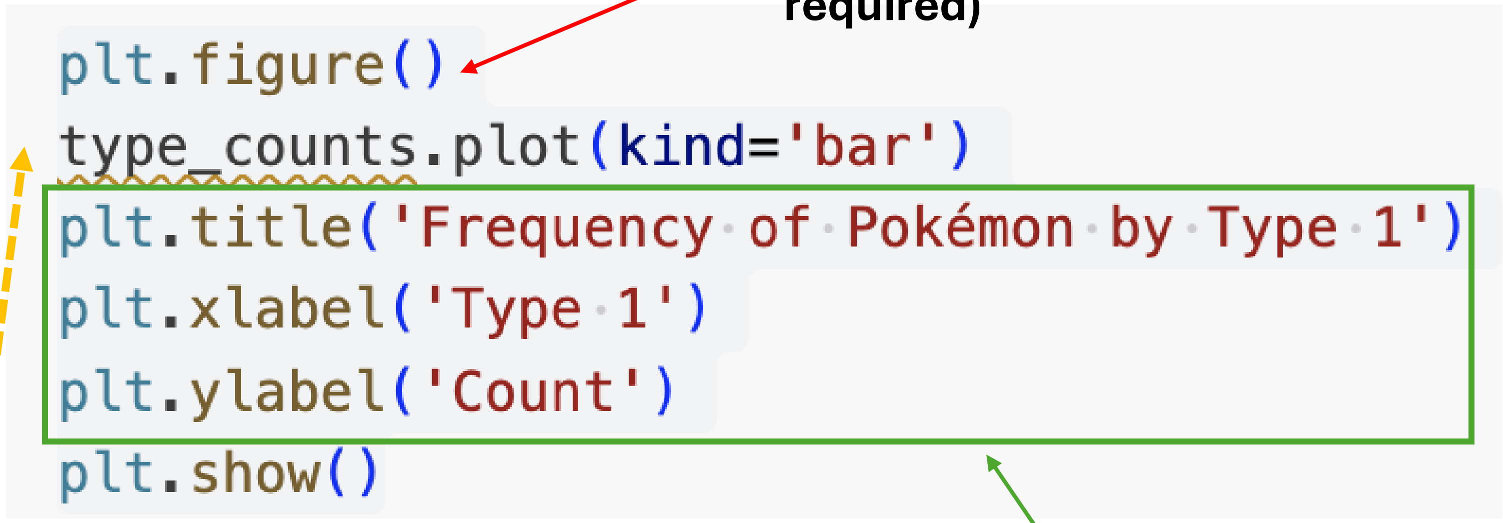
```
data[['Type 1']].value_counts()
```

- Okay, how do we do this?
  - From data, use the [[]] to select the Type 1 column. Then, we apply the option **value_counts()** to count the frequency of each type

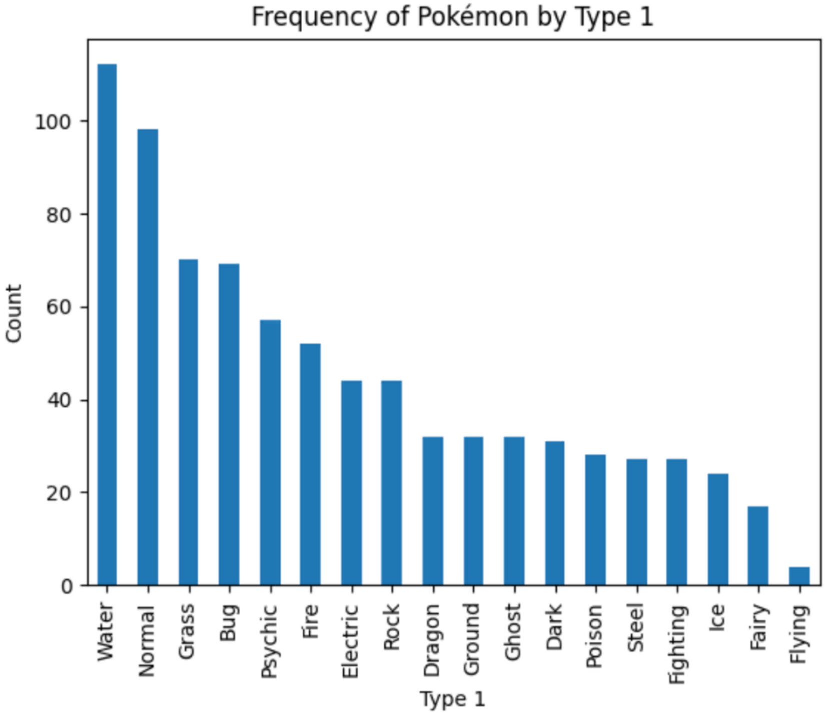
Pokemon images obtained from Bulbapedia

Plotting

Initial Declaration, (Semi-required)



```
plt.figure()
type_counts.plot(kind='bar')
plt.title('Frequency of Pokémon by Type 1')
plt.xlabel('Type 1')
plt.ylabel('Count')
plt.show()
```

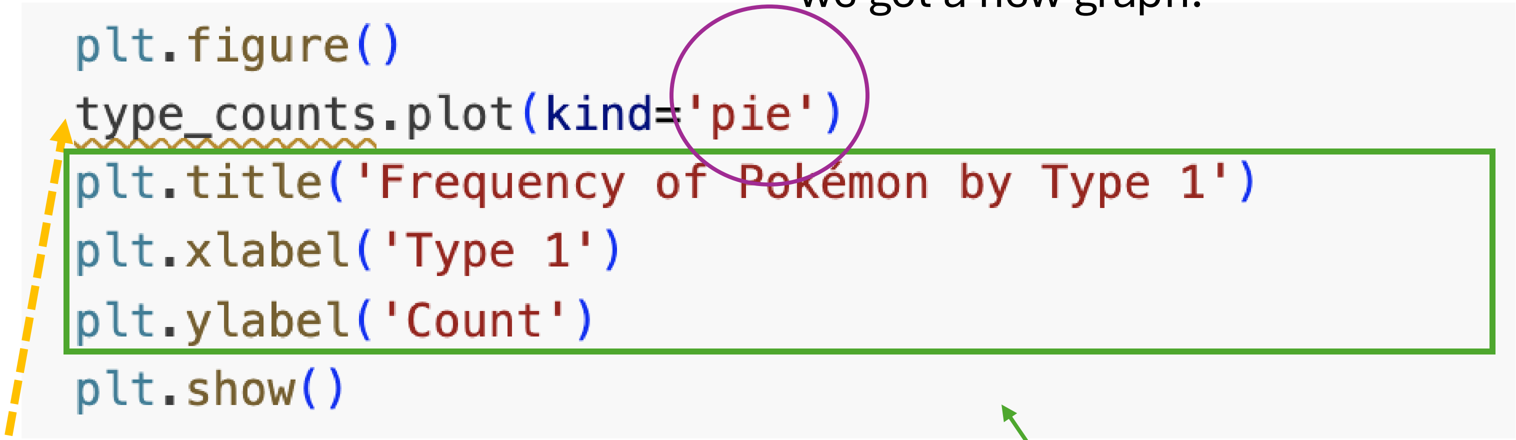


Required. Using the type_counts object, we use the function plot(). Inside plot, we use the option 'kind' to tell Python what kind of plot, in this case, a bar.

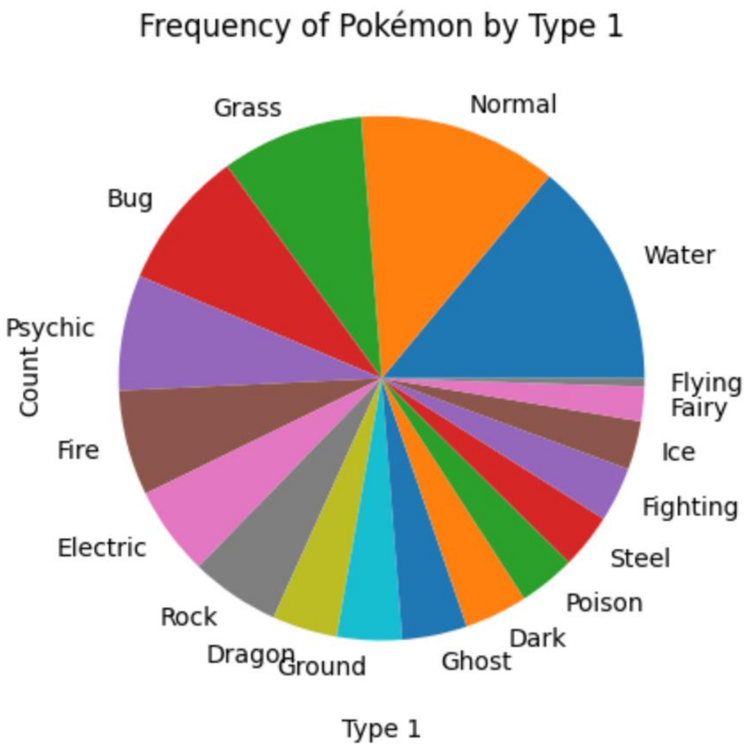
Optional. Using options in Matplotlib (i.e. plt) to specify the title and axes labels.

Easy to Change in Python!

This was all we changed, and we got a new graph!



```
plt.figure()
type_counts.plot(kind='pie')
plt.title('Frequency of Pokémon by Type 1')
plt.xlabel('Type 1')
plt.ylabel('Count')
plt.show()
```

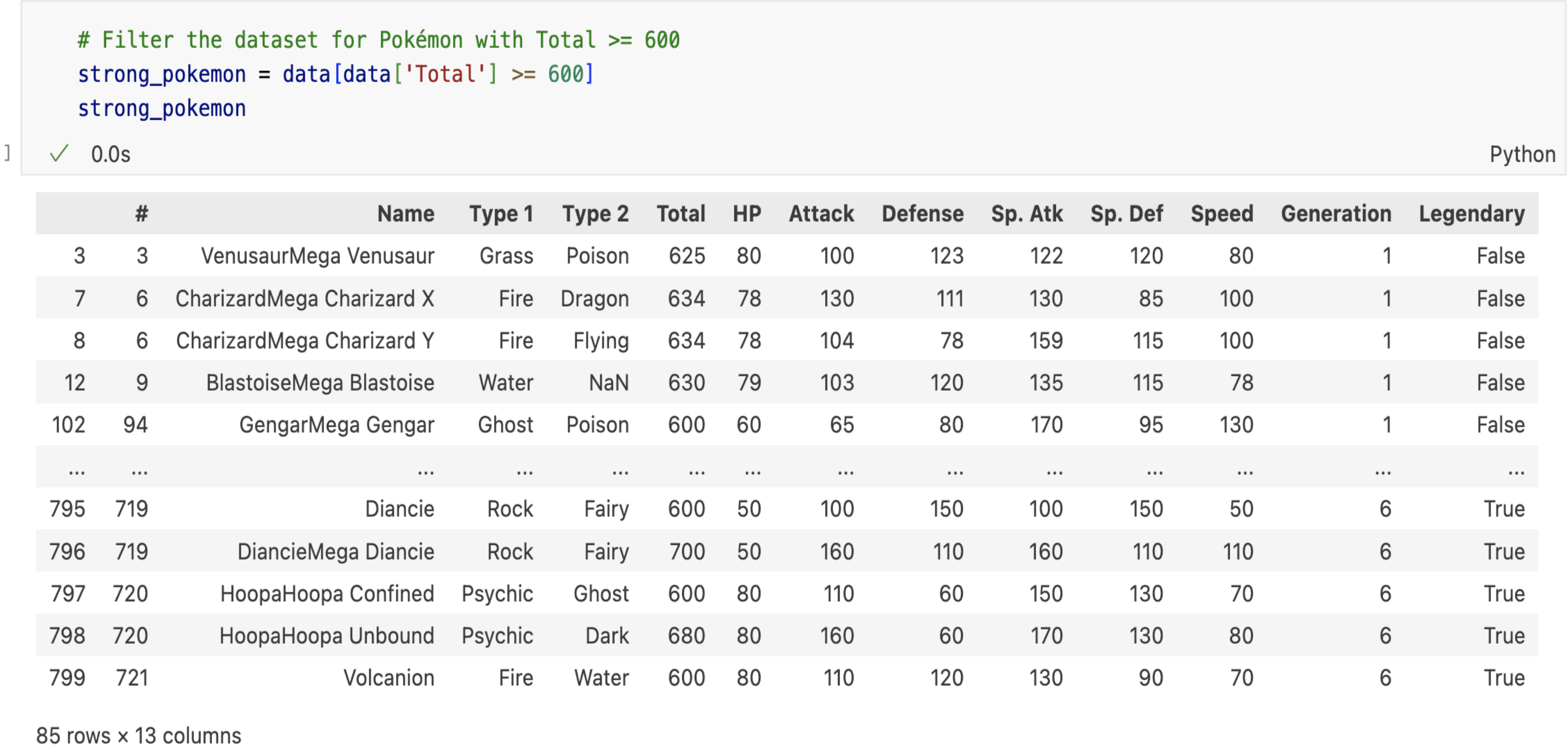


Required. Using the type_counts object, we use the function plot(). Inside plot, we use the option 'kind' to tell Python what kind of plot, in this case, a pie!

Optional. Using options in Matplotlib (i.e. plt) to specify the title and axes labels.

Some Basic Data Management

- One important skill is to make a sub-dataset from a bigger dataset.
- Suppose we wanted to look at just the strong Pokemon
    - Strong = Total Stat of above 600



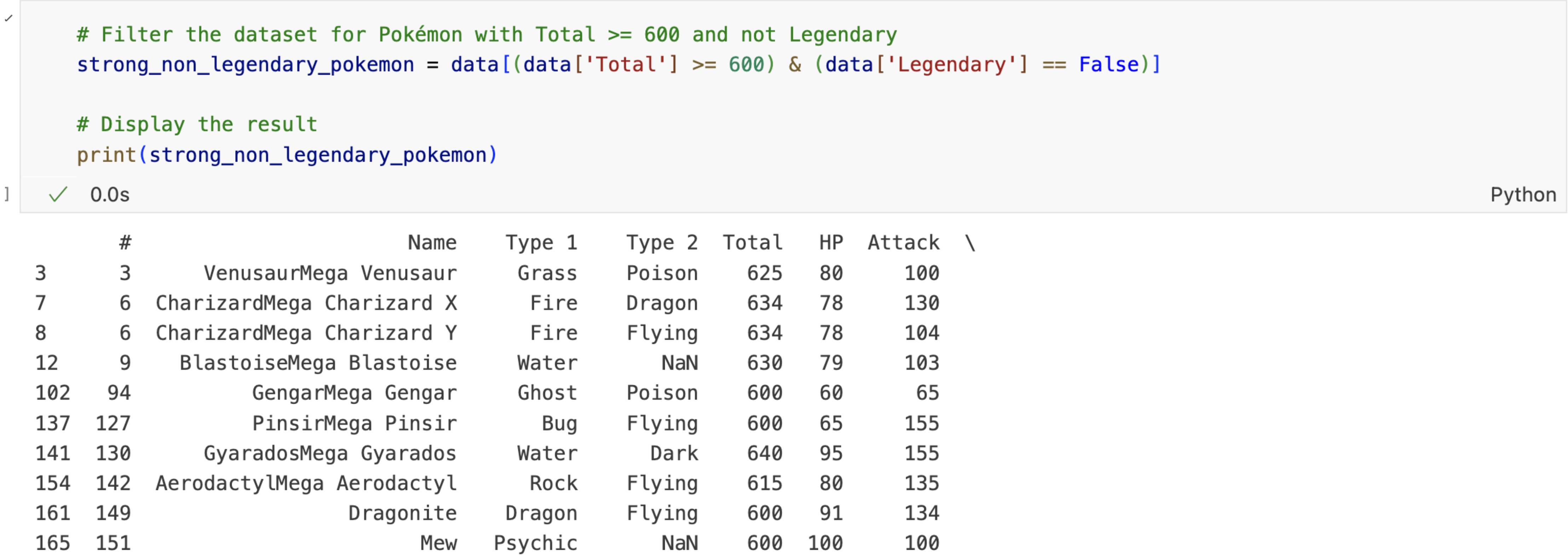
```python
# Filter the dataset for Pokémon with Total >= 600
strong_pokemon = data[data['Total'] >= 600]
strong_pokemon
```

✓ 0.0s Python

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False
12	9	BlastoiseMega Blastoise	Water	NaN	630	79	103	120	135	115	78	1	False
102	94	GengarMega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

85 rows × 13 columns

Two Conditions for Subsetting



```python
# Filter the dataset for Pokémon with Total >= 600 and not Legendary
strong_non_legendary_pokemon = data[(data['Total'] >= 600) & (data['Legendary'] == False)]

# Display the result
print(strong_non_legendary_pokemon)
```

✓ 0.0s Python

```
       #                       Name    Type 1    Type 2  Total   HP  Attack  \
3      3      VenusaurMega Venusaur     Grass    Poison    625   80     100
7      6  CharizardMega Charizard X      Fire    Dragon    634   78     130
8      6  CharizardMega Charizard Y      Fire    Flying    634   78     104
12     9    BlastoiseMega Blastoise     Water       NaN    630   79     103
102   94          GengarMega Gengar     Ghost    Poison    600   60      65
137  127          PinsirMega Pinsir       Bug    Flying    600   65     155
141  130      GyaradosMega Gyarados     Water      Dark    640   95     155
154  142  AerodactylMega Aerodactyl      Rock    Flying    615   80     135
161  149                  Dragonite    Dragon    Flying    600   91     134
165  151                        Mew   Psychic       NaN    600  100     100
```