Importing data into pandas

There are tons of ways you can get data into a pandas dataframe. Here are a few of the more common ones.

First, let's import pandas `as` pd.

In [12]:
```
import pandas as pd
```

From a CSV file

If your data file is delimited with something other than a comma, you'll need to specify that in the `sep` argument. For example, if you had a pipe-delimited file: `pd.read_csv('../data/my-delimited-file.txt', sep='|')`

Let's read in the MLB salary data.

In [13]:
```
csv_df = pd.read_csv('../data/mlb.csv')
```

In [14]:
```
csv_df.head()
```

Out[14]:

	NAME	TEAM	POS	SALARY	START_YEAR	END_YEAR	YEARS
0	Clayton Kershaw	LAD	SP	33000000	2014	2020	7
1	Zack Greinke	ARI	SP	31876966	2016	2021	6
2	David Price	BOS	SP	30000000	2016	2022	7
3	Miguel Cabrera	DET	1B	28000000	2014	2023	10
4	Justin Verlander	DET	SP	28000000	2013	2019	7

From a CSV file on the Internet

Just pass in the URL. This example uses animal complaints in Townsville.

The values that get returned aren't live -- like, if the results changed, your data frame would not update with new values. It reads in the data once.

In [16]:
```
csvi_df = pd.read_csv('https://data.gov.au/data/dataset/5a005841-f4f2-4c52-82db-8cce70715c
```

In [17]:
```
csvi_df.head()
```

Out[17]:

	Animal Type	Complaint Type	Date Received	Suburb	Electoral Division
0	dog	Aggressive Animal	June 2021	Alice River	Division 1
1	dog	Aggressive Animal	June 2021	Alice River	Division 1
2	dog	Aggressive Animal	June 2021	Alice River	Division 1
3	dog	Aggressive Animal	June 2021	Alice River	Division 1

	Animal Type	Complaint Type	Date Received	Suburb	Electoral Division
4	dog	Attack	June 2021	Alice River	Division 1

From an Excel file

To read an Excel file in pandas, use the `read_excel()` method. If you hadn't installed the `openpyxl` module, you'd need to do that, as well. (We've already done so here.)

You might also want to specify the `sheet_name` to select your worksheet of interest -- the default is "the first one."

Here, we're reading in a spreadsheet with data on murders in large cities.

In [18]:
```
xl_df = pd.read_excel('../data/Homicide2014.xlsx', sheet_name='Murders')
```

In [19]:
```
xl_df.head()
```

Out[19]:

	City	State	Population	Murders
0	New York City	NY	8473938	333
1	Los Angeles	CA	3906772	260
2	Chicago	IL	2724121	415
3	Houston	TX	2219933	242
4	Philadelphia	PA	1559062	248

From a Python data collection

Maybe the work you're doing in pandas happens downstream of some other Python processing, so the data exists as a native Python data collection -- say, a list of dictionaries. You can turn this (and other Python data collections, like a list of lists) into a pandas dataframe, too.

👉 For more details on Python data collections, see this notebook.

In [20]:
```
test_data = [
    {'name': 'Cody Winchester', 'job': 'Training director', 'location': 'Spearfish, SD'},
    {'name': 'Jacob Sanders', 'job': 'Developer', 'location': 'Pittsburgh, PA'},
    {'name': 'Guy Fieri', 'job': 'Gourmand', 'location': 'Flavortown'},
]

py_df = pd.DataFrame(test_data)
```

In [21]:
```
py_df.head()
```

Out[21]:

	name	job	location
0	Cody Winchester	Training director	Spearfish, SD
1	Jacob Sanders	Developer	Pittsburgh, PA
2	Guy Fieri	Gourmand	Flavortown

From an HTML table

OK SO.

This one requires you to install and specify the Python package that has the HTML parsing engine of your choice -- BeautifulSoup or lxml. The default is `lxml`, but here we're going to use BeautifulSoup.

Huge caveat! Pulling data directly from an HTML table can be hit and miss, depending on how hairy the underlying HTML is. And if you want to scrape data from a website, it's usually better practice to save the results to a local file, *then* load it up for analysis. But it's good to know that it's an option.

In this example, we've installed `BeautifulSoup` (alias `bs4`) and we're going to import a table of media witnesses to Texas death row executions.

We're going to pass four things to the pandas `read_html()` method:

1. The URL we want to scrape (in quotes, as a string)
2. The `flavor` of parser that we'd like to use to process the HTML (`bs4`)
3. The HTML attributes of the table we're targeting (in this case, the table has a `class` called `tdcj_table`)
4. The number of the list, in the list of lists that gets returned in a dataframe, that has the `header`? (Usually it's 0 -- the first one)

Reading through the documentation for this method, we also notice that this method returns a *list* of matching tables as dataframes, so we need to grab the *first* item in this list of tables returned. Our arguments were specific enough that there's only one item in the returned list, though, so we can just grab the first item with `[0]`.

👉For more details on selecting items from Python lists, see this notebook.

In [22]:
```
html_df = pd.read_html('http://www.tdcj.state.tx.us/death_row/dr_media_witness_list.html',
                       flavor='bs4',
                       attrs={'class': 'tdcj_table'},
                       header=0)[0]
```

In [23]:
```
html_df.head()
```

Out[23]:

	Execution	Link	Last Name	First Name	TDCJ Number	Date	Media Witness List
0	572	Inmate Information	Hummel	John	999567	6/30/2021	Michael Graczyk, Associated Press; Joseph Brow...
1	571	Inmate Information	Jones	Quintin	999379	5/19/2021	No media witnesses present.
2	570	Inmate Information	Wardlow	Billy	999137	7/8/2020	Michael Graczyk, Associated Press; Joseph Brow...
3	569	Inmate Information	Ochoa	Abel	999450	2/6/2020	Michael Graczyk, Associated Press; Joseph Brow...
4	568	Inmate Information	Gardner	John	999516	1/15/2020	Michael Graczyk, Associated Press; Joseph Brow...