Creating DataFrames

We will create a `DataFrame` object from a dataset in a CSV file using the **read_csv** method on the pandas - **pd** - object. While the read_csv method can read dataset directly from a web URL, it's best to download to your computer first to ensure the data is persisted.

About the Data

We will use the dataset on nobel lauretes available at: http://api.nobelprize.org/v1/laureates.csv. Download the file to your computer using: **curl -OL** http://api.nobelprize.org/v1/laureates.csv

Imports

```
import datetime
import numpy as np
import pandas as pd
```

Creating a `Series`

```
np.random.seed(0) # set a seed for reproducibility
pd.Series(np.random.rand(5), name='random')
```

```
0    0.548814
1    0.715189
2    0.602763
3    0.544883
4    0.423655
Name: random, dtype: float64
```

Creating a `DataFrame` from a `Series`

Use the `to_frame()` method:

```
pd.Series(np.linspace(0, 10, num=5)).to_frame()
```

	0
0	0.0
1	2.5
2	5.0
3	7.5
4	10.0

From a list of dictionaries

```
pd.DataFrame([
    {'mag' : 5.2, 'place' : 'California'},
    {'mag' : 1.2, 'place' : 'Alaska'},
    {'mag' : 0.2, 'place' : 'California'},
])
```

	mag	place
0	5.2	California
1	1.2	Alaska
2	0.2	California

From a NumPy array

```
pd.DataFrame(
    np.array([
        [0, 0, 0],
        [1, 1, 1],
        [2, 4, 8],
        [3, 9, 27],
        [4, 16, 64]
    ]), columns=['n', 'n_squared', 'n_cubed']
)
```

	n	n_squared	n_cubed
0	0	0	0
1	1	1	1
2	2	4	8
3	3	9	27
4	4	16	64

Creating a `DataFrame` by Reading in a CSV File

Finding information on the file before reading it in

Before attempting to read in a file, we can use the command line to see important information about the file that may determine how we read it in. We can run command line code from Jupyter Notebooks (thanks to IPython) by using `!` before the code.

Number of lines (row count)

For example, we can find out how many lines are in the file by using the `wc` utility (word count) and counting lines in the file (`-l`). Run the cell below to confirm the file has 1002 lines:

```
!wc -l laureates.csv  # this will not work on Windows commandline
# On windows the !dir command will show directory contents
```

```
1002 laureates.csv
```

We can even capture the result of a command and use it in our Python code:

Reading in the file

Our file is small in size, has headers in the first row, and is comma-separated, so we don't need to provide any additional arguments to read in the file with `pd.read_csv()`, but be sure to check the documentation for possible arguments. To read data from file we can use `pd.read_csv()` and for other delimited files, such as tab (\t), we can use the `read_csv()` function with the sep argument equal to the delimiter. We can use the `read_excel()` function for Excel files, the `read_json()` function for JSON (JavaScript Object Notation) files

```
import pandas as pd
df = pd.read_csv('laureates.csv')
```

Getting Documentation on Python elements

You can utilize the built-in `help()` function for documentation on Python elements. Simply run `help()`, passing in the package, module, class, object, method, or function. Assuming we aliased pandas as pd when we imported it, we can run `help(pd)` to see information on the pandas package; `help(pd.DataFrame)` for all the methods and attributes of a dataframe (note we can also pass in an already created DataFrame object instead); and `help(pd.read_csv)` to learn more about the pandas function for reading CSV files into Python and how to use it

Run the code cell below to see documentation for the DataFrame class.

```
import pandas as pd
help(pd.DataFrame)
```

```
 |  
 |  __lt__(self, other)
 |      Return self<value.
 |  
 |  __mod__(self, other)
 |  
 |  __mul__(self, other)
 |  
 |  __ne__(self, other)
 |      Return self!=value.
 |  
 |  __or__(self, other)
 |      Return self|value.
 |  
 |  __pow__(self, other)
 |  
 |  __radd__(self, other)
 |  
 |  __rand__(self, other)
 |  
 |  __rfloordiv__(self, other)
 |  
 |  __rmod__(self, other)
 |  
 |  __rmul__(self, other)
 |  
 |  __ror__(self, other)
 |      Return value|self.
 |  
 |  __rpow__(self, other)
 |  
 |  __rsub__(self, other)
 |  
 |  __rtruediv__(self, other)
 |  
 |  __rxor__(self, other)
 |  
 |  __sub__(self, other)
 |  
 |  __truediv__(self, other)
 |  
 |  __xor__(self, other)
 |  
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from pandas.core.arraylike.OpsMixin:
 |  
 |  __hash__ = None
```

Let's review summary statistics for the nobel laureates dataframe - df: use `describe()`.

df.describe()

`df.describe()` does not really tell us much. The `info()` method provides more information as you can check by running the code cell below.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 1000 non-null   int64 
 1   firstname          1000 non-null   object
 2   surname            968 non-null    object
 3   born               999 non-null    object
 4   died               1000 non-null   object
 5   bornCountry        969 non-null    object
 6   bornCountryCode    969 non-null    object
 7   bornCity           966 non-null    object
 8   diedCountry        653 non-null    object
 9   diedCountryCode    653 non-null    object
 10  diedCity           647 non-null    object
 11  gender             1000 non-null   object
 12  year               1000 non-null   int64 
 13  category           1000 non-null   object
 14  overallMotivation  23 non-null     object
 15  share              1000 non-null   int64 
 16  motivation         1000 non-null   object
 17  name               736 non-null    object
 18  city               731 non-null    object
 19  country            733 non-null    object
dtypes: int64(3), object(17)
memory usage: 156.4+ KB
```

We can use the dataframe `head()` or `tail()` method to view some actual entries. Without a numeric parameter, both methods return 5 entries!

Run the next 2 code cells below to see the first 10 and the last 10 entries

```
df.head(10)
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	bornCity
0	1	Wilhelm Conrad	Röntgen	1845-03-27	1923-02-10	Prussia (now Germany)	DE	Lennep (now Remscheid)
1	2	Hendrik A.	Lorentz	1853-07-18	1928-02-04	the Netherlands	NL	Arnhem
2	3	Pieter	Zeeman	1865-05-25	1943-10-09	the Netherlands	NL	Zonnemaire
3	4	Henri	Becquerel	1852-12-15	1908-08-25	France	FR	Paris
4	5	Pierre	Curie	1859-05-15	1906-04-19	France	FR	Paris
5	6	Marie	Curie	1867-11-07	1934-07-04	Russian Empire (now Poland)	PL	Warsaw
6	6	Marie	Curie	1867-11-07	1934-07-04	Russian Empire (now Poland)	PL	Warsaw
7	8	Lord	Rayleigh	1842-11-12	1919-06-30	United Kingdom	GB	Langford Grove Maldon Essex
8	9	Philipp	Lenard	1862-06-07	1947-05-20	Hungary (now Slovakia)	SK	Pressburg (now Bratislava)
9	10	J.J.	Thomson	1856-12-18	1940-08-30	United Kingdom	GB	Cheetham Hill

Next steps: Generate code with df View recommended plots

```
df.tail(10)
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	borr
990	1025	Drew	Weissman	1959-09-07	0000-00-00	USA	US	Lex
991	1026	Pierre	Agostini	1941-07-23	0000-00-00	French protectorate of Tunisia (now Tunisia)	TN	
992	1027	Ferenc	Krausz	1962-05-17	0000-00-00	Hungary	HU	
993	1028	Anne	L'Huillier	1958-08-16	0000-00-00	France	FR	
994	1029	Moungi	Bawendi	1961-00-00	0000-00-00	France	FR	
995	1030	Louis	Brus	1943-00-00	0000-00-00	USA	US	Clev
996	1031	Aleksey	Yekimov	1945-00-00	0000-00-00	USSR (now Russia)	RU	
997	1032	Jon	Fosse	1959-09-29	0000-00-00	Norway	NO	Haug
998	1033	Narges	Mohammadi	1972-04-21	0000-00-00	Iran	IR	Z
999	1034	Claudia	Goldin	1946-00-00	0000-00-00	USA	US	Nev

Both `head()` and `tail()` and indeed select statements will return the full set of attributes for the entries. How can we project to just a select set of attributes? This will be the equivalent of an SQL project operator listing the output columns.

Run the code cell below to show only 'firstname','surname','city' and 'country' columns for the first 10 entries.

```
df.head(10)[['firstname','surname','city','country']]
```

	firstname	surname	city	country
0	Wilhelm Conrad	Röntgen	Munich	Germany
1	Hendrik A.	Lorentz	Leiden	the Netherlands
2	Pieter	Zeeman	Amsterdam	the Netherlands
3	Henri	Becquerel	Paris	France
4	Pierre	Curie	Paris	France
5	Marie	Curie	NaN	NaN
6	Marie	Curie	Paris	France
7	Lord	Rayleigh	London	United Kingdom
8	Philipp	Lenard	Kiel	Germany
9	J.J.	Thomson	Cambridge	United Kingdom

Next steps: Generate code with df View recommended plots

Following the example of the last code cell, write a code cell below to return the last 10 records of nobel laureates but only showing their firstname, surname, year of birth, year of death and country

```
# Add your code below this line
df.head(10)[['firstname', 'surname', 'born', 'died', 'country']]
```

	firstname	surname	born	died	country
0	Wilhelm Conrad	Röntgen	1845-03-27	1923-02-10	Germany
1	Hendrik A.	Lorentz	1853-07-18	1928-02-04	the Netherlands
2	Pieter	Zeeman	1865-05-25	1943-10-09	the Netherlands
3	Henri	Becquerel	1852-12-15	1908-08-25	France
4	Pierre	Curie	1859-05-15	1906-04-19	France
5	Marie	Curie	1867-11-07	1934-07-04	NaN
6	Marie	Curie	1867-11-07	1934-07-04	France
7	Lord	Rayleigh	1842-11-12	1919-06-30	United Kingdom
8	Philipp	Lenard	1862-06-07	1947-05-20	Germany
9	J.J.	Thomson	1856-12-18	1940-08-30	United Kingdom

Next steps: Generate code with df View recommended plots

Querying & Locating Data in the DataFrame

One of the most useful tasks in pandas is locating data that satisfies desired criteria. For example, we can locate a Nobel laureate with a particular surname. Let's look at the record of Caltech's most beloved figure, physicist Richard Feynman (pronounced "FINE-men"). In addition to his groundbreaking work in theoretical physics (especially quantum electrodynamics and its associated Feynman diagrams), Feynman is known for The Feynman Lectures on Physics, which covers the elementary physics curriculum (mechanics, thermal physics, electrodynamics, etc.) in an unusually entertaining and insightful way. Let's use square brackets and a boolean criterion on the "surname" column to find Feynman's record in the laureates da.

What can you conclude from the format of this query below and results produce?a

```
df[df['surname'] == 'Feynman']
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	bornCity
86	86	Richard P.	Feynman	1918-05-11	1988-02-15	USA	US	New York NY

The DataFrame object can take a Boolean condition on columns in it's index and will return records that meet that condition!. The inner part of the syntax for query above returns a Series consisting of boolean values for every laureate, with True if the surname is equal to "Feynman" and

False otherwise.

By using the correct index (i.e., 86), we can confirm that the value in that case is True.

Run the code cell below to confirm only this index returns True.

```
(df["surname"] == "Feynman")[86]
```

```
True
```

The `loc` attribute can be used in place of brackets in many places and is generally a more flexible way to pull out data items of interest. Let's use the loc attribute to retrieve the year when Feyman won.

Run the code cell below for this result.

```
df.loc[df["surname"] == "Feynman", "year"]
```

```
86    1965
Name: year, dtype: int64
```

Use the `loc` attribute illustrated above to code a query on this dataset to:

Find all Nobel laureates named named 'Curie'
Tip: Use the Boolean condition: df["surname"].str.contains("Curie", na=False)

```
# Code and test your scriplet to find all 'Curies'
# Assigned the results to a variable named curies.
curies = df.loc[df['surname'].str.contains("Curie", na=False)]
#View output
curies
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	bornCit
4	5	Pierre	Curie	1859-05-15	1906-04-19	France	FR	Pari
5	6	Marie	Curie	1867-11-07	1934-07-04	Russian Empire (now Poland)	PL	Warsaw
6	6	Marie	Curie	1867-11-07	1934-07-04	Russian Empire (now Poland)	PL	Warsaw
191	194	Irène	Joliot-Curie	1897-09-12	1956-03-17	France	FR	Pari

Let's **find all the winners of multiple nobel prizes** using the `groupby` method

```
laureates = df.groupby(["id", "firstname", "surname"])
sizes = laureates.size()
sizes[sizes > 1]       # result should show 5 winners
```

```
id   firstname  surname
6    Marie      Curie        2
66   John       Bardeen      2
217  Linus      Pauling      2
222  Frederick  Sanger       2
743  Barry      Sharpless    2
dtype: int64
```

Selecting Dates & Time information Pandas provides good support fordatetimee. Let’s search for laureates by exact birthday as a strin - the default pandas storage format for dates.

Run the code cell below to get the result entry for Eistein born 3/14/1879 fondly known as Pi Day.

```
df.loc[df['born'] == '1879-03-14']
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	bornCity
25	26	Albert	Einstein	1879-03-14	1955-04-18	Germany	DE	Ulm

Like the query above let's see if there are any laureates born 6/28 known as Tau Day

Run the code cell below to confirm!

```
df.loc[df['born'].str.contains('06-08', na=False)]
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	born
120	121	Kenneth G.	Wilson	1936-06-08	2013-06-15	USA	US	Walthan
370	372	Francis	Crick	1916-06-08	2004-07-28	United Kingdom	GB	Northam
452	454	Eric F.	Wieschaus	1947-06-08	0000-00-00	USA	US	South I
667	683	Wassily	Leontief	1906-08-05	1999-02-05	Russia	RU	Peters
781	799	Robert J.	Aumann	1930-06-08	0000-00-00	Germany	DE	Frankfur the-
918	944	Jacques	Dubochet	1942-06-08	0000-00-00	Switzerland	CH	

Rewrite the query above to filter the result to just laureates in Physics born on Tau Day ... Tip: use (df['category'] == "physics") as the other part of your Boolean function!

```
# Your code for the Physics category Tau Day Nobel Laureates below
tau_day_physics_curies = df.loc[
    (df['category'] == "physics") &
    (df['born'].str.contains('06-28', na=False))
]

tau_day_physics_curies
```

	id	firstname	surname	born	died	bornCountry	bornCountryCode	bornCity	diedCountry	diedCountryCode	diedCity	g
79	79	Maria	Goeppert Mayer	1906-06-28	1972-02-20	Germany (now Poland)	PL	Kattowitz (now Katowice)	USA	US	San Diego CA	
125	126	Klaus	von Klitzing	1943-06-28	0000-00-00	German-occupied Poland (now Poland)	PL	Schroda	NaN	NaN	NaN	

```
Start coding or generate with AI.
```