Take a seat!
We'll start at
7:05pm <3

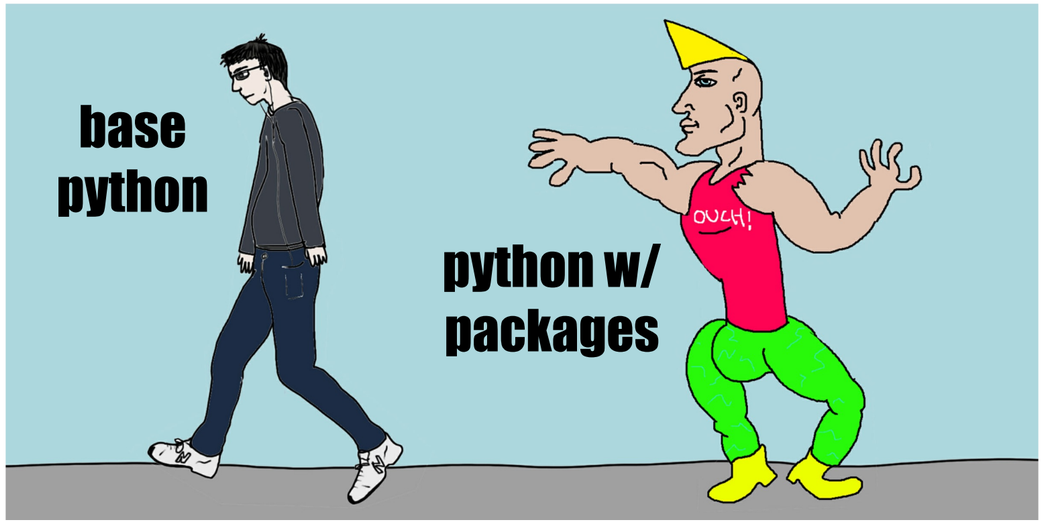
# DATA FOR DUMMIES®

## Data Analysis w/ Numpy & Pandas

# What are Packages?

**Base Python**: lets us use the basic functionalities from last workshop: creating variables, basic math, importing data

**Packages** are things we can install into Python to do new and cool things, similar to how you might add mods to a video game




# What are Packages?

There are two fundamental packages in Python that are used for Data Science/Analytics

**Pandas**: allows us to load data, examine it, and modify/clean it to fit our needs

**NumPy** (Numerical Python): allows us to store and work with numbers efficiently


# What are Packages?



```
np.array(my_list)
```

```
my_array1 + my_array2
```

```
my_array.sum()
```

```
data = pd.read_csv()
data.head()
```

```
data["column_name"].mean()
```

```
data.info()
```

# Installing Packages

We can install packages into our Python environment using the following conventions

```
import pandas as pd
```

Actual package name

Our nickname for it

```
import numpy as np
```

# NumPy Data Types

Quick Review: **Lists**

A collection of items (can contain any data type)

```
my_list = [1, "Hello", 2.125]
```

**NumPy Array**

A collection of items with the same data type, most commonly numbers

```
np.array([1, 2, 3, 4, 5])
```

We use Arrays instead of Lists for collections of numbers because it is faster, more memory efficient and easier to work with

# NumPy Data Types

A NumPy Array can be created by first making a List, then applying np.array()

```
np.array([1, 2, 3, 4, 5])
```

```
array([1, 2, 3, 4, 5])
```

```
my_list = [1, 2, 3, 4, 5]
np.array(my_list)
```

```
array([1, 2, 3, 4, 5])
```

# NumPy Data Types

An array made up of just one list is **one dimensional**

```
np.array([1, 2, 3, 4, 5])
```

```
array([1, 2, 3, 4, 5])
```

This example has 1 row and 5 columns, so it is a 1x5 array

We can make **multi-dimensional** arrays using more lists

```
np.array([[1,2,3],
          [4,5,6],
          [7,8,9]])
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

This example has 3 rows and 3 columns, so it is a 3x3 array

# NumPy Data Types

We can also index arrays the same way as lists!
Remember that we start counting at 0!

```
np.array([1, 2, 3, 4, 5])
```

0 1 2 3 4

```
my_array[0]
```

```
1
```

```
my_array[2]
```

```
3
```

We can retrieve multiple elements as well

```
my_array[0:3]
```

```
array([1, 2, 3])
```

Note: the last index is exclusive

# Pandas Data Types

Quick Review: **Dictionaries**

Collections of items organized to labels (called “keys”)

```
my_dict = {
    'colors': ['red', 'blue', 'green'],
    'numbers': [1, 2, 3],
    'favorite food': 'pizza'
}
```

## Pandas DataFrame

Very similar to a dictionary, but allows us to look at data as a table (each key/column must have the same number of values)

```
pd.DataFrame({
    'colors': ['red', 'blue', 'green'],
    'numbers': [1,2,3],
    'favorite food': ['pizza', 'spaghetti', 'apples']
})
```

# Pandas Data Types

Data frames allow us to make **tidy data**, where each row is an **observation** and each column is a **feature/variable**

```
class_info = pd.DataFrame({
    "name": ["Cyrus N.", "Owen F.", "Evan J."],
    "GPA": [4.0, 3.5, 3.6],
    "grade": ["Senior", "Senior", "Junior"]
})
```

| | name | GPA | grade |
|---|---|---|---|
| 0 | Cyrus N. | 4.0 | Senior |
| 1 | Owen F. | 3.5 | Senior |
| 2 | Evan J. | 3.6 | Junior |

```
class_dict = {
    "name": ["Cyrus N.", "Owen F.", "Evan J."],
    "GPA": [4.0, 3.5, 3.6],
    "grade": ["Senior", "Senior", "Junior"]
}

pd.DataFrame(class_dict)
```

| | name | GPA | grade |
|---|---|---|---|
| 0 | Cyrus N. | 4.0 | Senior |
| 1 | Owen F. | 3.5 | Senior |
| 2 | Evan J. | 3.6 | Junior |

Data frames are made by surrounding a dictionary with pd.DataFrame()

# Pandas Data Types

Data frames allow us to index in two ways based on our needs

| | name | GPA | grade |
|---|---|---|---|
| 0 | Cyrus N. | 4.0 | Senior |
| 1 | Owen F. | 3.5 | Senior |
| 2 | Evan J. | 3.6 | Junior |

Retrieving a **column**:

```
class_info["GPA"]
```

| | GPA |
|---|---|
| 0 | 4.0 |
| 1 | 3.5 |
| 2 | 3.6 |

# Pandas Data Types

Data frames allow us to index in two ways based on our needs

| | name | GPA | grade |
|---|---|---|---|
| 0 | Cyrus N. | 4.0 | Senior |
| 1 | Owen F. | 3.5 | Senior |
| 2 | Evan J. | 3.6 | Junior |

Retrieving a **row**:

```
class_info.iloc[0]
```

| | 0 |
|---|---|
| name | Cyrus N. |
| GPA | 4.0 |
| grade | Senior |

# Pandas Data Types

## Pandas Series

A Data Frame with only one column, typically seen when we index a column

| | 0 |
|---|---|
| name | Cyrus N. |
| GPA | 4.0 |
| grade | Senior |

You may encounter errors or unexpected outcomes if you confuse these for Data Frames

Recall: you can use type() to check!

# Your Turn!

**Dataset:** http://bit.ly/47kMlv0

**Worksheet:** http://bit.ly/4qljfVd