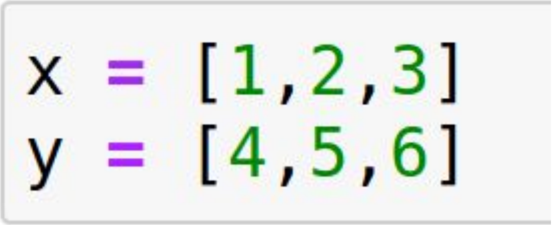
Python, *pandas*, and *NumPy*

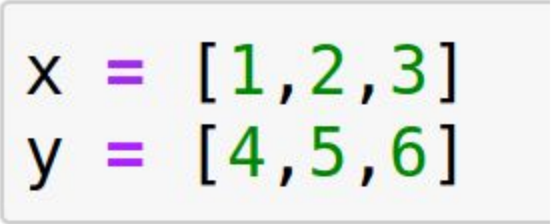
Lists, arrays, Series, and DataFrames



```
x = [1,2,3]
y = [4,5,6]
```

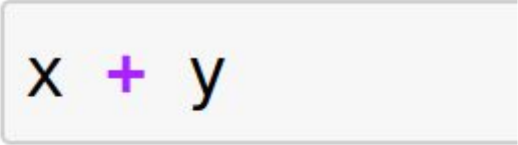
What are *x* and *y* ?

Lists, arrays, Series, and DataFrames



```
x = [1,2,3]
y = [4,5,6]
```

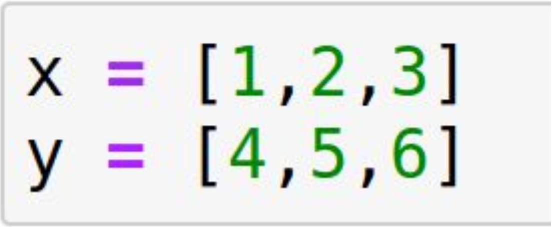
What are *x* and *y* ?



```
x + y
```

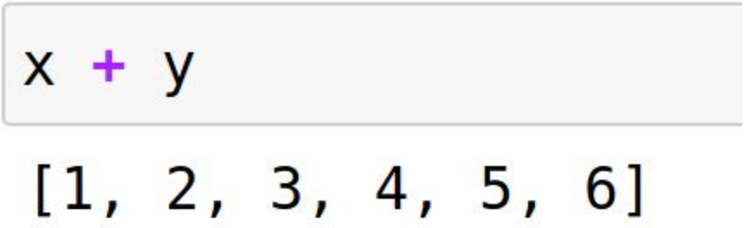
What happens when I do this?

Lists, arrays, Series, and DataFrames



```
x = [1,2,3]
y = [4,5,6]
```

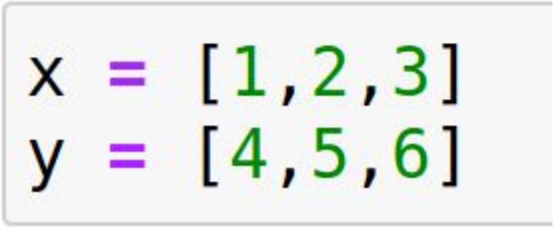
What are *x* and *y* ?



```
x + y
```

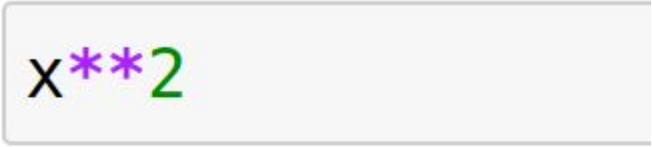
```
[1, 2, 3, 4, 5, 6]
```

Lists, arrays, Series, and DataFrames



```
x = [1,2,3]
y = [4,5,6]
```

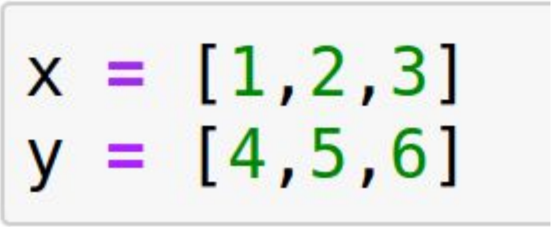
What are *x* and *y* ?



```
x**2
```

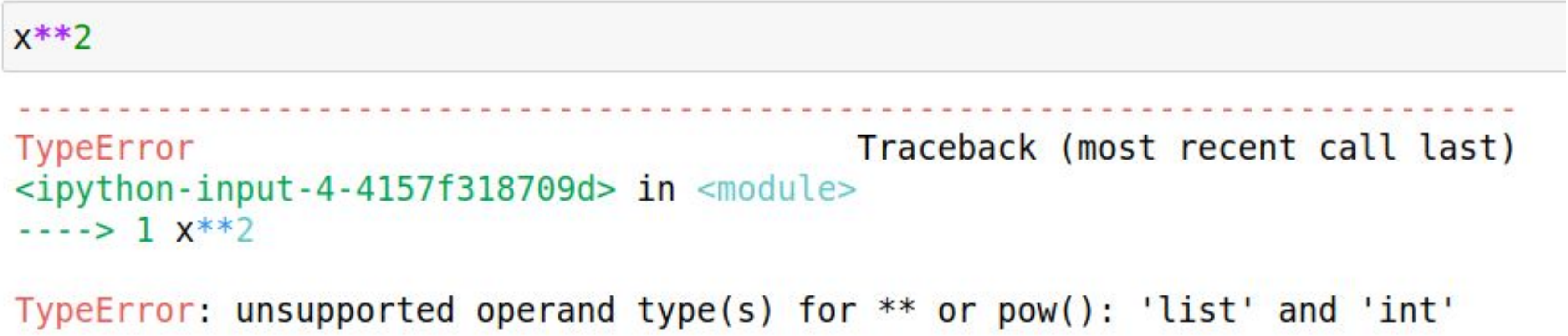
What happens when I do this?

Lists, arrays, Series, and DataFrames



```
x = [1,2,3]
y = [4,5,6]
```

What are *x* and *y* ?



```
x**2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-4157f318709d> in <module>
----> 1 x**2

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

Lists, arrays, Series, and DataFrames

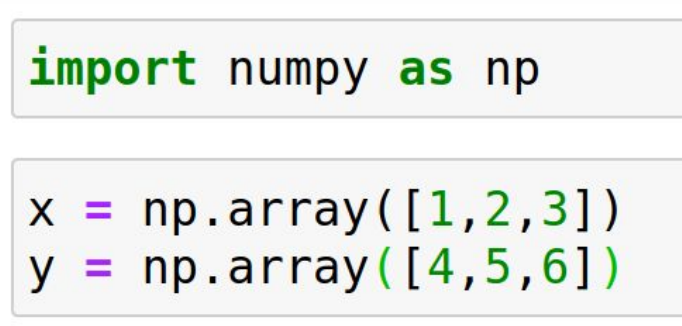
What if I want to be able to do element-wise operations?

Lists, arrays, Series, and DataFrames

What if I want to be able to do element-wise operations?

One option is to use NumPy arrays

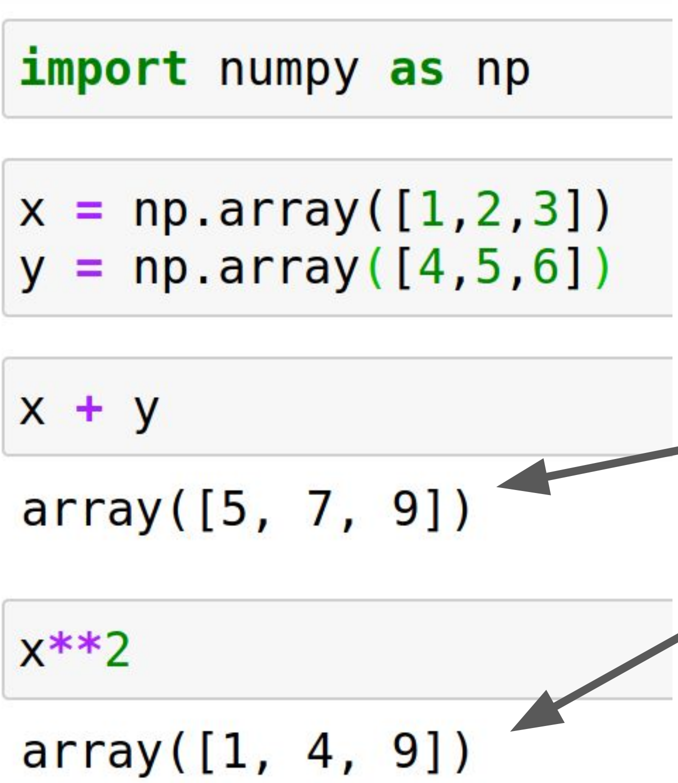
Lists, arrays, Series, and DataFrames



```
import numpy as np
```

```
x = np.array([1,2,3])
y = np.array([4,5,6])
```

Lists, arrays, Series, and DataFrames



```
import numpy as np
```

```
x = np.array([1,2,3])
y = np.array([4,5,6])
```

```
x + y
```

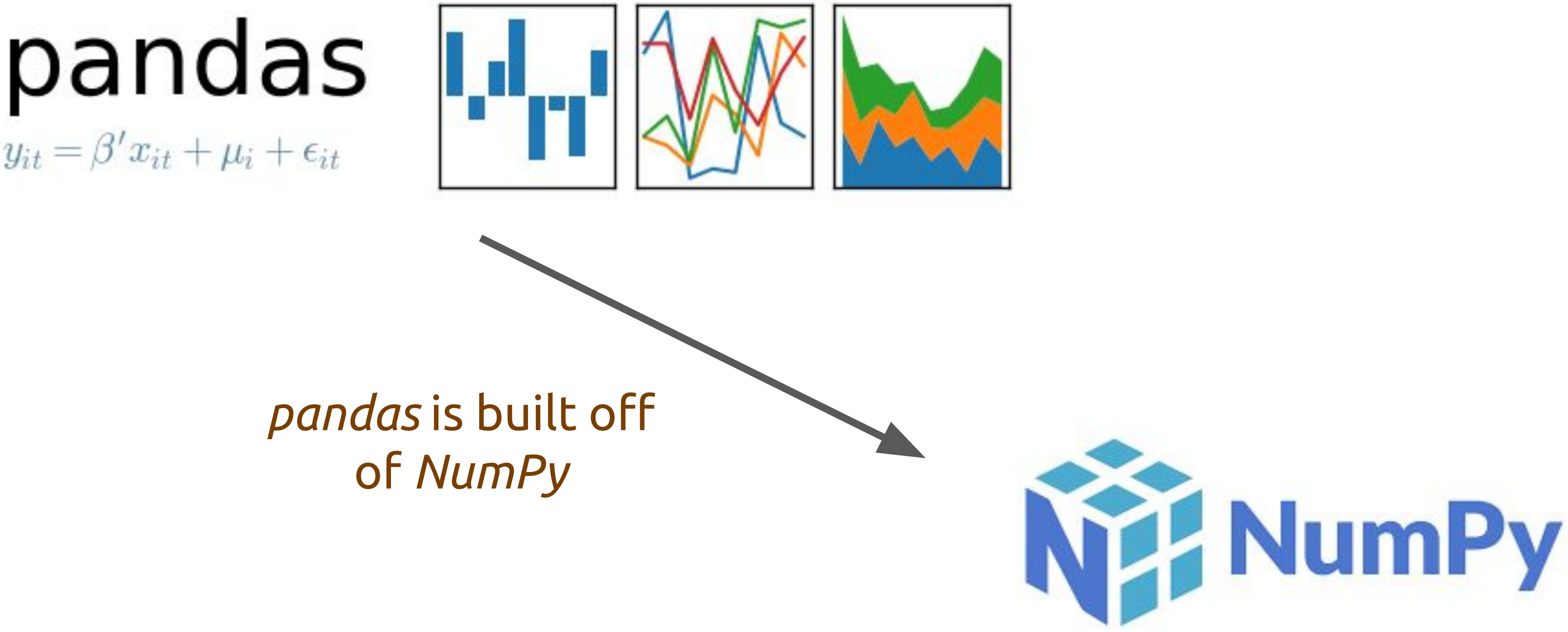
```
array([5, 7, 9])
```

```
x**2
```

```
array([1, 4, 9])
```

Notice that the results of these operations are arrays.

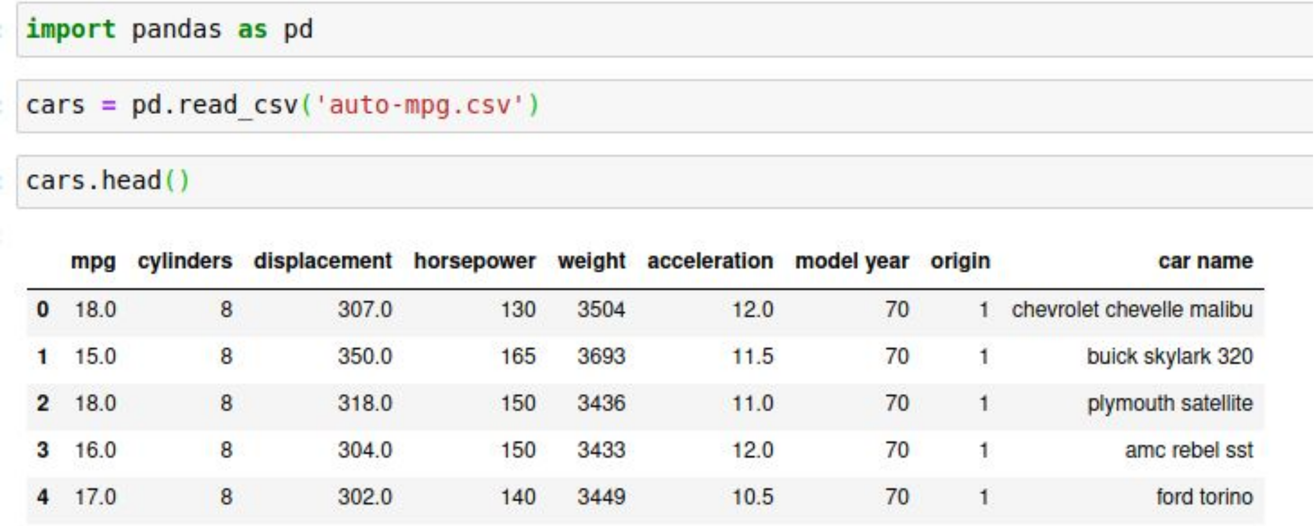
Lists, arrays, Series, and DataFrames



pandas is built off
of *NumPy*

Lists, arrays, Series, and DataFrames

pandas DataFrame



```
import pandas as pd
```

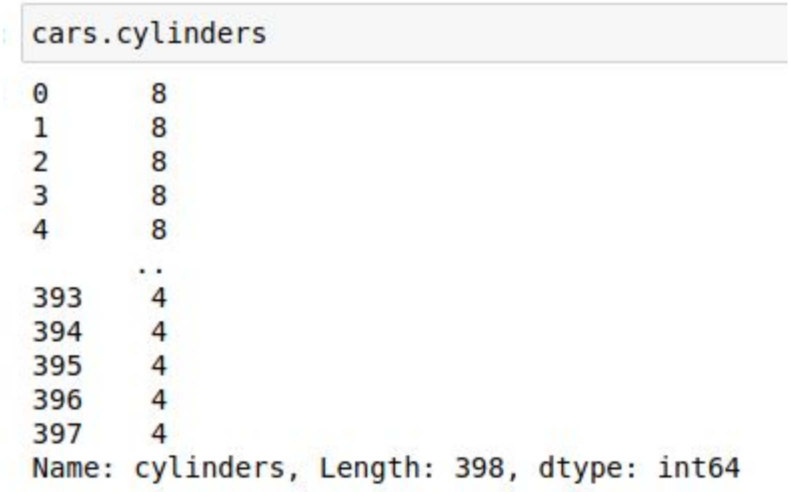
```
cars = pd.read_csv('auto-mpg.csv')
```

```
cars.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Lists, arrays, Series, and DataFrames

pandas Series

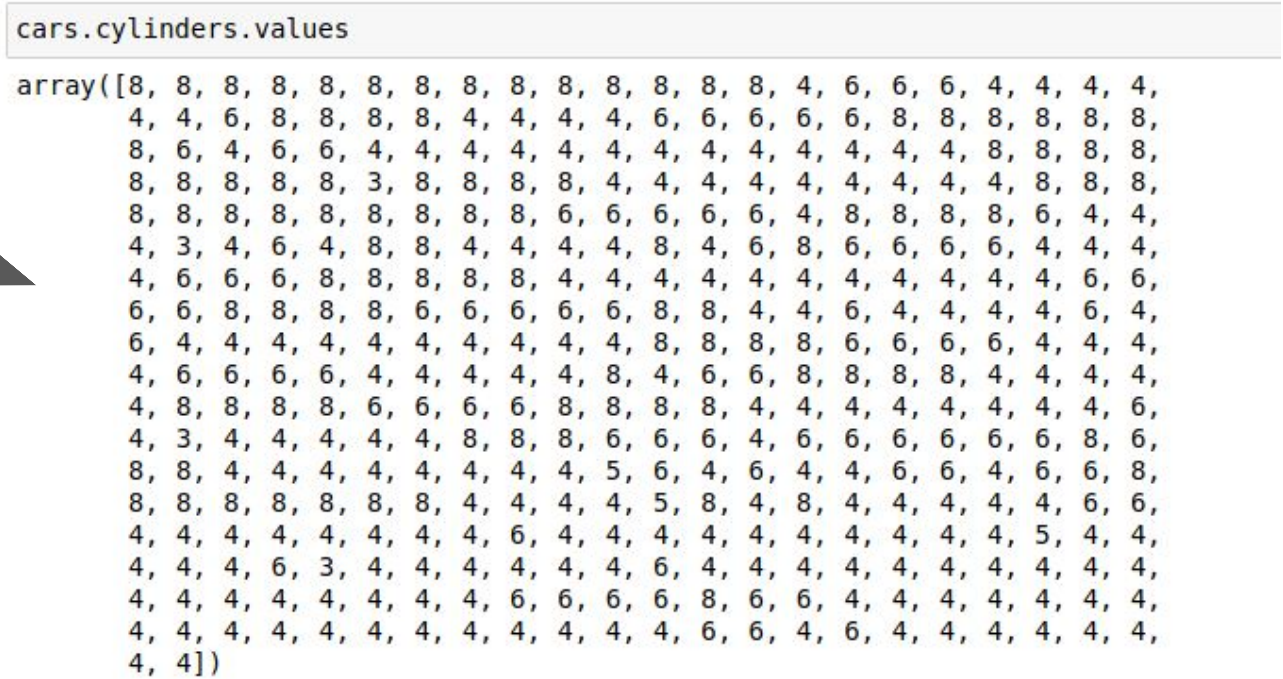


```
cars.cylinders
```

```
0      8
1      8
2      8
3      8
4      8
      ..
393    4
394    4
395    4
396    4
397    4
Name: cylinders, Length: 398, dtype: int64
```

Lists, arrays, Series, and DataFrames

NumPy array

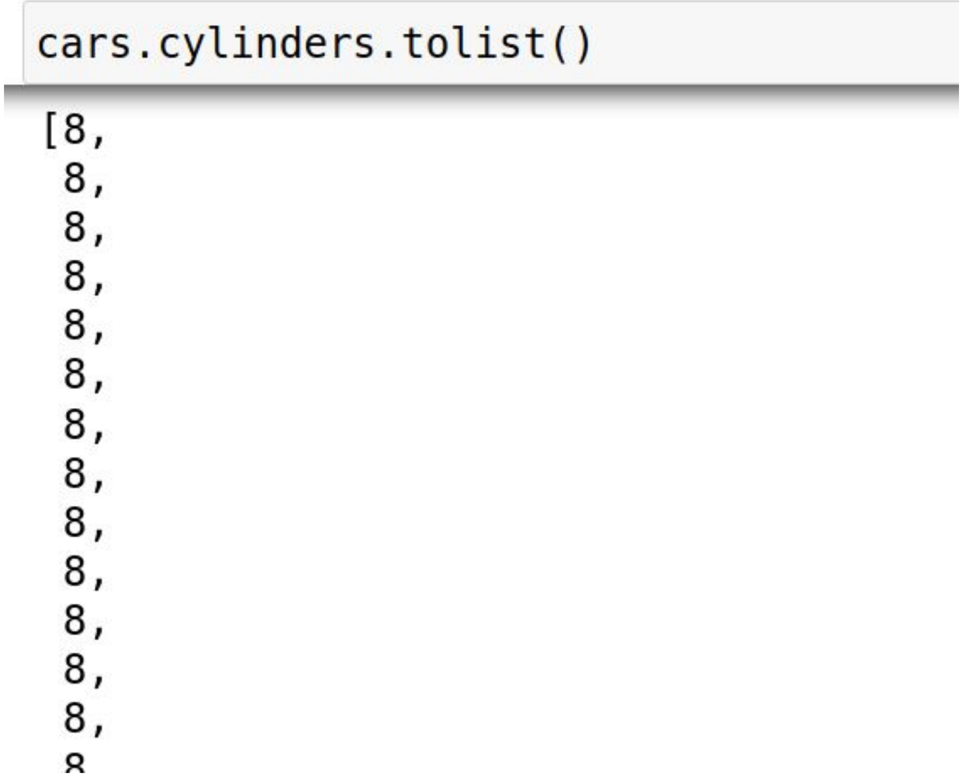


```
cars.cylinders.values
```

```
array([8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 6, 6, 6, 4, 4, 4, 4,
       4, 4, 6, 8, 8, 8, 8, 4, 4, 4, 4, 6, 6, 6, 6, 6, 8, 8, 8, 8, 8, 8,
       8, 6, 4, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8, 8, 8, 8,
       8, 8, 8, 8, 8, 3, 8, 8, 8, 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8, 8, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 6, 6, 6, 6, 4, 8, 8, 8, 8, 6, 4, 4,
       4, 3, 4, 6, 4, 8, 8, 4, 4, 4, 4, 8, 4, 6, 8, 6, 6, 6, 6, 4, 4, 4,
       4, 6, 6, 6, 8, 8, 8, 8, 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6,
       6, 6, 8, 8, 8, 8, 6, 6, 6, 6, 6, 8, 8, 4, 4, 6, 4, 4, 4, 4, 6, 4,
       6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 8, 8, 8, 8, 6, 6, 6, 6, 4, 4, 4,
       4, 6, 6, 6, 6, 4, 4, 4, 4, 4, 8, 4, 6, 6, 8, 8, 8, 8, 4, 4, 4, 4,
       4, 8, 8, 8, 8, 6, 6, 6, 6, 8, 8, 8, 8, 4, 4, 4, 4, 4, 4, 4, 4, 6,
       4, 3, 4, 4, 4, 4, 4, 8, 8, 8, 6, 6, 6, 4, 6, 6, 6, 6, 6, 6, 8, 6,
       8, 8, 4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 4, 6, 4, 4, 6, 6, 4, 6, 6, 8,
       8, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 5, 8, 4, 8, 4, 4, 4, 4, 4, 6, 6,
       4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4,
       4, 4, 4, 6, 3, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 8, 6, 6, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 4, 6, 4, 4, 4, 4, 4, 4,
       4, 4])
```

Lists, arrays, Series, and DataFrames

list

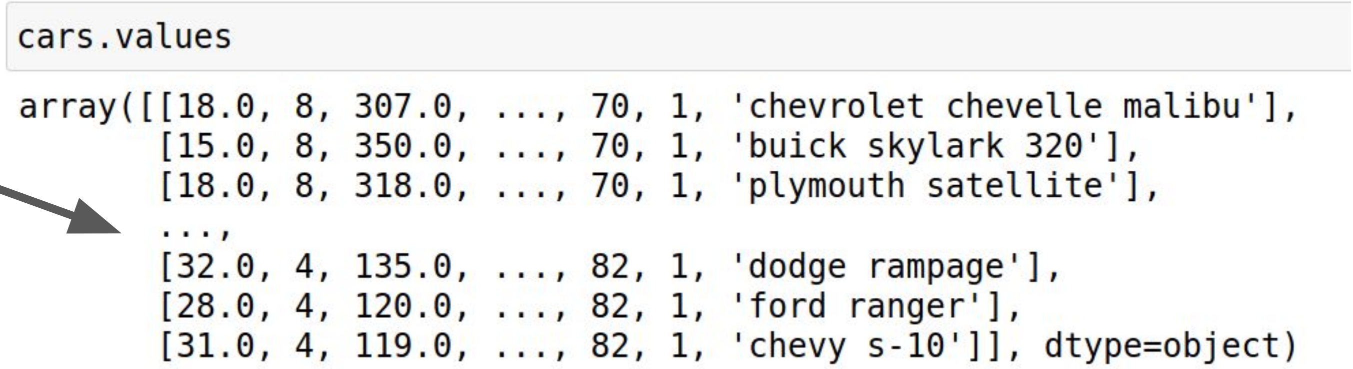


```
cars.cylinders.tolist()
```

```
[8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8,
 8
```

Lists, arrays, Series, and DataFrames

NumPy array



```
cars.values
```

```
array([[18.0, 8, 307.0, ..., 70, 1, 'chevrolet chevelle malibu'],
       [15.0, 8, 350.0, ..., 70, 1, 'buick skylark 320'],
       [18.0, 8, 318.0, ..., 70, 1, 'plymouth satellite'],
       ...,
       [32.0, 4, 135.0, ..., 82, 1, 'dodge rampage'],
       [28.0, 4, 120.0, ..., 82, 1, 'ford ranger'],
       [31.0, 4, 119.0, ..., 82, 1, 'chevy s-10']], dtype=object)
```

Lists, arrays, Series, and DataFrames

Question: Which of the following are homogeneous (all elements are the same type)?

A. pandas DataFrames
B. NumPy Arrays
C. Lists

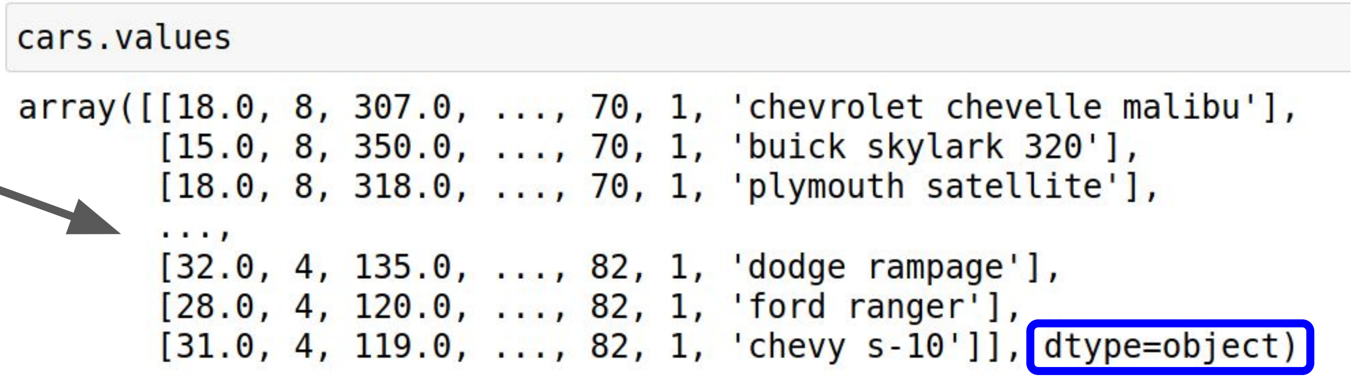
Lists, arrays, Series, and DataFrames

Question: Which of the following are homogeneous (all elements are the same type)?

A. pandas DataFrames
B. NumPy Arrays
C. Lists

Lists, arrays, Series, and DataFrames

NumPy array

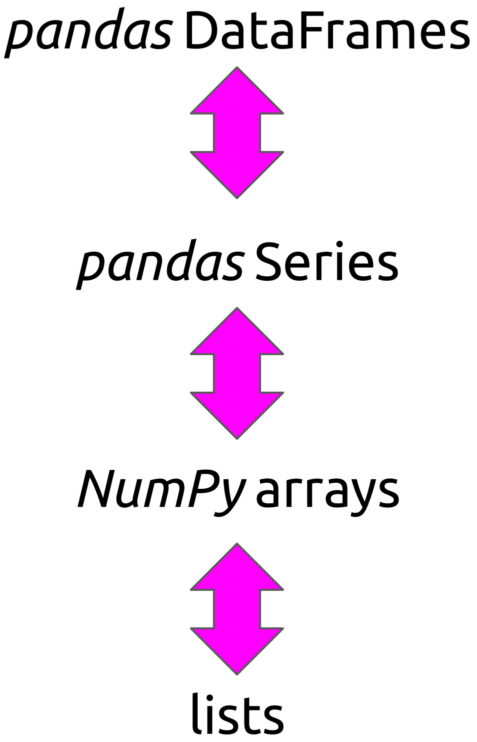


```
cars.values
```

```
array([[18.0, 8, 307.0, ..., 70, 1, 'chevrolet chevelle malibu'],
       [15.0, 8, 350.0, ..., 70, 1, 'buick skylark 320'],
       [18.0, 8, 318.0, ..., 70, 1, 'plymouth satellite'],
       ...,
       [32.0, 4, 135.0, ..., 82, 1, 'dodge rampage'],
       [28.0, 4, 120.0, ..., 82, 1, 'ford ranger'],
       [31.0, 4, 119.0, ..., 82, 1, 'chevy s-10']], dtype=object)
```

Everything is treated as an object.

Lists, arrays, Series, and DataFrames



pandas DataFrames

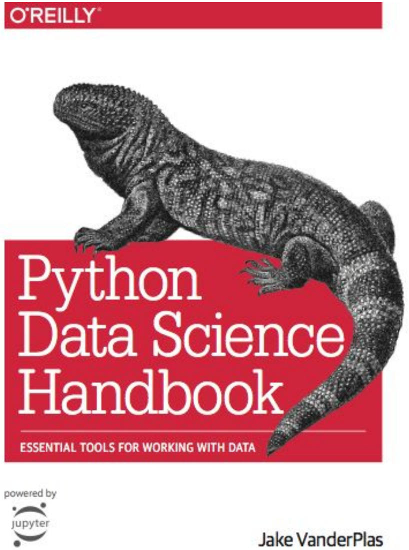
pandas Series

NumPy arrays

lists

Lists, arrays, Series, and DataFrames

For another reference for this topic (and lots of other Python data analysis topics), see section 3.1 of The Python Data Science Handbook


O'REILLY
Python Data Science Handbook
ESSENTIAL TOOLS FOR WORKING WITH DATA
powered by
jupyter
Jake VanderPlas