



INTRO TO PYTHON FOR DATA SCIENCE

NumPy



Lists Recap

- Powerful
- Collection of values
- Hold different types
- Change, add, remove
- Need for Data Science
 - Mathematical operations over collections
 - Speed



Illustration

```
In [1]: height = [1.73, 1.68, 1.71, 1.89, 1.79]
```

```
In [2]: height
```

```
Out[2]: [1.73, 1.68, 1.71, 1.89, 1.79]
```

```
In [3]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]
```

```
In [4]: weight
```

```
Out[4]: [65.4, 59.2, 63.6, 88.4, 68.7]
```

```
In [5]: weight / height ** 2
```

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



Solution: NumPy

- Numeric Python
- Alternative to Python List: NumPy Array
- Calculations over entire arrays
- Easy and Fast
- Installation
 - In the terminal: `pip3 install numpy`



NumPy

```
In [6]: import numpy as np
```

```
In [7]: np_height = np.array(height)
```

```
In [8]: np_height
```

```
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])
```

```
In [9]: np_weight = np.array(weight)
```

```
In [10]: np_weight
```

```
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])
```

```
In [11]: bmi = np_weight / np_height ** 2
```

```
In [12]: bmi
```

```
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```



NumPy

```
In [6]: import numpy as np
```

Element-wise calculations

```
In [7]: np_height = np.array(height)
```

```
In [8]: np_height
```

```
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])
```

```
In [9]: np_weight = np.array(weight)
```

```
In [10]: np_weight
```

```
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])
```

```
In [11]: bmi = np_weight / np_height ** 2
```

```
In [12]: bmi
```

```
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

```
= 65.5/1.73 ** 2
```



Comparison

```
In [13]: height = [1.73, 1.68, 1.71, 1.89, 1.79]
```

```
In [14]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]
```

```
In [15]: weight / height ** 2
```

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

```
In [16]: np_height = np.array(height)
```

```
In [17]: np_weight = np.array(weight)
```

```
In [18]: np_weight / np_height ** 2
```

```
Out[18]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```



NumPy: remarks

```
In [19]: np.array([1.0, "is", True])  
Out[19]:  
array(['1.0', 'is', 'True'],  
      dtype='<U32')
```

NumPy arrays: contain only one type

```
In [20]: python_list = [1, 2, 3]
```

```
In [21]: numpy_array = np.array([1, 2, 3])
```

Different types: different behavior!

```
In [22]: python_list + python_list  
Out[22]: [1, 2, 3, 1, 2, 3]
```

```
In [23]: numpy_array + numpy_array  
Out[23]: array([2, 4, 6])
```




NumPy Subsetting

```
In [24]: bmi
Out[24]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [25]: bmi[1]
Out[25]: 20.975

In [26]: bmi > 23
Out[26]: array([False, False, False,  True, False], dtype=bool)

In [27]: bmi[bmi > 23]
Out[27]: array([ 24.747])
```



INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!



INTRO TO PYTHON FOR DATA SCIENCE

2D NumPy Arrays



Type of NumPy Arrays

```
In [1]: import numpy as np
```

```
In [2]: np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])
```

```
In [3]: np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])
```

```
In [4]: type(np_height)
```

```
Out[4]: numpy.ndarray
```

ndarray = N-dimensional array

```
In [5]: type(np_weight)
```

```
Out[5]: numpy.ndarray
```



2D NumPy Arrays

```
In [6]: np_2d = np.array([[1.73, 1.68, 1.71, 1.89, 1.79],  
                          [65.4, 59.2, 63.6, 88.4, 68.7]])
```

```
In [7]: np_2d  
Out[7]:  
array([[ 1.73,   1.68,   1.71,   1.89,   1.79],  
       [ 65.4 ,   59.2 ,   63.6 ,   88.4 ,   68.7 ]])
```

```
In [8]: np_2d.shape  
Out[8]: (2, 5) 2 rows, 5 columns
```

```
In [9]: np.array([[1.73, 1.68, 1.71, 1.89, 1.79],  
                  [65.4, 59.2, 63.6, 88.4, "68.7"]])
```

```
Out[9]:  
array([[ '1.73', '1.68', '1.71', '1.89', '1.79'],  
       [ '65.4', '59.2', '63.6', '88.4', '68.7']],  
      dtype='<U32')
```

Single type!



Subsetting

	0	1	2	3	4	
array([[1.73,	1.68,	1.71,	1.89,	1.79],	0
	65.4,	59.2,	63.6,	88.4,	68.7]])	1

```
In [10]: np_2d[0]
```

```
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])
```

```
In [11]: np_2d[0][2]
```

```
Out[11]: 1.71
```

```
In [12]: np_2d[0,2]
```

```
Out[12]: 1.71
```



Subsetting

	0	1	2	3	4	
array([[1.73,	1.68,	1.71,	1.89,	1.79],	0
	65.4,	59.2,	63.6,	88.4,	68.7]])	1

```
In [10]: np_2d[0]
```

```
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])
```

```
In [11]: np_2d[0][2]
```

```
Out[11]: 1.71
```

```
In [12]: np_2d[0,2]
```

```
Out[12]: 1.71
```

```
In [13]: np_2d[:,1:3]
```

```
Out[13]:
```

```
array([[ 1.68,  1.71],
       [ 59.2 ,  63.6 ]])
```



Subsetting

	0	1	2	3	4	
array([[1.73,	1.68,	1.71,	1.89,	1.79],	0
[65.4,	59.2,	63.6,	88.4,	68.7])	1

```
In [10]: np_2d[0]
```

```
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])
```

```
In [11]: np_2d[0][2]
```

```
Out[11]: 1.71
```

```
In [12]: np_2d[0,2]
```

```
Out[12]: 1.71
```

```
In [13]: np_2d[:,1:3]
```

```
Out[13]:
```

```
array([[ 1.68,  1.71],
       [ 59.2 ,  63.6 ]])
```

```
In [14]: np_2d[1,:]
```

```
Out[14]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])
```




INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!



INTRO TO PYTHON FOR DATA SCIENCE

NumPy: Basic Statistics



Data analysis

- Get to know your data
- Little data → simply look at it
- Big data → ?



City-wide survey

```
In [1]: import numpy as np
```

```
In [2]: np_city = ... # Implementation left out
```

```
In [3]: np_city
```

```
Out[3]:
```

```
array([[ 1.64,  71.78],  
       [ 1.37,  63.35],  
       [ 1.6 ,  55.09],  
       ...,  
       [ 2.04,  74.85],  
       [ 2.04,  68.72],  
       [ 2.01,  73.57]])
```



NumPy

```
In [4]: np.mean(np_city[:,0])  
Out[4]: 1.7472
```

```
In [5]: np.median(np_city[:,0])  
Out[5]: 1.75
```

```
In [6]: np.corrcoef(np_city[:,0], np_city[:,1])  
Out[6]:  
array([[ 1.         , -0.01802],  
       [-0.01803,  1.         ]])
```

```
In [7]: np.std(np_city[:,0])  
Out[7]: 0.1992
```

- `sum()`, `sort()`, ...
- Enforce single data type: speed!



Generate data

distribution
mean

distribution
standard dev.

number of
samples

```
In [8]: height = np.round(np.random.normal(1.75, 0.20, 5000), 2)
```

```
In [9]: weight = np.round(np.random.normal(60.32, 15, 5000), 2)
```

```
In [10]: np_city = np.column_stack((height, weight))
```



INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!