USA.gov and .mil data from bit.ly

...

US Government and US Military data

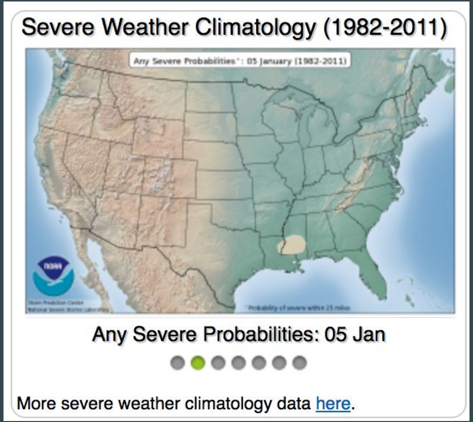
McKinney Ch. 14

USA.gov Data from Bitly

In 2011, URL shortening service Bitly partnered with the US government website USA.gov to provide a feed of anonymous data gathered from users who shorten links ending with .gov or .mil.

.gov and .mil U.S. Government Site Examples

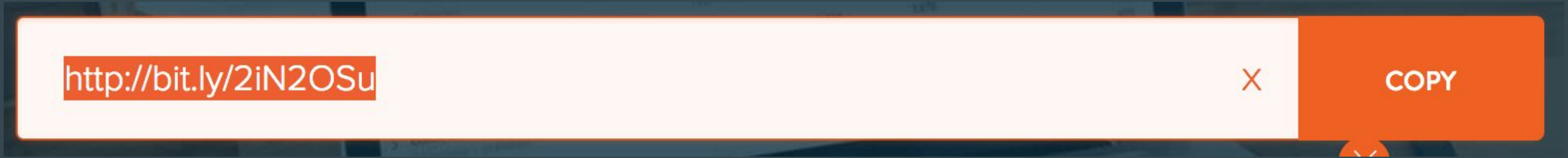
Storm Prediction Center	NASA's Journey to Mars	U.S. Navy
http://www.spc.noaa.gov/	http://mars.nasa.gov/	http://www.navy.mil/


Severe Weather Climatology (1982-2011)
Any Severe Probabilities: 05 January (1982-2011)
Any Severe Probabilities: 05 Jan
More severe weather climatology data here.

http://www.bitly.com is a URL Shortener

https://www.army.mil/article/180289/finance_guru_suze_orman_investing_time_to_train_soldiers

SHORTEN


http://bit.ly/2iN2OSu

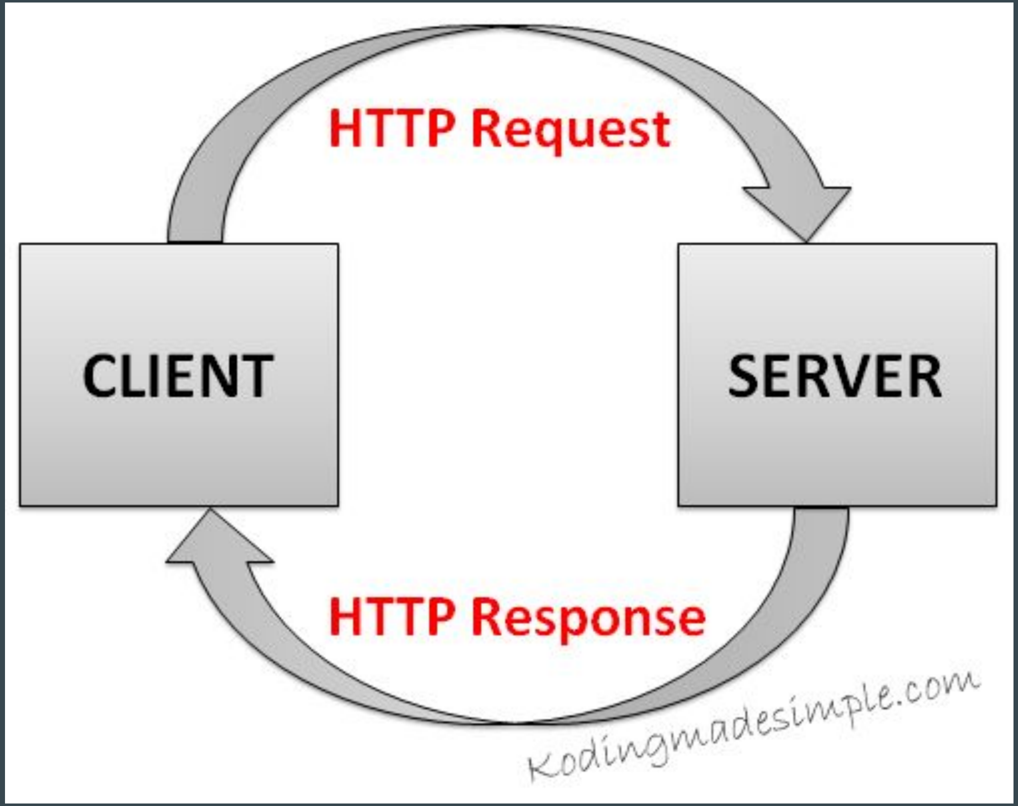
Question 1. What are the top occuring time zones?

Question 2. For the top occuring time zones, Windows or not Windows?

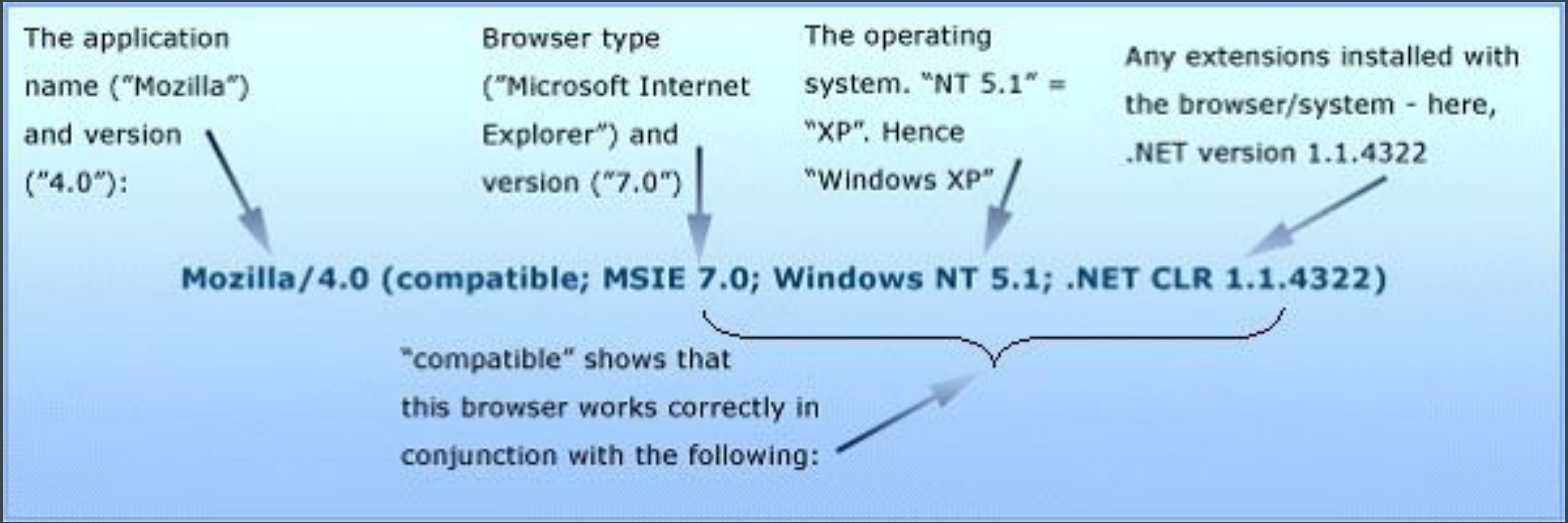
A User Agent (ua)

is a string in the header of HTTP Request
that contains the client
operating system and / or
browser / device requesting content.

Agent string is located in the header of HTTP Request.



Deciphering User Agent string



The application name ("Mozilla") and version ("4.0"):
Browser type ("Microsoft Internet Explorer") and version ("7.0")
The operating system. "NT 5.1" = "XP". Hence "Windows XP"
Any extensions installed with the browser/system - here, .NET version 1.1.4322
"compatible" shows that this browser works correctly in conjunction with the following:

```
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)
```

https://developers.whatismybrowser.com/useragents/parse/

For example, a UA string from a Firefox browser would include the word *Mozilla*, as well as version data and other details.

Mozilla Foundation is the creator of Firefox web browser

But I am using IE! Why does my *ua* start with ‘Mozilla? Read about Microsoft hack.

Agenda

List Comprehensions

JSON Data

Series Value Counts

Handling Missing Data

Indirect sorts with **argsort()**

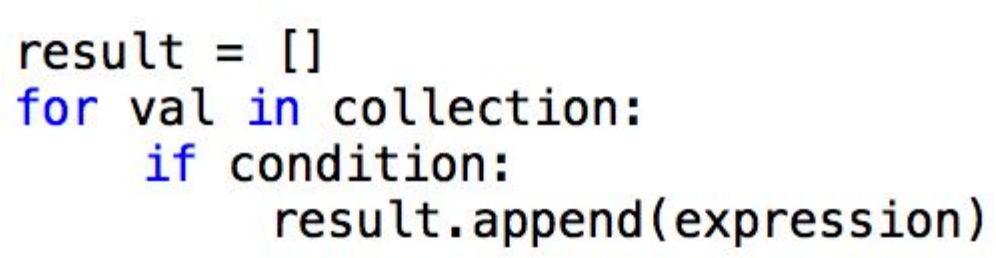
List Comprehensions

McKinney Edition 2. **Chapter 3.1.** List, Set, and Dict Comprehensions.

Online Tutorial

[expression for val in collection if condition]

Is equivalent to the following *for* loop:



```
result = []
for val in collection:
    if condition:
        result.append(expression)
```

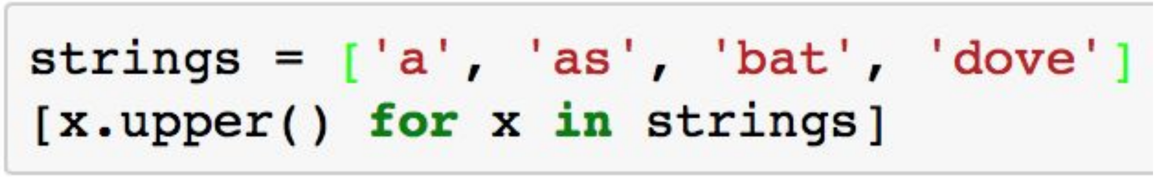
List Comprehension:

Creates a new list from an existing list.

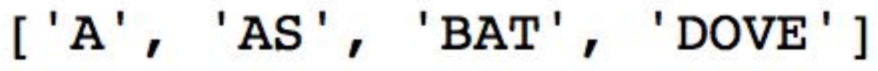
Elements can be conditionally included in the new list.

Each element can be transformed as needed.

List Comprehension Example



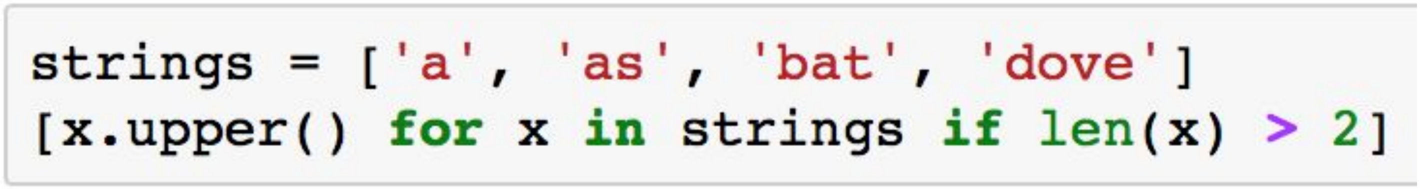
```python
strings = ['a', 'as', 'bat', 'dove']
[x.upper() for x in strings]
```



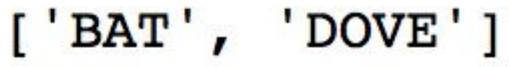
```
['A', 'AS', 'BAT', 'DOVE']
```

List Comprehension Example

If an element’s length is greater than 2 characters, than add it to the list.

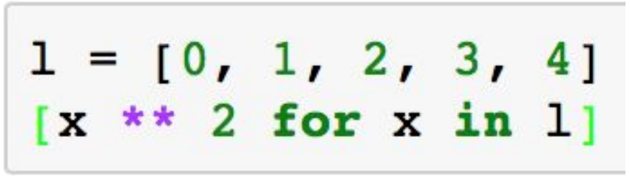


```
strings = ['a', 'as', 'bat', 'dove']
[x.upper() for x in strings if len(x) > 2]
```



```
['BAT', 'DOVE']
```

List Comprehension Example

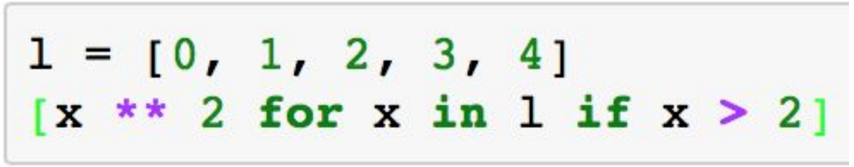


```python
l = [0, 1, 2, 3, 4]
[x ** 2 for x in l]
```

```
[0, 1, 4, 9, 16]
```

List Comprehension Example

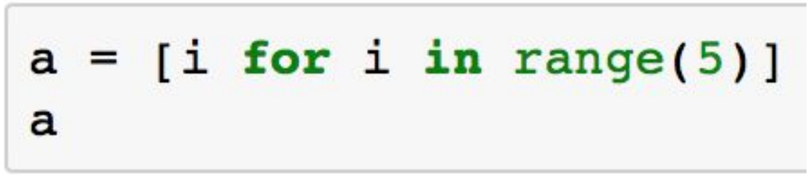
If an element in the list is greater than 2, then add it to the new list.



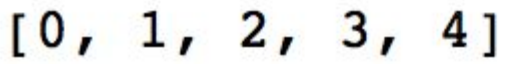
```
l = [0, 1, 2, 3, 4]
[x ** 2 for x in l if x > 2]
```

```
[9, 16]
```

range() in List Comprehensions



```
a = [i for i in range(5)]
a
```



```
[0, 1, 2, 3, 4]
```

JSON Data

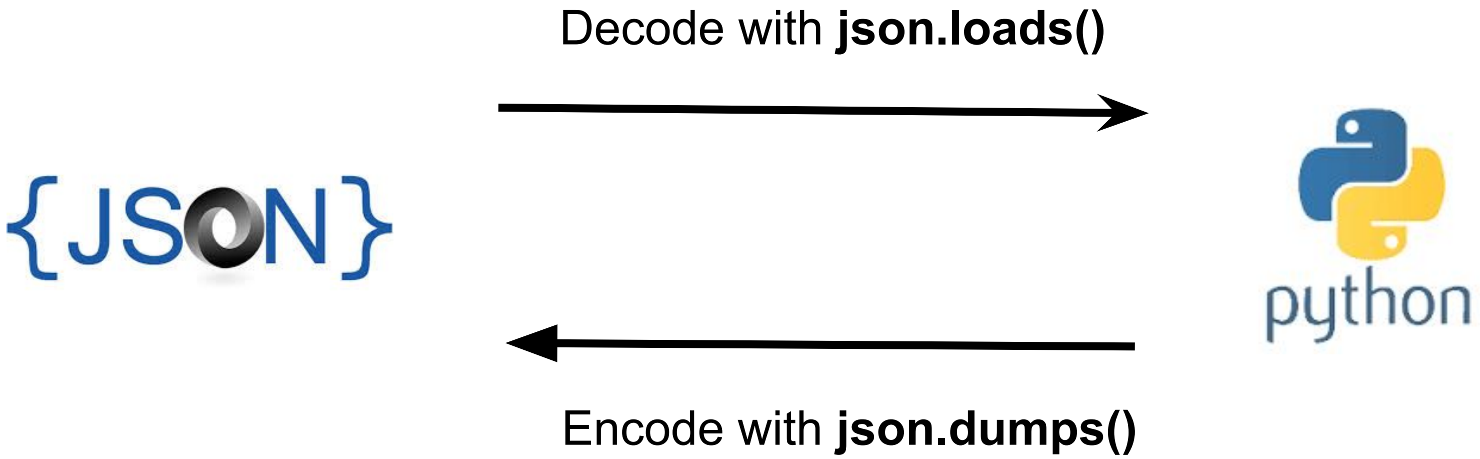
short for **JavaScript Object Notation**

McKinney Editions 2. Chapter 6. JSON Data.

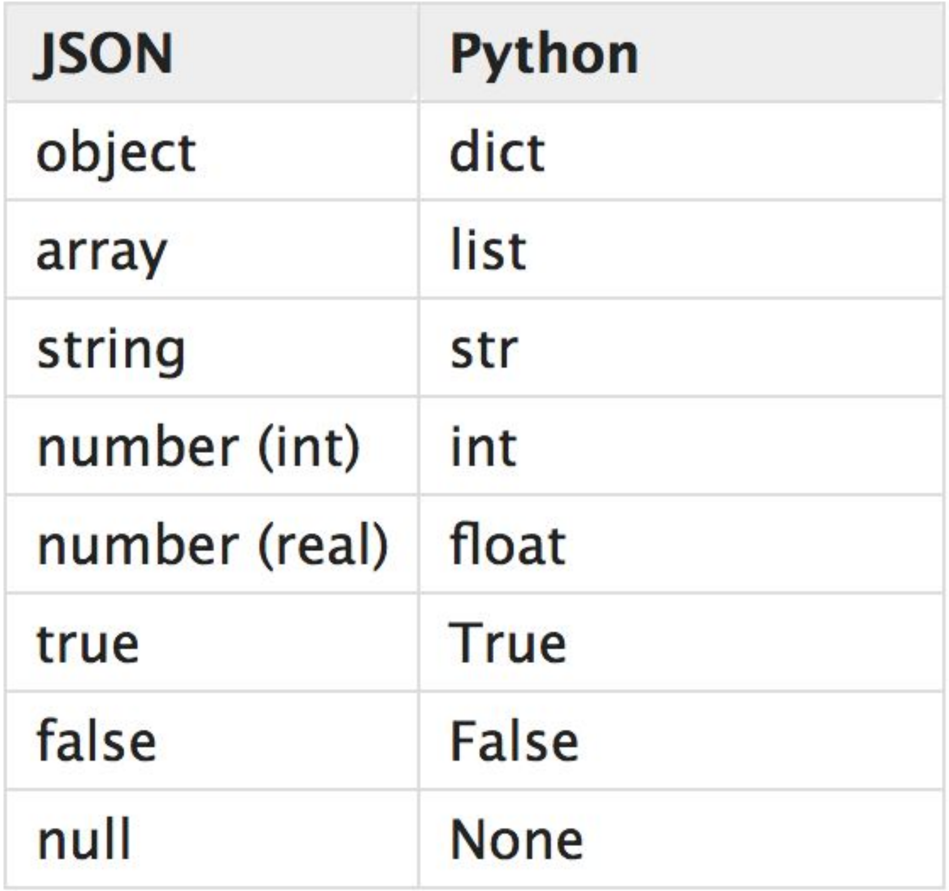
json.org

JSON an exchange format widely used when moving data across applications, especially when data needs to be treated in a language or platform-agnostic way.

json module

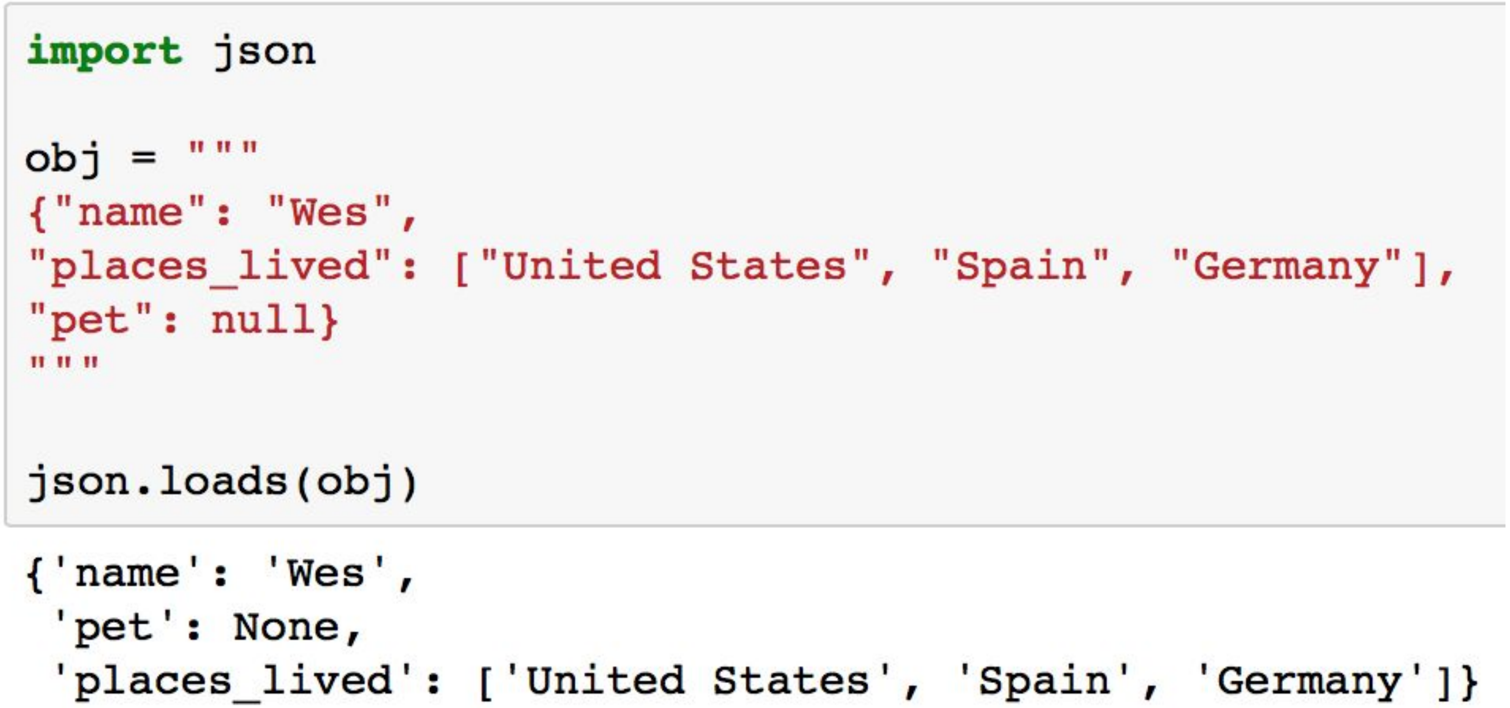


JSON to Python Conversion



JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

JSON load to Python dictionary



```
import json

obj = """
{"name": "Wes",
"places_lived": ["United States", "Spain", "Germany"],
"pet": null}
"""


json.loads(obj)
```

```
{'name': 'Wes',
 'pet': None,
 'places_lived': ['United States', 'Spain', 'Germany']}
```

Encode back to JSON string

```
result
```

```
{'name': 'Wes',
 'pet': None,
 'places_lived': ['United States', 'Spain', 'Germany']}
```

```
json.dumps(result)
```

```
'{"name": "Wes", "places_lived": ["United States", "Spain", "Germany"], "pet": null}'
```

pandas.Series.value_counts

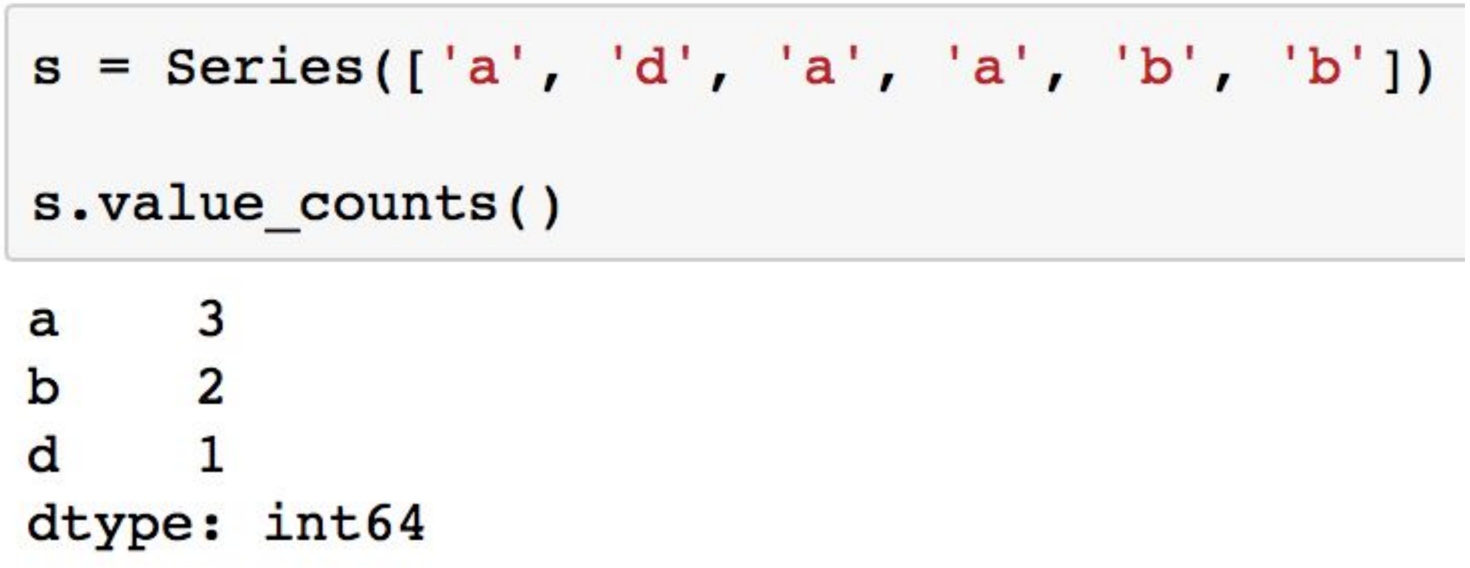
McKinney Edition 1-2. **Chapter 5**. Unique Values, Value Counts, and Membership.

Returns a series object that contains counts of unique values.

The resulting object will be in descending order so that the first element is the most frequently-occurring element.

Excludes NA values by default.

Series value_counts() example



```
s = Series(['a', 'd', 'a', 'a', 'b', 'b'])

s.value_counts()
```

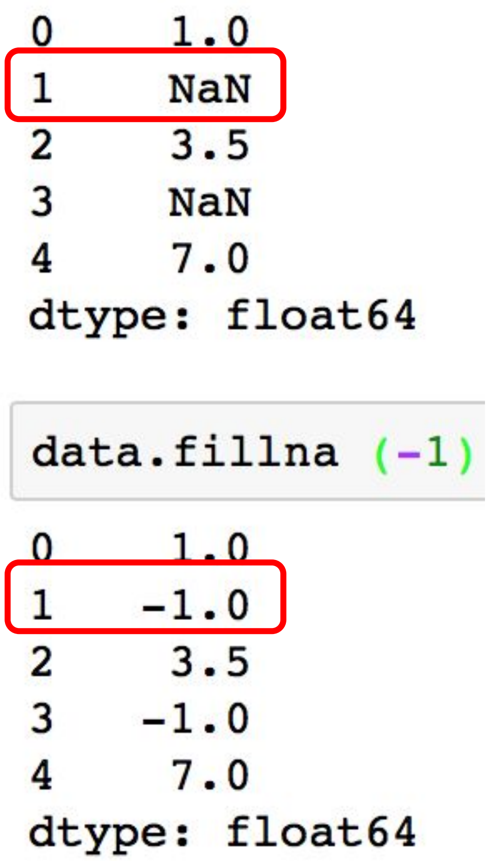
```
a     3
b     2
d     1
dtype: int64
```

Handling Missing Data

McKinney **Chapter 7. 1.** Handling Missing Data.

pandas.DataFrame.fillna()
pandas.Series.fillna()
pandas.DataFrame.dropna()
pandas.Series.dropna()

Series.fillna()
fills NA / NaN values



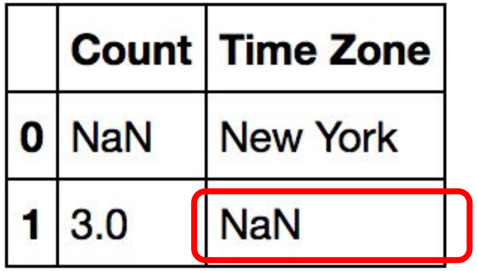
```
0      1.0
1      NaN
2      3.5
3      NaN
4      7.0
dtype: float64
```

```
data.fillna (-1)
```

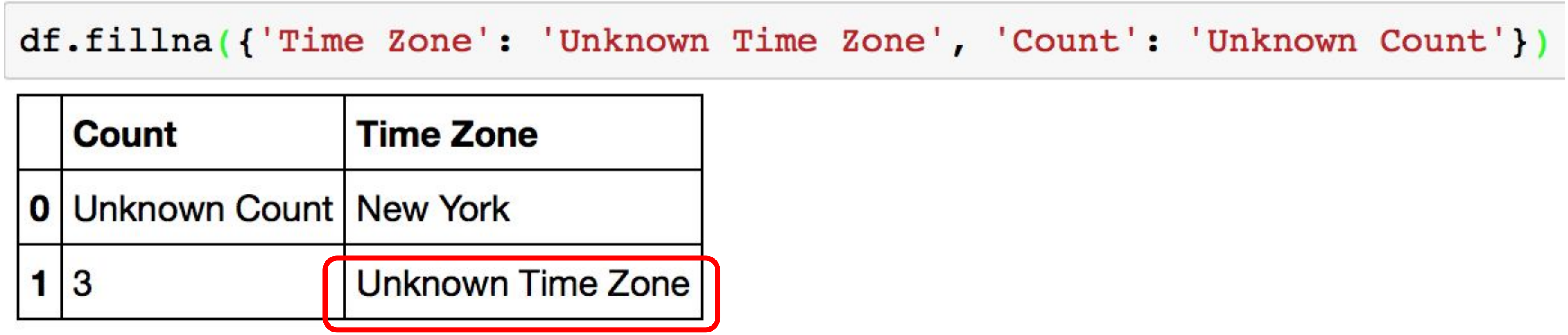
```
0      1.0
1     -1.0
2      3.5
3     -1.0
4      7.0
dtype: float64
```

Replace missing values with a constant.

DataFrame.fillna()
fills NA / NaN values



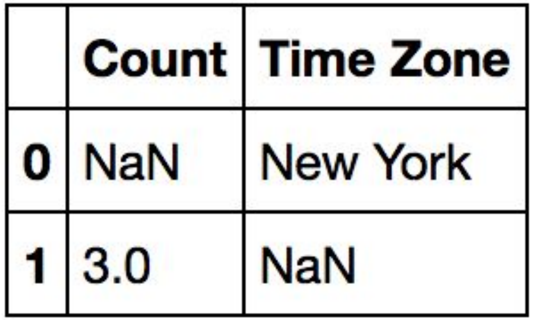
	Count	Time Zone
0	NaN	New York
1	3.0	NaN



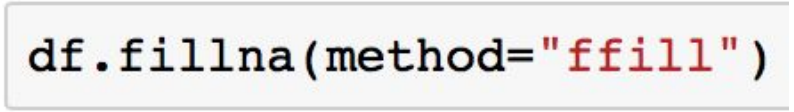
```
df.fillna({'Time Zone': 'Unknown Time Zone', 'Count': 'Unknown Count'})
```

	Count	Time Zone
0	Unknown Count	New York
1	3	Unknown Time Zone

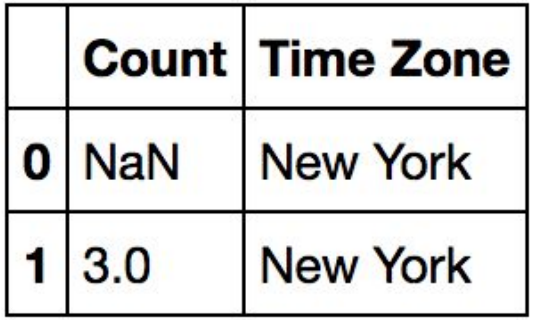
Fill value is different for every column.



	Count	Time Zone
0	NaN	New York
1	3.0	NaN

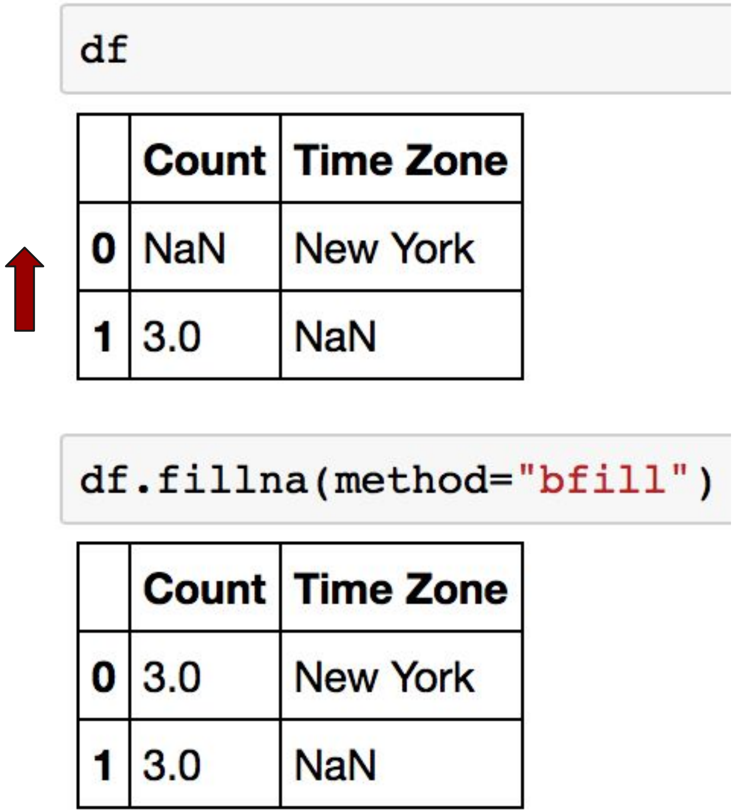


```
df.fillna(method="ffill")
```



	Count	Time Zone
0	NaN	New York
1	3.0	New York

Forward Fill (‘ffill’) will propagate last valid observation forward to next valid observation.



```
df
```

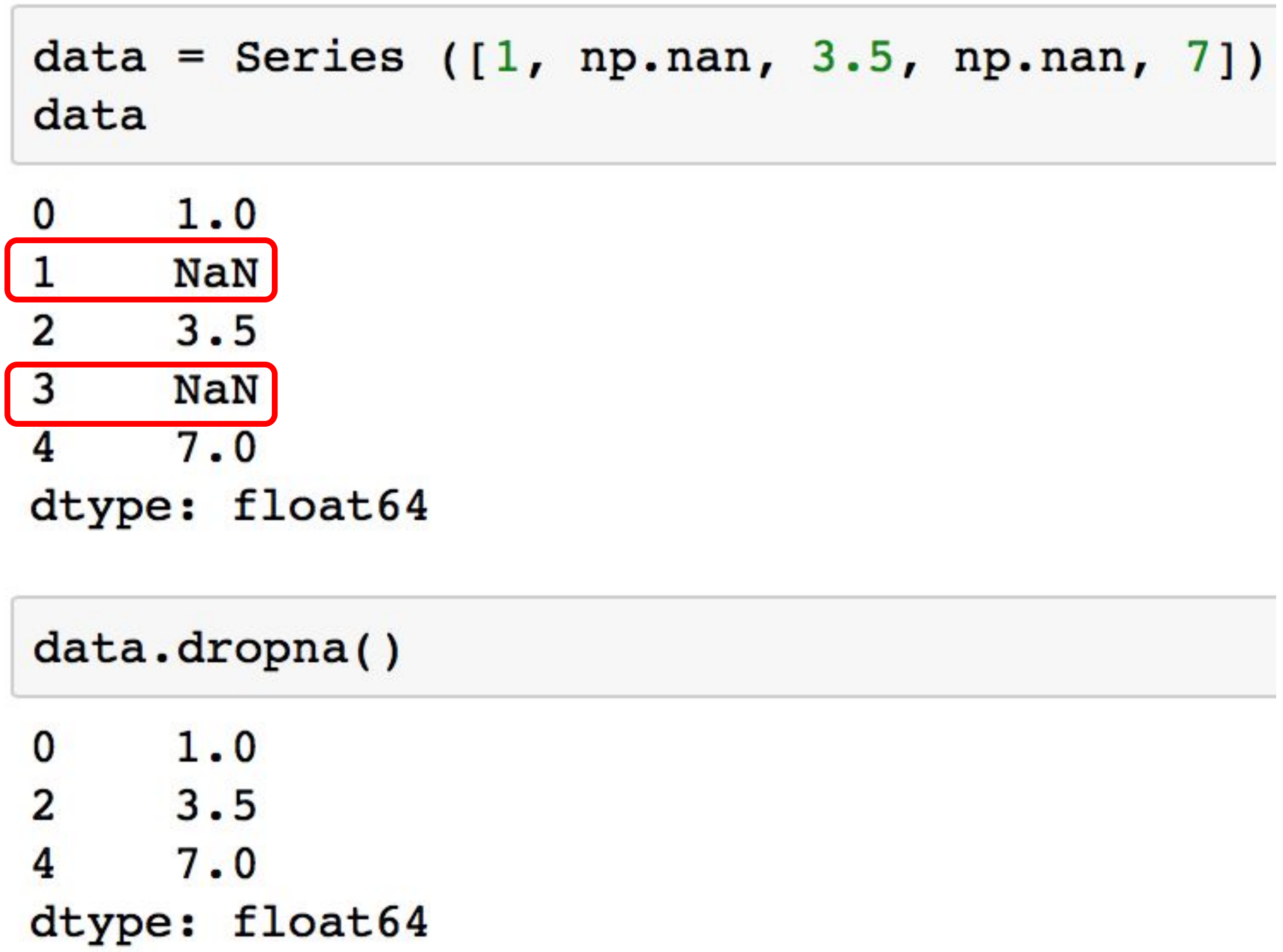
	Count	Time Zone
0	NaN	New York
1	3.0	NaN

```
df.fillna(method="bfill")
```

	Count	Time Zone
0	3.0	New York
1	3.0	NaN

Back Fill (‘bfill’) use NEXT valid observation to fill the gap.

Series.dropna()
returns Series without null values.



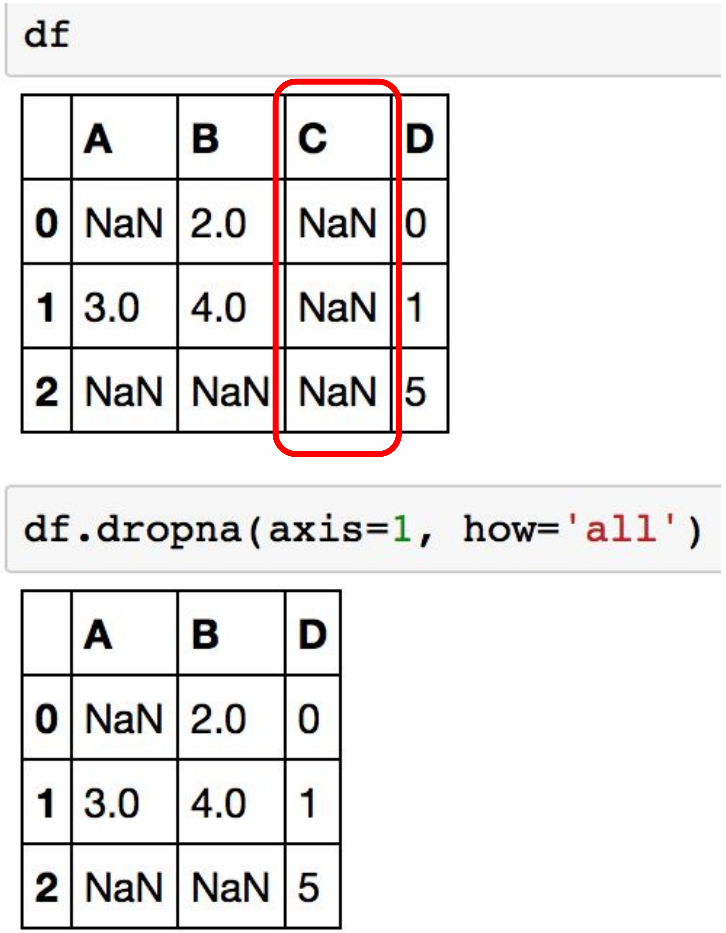
```
data = Series ([1, np.nan, 3.5, np.nan, 7])
data
```

```
0      1.0
1      NaN
2      3.5
3      NaN
4      7.0
dtype: float64
```

```
data.dropna()
```

```
0      1.0
2      3.5
4      7.0
dtype: float64
```

DataFrame.dropna()
returns DataFrame with NA entries dropped from it.



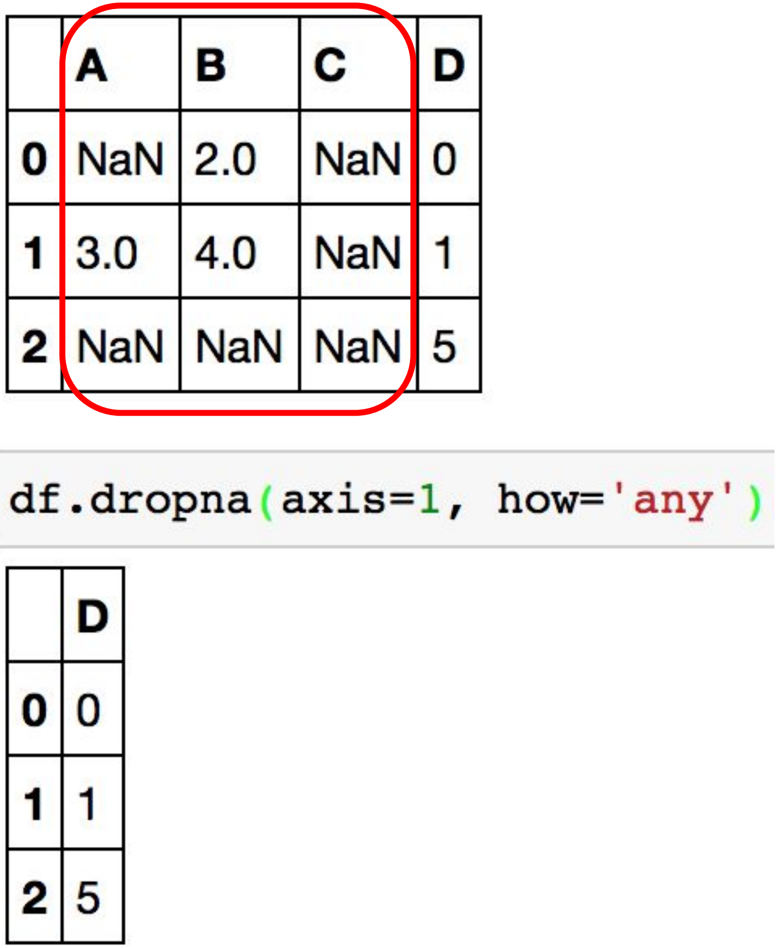
```python
df
```

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

```python
df.dropna(axis=1, how='all')
```

	A	B	D
0	NaN	2.0	0
1	3.0	4.0	1
2	NaN	NaN	5

Column ‘C’ is dropped because it contains all NaN values.



	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

```
df.dropna(axis=1, how='any')
```

	D
0	0
1	1
2	5

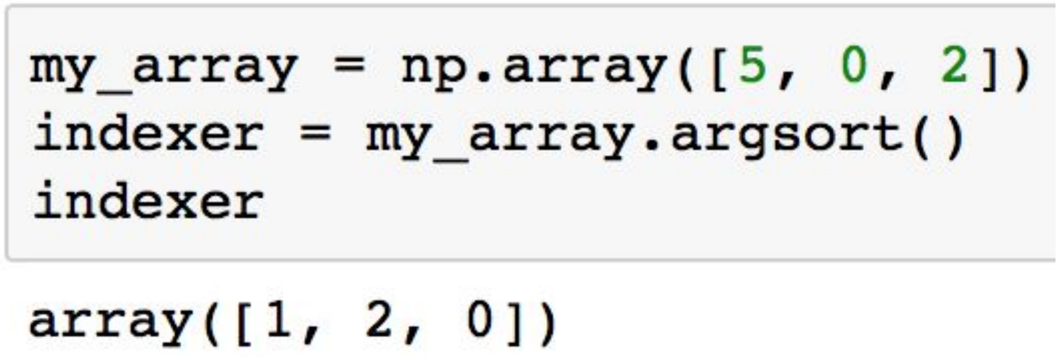
Any column that contains NaN values is dropped.

numpy.argsort()

McKinney Edition 2. **A6.** Indirect Sorts: argsort and lexsort.
McKinney Edition 1. **Chapter 12**. Indirect Sorts: argsort and lexsort.

argsort returns an array of indices that would sort an array.

Create an indexer with numpy.argsort()

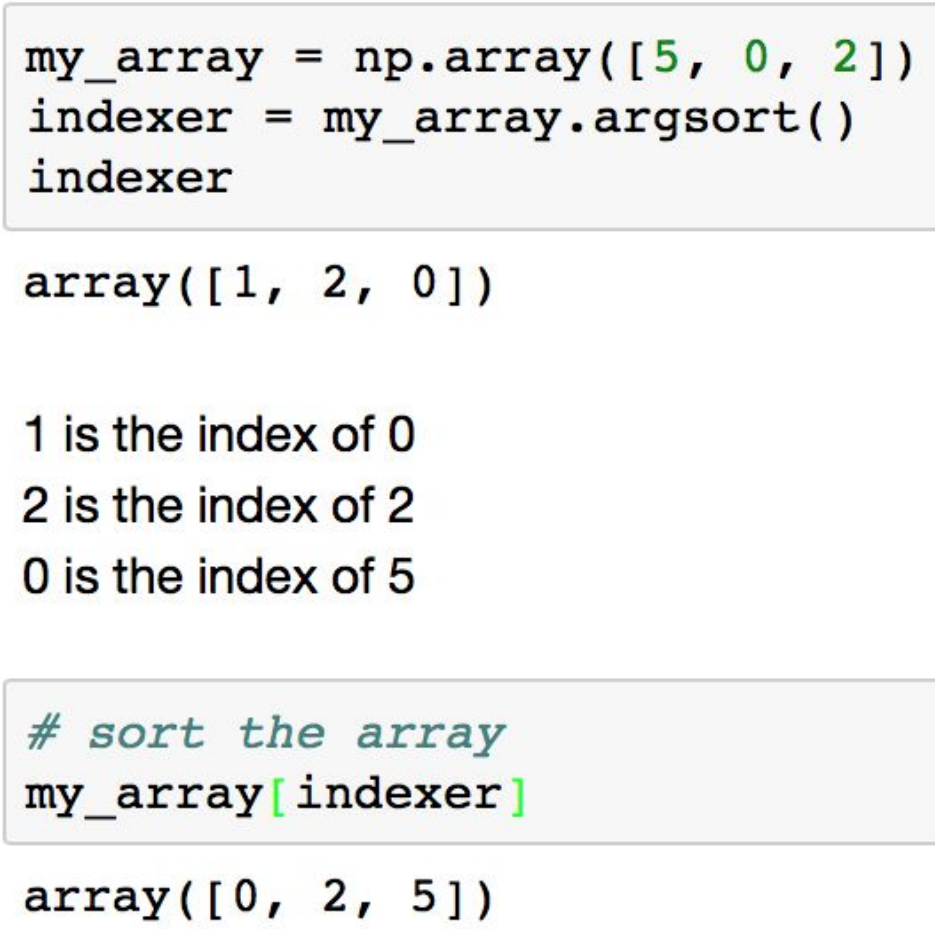


```
my_array = np.array([5, 0, 2])
indexer = my_array.argsort()
indexer
```

```
array([1, 2, 0])
```

Full Example

Sorts an array



```
my_array = np.array([5, 0, 2])
indexer = my_array.argsort()
indexer
```

```
array([1, 2, 0])
```

1 is the index of 0

2 is the index of 2

0 is the index of 5

```
# sort the array
my_array[indexer]
```

```
array([0, 2, 5])
```