Chapter 1

Introduction to Python for data analysis

Objectives

Applied

1. Write Python statements that

   import a module

   call and chain methods

   use slices, lists, tuples, dictionaries, and list comprehensions

2. Use JupyterLab to

   open an existing Notebook or to start a new Notebook

   edit and run the cells of a Notebook

   create and modify the headings in a Notebook

3. Use JupyterLab’s Tab completion and tooltip features.

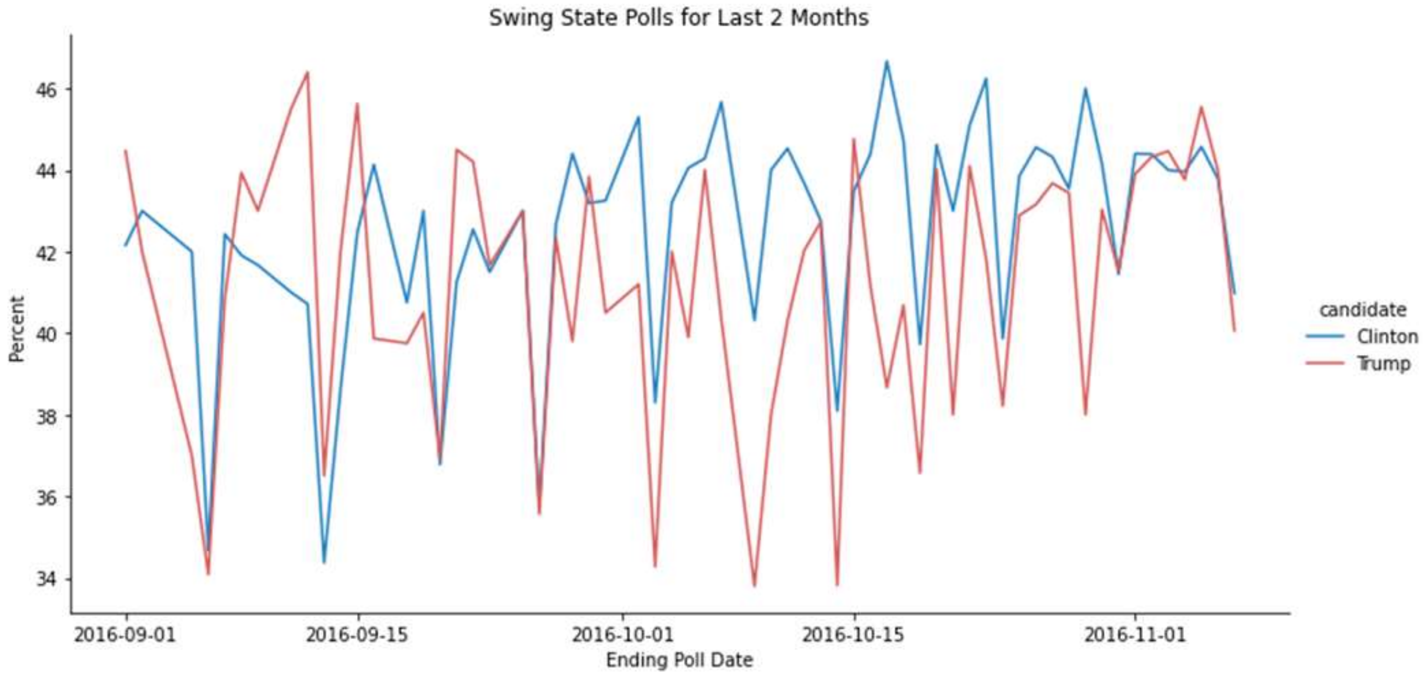
4. Use JupyterLab’s Magic Commands to time statements and display the current variables.

Objectives (continued)

Knowledge

1. Describe data analysis and data visualization.
2. List the five phases of data analysis and visualization.
3. Describe the Pandas and Seaborn modules that are in the Anaconda distribution.
4. Distinguish between runtime errors and syntax errors.

Data visualization often provides the best insights into the data



What data analysis includes

- Data analysis
- Data visualization (data viz)
- Data modeling (predictive analysis)

Related terms

- Data analytics
- Business analytics
- Sports analytics

What to do before starting an analysis

Set your goals

- The *goals of analysis* can be well-defined, like trying to answer specific questions, or more general, like trying to extract useful information from large volumes of data.

Define your target audience

- If you’re going to present your findings to other people like managers or clients, you also need to define your *target audience* before you start your analysis.

The five phases of data analysis and visualization

Get the data

- Find the data on a website or in one of your company’s databases or spreadsheets.
- Read the data into a DataFrame or build a DataFrame from the data.

Clean the data

- Remove unnecessary rows and columns.
- Handle invalid or missing values.
- Change object data types to datetime or numeric data types.

The five phases of data analysis and visualization (continued)

Prepare the data

- Add columns that are derived from other columns.
- Shape the data into the forms that are needed for your analysis.
- Make preliminary visualizations to better understand the data.

Analyze the data

- Get new views of the data by grouping and aggregating.
- Make visualizations that provide insights and show relationships.
- Model the data as part of predictive analysis.

Visualize the data

- Enhance your visualizations so they're appropriate for your target audience.

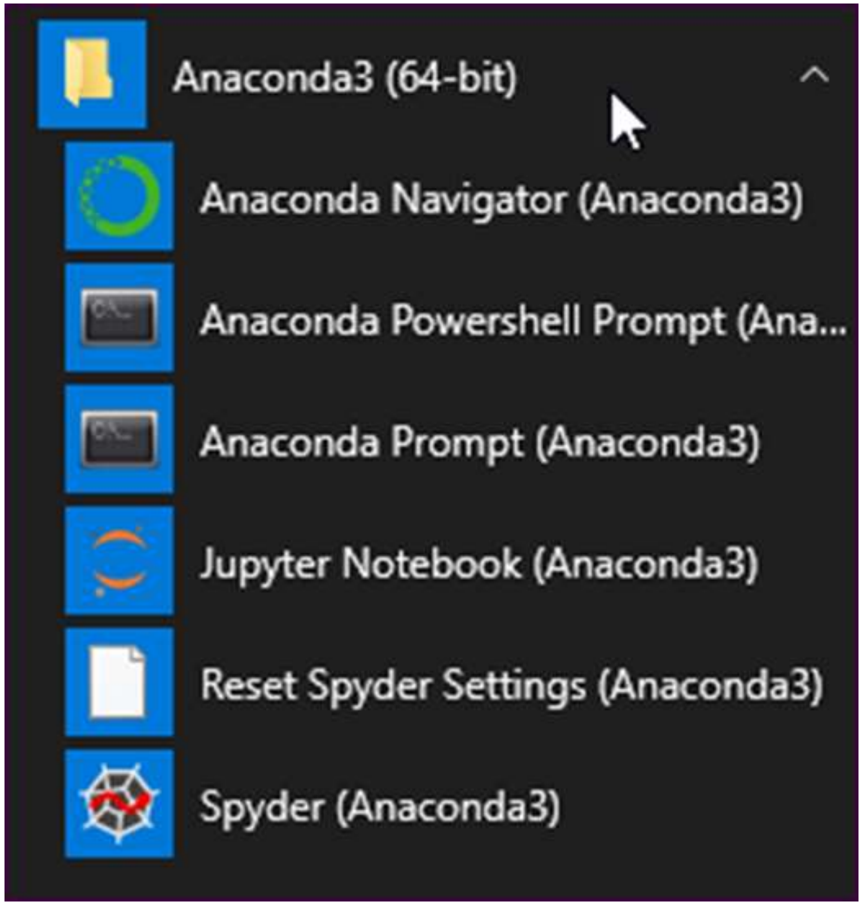
Three of the IDEs for Python data analysis

- Jupyter Notebook
- JupyterLab
- VS Code

Our recommendations for Python distributions and IDEs

- Use the Anaconda distribution of Python.
- Use JupyterLab as your IDE.

The programs that are installed by the Anaconda distribution


Anaconda3 (64-bit)
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Ana...
Anaconda Prompt (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)
Spyder (Anaconda3)

Modules included with the Anaconda distribution

Module	Abbreviation	Provides methods for
`pandas`	`pd`	Data analysis and visualization
`numpy`	`np`	Numerical computing
`seaborn`	`sns`	Data visualization
`datetime`	`dt`	Working with datetime objects
`urllib`		Getting files from the web
`zipfile`		Working with zip files
`sqlite3`		Working with a SQLite database
`json`		Working with JSON data
`sklearn`		Regression analysis

The modules you need to install for this book

Module	Chapter	Provides methods for
`pyreadstat`	5	Reading Stata files
`geopandas`	12	Plotting geographic data

Two ways to install a module

Use the conda command from the Anaconda prompt

```
conda install pandas --yes
```

Use the conda command with a different channel

```
conda install --channel conda-forge pyreadstat --yes
```

How to import modules

How to import one module into the namespace specified by the as clause

```
import pandas as pd
```

How to import one submodule from a module

```
from urllib import request
```

How to call methods

How to call a method in a module

```
import pandas as pd
polls_url = \
    'http://projects.fivethirtyeight.com/.../president_general_polls_2016.csv'
polls = pd.read_csv(poll_url)
```

How to call a method from a DataFrame object

```
polls.sort_values('startdate')
```

How to chain methods

How to chain the sort_values() and head() methods

```
polls.sort_values('startdate').head()
```

How to chain the query() and plot() methods

```
polls.query('state != "U.S."') \
    .plot(x='startdate', y=['Clinton_pct','Trump_pct'])
```

How to call a method with positional and keyword parameters

The signature for the sort_values() method

```
sort_values(by, axis=0, ascending=True, inplace=False,
                kind='quicksort', na_position='last')
```

The sort_values() method with positional and keyword parameters

```
polls.sort_values('startdate', ascending=False, inplace=True)
```

The syntax for coding lists, slices, tuples, and dictionary objects

A list is a sequence of items within brackets

```
[item1,item2,...]
```

A tuple is coded like a list but in parentheses

```
(item1,item2,...)
```

A dictionary is a sequence of key/value pairs within braces

```
{key1:value1, key2:value2, ...}
```

A slice sets the start and stop values and an optional step value

```
start:stop:step
```

How to use lists, slices, tuples, and dictionary objects

A list used as a keyword parameter

```
polls.drop(columns=['cycle','branch','matchup','forecastdate'],
    inplace=True)
```

A tuple used as a keyword parameter

```
polls.plot.line(xlim=('2016-06','2016-11'))
```

A dictionary used as a keyword parameter

```
polls.rename(columns={'adjpoll_clinton':'Clinton',
                     'adjpoll_trump':'Trump'})
```

Two slices used in a loc[ ] accessor

```
polls.loc[0:100:10,'state':'grade']
```

How to code a list comprehension

The syntax

```
[expression for member in iterable]
```

A list comprehension used to provide the list for a keyword parameter

```
xticks = [x for x in range(1900,1920,2)]
```

The resulting list

```
[1900, 1902, 1904, 1906, 1908, 1910, 1912, 1914, 1916, 1918]
```

Two ways to continue a statement

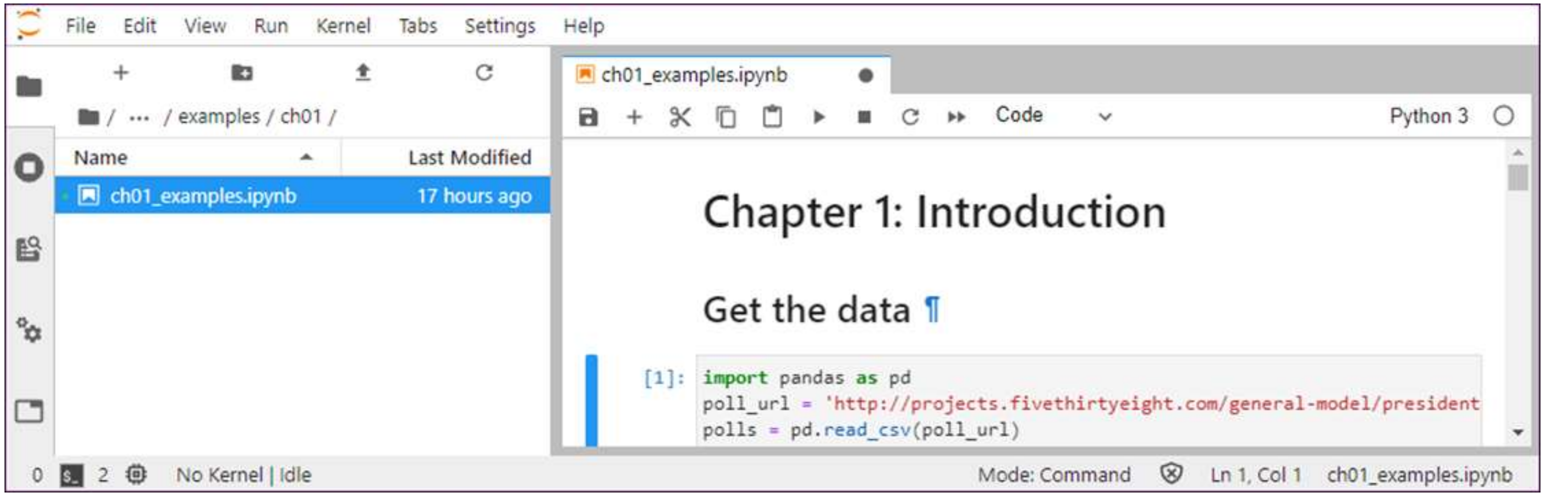
With implicit continuation

```
polls.sort_values(
    ['state','startdate'],
    ascending=False,
    inplace=True)
```

With explicit continuation

```
polls.sort_values(['state','startdate'], \
                  ascending=False, \
                  inplace=True)
```

One Notebook in JupyterLab with the File Browser open

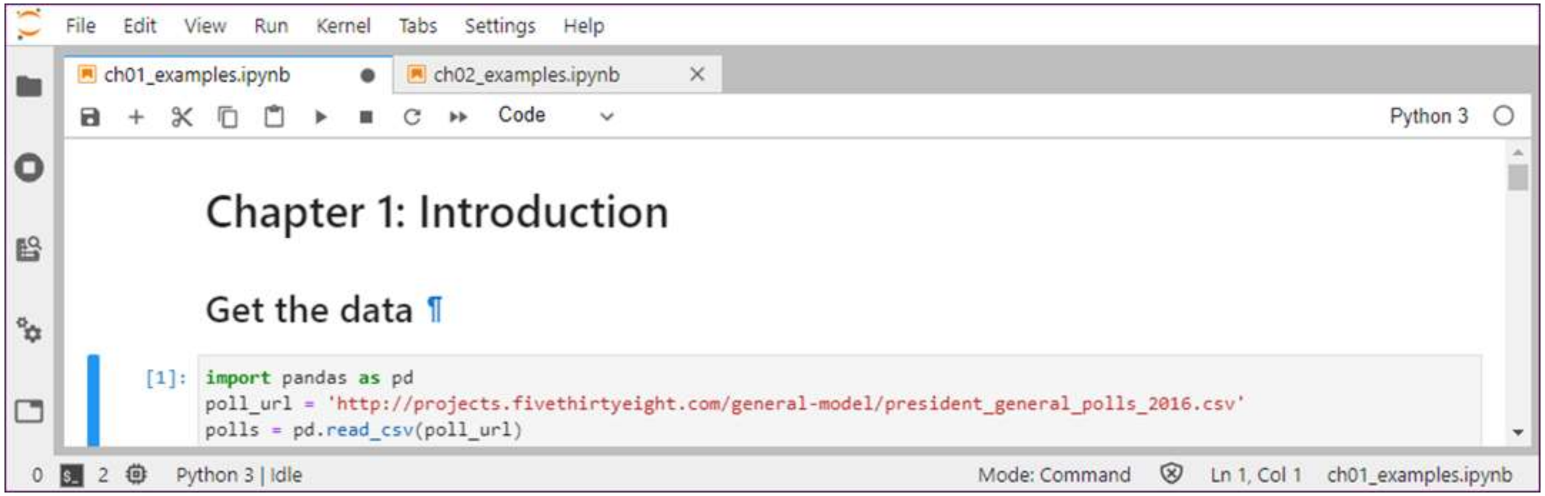


Chapter 1: Introduction

Get the data ¶

```
[1]: import pandas as pd
     poll_url = 'http://projects.fivethirtyeight.com/general-model/president
     polls = pd.read_csv(poll_url)
```

Two Notebooks in JupyterLab with the File Browser closed



Chapter 1: Introduction

Get the data ¶

```
[1]: import pandas as pd
     poll_url = 'http://projects.fivethirtyeight.com/general-model/president_general_polls_2016.csv'
     polls = pd.read_csv(poll_url)
```

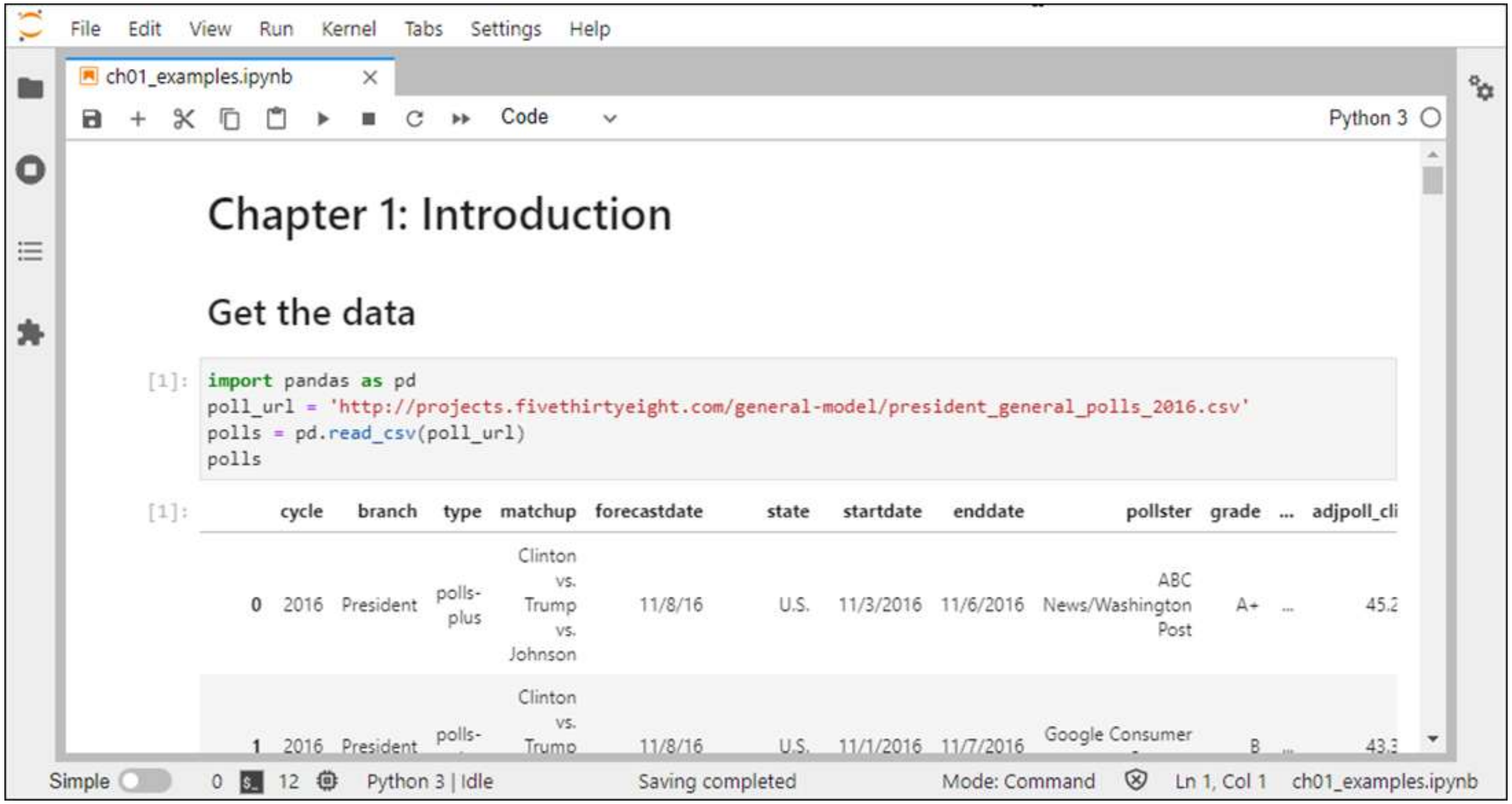
How to start JupyterLab

- Use the Start menu to start the Anaconda Navigator, and then launch JupyterLab.

How to work with Notebooks

- To open or close the File Browser, click the File Browser icon in the upper left corner.
- To open a Notebook, browse to the file you want to open and double-click on it.
- To start a new Notebook, select File→New Launcher to open a Launcher tab. Then, click the Python 3 icon.
- To save, close, or rename a Notebook, use the File menu. To save the active Notebook, click on the Save icon in the toolbar for the tab.
- To restore a Notebook to the last checkpoint, select File→Revert Notebook to Checkpoint.

A cell and its output when the cell is run



Chapter 1: Introduction

Get the data

```
[1]: import pandas as pd
     poll_url = 'http://projects.fivethirtyeight.com/general-model/president_general_polls_2016.csv'
     polls = pd.read_csv(poll_url)
     polls
```

[1]:

	cycle	branch	type	matchup	forecastdate	state	startdate	enddate	pollster	grade	...	adjpoll_cli
0	2016	President	polls-plus	Clinton vs. Trump vs. Johnson	11/8/16	U.S.	11/3/2016	11/6/2016	ABC News/Washington Post	A+	...	45.2
1	2016	President	polls-	Clinton vs. Trump	11/8/16	U.S.	11/1/2016	11/7/2016	Google Consumer	B	...	43.3

Simple 0 12 Python 3 | Idle Saving completed Mode: Command Ln 1, Col 1 ch01_examples.ipynb

How to select one or more cells

- To select one cell, position the pointer in the left margin of the cell so it becomes a crosshair, and then click so a blue line is displayed.
- To select more than one cell, select the first cell, hold down the Shift key, and select the last cell.

How to copy, delete, merge, or move the selected cells

- Use the buttons in the toolbar or the items in the Edit or shortcut menu.

How to add a cell after the current cell

- Use the + button in the toolbar.

How to run the code in one cell

- Press Shift+Enter or click the Run button in the toolbar.

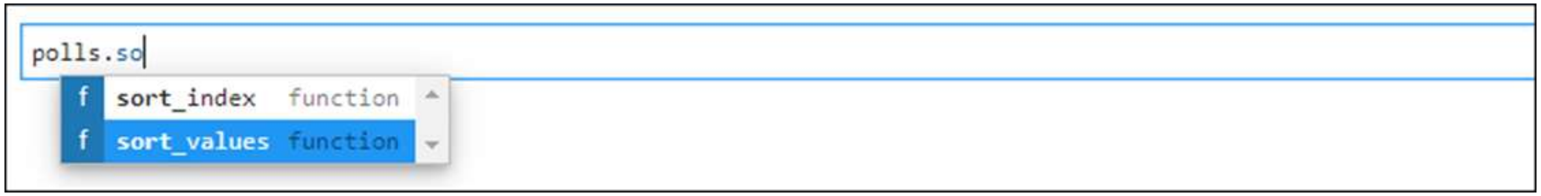
How to run the code in selected cells or all cells

- Use the Run button in the toolbar or the items in the Run menu.

How to interrupt, restart, or shutdown the kernel

- Use the items in the Kernel menu.

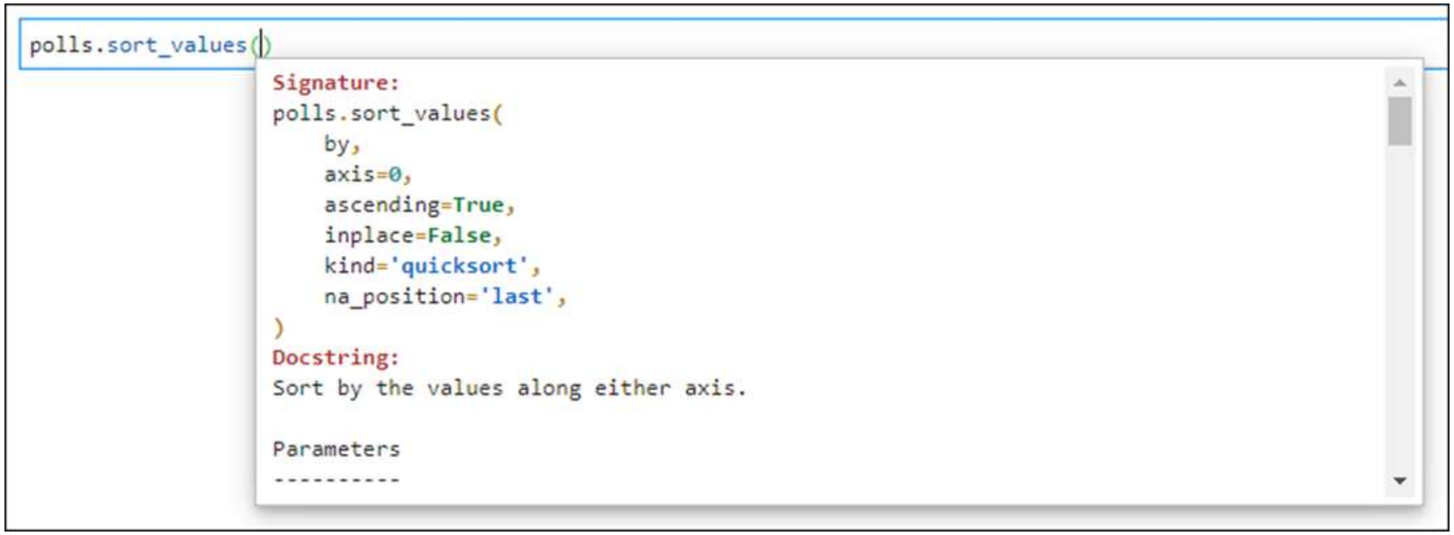
The Tab completion feature is activated when you press the Tab key



```
polls.so
```

The tooltip feature is activated when you press the Shift+Tab key

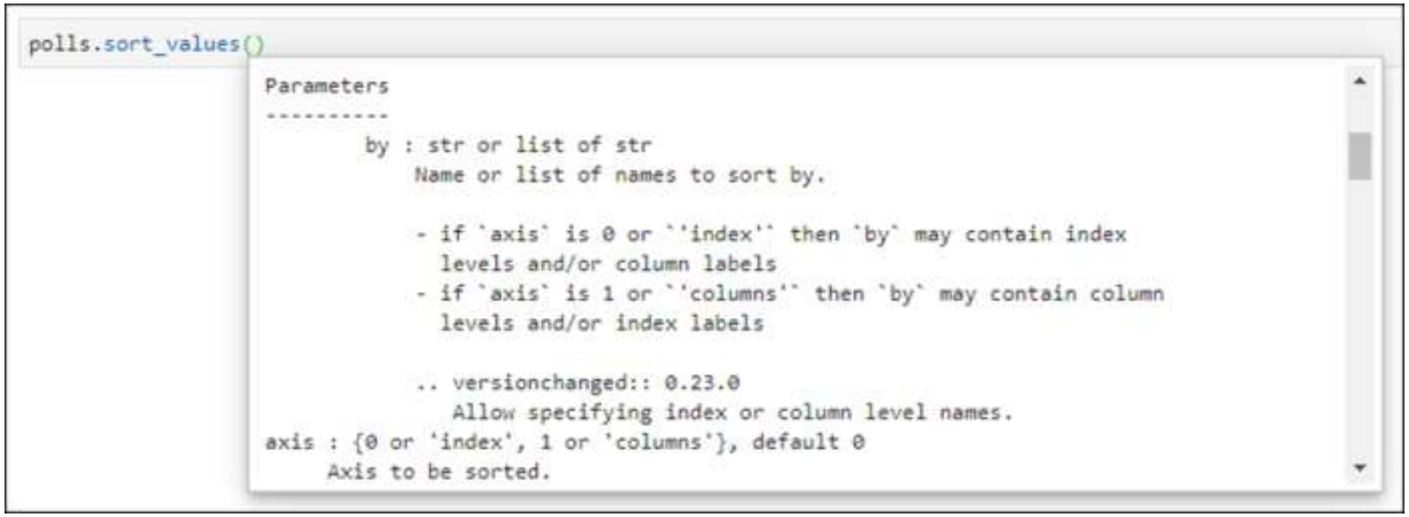
The start of the tooltip for the sort_values() method



```
polls.sort_values()
Signature:
polls.sort_values(
    by,
    axis=0,
    ascending=True,
    inplace=False,
    kind='quicksort',
    na_position='last',
)
Docstring:
Sort by the values along either axis.

Parameters
----------
```

The tooltip feature (continued)

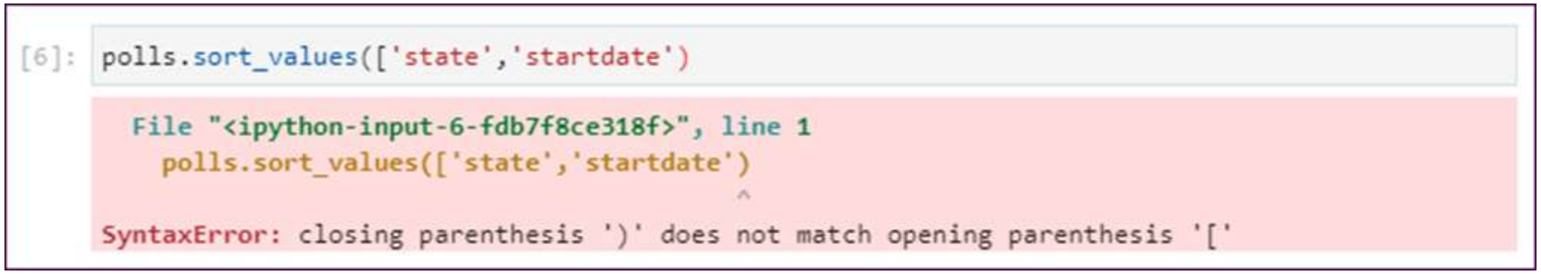
More of the tooltip after scrolling down to the start of the parameters



```
polls.sort_values()
```

```
Parameters
----------
        by : str or list of str
            Name or list of names to sort by.

            - if `axis` is 0 or `'index'` then `by` may contain index
              levels and/or column labels
            - if `axis` is 1 or `'columns'` then `by` may contain column
              levels and/or index labels

            .. versionchanged:: 0.23.0
               Allow specifying index or column level names.
axis : {0 or 'index', 1 or 'columns'}, default 0
     Axis to be sorted.
```

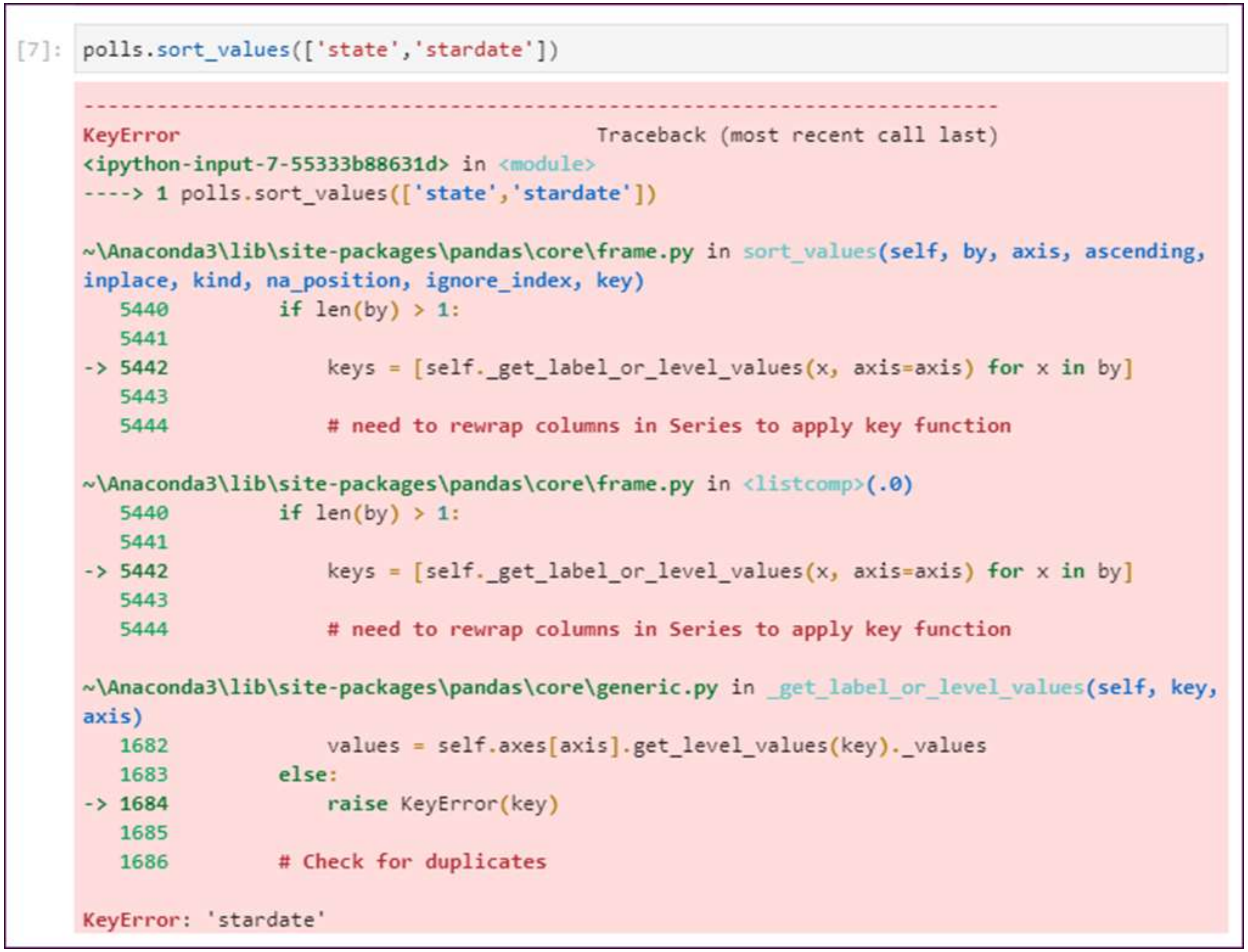
A syntax error in a Notebook



```
[6]: polls.sort_values(['state','startdate')
```

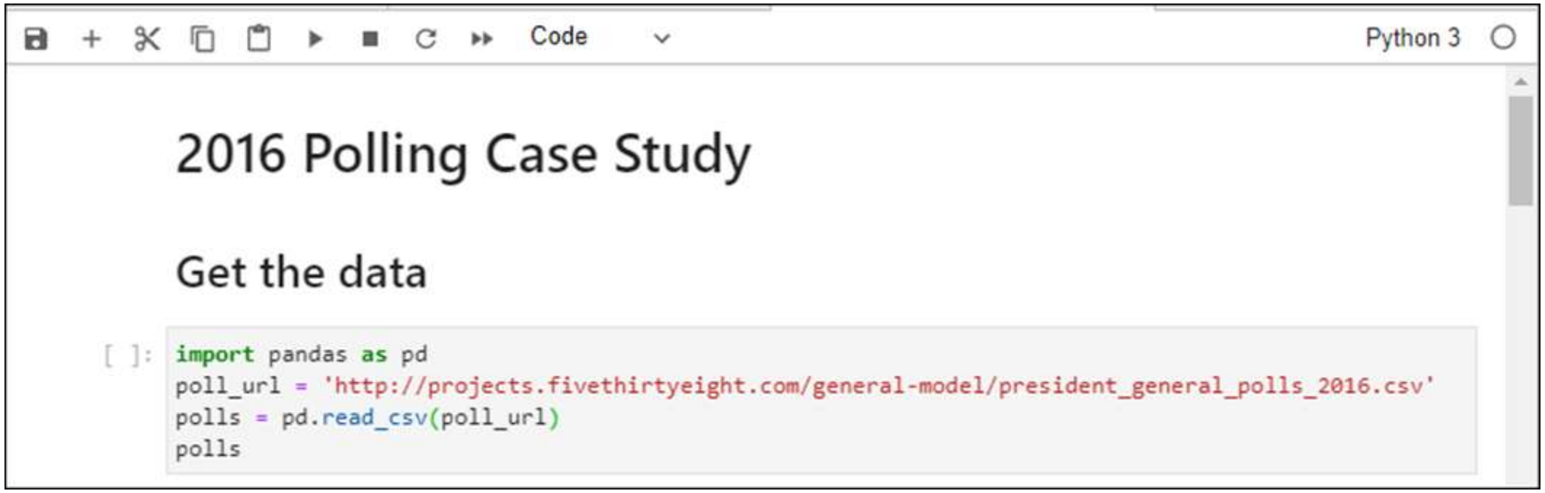
```
  File "<ipython-input-6-fdb7f8ce318f>", line 1
    polls.sort_values(['state','startdate')
                                          ^
SyntaxError: closing parenthesis ')' does not match opening parenthesis '['
```

A runtime error in a Notebook



```
[7]: polls.sort_values(['state','stardate'])

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-55333b88631d> in <module>
----> 1 polls.sort_values(['state','stardate'])

~\Anaconda3\lib\site-packages\pandas\core\frame.py in sort_values(self, by, axis, ascending,
inplace, kind, na_position, ignore_index, key)
   5440         if len(by) > 1:
   5441
-> 5442             keys = [self._get_label_or_level_values(x, axis=axis) for x in by]
   5443
   5444             # need to rewrap columns in Series to apply key function

~\Anaconda3\lib\site-packages\pandas\core\frame.py in <listcomp>(.0)
   5440         if len(by) > 1:
   5441
-> 5442             keys = [self._get_label_or_level_values(x, axis=axis) for x in by]
   5443
   5444             # need to rewrap columns in Series to apply key function

~\Anaconda3\lib\site-packages\pandas\core\generic.py in _get_label_or_level_values(self, key,
axis)
   1682             values = self.axes[axis].get_level_values(key)._values
   1683         else:
-> 1684             raise KeyError(key)
   1685
   1686         # Check for duplicates

KeyError: 'stardate'
```

A Notebook with headings

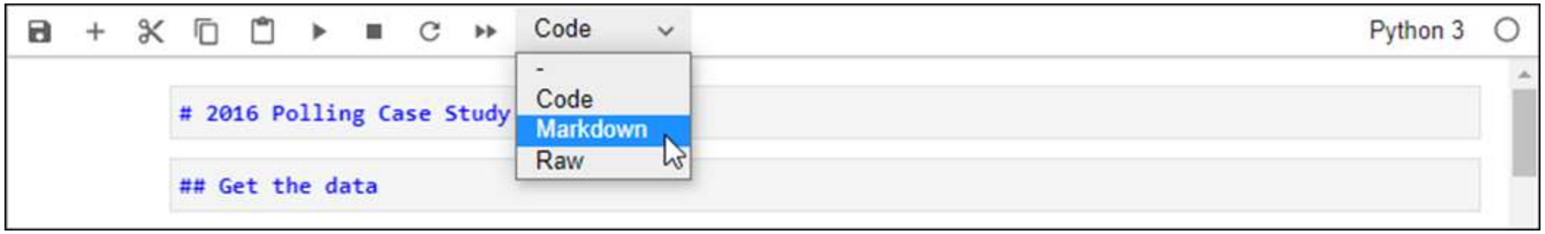


2016 Polling Case Study

Get the data

```
[ ]: import pandas as pd
     poll_url = 'http://projects.fivethirtyeight.com/general-model/president_general_polls_2016.csv'
     polls = pd.read_csv(poll_url)
     polls
```

The Markdown language for the headings



```
# 2016 Polling Case Study

## Get the data
```

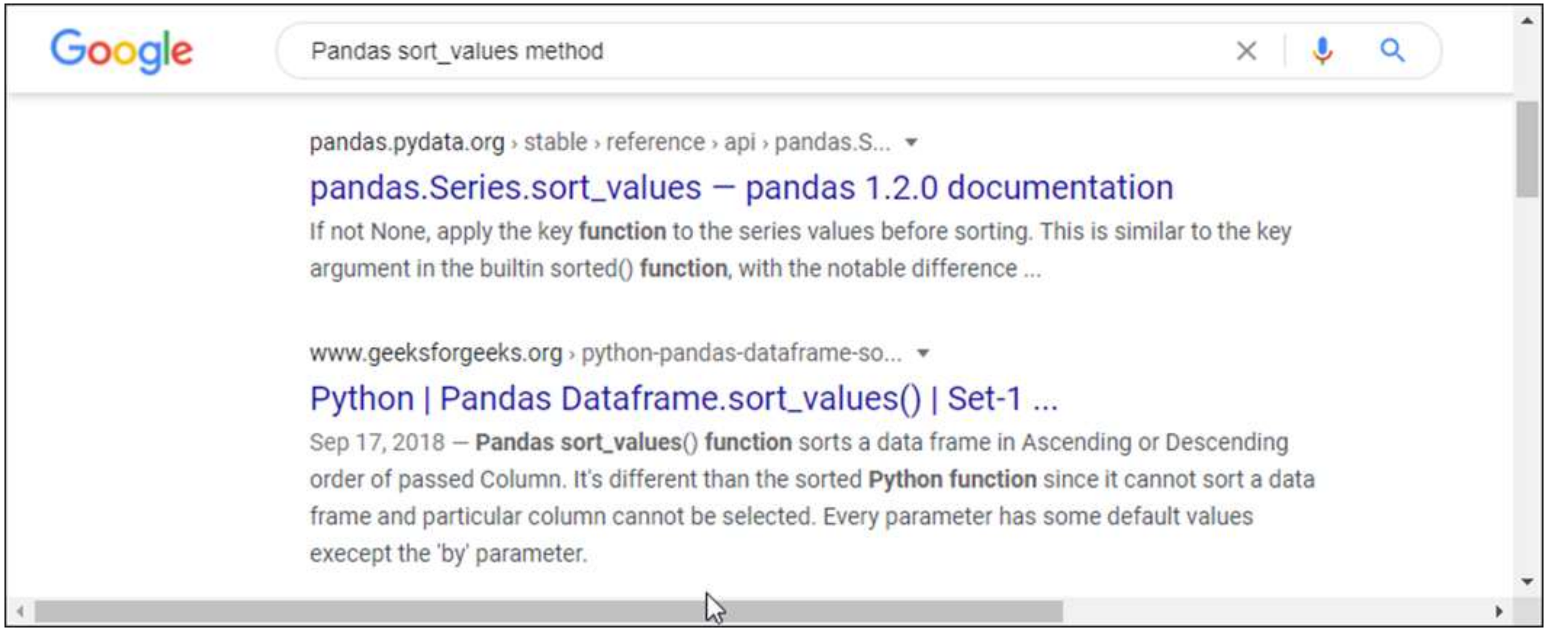
How to create a heading by using Markdown language

- With the cursor in a new cell, change the drop-down list from Code to Markdown.
- Type the text for a heading into the cell preceded by from one to five # signs. The number of signs determines the level of the heading.
- Run the cell to convert the Markdown language to the heading.

How to modify a heading in a Notebook cell

- Double-click in the cell to display the Markdown language, modify it, and run the cell.

A search for the Pandas sort_values() method



Pandas sort_values method

pandas.pydata.org › stable › reference › api › pandas.S...

pandas.Series.sort_values — pandas 1.2.0 documentation

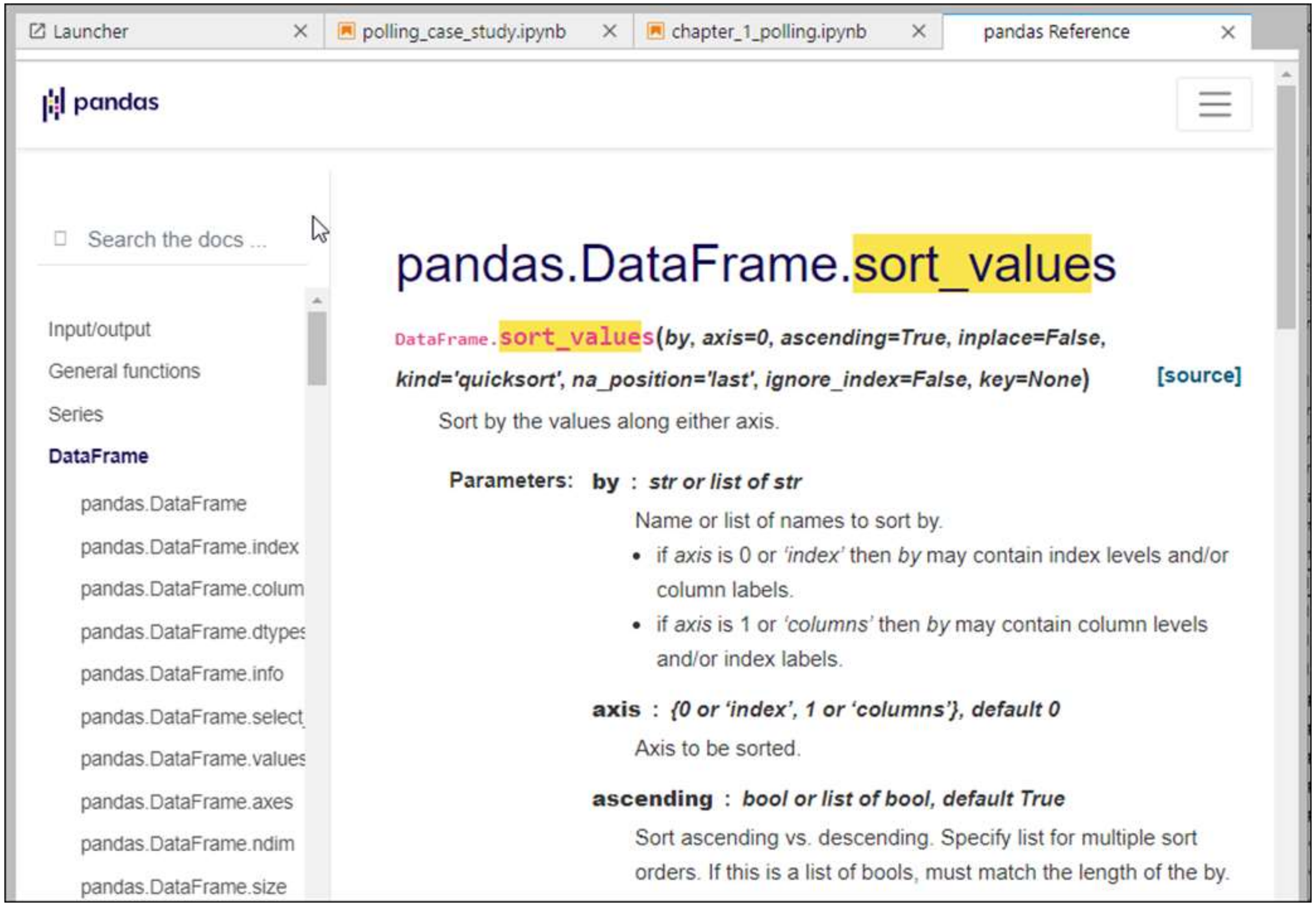
If not None, apply the key **function** to the series values before sorting. This is similar to the key argument in the builtin sorted() **function**, with the notable difference ...

www.geeksforgeeks.org › python-pandas-dataframe-so...

Python | Pandas Dataframe.sort_values() | Set-1 ...

Sep 17, 2018 — **Pandas sort_values**() **function** sorts a data frame in Ascending or Descending order of passed Column. It's different than the sorted **Python function** since it cannot sort a data frame and particular column cannot be selected. Every parameter has some default values execept the 'by' parameter.

The JupyterLab Help menu and a page in the Pandas reference



pandas.DataFrame.sort_values

`DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None)` [source]

Sort by the values along either axis.

Parameters: **by** : *str or list of str*

Name or list of names to sort by.

- if *axis* is 0 or *'index'* then *by* may contain index levels and/or column labels.
- if *axis* is 1 or *'columns'* then *by* may contain column levels and/or index labels.

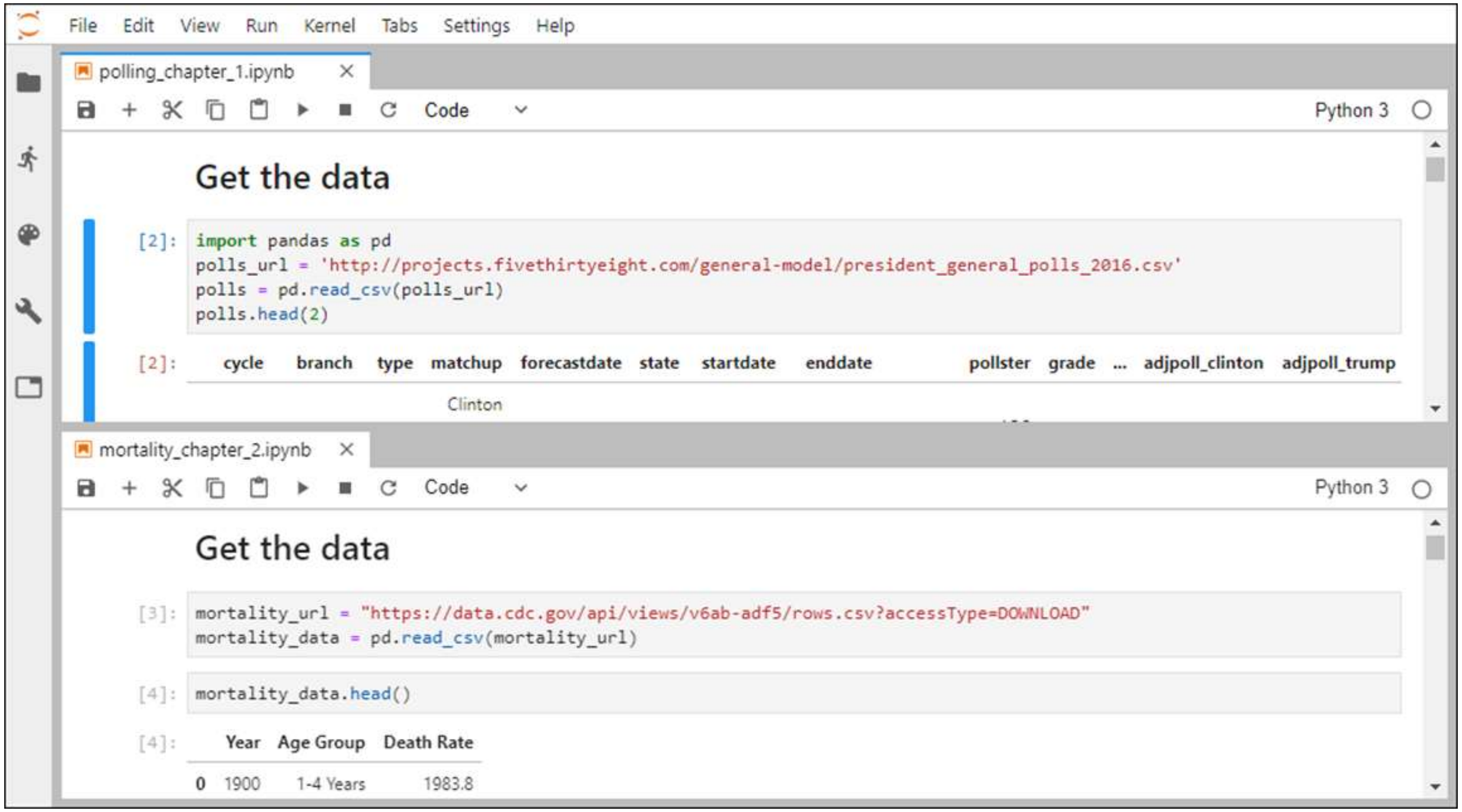
axis : *{0 or 'index', 1 or 'columns'}, default 0*

Axis to be sorted.

ascending : *bool or list of bool, default True*

Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

JupyterLab with two Notebooks in a horizontally split screen



Get the data

```
[2]: import pandas as pd
     polls_url = 'http://projects.fivethirtyeight.com/general-model/president_general_polls_2016.csv'
     polls = pd.read_csv(polls_url)
     polls.head(2)
```

```
[2]:   cycle  branch  type  matchup  forecastdate  state  startdate  enddate    pollster  grade  ...  adjpoll_clinton  adjpoll_trump
                              Clinton
```

Get the data

```
[3]: mortality_url = "https://data.cdc.gov/api/views/v6ab-adf5/rows.csv?accessType=DOWNLOAD"
     mortality_data = pd.read_csv(mortality_url)

[4]: mortality_data.head()
```

```
[4]:     Year  Age Group  Death Rate
      0  1900  1-4 Years      1983.8
```

How to split the screen

- To split the screen vertically between two open Notebooks, drag the second tab down and to the right and drop it.
- To split the screen horizontally between two open Notebooks, drag the second tab down and drop it.

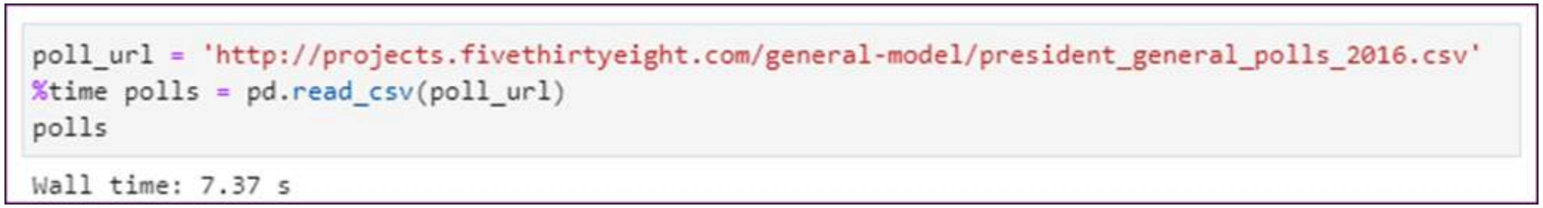
How to restore the screen

- Drag the tab of the bottom or right Notebook until it's next to the tab of the other Notebook and drop it.
- If you have trouble restoring the screen by dragging the split tab, you can close one of the Notebooks and then reopen it.

Four of the most useful Magic Commands

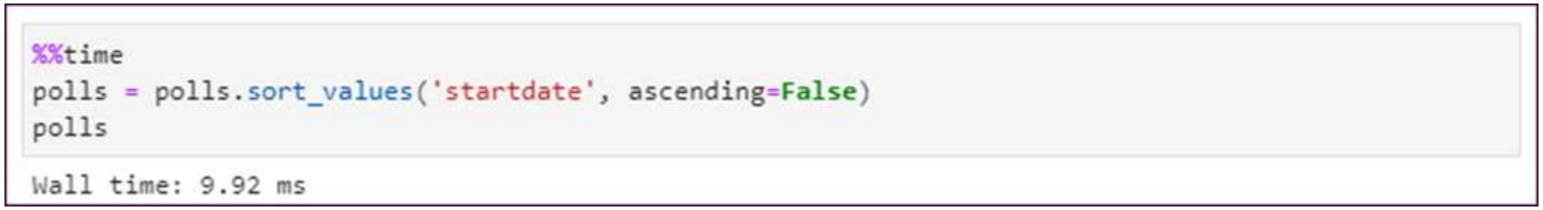
Command	Description
`%time`	Displays the time that it takes for a statement to run.
`%%time`	Displays the time that it takes for all the statements in a cell to run.
`%whos`	Displays the variables that are in the namespace along with their data types.
`%magic`	Displays a reference for all of the Magic commands.

How the %time command works



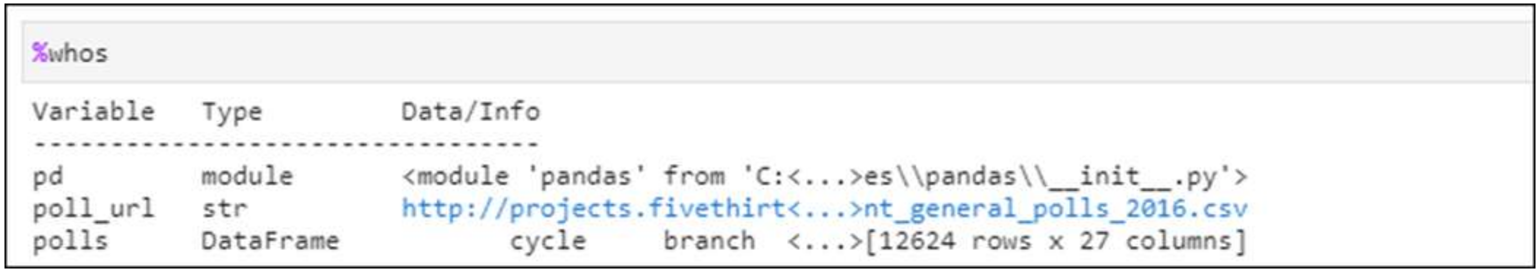
```
poll_url = 'http://projects.fivethirtyeight.com/general-model/president_general_polls_2016.csv'
%time polls = pd.read_csv(poll_url)
polls

Wall time: 7.37 s
```

How the %%time command works



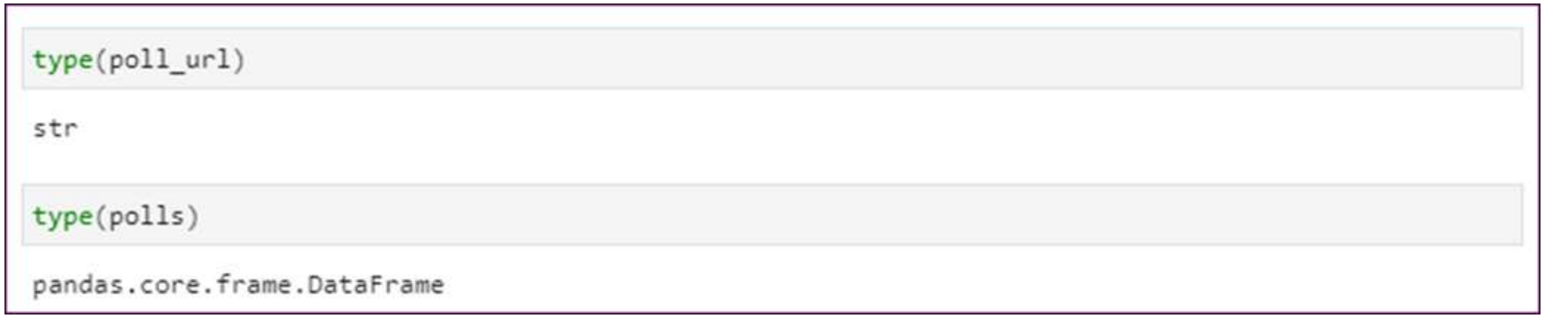
```
%%time
polls = polls.sort_values('startdate', ascending=False)
polls

Wall time: 9.92 ms
```

How the %whos command works



```
%whos

Variable   Type         Data/Info
----------------------------------
pd         module       <module 'pandas' from 'C:<...>es\\pandas\\__init__.py'>
poll_url   str          http://projects.fivethirt<...>nt_general_polls_2016.csv
polls      DataFrame          cycle      branch   <...>[12624 rows x 27 columns]
```

How to use the Python type() function to check the data type of a variable



```
type(poll_url)
```

```
str
```

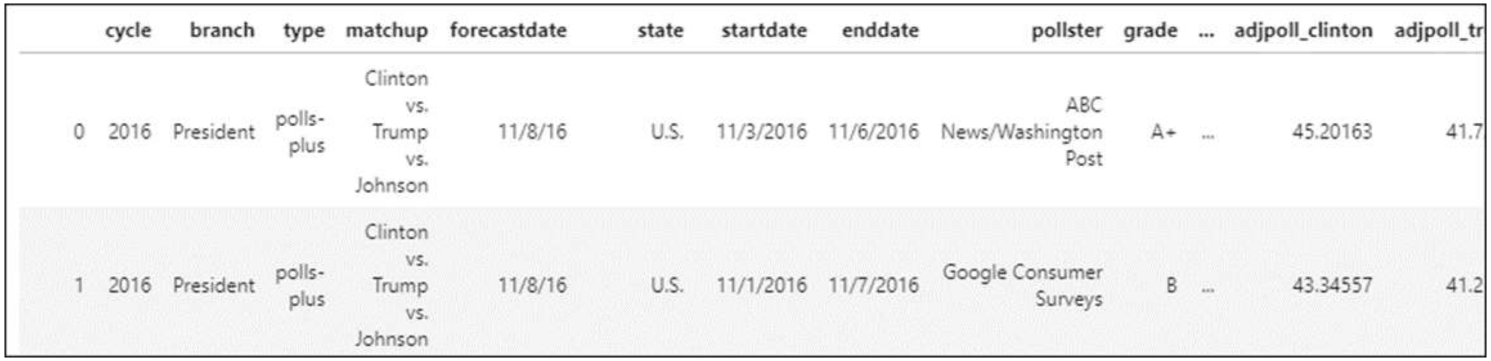
```
type(polls)
```

```
pandas.core.frame.DataFrame
```

The URL for the Polling data

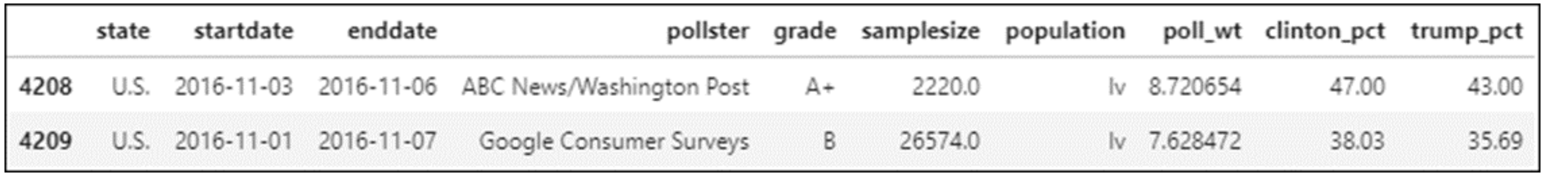
```
http://projects.fivethirtyeight.com/general-model/
president_general_polls_2016.csv
```

The imported DataFrame (12,624 rows and 27 columns)



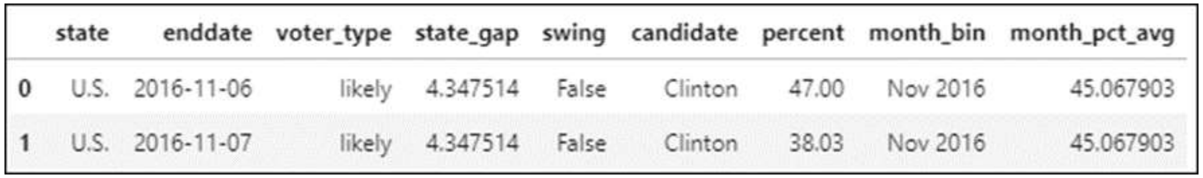
	cycle	branch	type	matchup	forecastdate	state	startdate	enddate	pollster	grade	...	adjpoll_clinton	adjpoll_tr
0	2016	President	polls-plus	Clinton vs. Trump vs. Johnson	11/8/16	U.S.	11/3/2016	11/6/2016	ABC News/Washington Post	A+	...	45.20163	41.7
1	2016	President	polls-plus	Clinton vs. Trump vs. Johnson	11/8/16	U.S.	11/1/2016	11/7/2016	Google Consumer Surveys	B	...	43.34557	41.2

The cleaned DataFrame (4,116 rows and 10 columns)



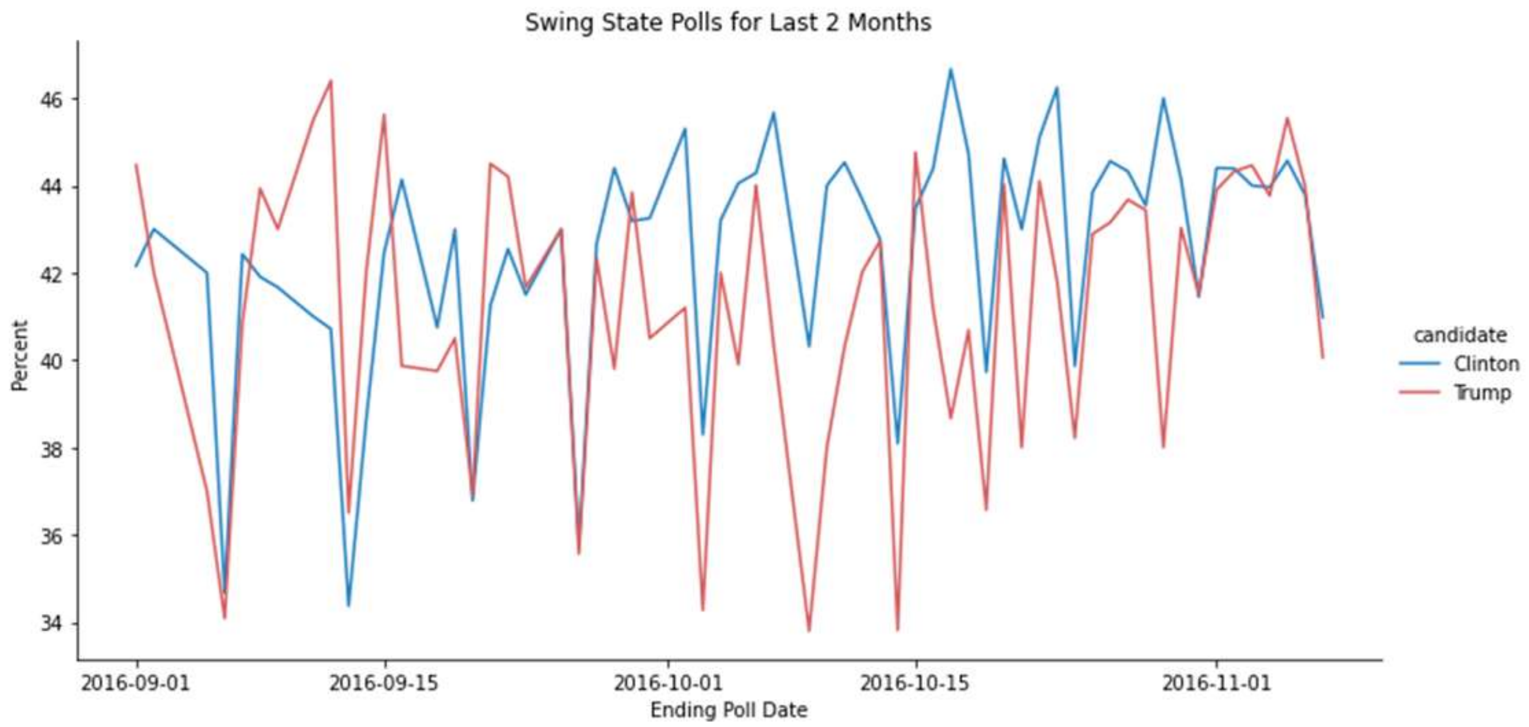
	state	startdate	enddate	pollster	grade	samplesize	population	poll_wt	clinton_pct	trump_pct
4208	U.S.	2016-11-03	2016-11-06	ABC News/Washington Post	A+	2220.0	lv	8.720654	47.00	43.00
4209	U.S.	2016-11-01	2016-11-07	Google Consumer Surveys	B	26574.0	lv	7.628472	38.03	35.69

The prepared DataFrame (8,232 rows and 9 columns)



	state	enddate	voter_type	state_gap	swing	candidate	percent	month_bin	month_pct_avg
0	U.S.	2016-11-06	likely	4.347514	False	Clinton	47.00	Nov 2016	45.067903
1	U.S.	2016-11-07	likely	4.347514	False	Clinton	38.03	Nov 2016	45.067903

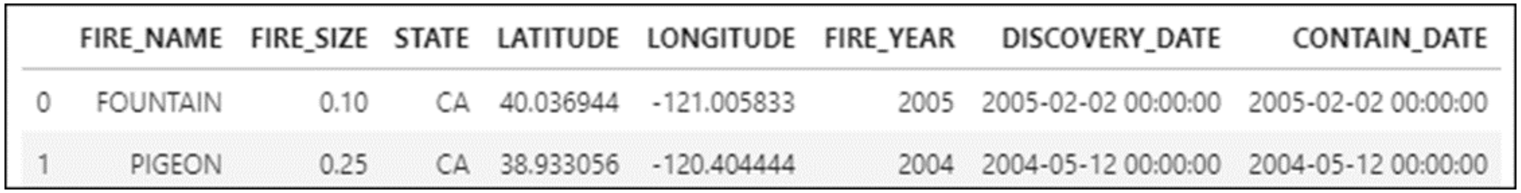
A Seaborn plot of the swing state polls in the 2 months before the election



The URL for the Fires data

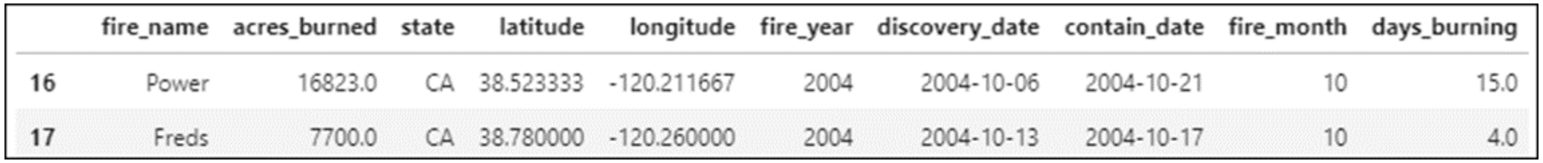
```
https://www.fs.usda.gov/rds/archive/products/RDS-2013-0009.4/
RDS-2013-0009.4_SQLITE.zip
```

The imported DataFrame (1,880,465 rows and 8 columns)



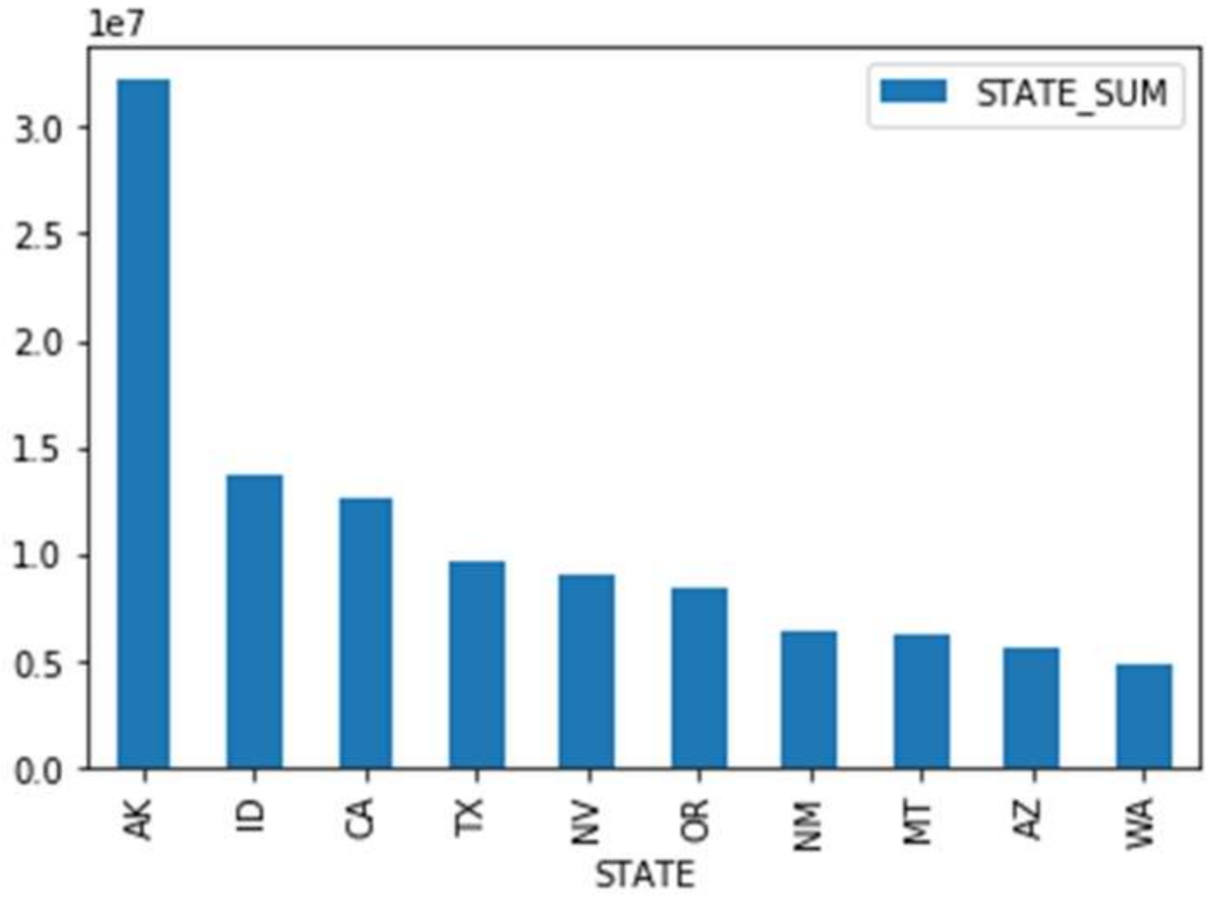
	FIRE_NAME	FIRE_SIZE	STATE	LATITUDE	LONGITUDE	FIRE_YEAR	DISCOVERY_DATE	CONTAIN_DATE
0	FOUNTAIN	0.10	CA	40.036944	-121.005833	2005	2005-02-02 00:00:00	2005-02-02 00:00:00
1	PIGEON	0.25	CA	38.933056	-120.404444	2004	2004-05-12 00:00:00	2004-05-12 00:00:00

The prepared DataFrame (247,123 rows and 10 columns)

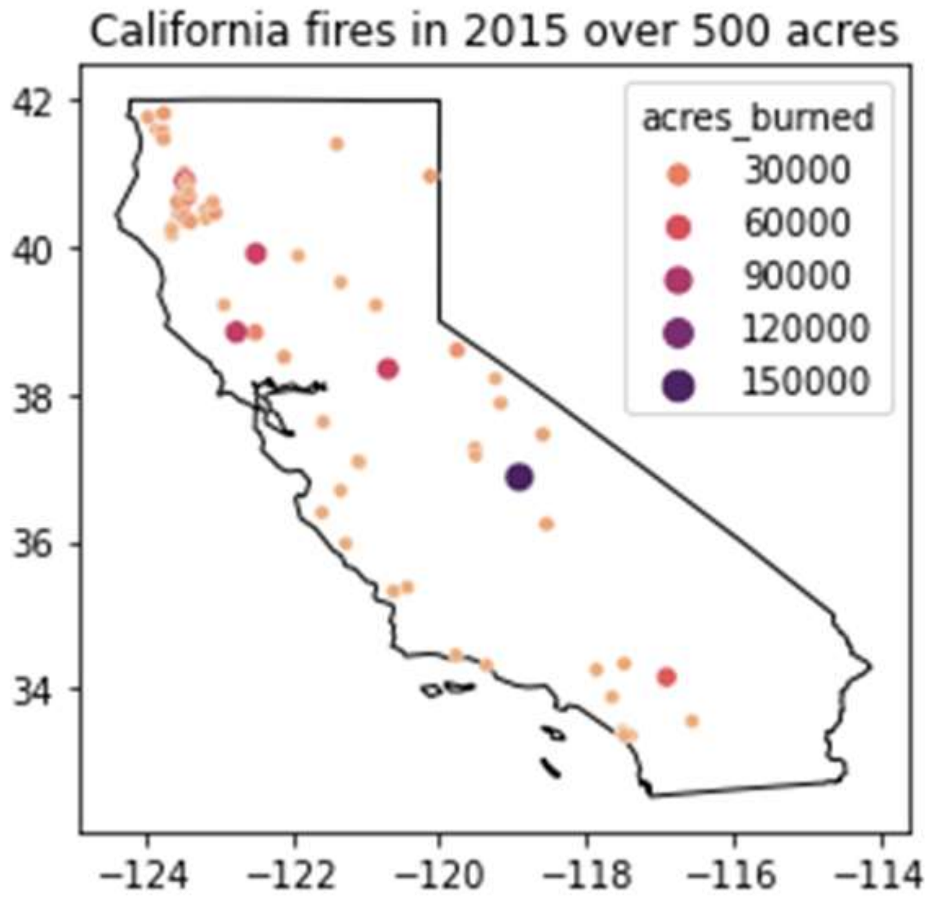


	fire_name	acres_burned	state	latitude	longitude	fire_year	discovery_date	contain_date	fire_month	days_burning
16	Power	16823.0	CA	38.523333	-120.211667	2004	2004-10-06	2004-10-21	10	15.0
17	Freds	7700.0	CA	38.780000	-120.260000	2004	2004-10-13	2004-10-17	10	4.0

A Pandas plot of the total acres burned in the top 10 fire states in 2015



A GeoPandas and Seaborn plot of the California fires over 500 acres in 2015



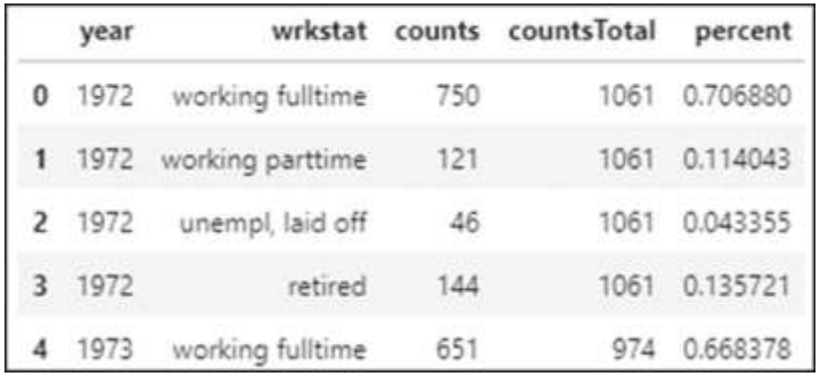
The URL for the Social Survey data

```
http://gss.norc.org/Documents/stata/gss_stata_with_codebook.zip
```

The starting dataset

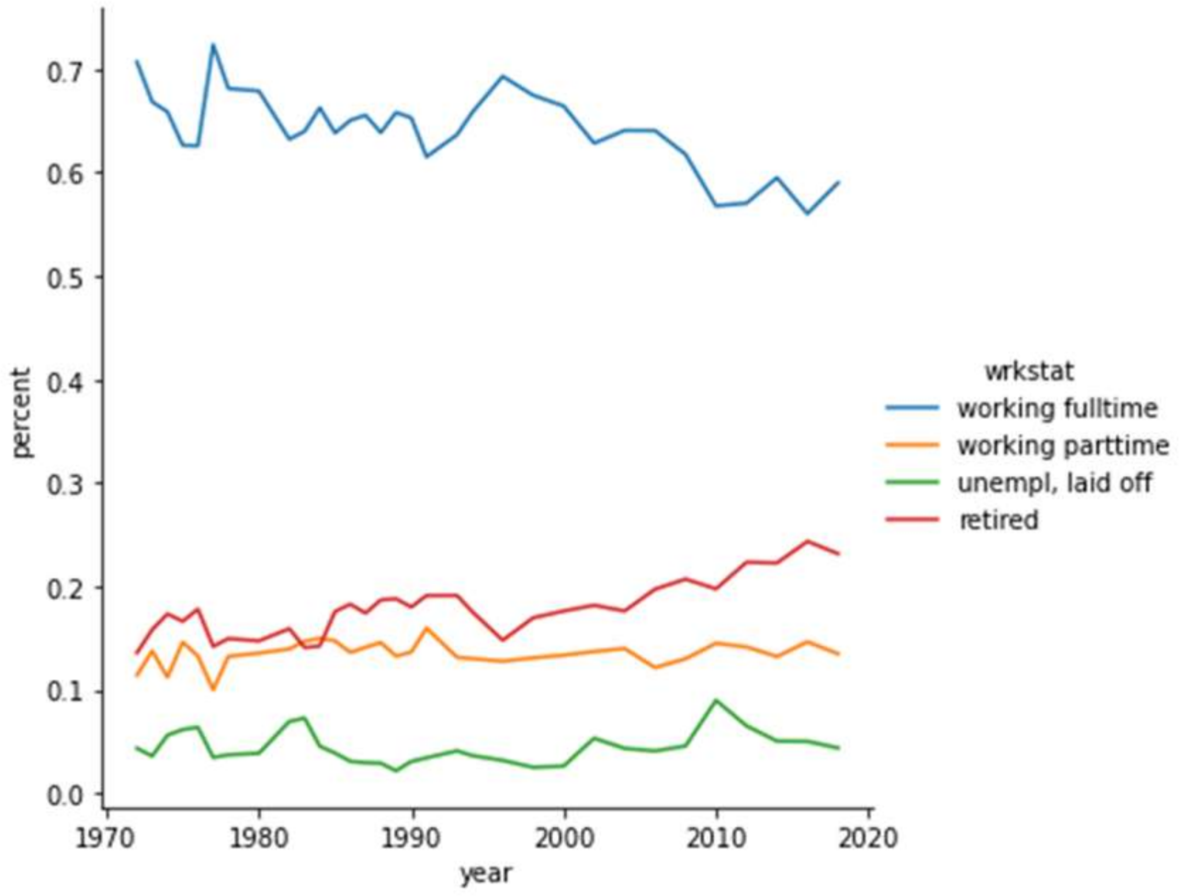
- 64,814 rows and 6,110 columns
- This dataset is so large that importing it requires 3 gigabytes of memory. So instead of importing the entire file, you should import just the subsets of data that you need for your analyses.

One of the DataFrames that's prepared for analysis (128 rows, 5 columns)



	year	wrkstat	counts	countsTotal	percent
0	1972	working fulltime	750	1061	0.706880
1	1972	working parttime	121	1061	0.114043
2	1972	unempl, laid off	46	1061	0.043355
3	1972	retired	144	1061	0.135721
4	1973	working fulltime	651	974	0.668378

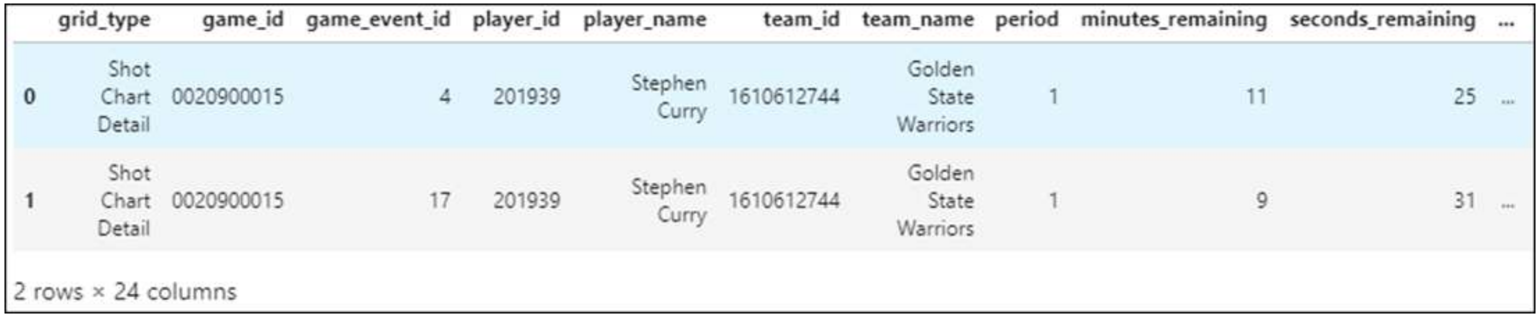
A Seaborn plot derived from the DataFrame



The URL for the Sports Analytics data

```
https://www.murach.com/python_analysis/shots.json
```

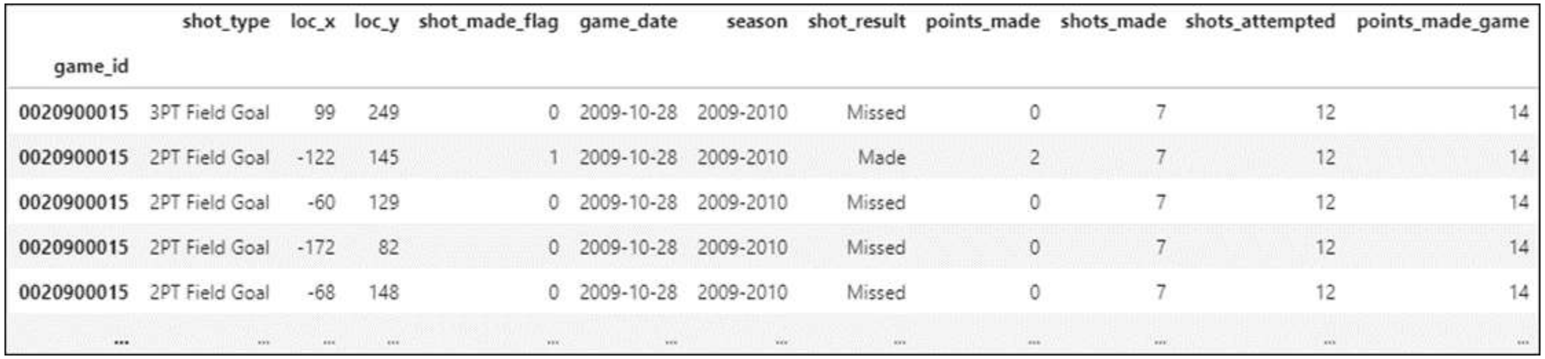
The imported DataFrame (11,846 rows and 24 columns)



	grid_type	game_id	game_event_id	player_id	player_name	team_id	team_name	period	minutes_remaining	seconds_remaining	...
0	Shot Chart Detail	0020900015	4	201939	Stephen Curry	1610612744	Golden State Warriors	1	11	25	...
1	Shot Chart Detail	0020900015	17	201939	Stephen Curry	1610612744	Golden State Warriors	1	9	31	...

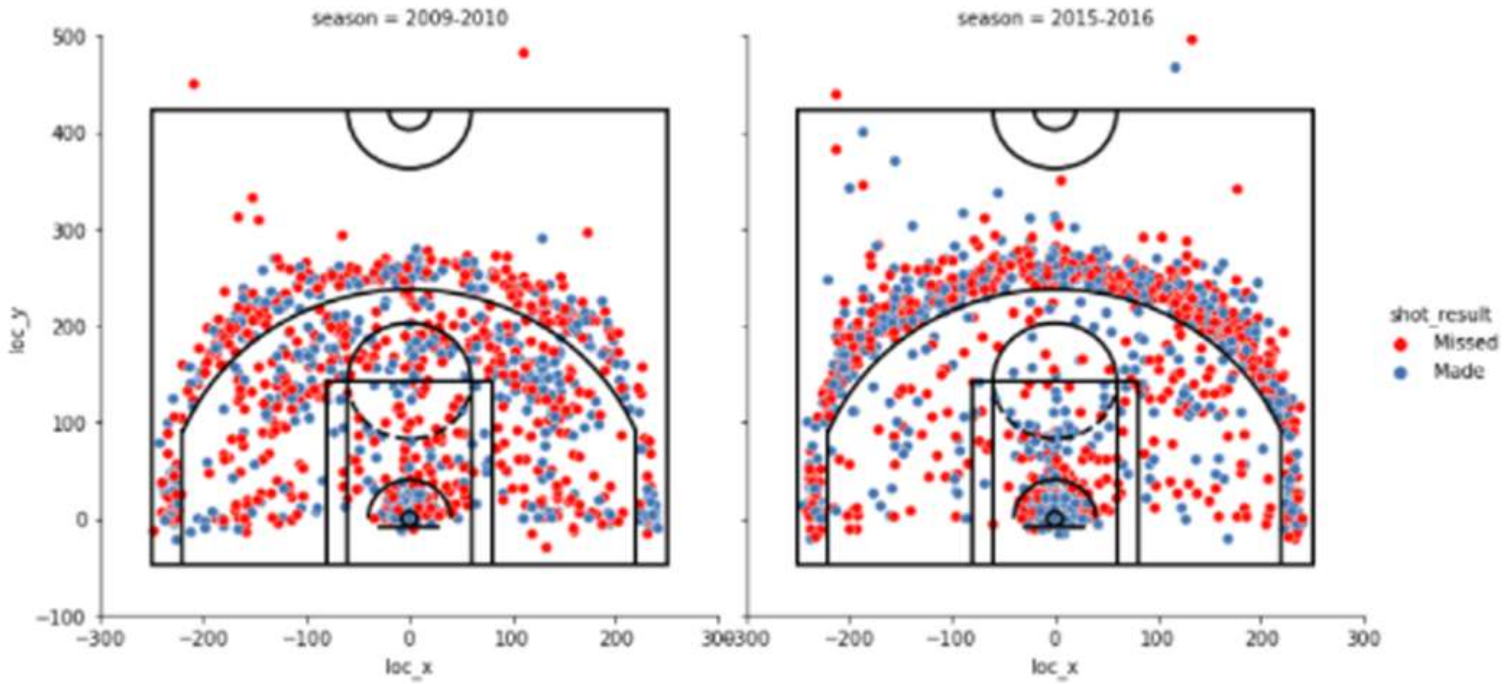
2 rows × 24 columns

The prepared DataFrame (11,753 rows and 12 columns)



	shot_type	loc_x	loc_y	shot_made_flag	game_date	season	shot_result	points_made	shots_made	shots_attempted	points_made_game
game_id											
0020900015	3PT Field Goal	99	249	0	2009-10-28	2009-2010	Missed	0	7	12	14
0020900015	2PT Field Goal	-122	145	1	2009-10-28	2009-2010	Made	2	7	12	14
0020900015	2PT Field Goal	-60	129	0	2009-10-28	2009-2010	Missed	0	7	12	14
0020900015	2PT Field Goal	-172	82	0	2009-10-28	2009-2010	Missed	0	7	12	14
0020900015	2PT Field Goal	-68	148	0	2009-10-28	2009-2010	Missed	0	7	12	14
...	...	...	...	...	...	...	...	...	...	...	...

A Seaborn plot for how Curry's shot selection changed


season = 2009-2010
season = 2015-2016
loc_y
loc_x
shot_result
Missed
Made