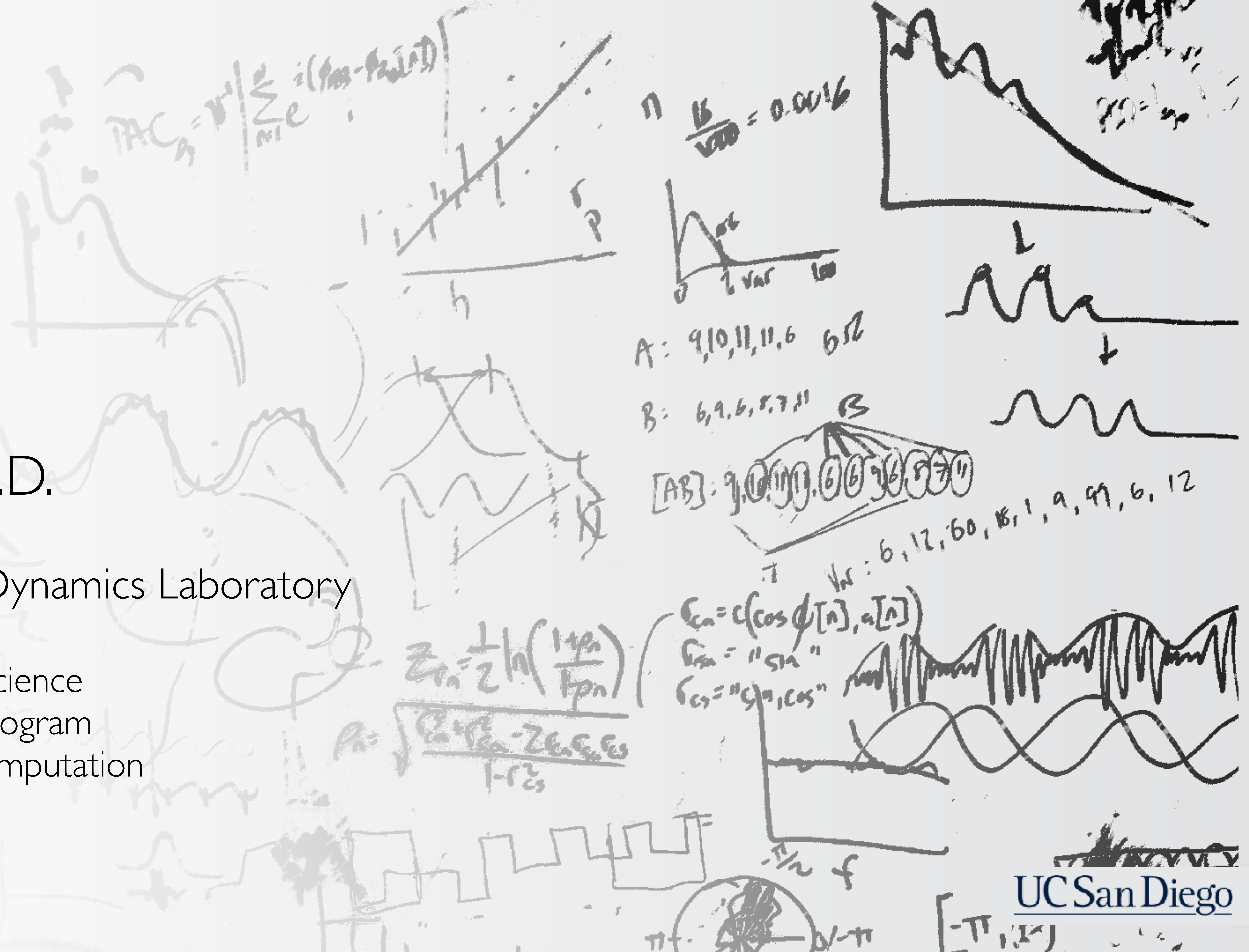


Bradley Voytek, Ph.D.

UC San Diego
Cognitive and Neural Dynamics Laboratory

Department of Cognitive Science
Neurosciences Graduate Program
The Institute for Neural Computation

bvoytek@ucsd.edu
@bradleyvoytek

UC San Diego

Administrative stuff

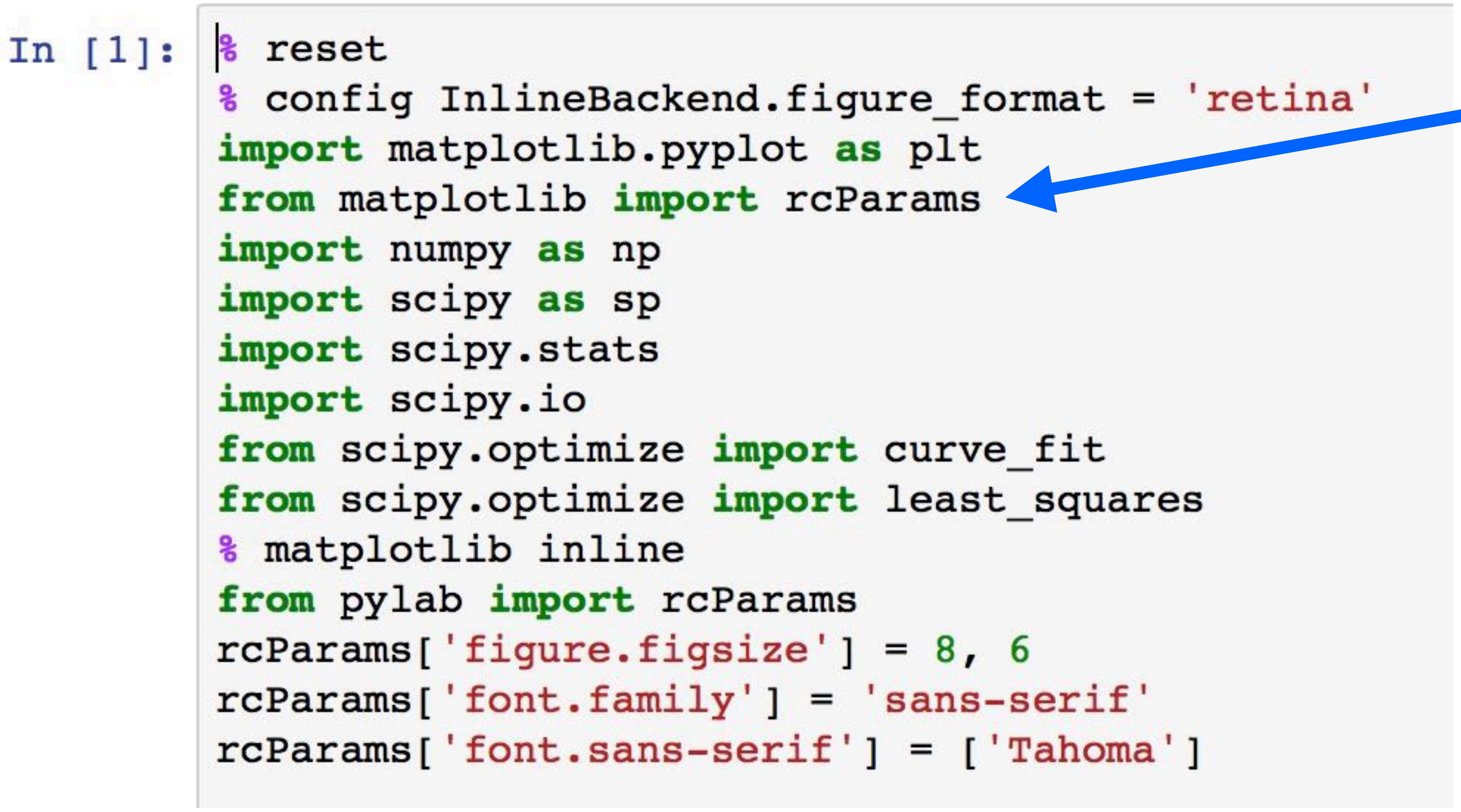
- NO CLASS NEXT FRIDAY (Apr 21)
- Attendance will only be 5% (not 10%)
- Final projects will use GitHub classroom: allows for private group repos; project deliverables are an organized repo (readme and narrative notebook with visualizations); provides a project deliverable for your portfolio; also allows you to make project web pages using GitHub pages, Jupyter slides, etc.

COGS 108
Data Science in Practice

Data Science in Python

UC San Diego

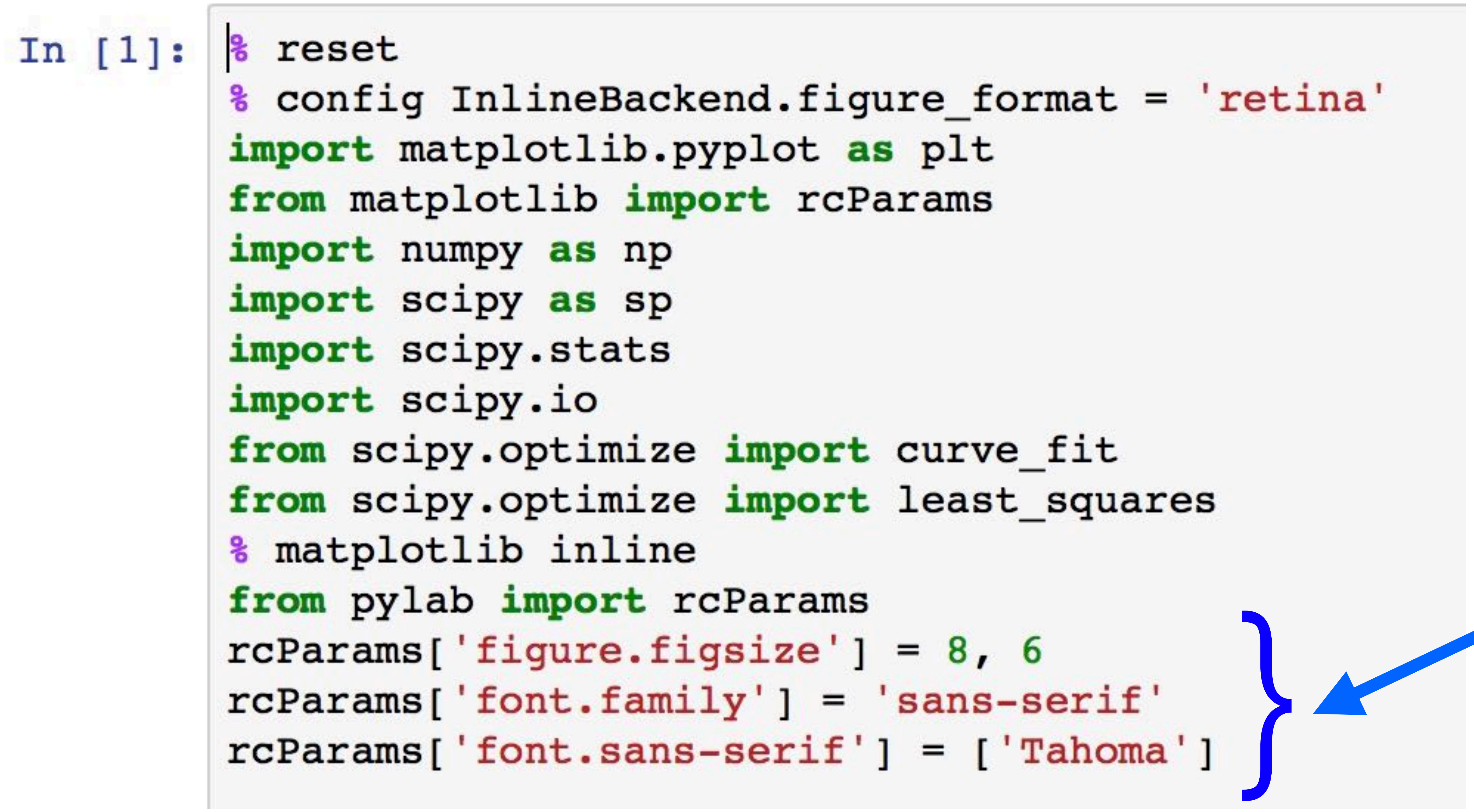
Jupyter - Beginning an analysis



```
In [1]: % reset
        % config InlineBackend.figure_format = 'retina'
        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import numpy as np
        import scipy as sp
        import scipy.stats
        import scipy.io
        from scipy.optimize import curve_fit
        from scipy.optimize import least_squares
        % matplotlib inline
        from pylab import rcParams
        rcParams['figure.figsize'] = 8, 6
        rcParams['font.family'] = 'sans-serif'
        rcParams['font.sans-serif'] = ['Tahoma']
```

plotting parameters

Jupyter - Beginning an analysis

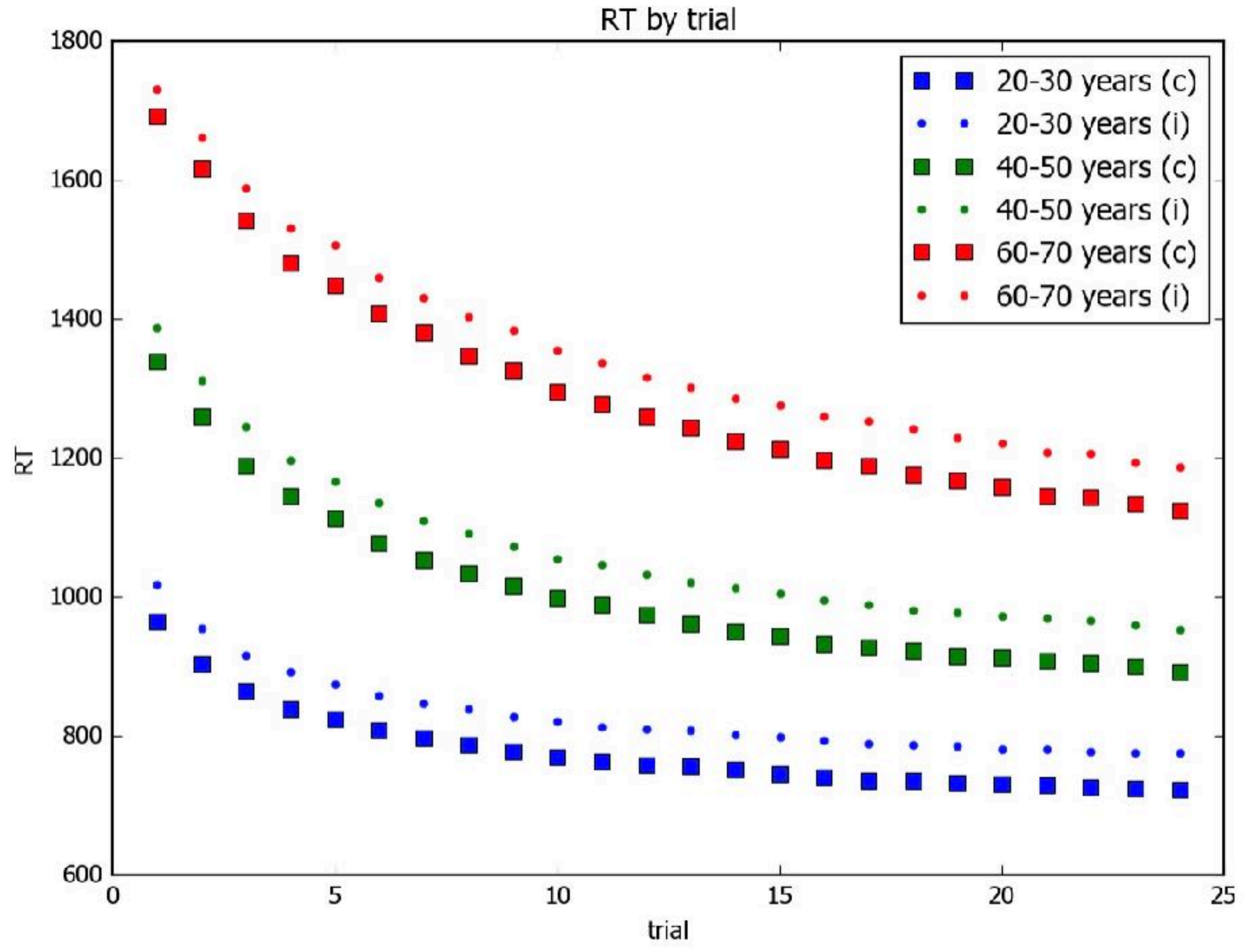


```
In [1]: % reset
        % config InlineBackend.figure_format = 'retina'
        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import numpy as np
        import scipy as sp
        import scipy.stats
        import scipy.io
        from scipy.optimize import curve_fit
        from scipy.optimize import least_squares
        % matplotlib inline
        from pylab import rcParams
        rcParams['figure.figsize'] = 8, 6
        rcParams['font.family'] = 'sans-serif'
        rcParams['font.sans-serif'] = ['Tahoma']
```

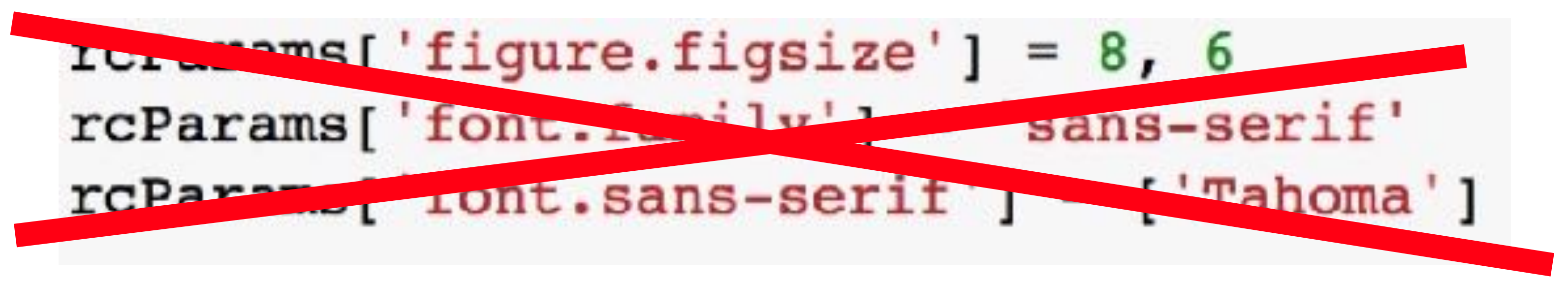
figure parameters

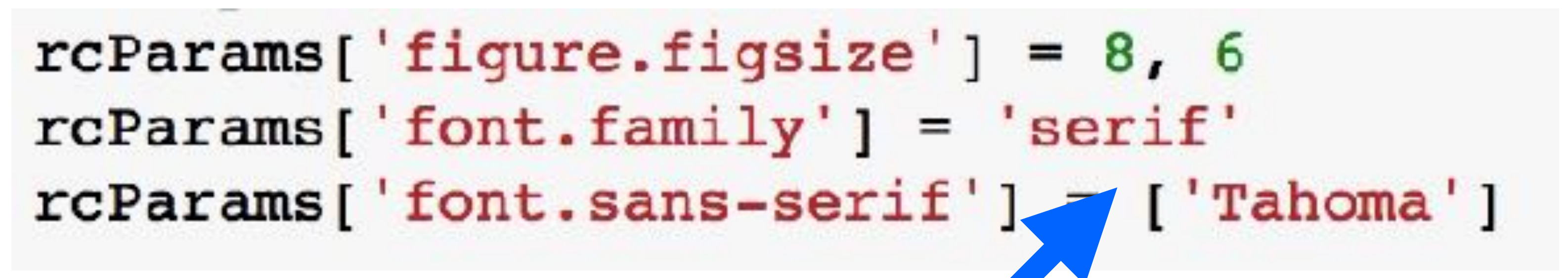
Jupyter - Figure parameters

```
rcParams['figure.figsize'] = 8, 6
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Tahoma']
```

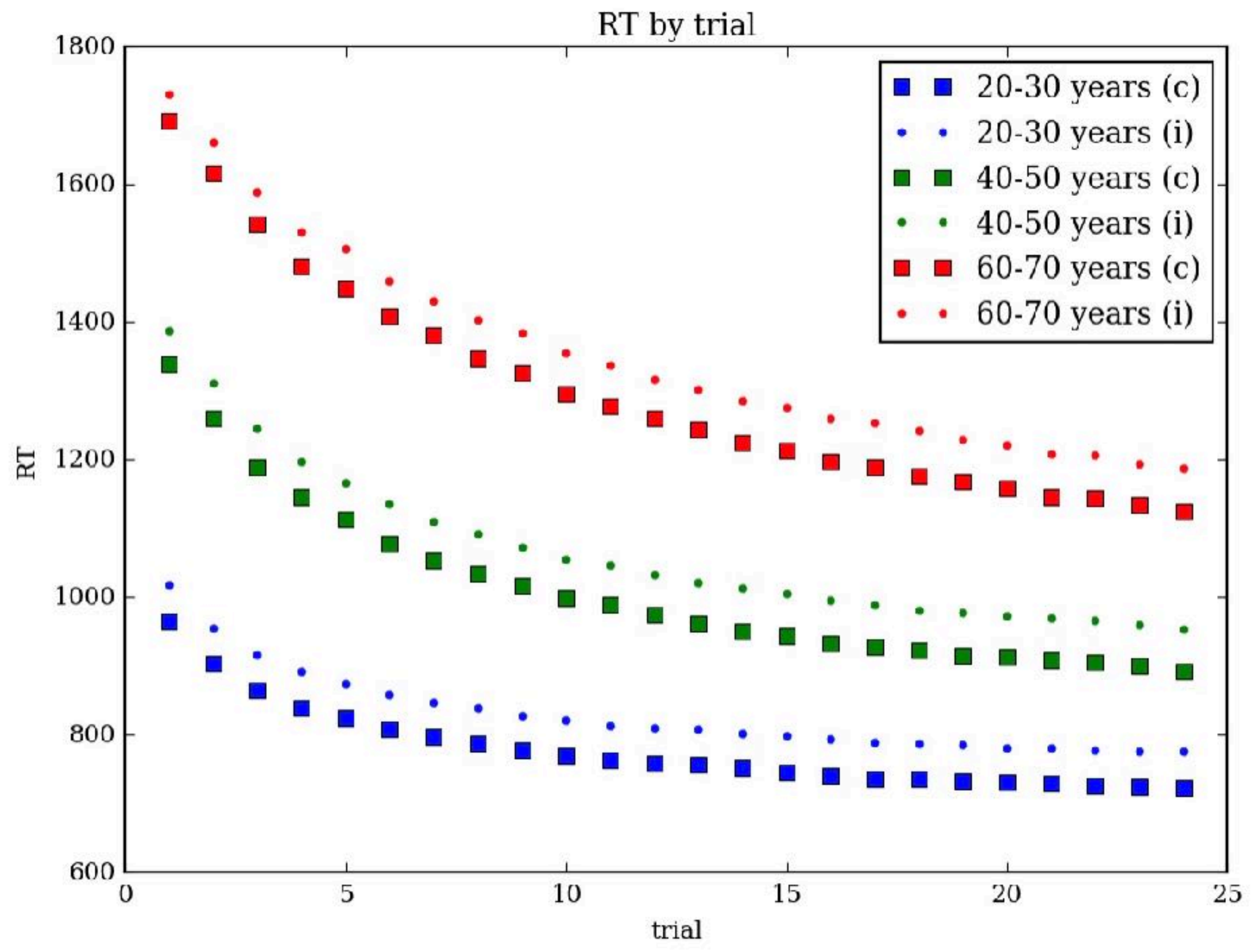


Jupyter - Figure parameters


~~rcParams['figure.figsize'] = 8, 6~~
~~rcParams['font.family'] = 'sans-serif'~~
~~rcParams['font.sans-serif'] = ['Tahoma']~~


```
rcParams['figure.figsize'] = 8, 6
rcParams['font.family'] = 'serif'
rcParams['font.sans-serif'] = ['Tahoma']
```

serif font now

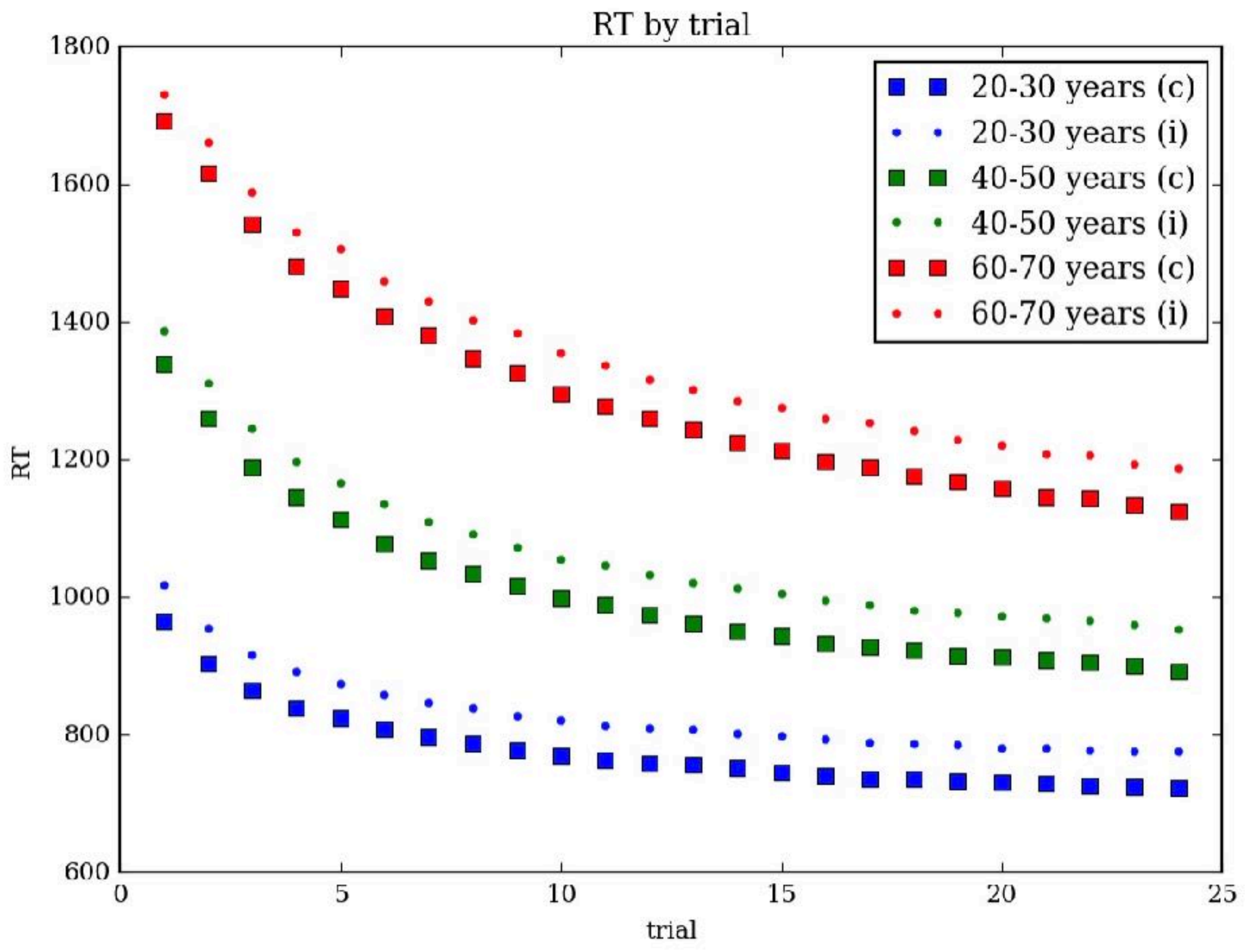


Jupyter - Figure parameters

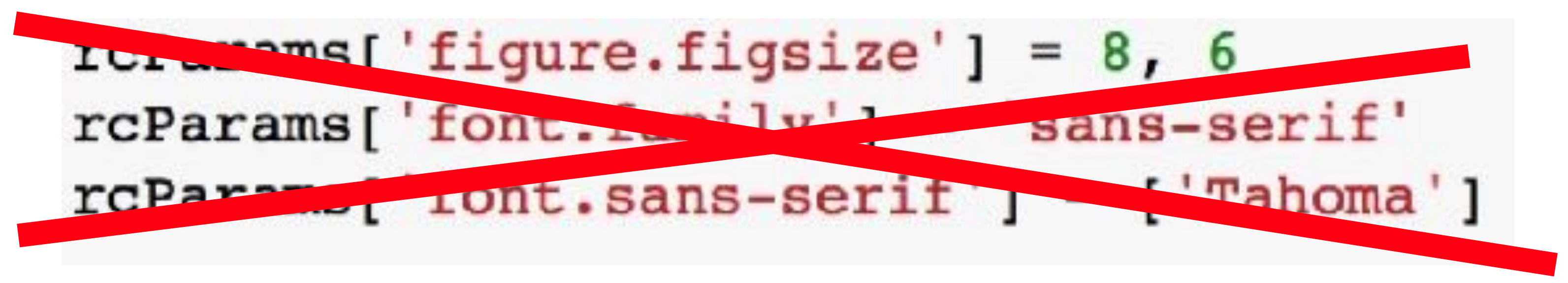
~~rcParams['figure.figsize'] = 8, 6~~
~~rcParams['font.family'] = 'sans-serif'~~
~~rcParams['font.sans-serif'] = ['Tahoma']~~

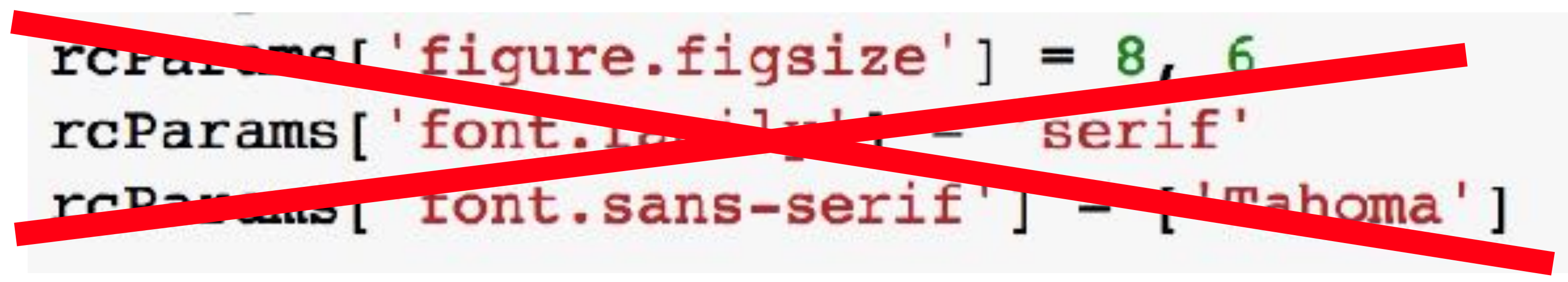
```
rcParams['figure.figsize'] = 8, 6
rcParams['font.family'] = 'serif'
rcParams['font.sans-serif'] = ['Tahoma']
```

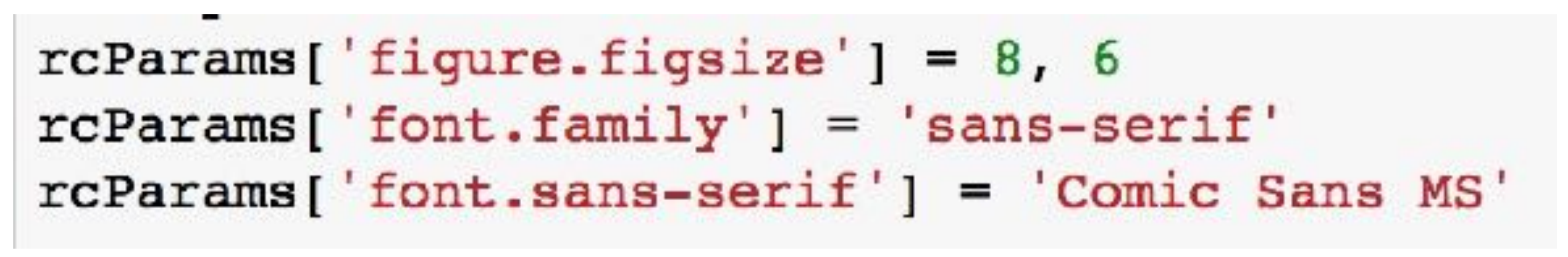
despite this saying sans serif



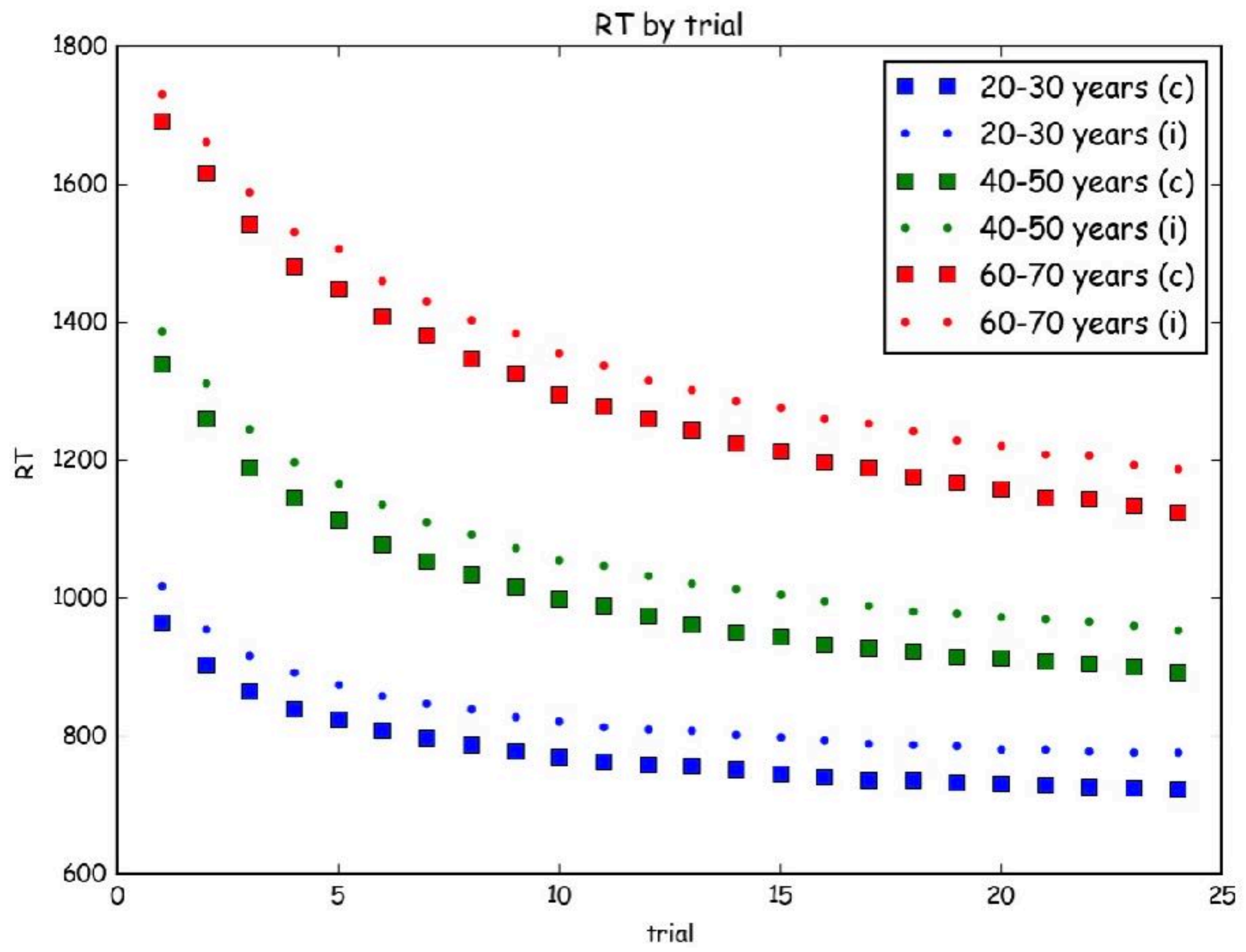
Jupyter - Figure parameters


~~rcParams['figure.figsize'] = 8, 6~~
~~rcParams['font.family'] = 'sans-serif'~~
~~rcParams['font.sans-serif'] = ['Tahoma']~~


~~rcParams['figure.figsize'] = 8, 6~~
~~rcParams['font.family'] = 'serif'~~
~~rcParams['font.sans-serif'] = ['Tahoma']~~

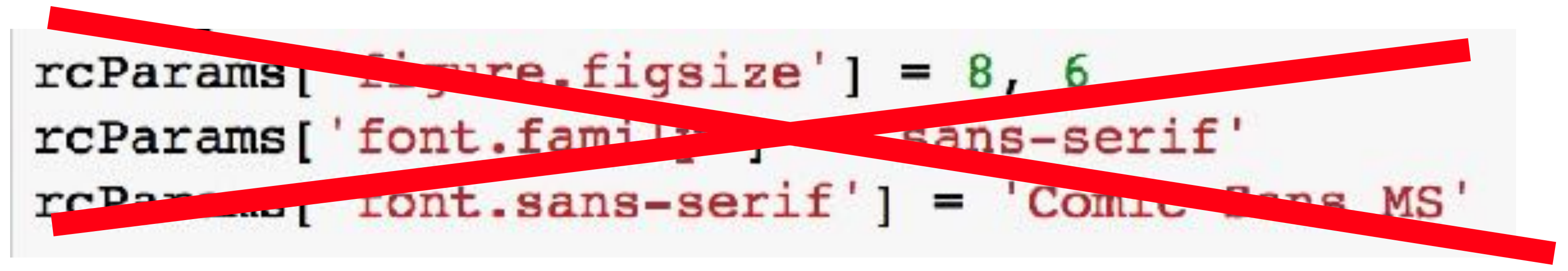

```
rcParams['figure.figsize'] = 8, 6
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = 'Comic Sans MS'
```

comic sans ftw!

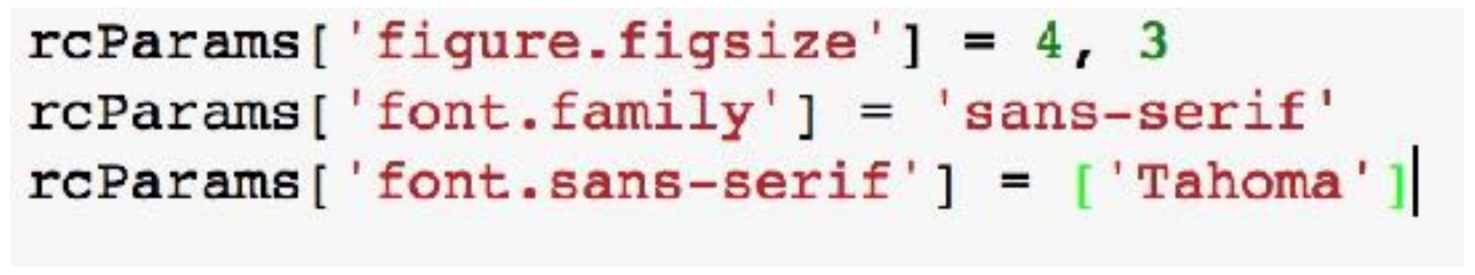


Jupyter - Figure parameters

NOTE! I didn't restart the jupyter kernel before plotting again, meaning it's still plotting in comic sans!



~~rcParams['figure.figsize'] = 8, 6~~
~~rcParams['font.family'] = 'sans-serif'~~
~~rcParams['font.sans-serif'] = 'Comic Sans MS'~~



```
rcParams['figure.figsize'] = 4, 3
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Tahoma']
```

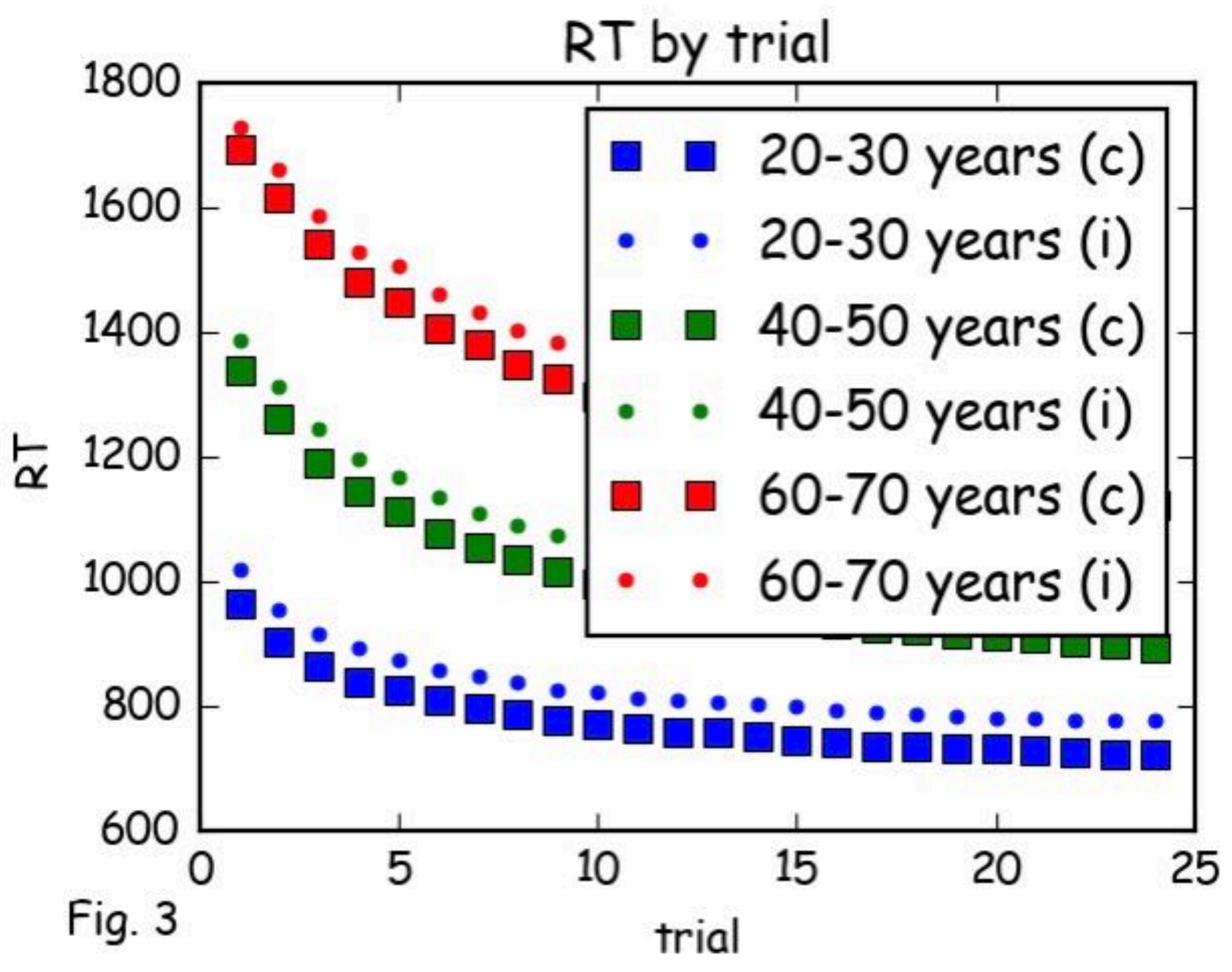
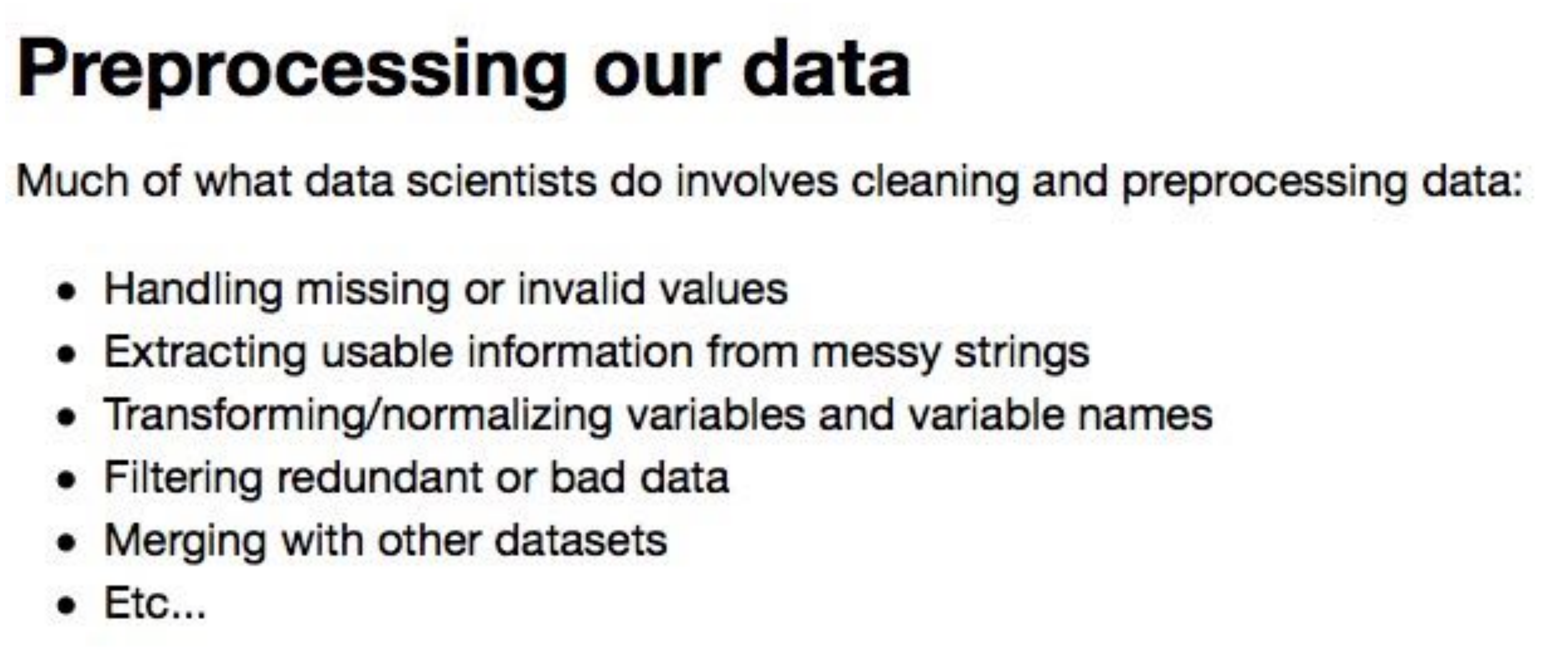


Fig. 3

Pandas

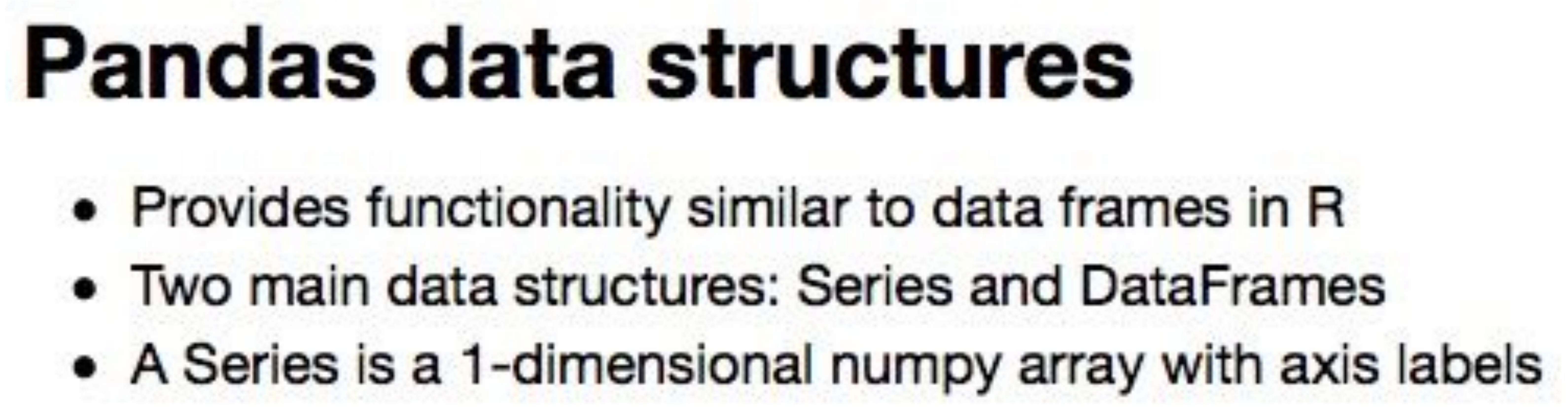


Preprocessing our data

Much of what data scientists do involves cleaning and preprocessing data:

- Handling missing or invalid values
- Extracting usable information from messy strings
- Transforming/normalizing variables and variable names
- Filtering redundant or bad data
- Merging with other datasets
- Etc...

Pandas

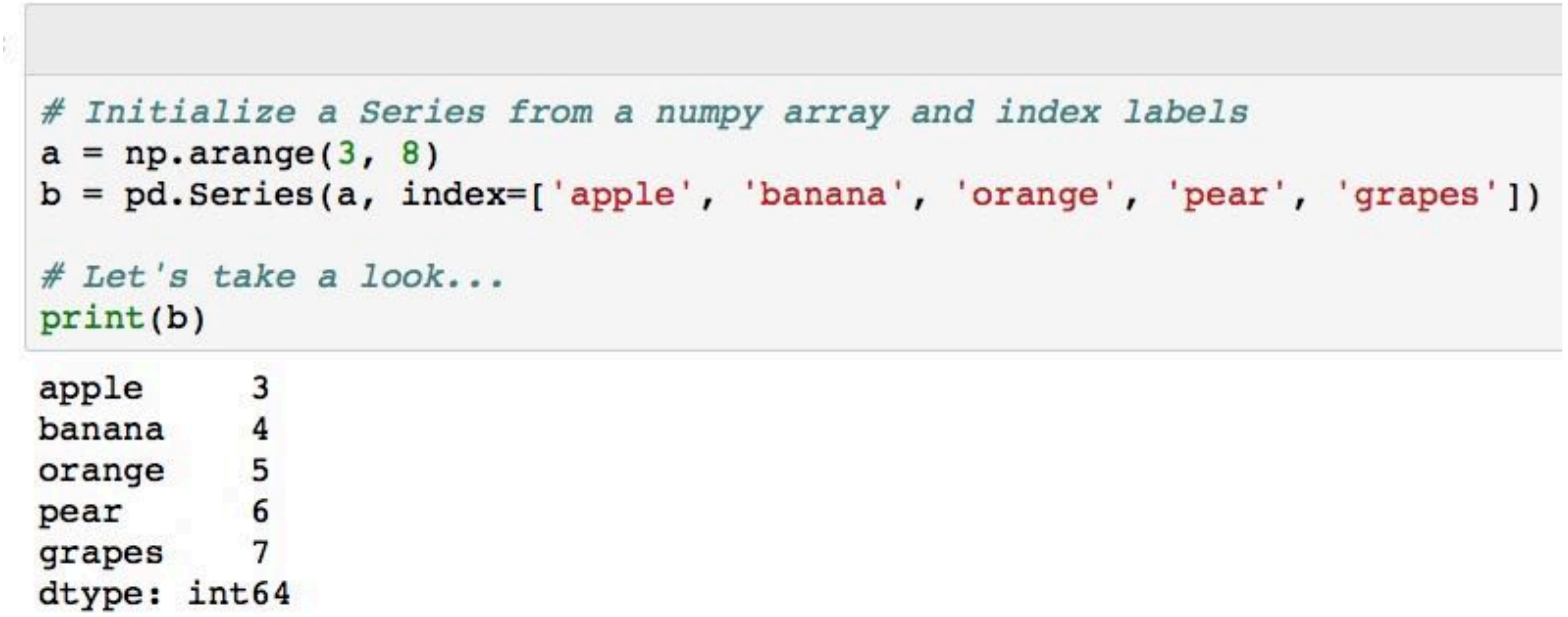


Pandas data structures

- Provides functionality similar to data frames in R
- Two main data structures: Series and DataFrames
- A Series is a 1-dimensional numpy array with axis labels

Source: Tal Yarkoni

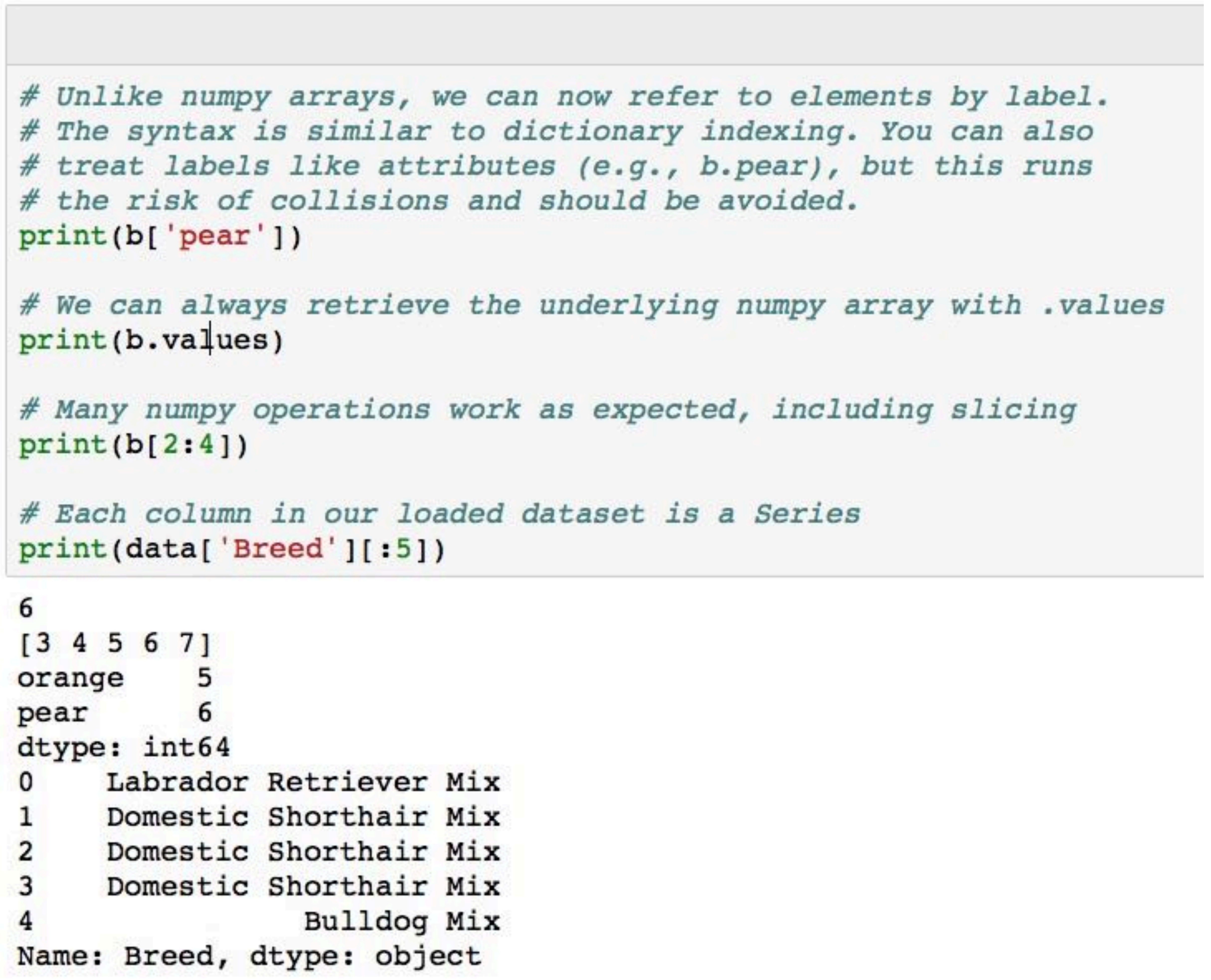
Pandas



```python
# Initialize a Series from a numpy array and index labels
a = np.arange(3, 8)
b = pd.Series(a, index=['apple', 'banana', 'orange', 'pear', 'grapes'])

# Let's take a look...
print(b)
```

```
apple     3
banana    4
orange    5
pear      6
grapes    7
dtype: int64
```

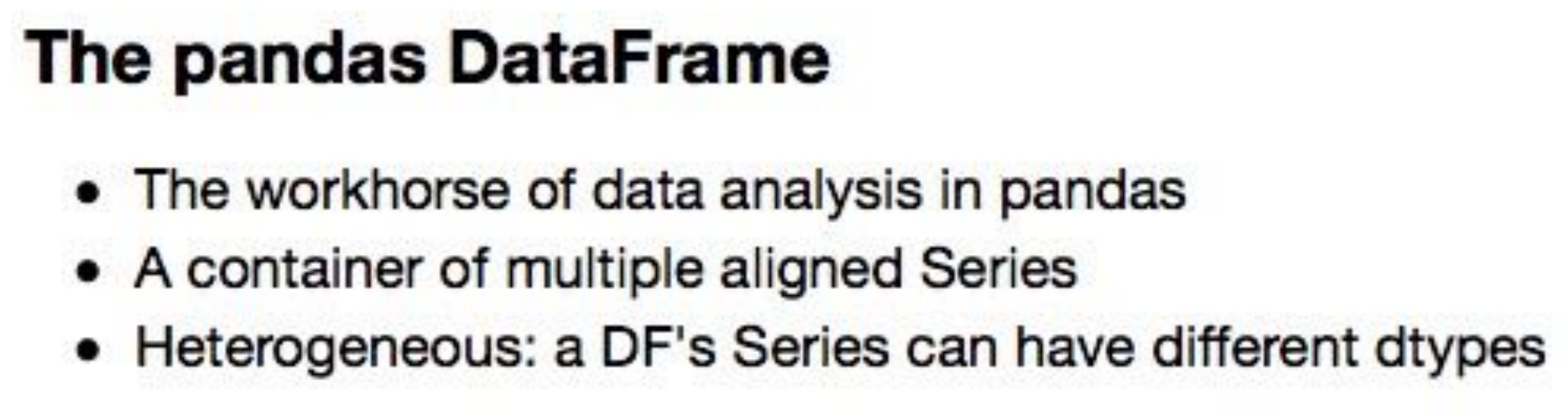
Pandas



```
# Unlike numpy arrays, we can now refer to elements by label.
# The syntax is similar to dictionary indexing. You can also
# treat labels like attributes (e.g., b.pear), but this runs
# the risk of collisions and should be avoided.
print(b['pear'])

# We can always retrieve the underlying numpy array with .values
print(b.values)

# Many numpy operations work as expected, including slicing
print(b[2:4])

# Each column in our loaded dataset is a Series
print(data['Breed'][:5])
```

```
6
[3 4 5 6 7]
orange    5
pear      6
dtype: int64
0    Labrador Retriever Mix
1    Domestic Shorthair Mix
2    Domestic Shorthair Mix
3    Domestic Shorthair Mix
4               Bulldog Mix
Name: Breed, dtype: object
```

Pandas

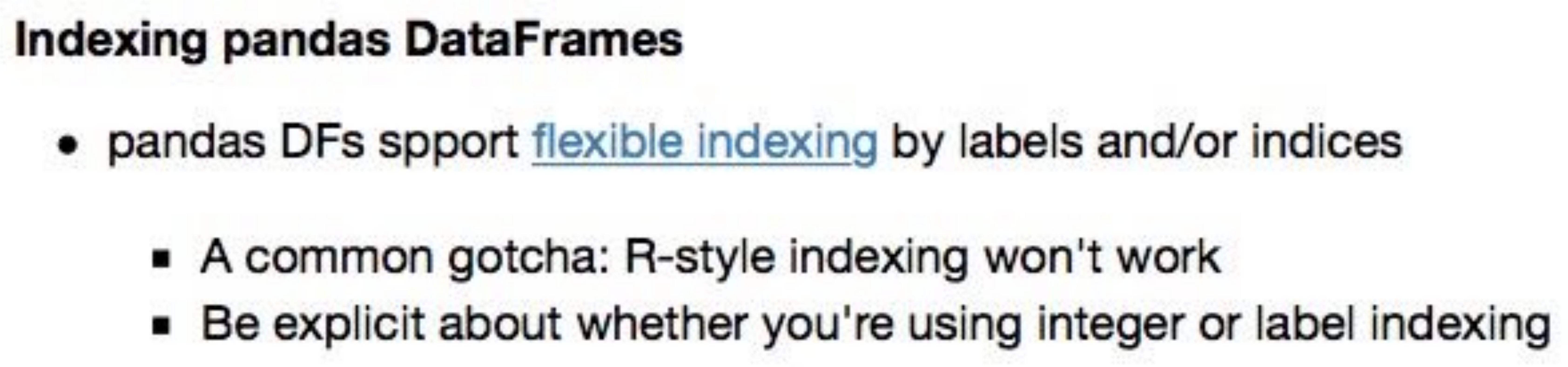


The pandas DataFrame

- The workhorse of data analysis in pandas
- A container of multiple aligned Series
- Heterogeneous: a DF's Series can have different dtypes

Source: Tal Yarkoni

Pandas

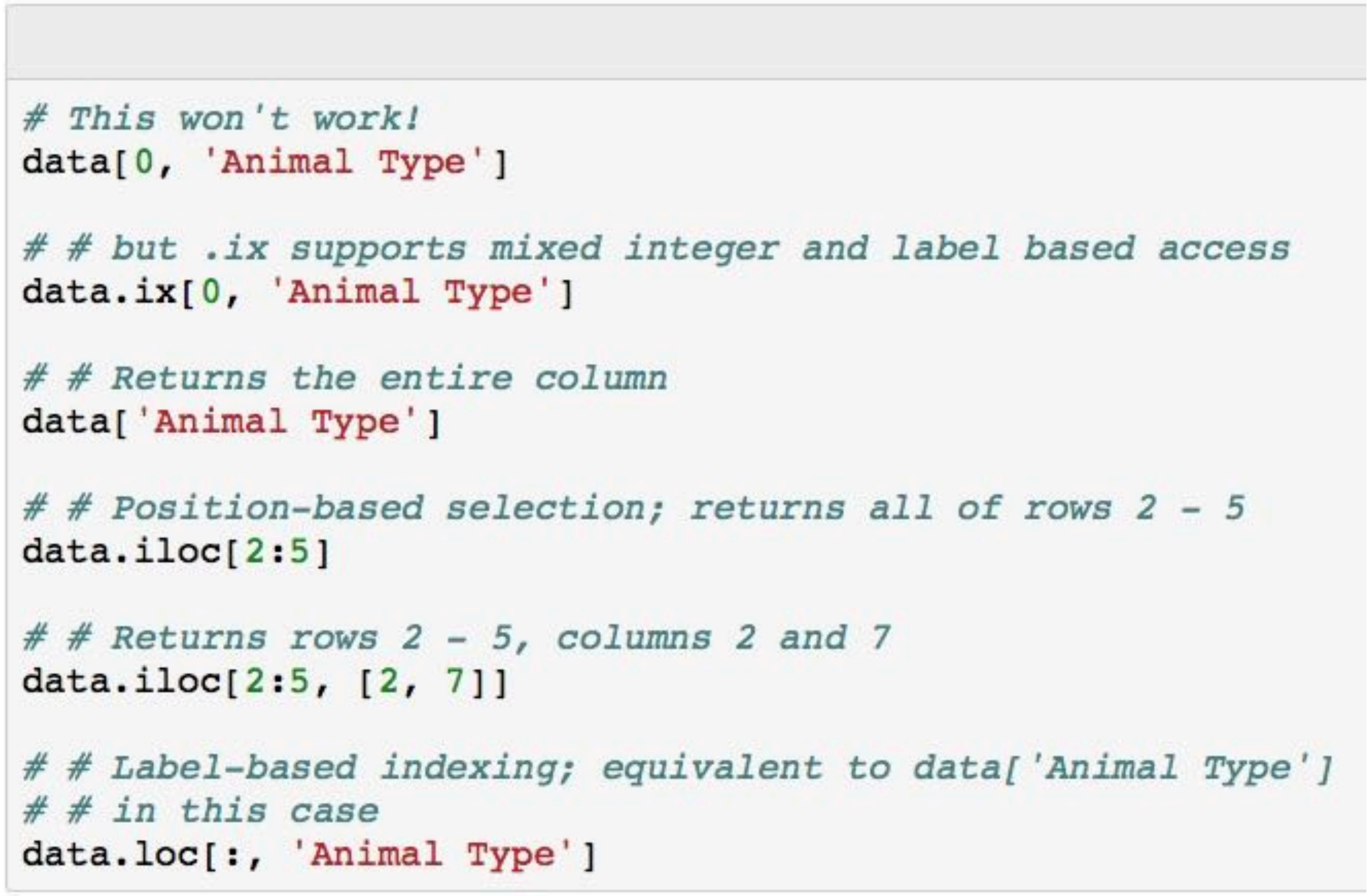


Indexing pandas DataFrames

- pandas DFs spport flexible indexing by labels and/or indices
  - A common gotcha: R-style indexing won't work
  - Be explicit about whether you're using integer or label indexing

Source: Tal Yarkoni

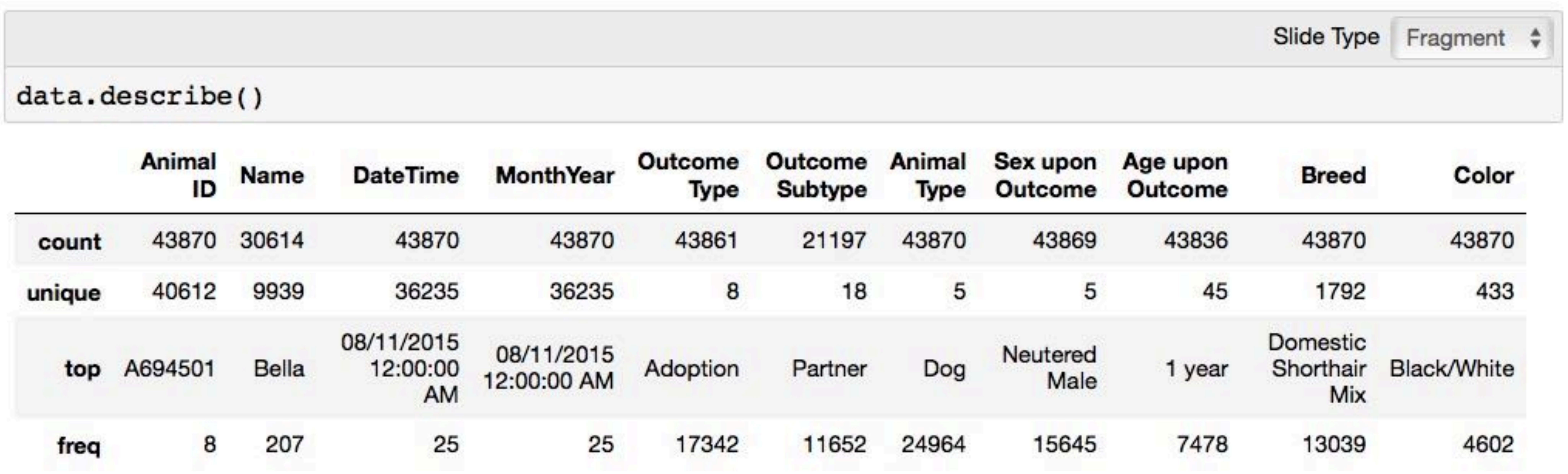
Pandas



```
# This won't work!
data[0, 'Animal Type']

# # but .ix supports mixed integer and label based access
data.ix[0, 'Animal Type']

# # Returns the entire column
data['Animal Type']

# # Position-based selection; returns all of rows 2 - 5
data.iloc[2:5]

# # Returns rows 2 - 5, columns 2 and 7
data.iloc[2:5, [2, 7]]

# # Label-based indexing; equivalent to data['Animal Type']
# # in this case
data.loc[:, 'Animal Type']
```

Source: Tal Yarkoni

Pandas



Slide Type Fragment

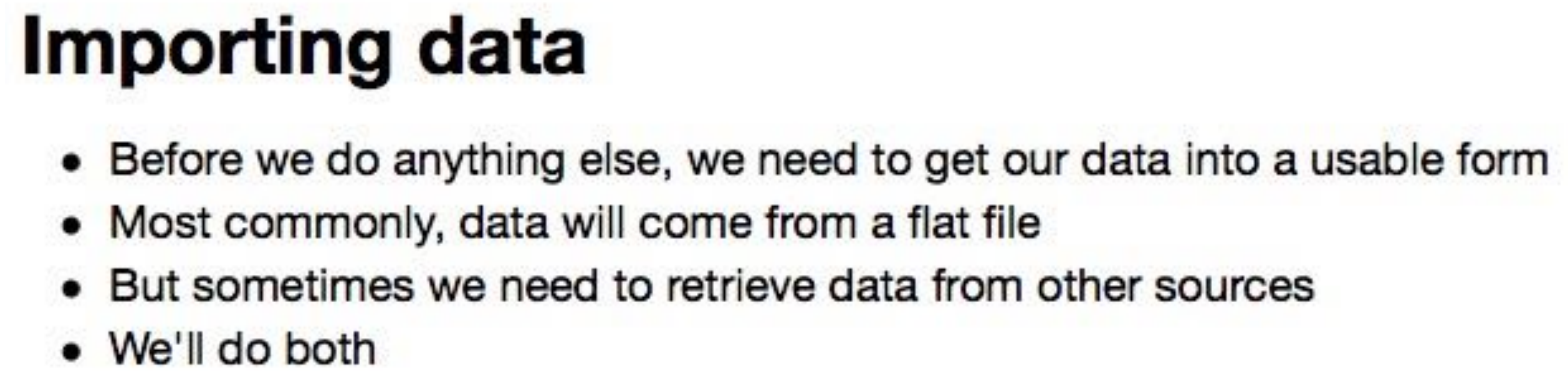
```
data.describe()
```

	Animal ID	Name	DateTime	MonthYear	Outcome Type	Outcome Subtype	Animal Type	Sex upon Outcome	Age upon Outcome	Breed	Color
count	43870	30614	43870	43870	43861	21197	43870	43869	43836	43870	43870
unique	40612	9939	36235	36235	8	18	5	5	45	1792	433
top	A694501	Bella	08/11/2015 12:00:00 AM	08/11/2015 12:00:00 AM	Adoption	Partner	Dog	Neutered Male	1 year	Domestic Shorthair Mix	Black/White
freq	8	207	25	25	17342	11652	24964	15645	7478	13039	4602

Source: Tal Yarkoni

UC San Diego

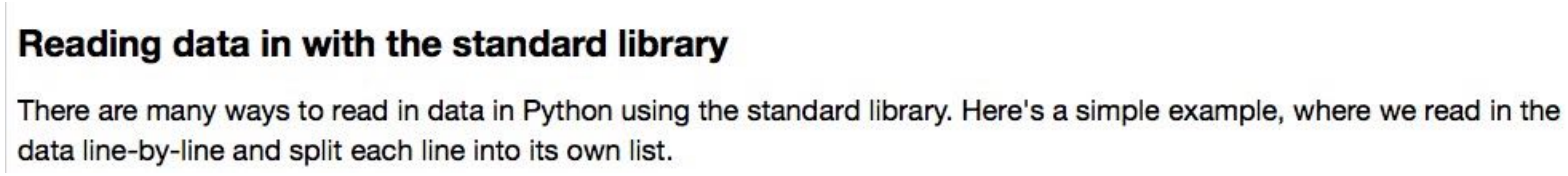
Pandas



Importing data

- Before we do anything else, we need to get our data into a usable form
- Most commonly, data will come from a flat file
- But sometimes we need to retrieve data from other sources
- We'll do both

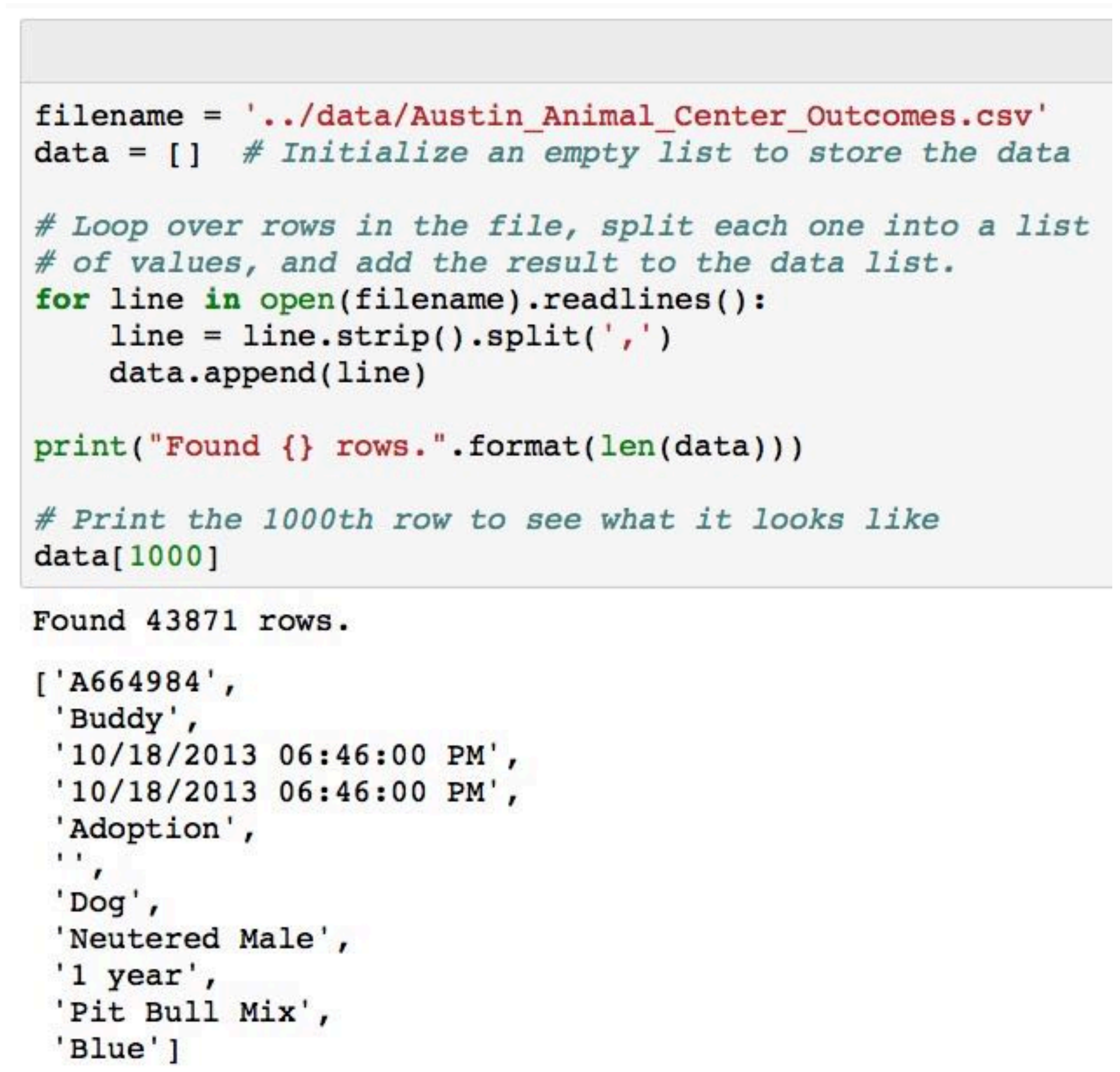
Not Pandas



Reading data in with the standard library

There are many ways to read in data in Python using the standard library. Here's a simple example, where we read in the data line-by-line and split each line into its own list.

Not Pandas

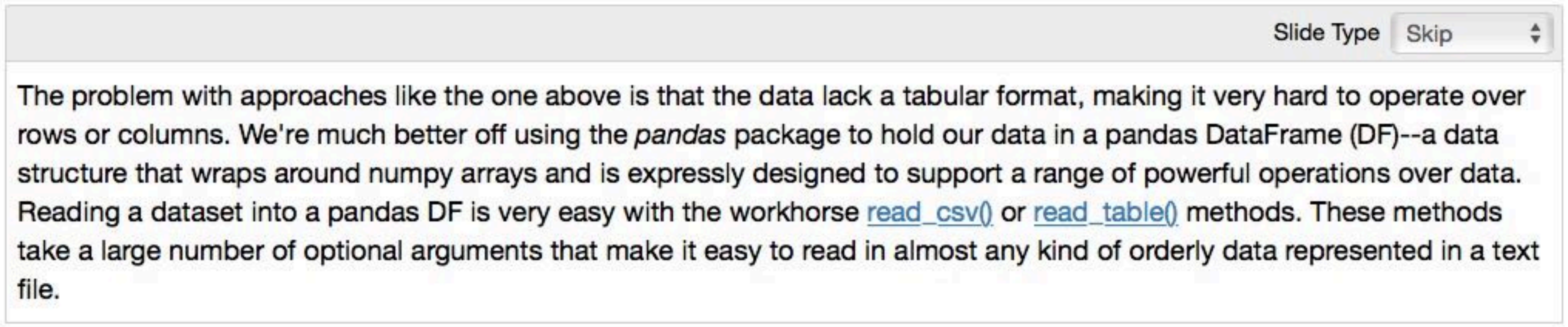


```
filename = '../data/Austin_Animal_Center_Outcomes.csv'
data = []  # Initialize an empty list to store the data

# Loop over rows in the file, split each one into a list
# of values, and add the result to the data list.
for line in open(filename).readlines():
    line = line.strip().split(',')
    data.append(line)

print("Found {} rows.".format(len(data)))

# Print the 1000th row to see what it looks like
data[1000]
```

```
Found 43871 rows.

['A664984',
 'Buddy',
 '10/18/2013 06:46:00 PM',
 '10/18/2013 06:46:00 PM',
 'Adoption',
 '',
 'Dog',
 'Neutered Male',
 '1 year',
 'Pit Bull Mix',
 'Blue']
```

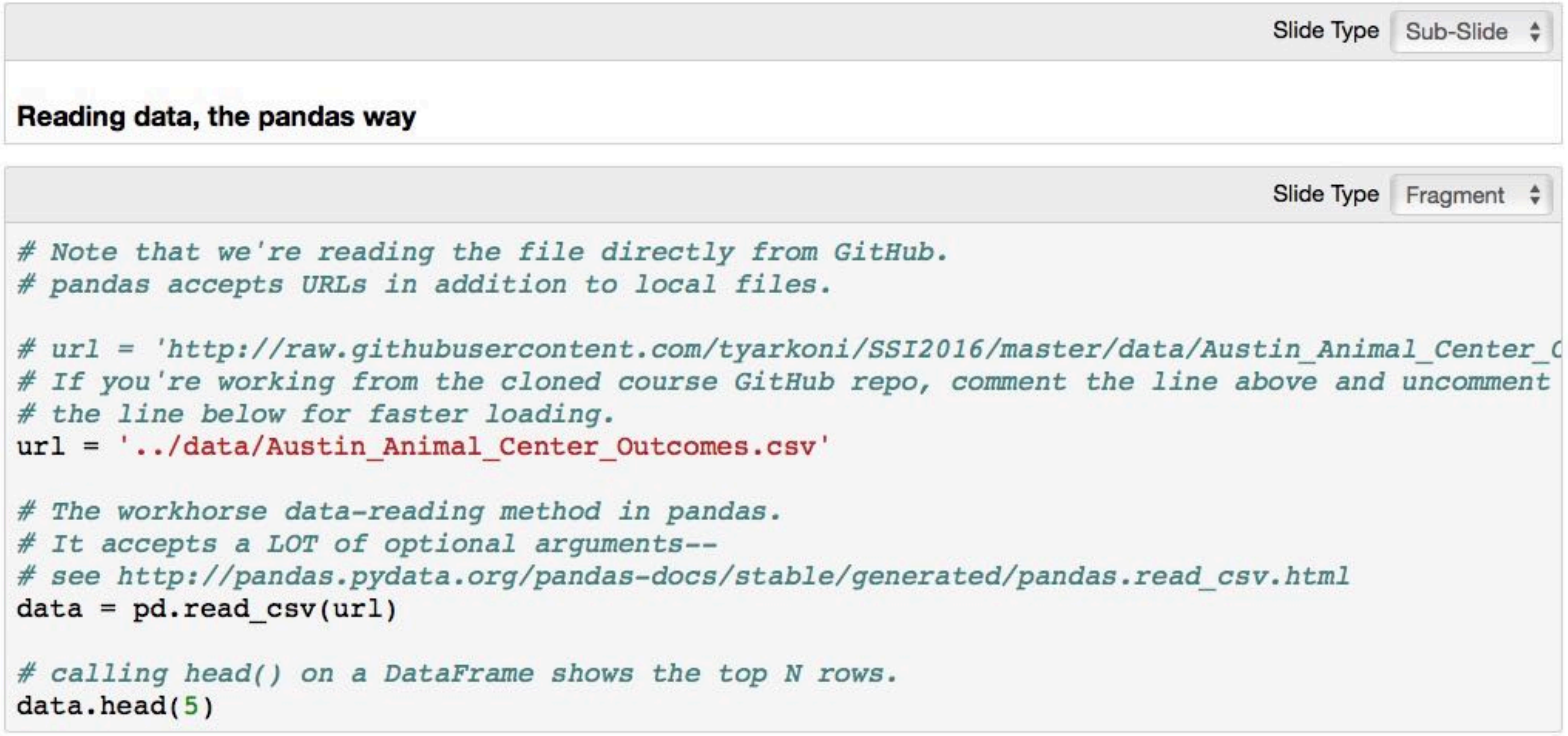
Source: Tal Yarkoni

Pandas


Slide Type Skip

The problem with approaches like the one above is that the data lack a tabular format, making it very hard to operate over rows or columns. We're much better off using the *pandas* package to hold our data in a pandas DataFrame (DF)--a data structure that wraps around numpy arrays and is expressly designed to support a range of powerful operations over data. Reading a dataset into a pandas DF is very easy with the workhorse read_csv() or read_table() methods. These methods take a large number of optional arguments that make it easy to read in almost any kind of orderly data represented in a text file.

Pandas



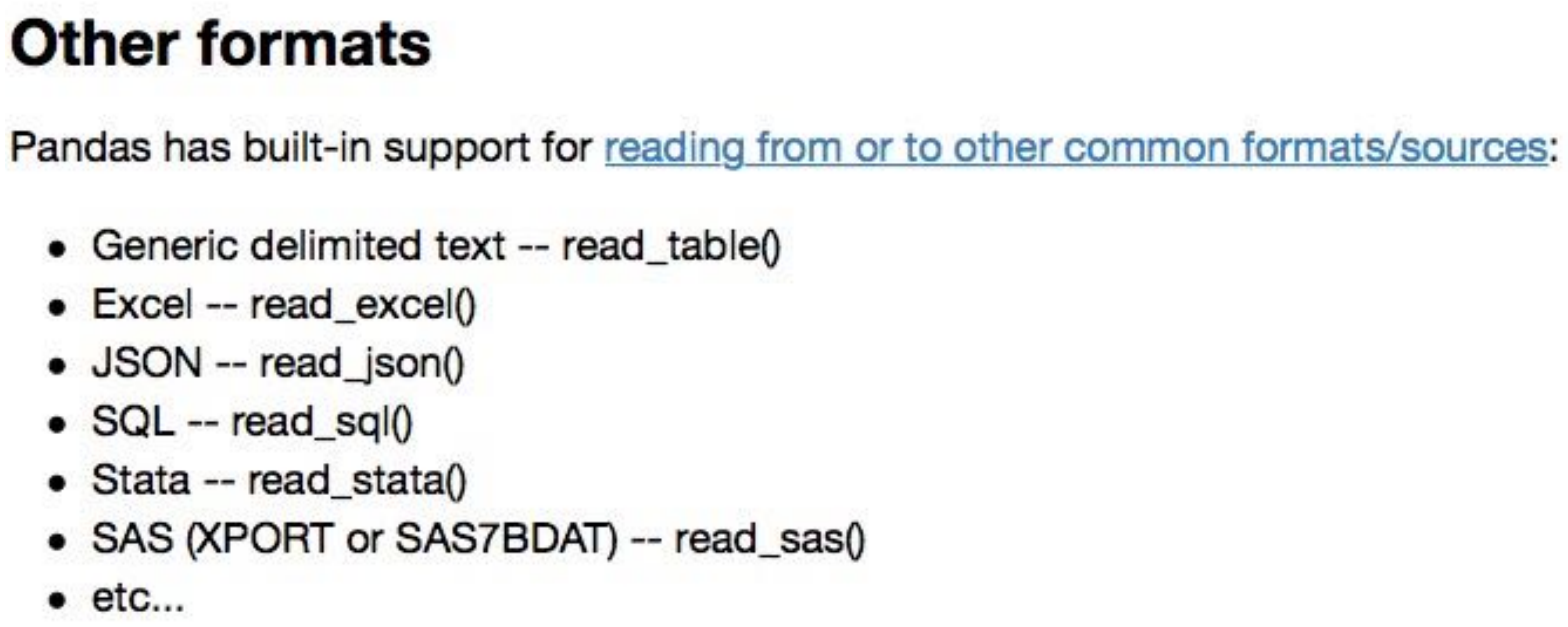
Reading data, the pandas way

```
# Note that we're reading the file directly from GitHub.
# pandas accepts URLs in addition to local files.

# url = 'http://raw.githubusercontent.com/tyarkoni/SSI2016/master/data/Austin_Animal_Center_O
# If you're working from the cloned course GitHub repo, comment the line above and uncomment
# the line below for faster loading.
url = '../data/Austin_Animal_Center_Outcomes.csv'

# The workhorse data-reading method in pandas.
# It accepts a LOT of optional arguments--
# see http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html
data = pd.read_csv(url)

# calling head() on a DataFrame shows the top N rows.
data.head(5)
```

UC San Diego

Source: Tal Yarkoni

Pandas



Other formats

Pandas has built-in support for reading from or to other common formats/sources:

- Generic delimited text -- read_table()
- Excel -- read_excel()
- JSON -- read_json()
- SQL -- read_sql()
- Stata -- read_stata()
- SAS (XPORT or SAS7BDAT) -- read_sas()
- etc...

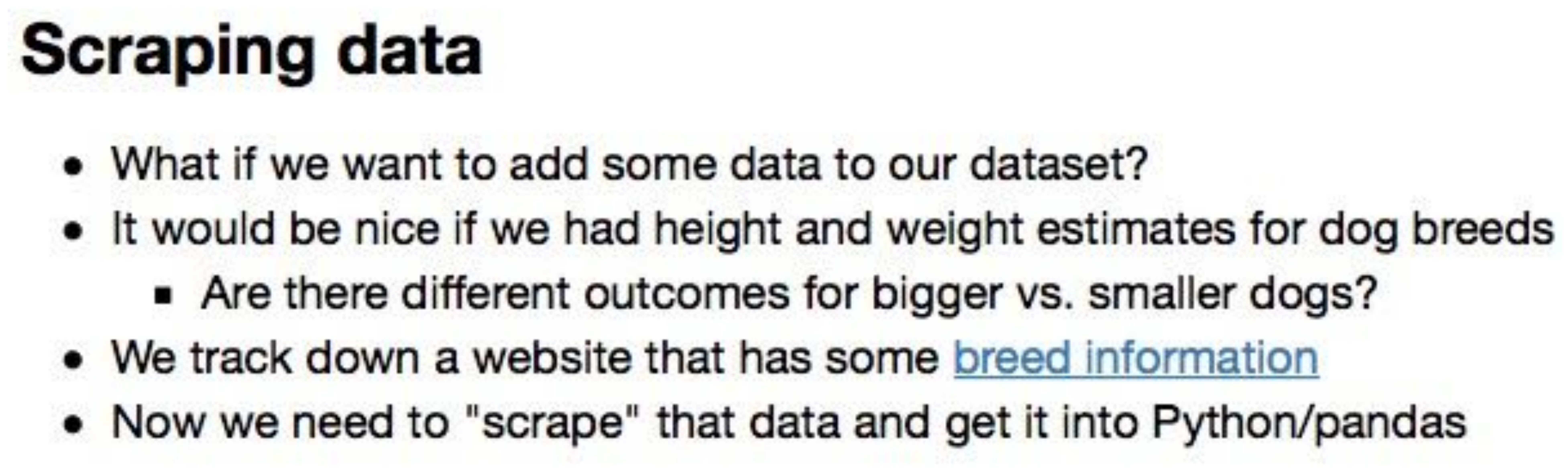
Source: Tal Yarkoni

COGS 108
Data Science in Practice

Data gathering

UC San Diego

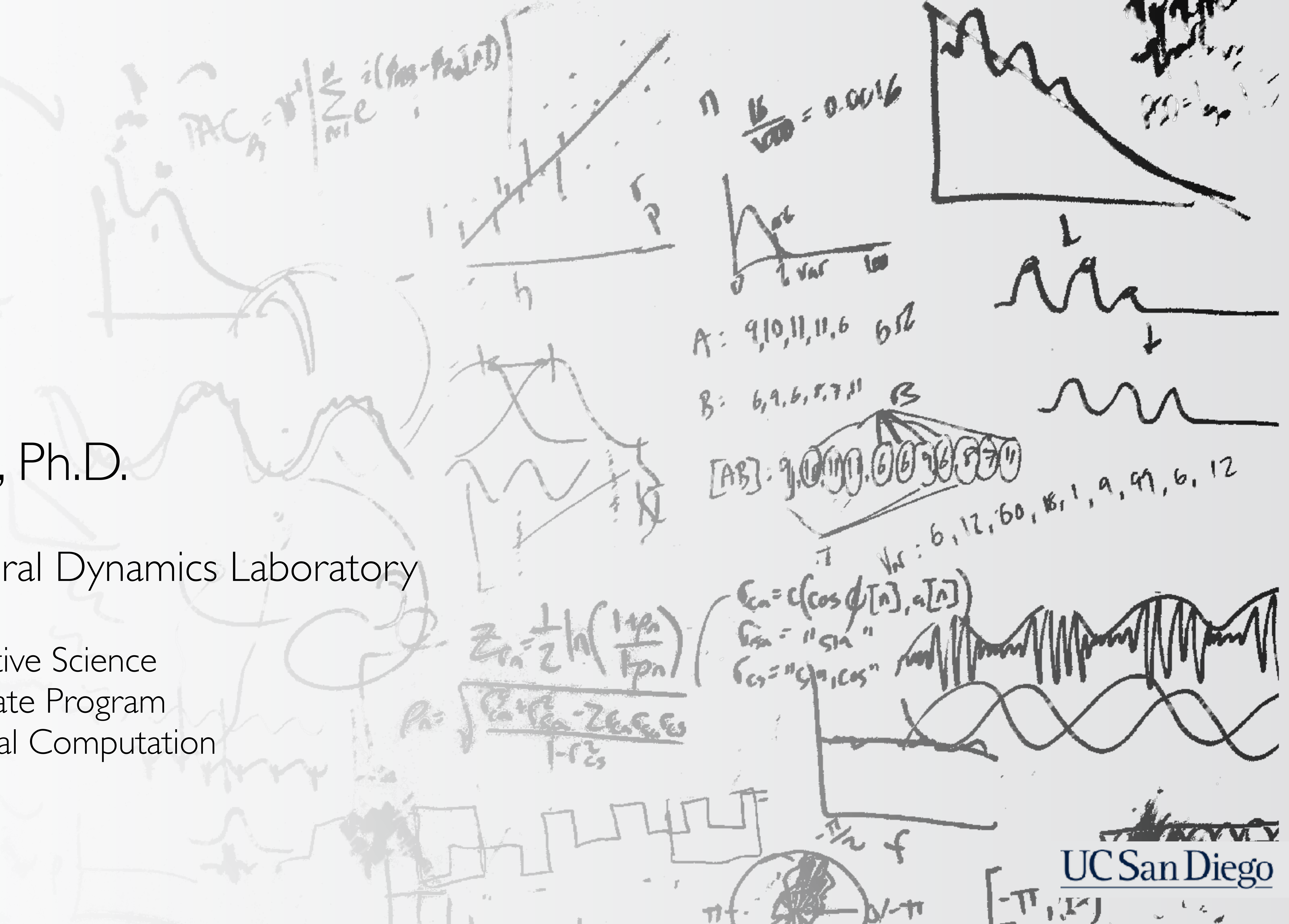
Pandas



Scraping data

- What if we want to add some data to our dataset?
- It would be nice if we had height and weight estimates for dog breeds
  - Are there different outcomes for bigger vs. smaller dogs?
- We track down a website that has some breed information
- Now we need to "scrape" that data and get it into Python/pandas

Source: Tal Yarkoni



Bradley Voytek, Ph.D.

UC San Diego
Cognitive and Neural Dynamics Laboratory

Department of Cognitive Science
Neurosciences Graduate Program
The Institute for Neural Computation

bvoytek@ucsd.edu
@bradleyvoytek