7 Secrets to master Numpy

You need to master Numpy

- Data manipulation is extremely important
  - Foundation for almost everything else in data science and machine learning.

- To be a great data scientist, you need to master data manipulation tools.

- And if you're doing data manipulation in Python, that means that you need to learn Numpy.

- Numpy is easy in many ways, but many people still get stuck on it.

- So in this tutorial, I'll give you 7 tips to help you learn Numpy.

1. Use the 80/20 rule

SHARP SIGHT

The 80/20 rule helps you simplify NumPy

- Numpy has over 100 functions, methods, and tools
  - BUT, you'll use a few over and over

- Identify, learn, and master the most commonly used tools

Here's a quick list of the numpy tools I use most often

- -np.array
- -np.arange
- -np.random.seed
- -np.random.uniform
- -np.random.normal
- -np.linspace
- -np.transpose
- -np.meshgrid
- -np.where
- -np.dot
- -np.exp
- -np.min
- -np.max
- -np.argmin
- -np.argmax
- -.ravel
- -.reshape

2. Learn how arrays are structured

SHARP SIGHT

Learn how arrays are structured

- Numpy arrays store numeric data in a row-and-column structure
- They can be 1-dimensional, 2-dimensional, or multi-dimensional.
- The shape and dimensions matter, because some tools work differently based on size and dims.
  - check the shape with `my_array.shape`

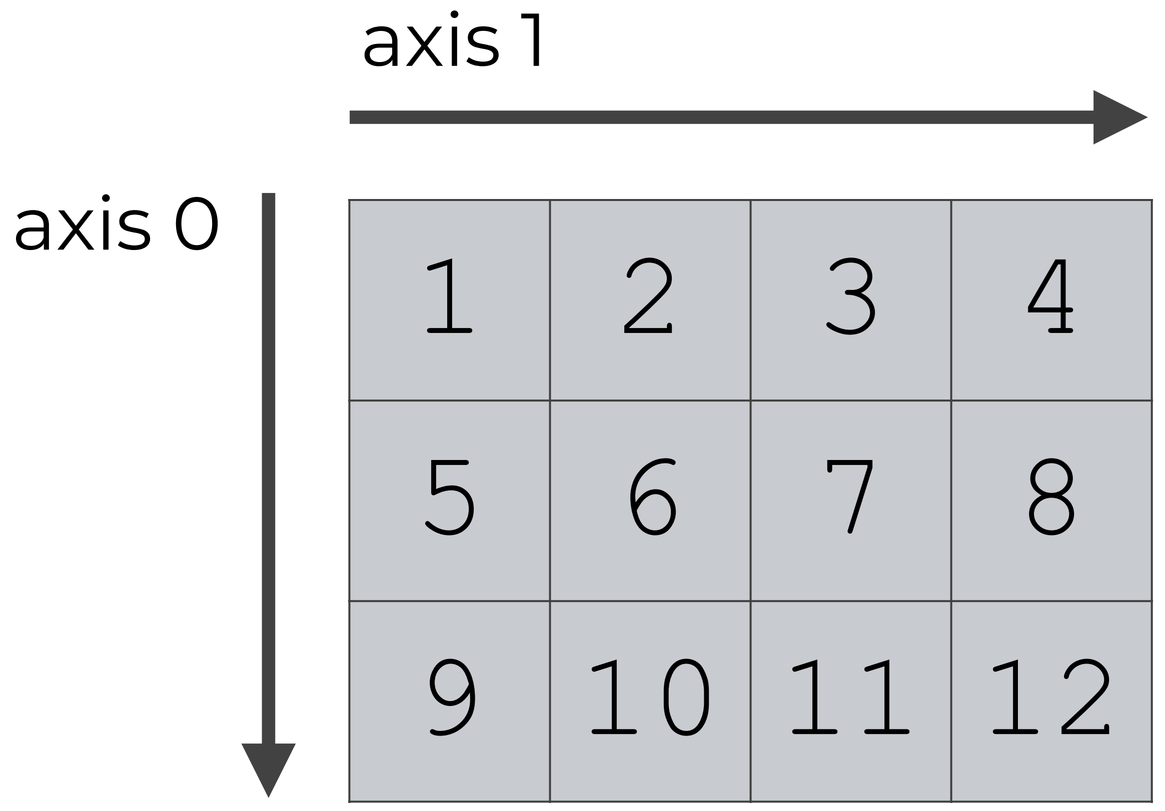
Numpy Arrays store numeric data in a row-and-column structure

4	12	0	4
6	11	8	4
18	14	13	7

3: LEARN NUMPY AXES

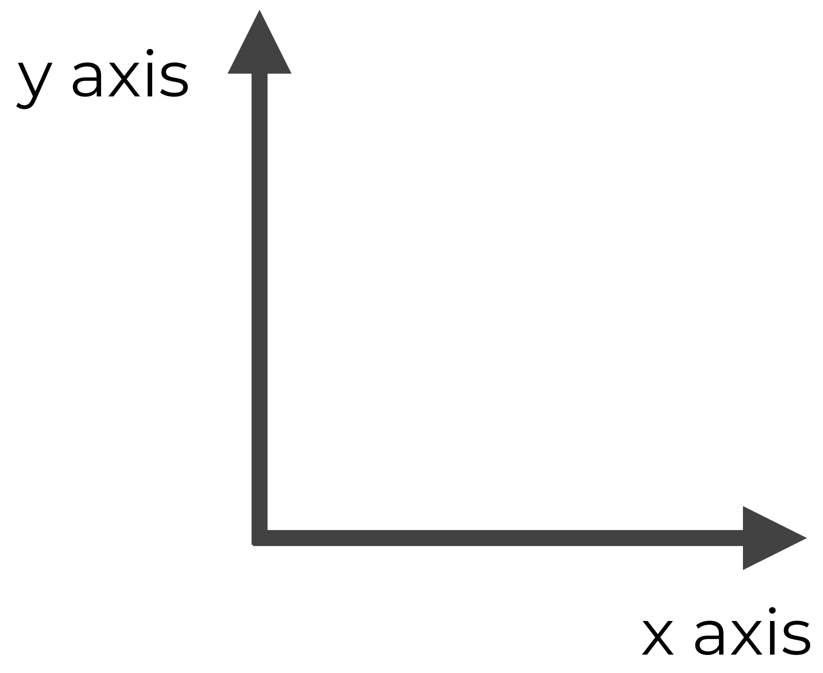
SHARP SIGHT

NUMPY ARRAYS HAVE *AXES*



Array axes are very similar to axes in coordinate systems

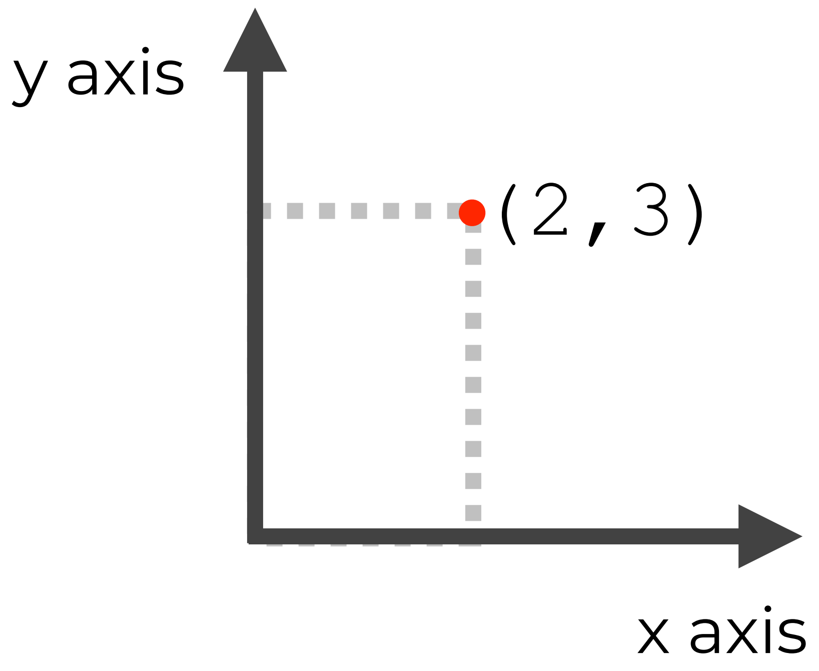
Array axes are like axes in a coordinate system



For example, a Cartesian coordinate system has an x axis and y axis

These axes are like directions in space

In a coordinate system, points can be defined by values along the axes

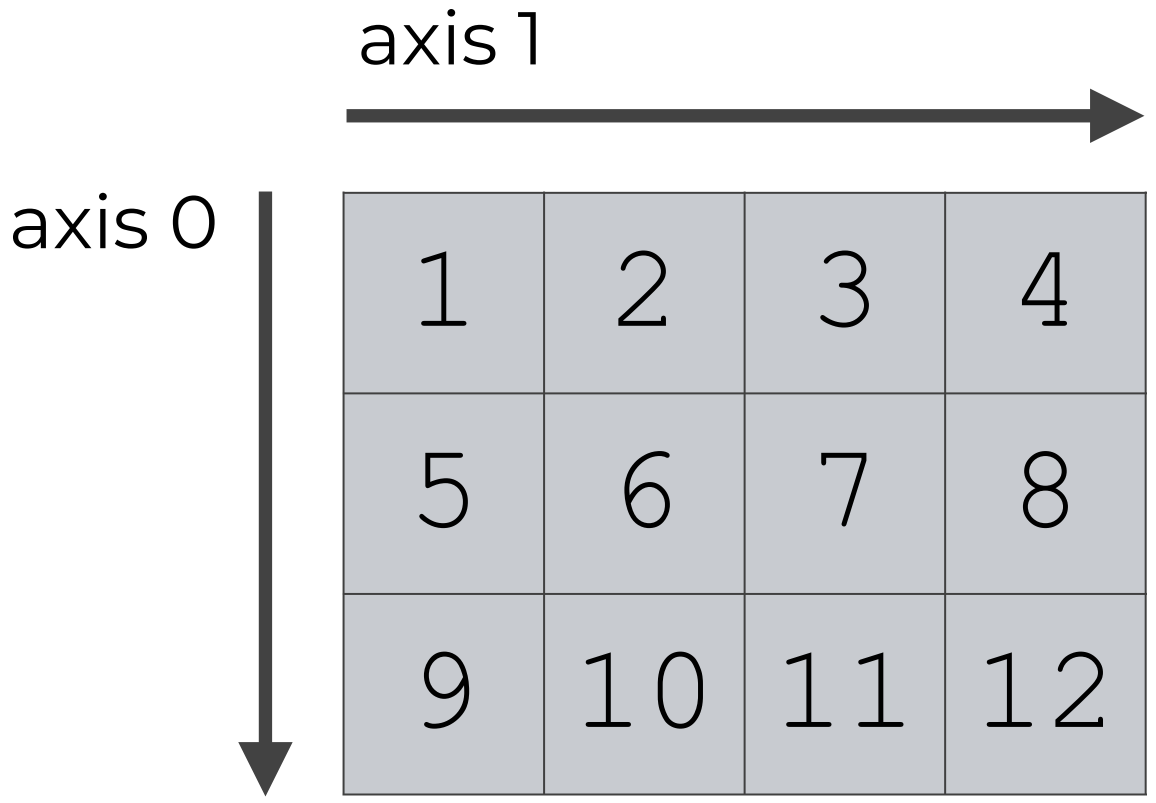


Here, the point lies at
2 units along the x axis,
and 3 units along the y axis

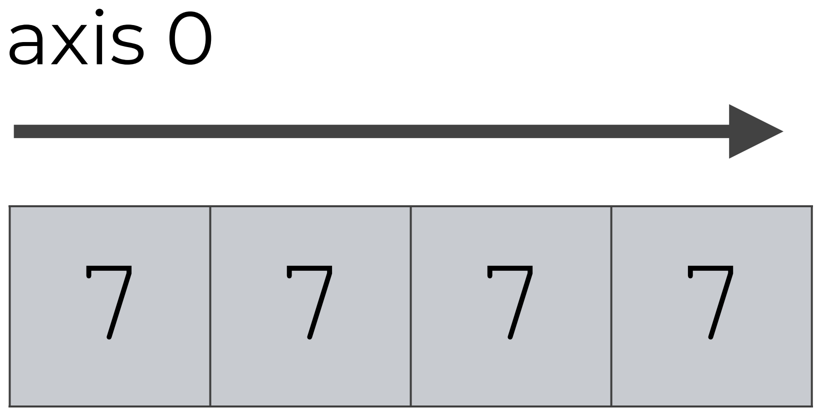
Axes are like *directions* along a Numpy array

Axis-1 is the direction that runs horizontally across the columns

Axis-0 is the direction that runs downward down the rows



In a 1-dimensional array, the first axis is axis-0

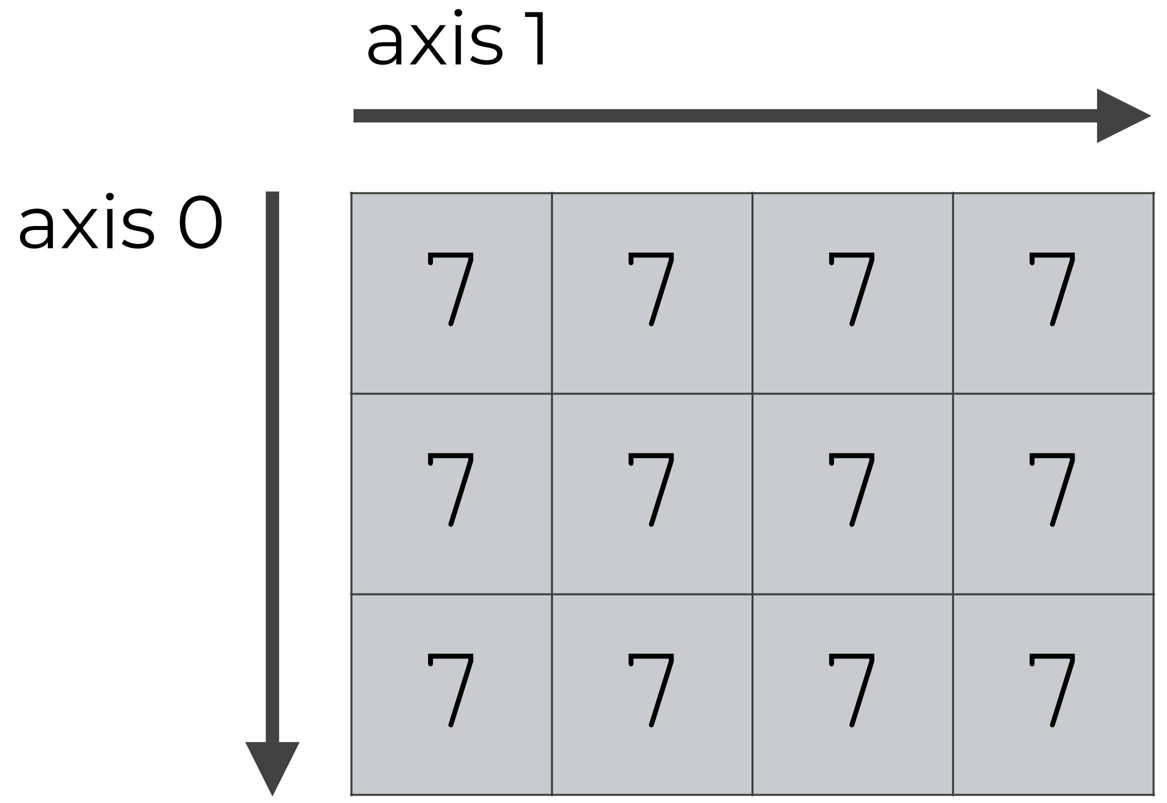


Just remember, 1-dimensional arrays are a little different

In a 2-dimensional array, axis-0 is down and axis-1 is across

Axis-1 is the direction that runs horizontally across the columns

Axis-0 is the direction that runs downward down the rows



Numpy axes are important!

- We will use axes when we use many functions
  - `np.sum()`
  - `np.mean()`
  - `np.concatenate()`
  - `np.sort()`
  - etc

- We commonly use axes when we need to aggregate, sort, or manipulate

- Make sure you understand them!
  - It's best to memorize them

4: LEARN THE IMPORTANT ARRAY CREATION METHODS

SHARP SIGHT

You need to know the main ways to create arrays

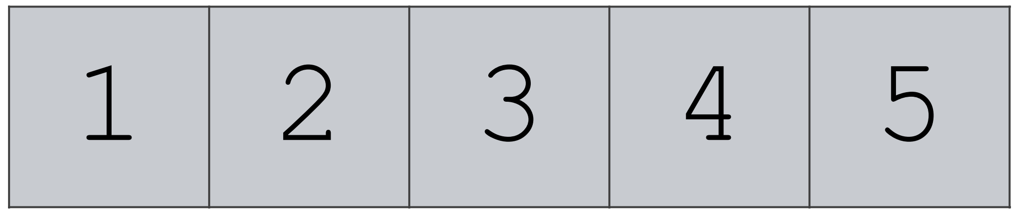
- Once you understand the essentials of array structure and array axes, you need to know how to create arrays.

- There are many, many ways to create Numpy arrays, but the ones that I use most often are:
  - the Numpy array function
  - Numpy arange
  - Numpy linspace
  - numpy ones and Numpy zeroes
  - a few of the Numpy random functions like Numpy random normal and Numpy random choice

NP.ARRAY CREATES AN ARRAY FROM A LIST

```
np_array_1d = np.array([1,2,3,4,5])

print(np_array_1d)
```

OUT:

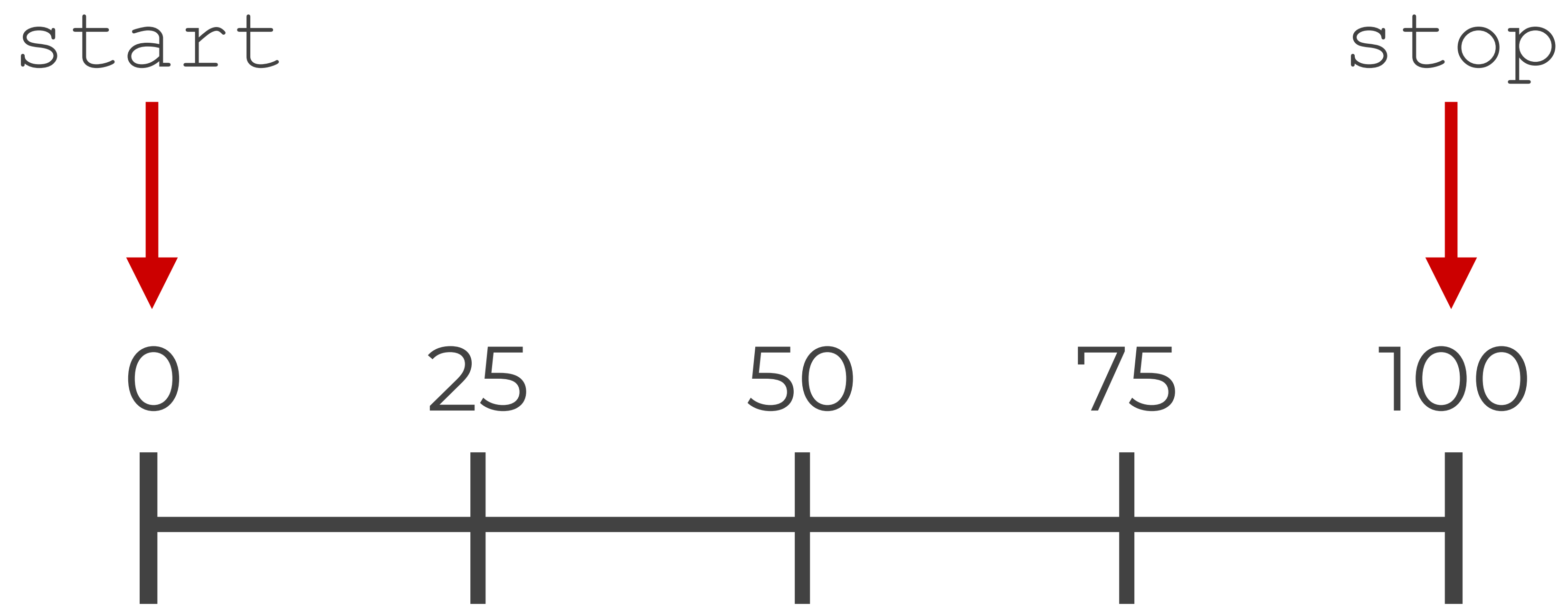


NP.ARANGE CREATES ARRAYS WITHIN SPECIFIC RANGES OF NUMBERS

Here, we're using the `start` parameter and the `stop` parameter to specify the start and stop point of the sequence

```
np.arange(start = 5, stop = 11)
array([ 5,  6,  7,  8,  9, 10])
```

Numpy linspace creates ranges of equally spaced values



NP.LINSPACE CREATES NUMPY ARRAYS WITH EVENLY SPACED VALUES WITHIN A RANGE

Five values between
0 and 1

0	.25	.5	.75	1

Five values between
0 and 100

0	25	50	75	100

Four values between
10 and 40

10	20	30	40

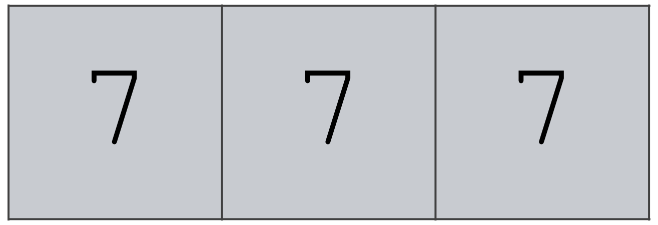
5: LEARN HOW TO RESHAPE ARRAYS

SHARP SIGHT

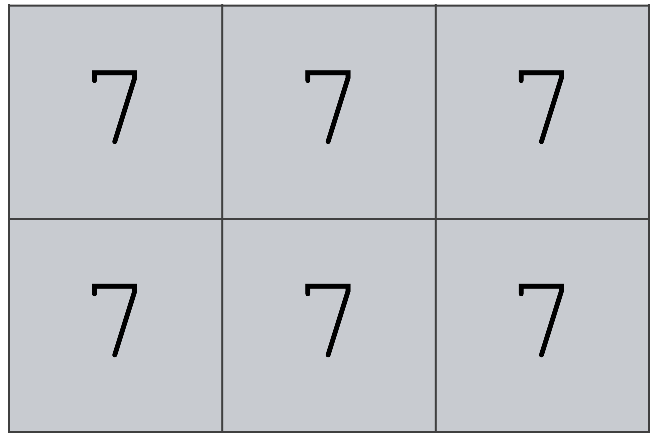
REMEMBER: ALL NUMPY ARRAYS HAVE A "SHAPE"

The `shape` is essentially the number of rows and columns

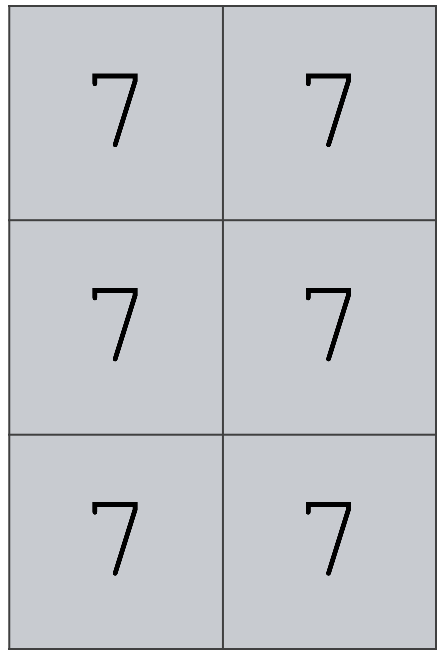
`shape (3,)`



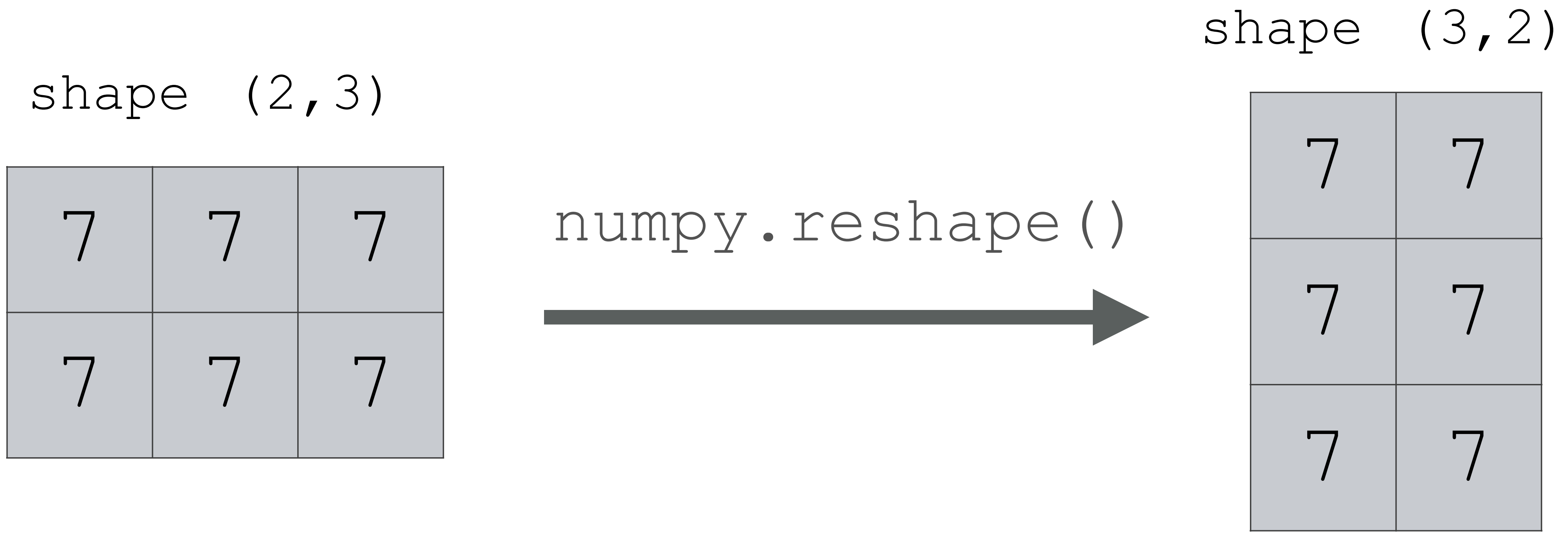
`shape (2,3)`



`shape (3,2)`

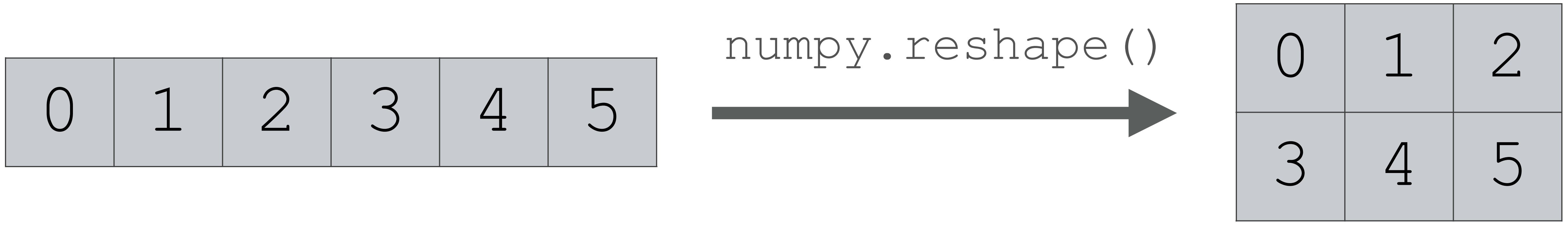


NUMPY RESHAPE CHANGES THE SHAPE OF A NUMPY ARRAY



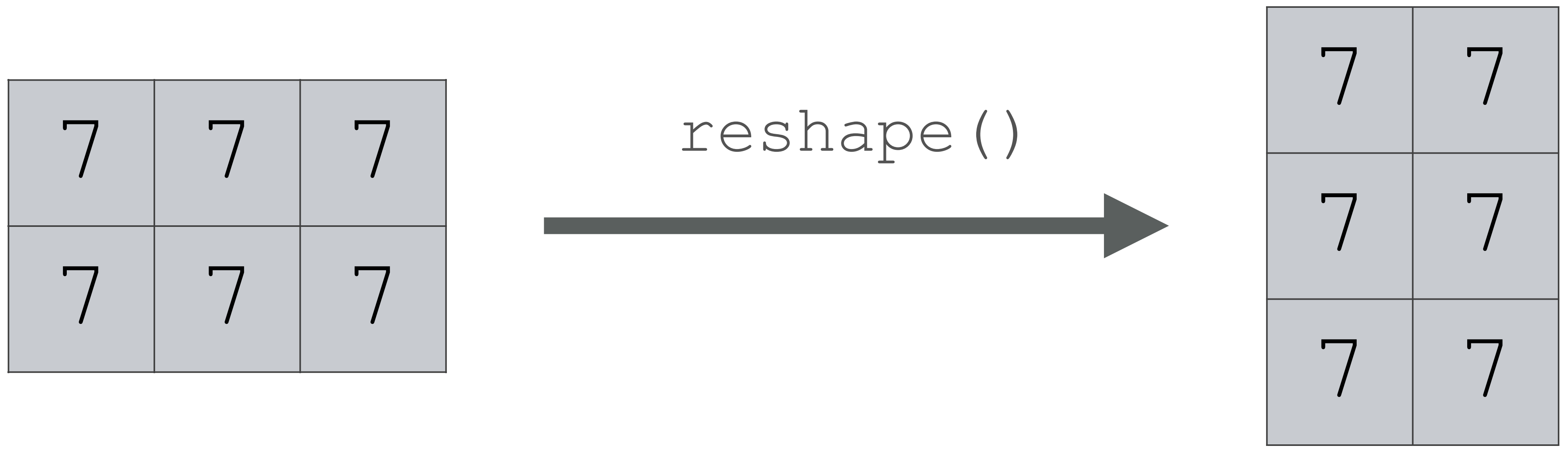
Reshape from 1-dimensional to 2-dimensional

```
np_array_1d = np.arange(start = 0, stop = 6)


np.reshape(a = np_array_1d, newshape = (2,3))
```



Reshape a 2-dimensional array to a new shape

```
np_array_2x3 = np.full(shape = (2,3),fill_value = 7)

np.reshape(a = np_array_2x3, newshape = (3,2))
```



6: LEARN NUMPY AGGREGATION TECHNIQUES

SHARP SIGHT

You often need to aggregate arrays for data cleaning and analysis

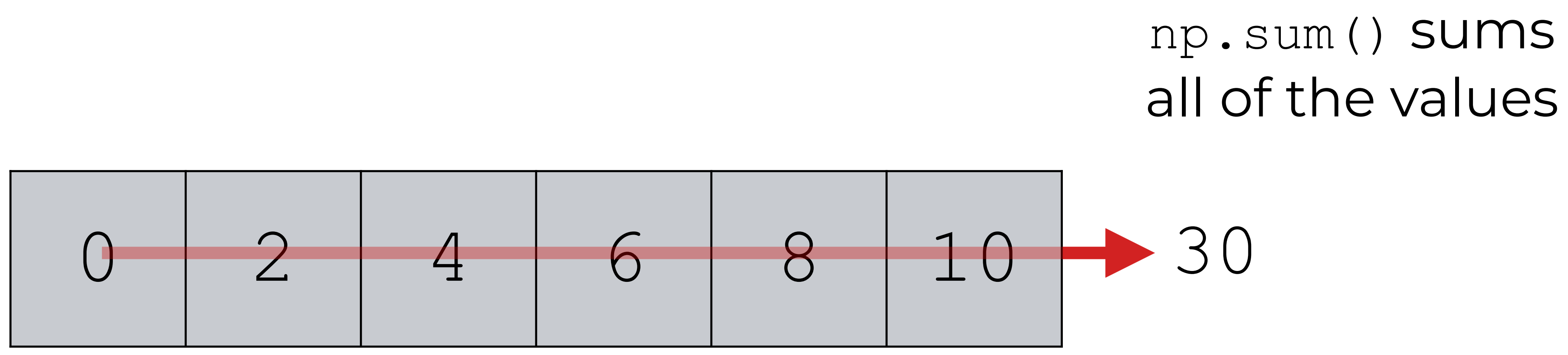
- Next, you should learn some array aggregation techniques.

- You'll commonly use these techniques when you need to analyze the numbers in your array by computing summary statistics.

- The most common:
  - Numpy sum
  - Numpy mean
  - Numpy median
  - Numpy min
  - Numpy max

NUMPY SUM SUMS THE VALUES OF A NUMPY ARRAY

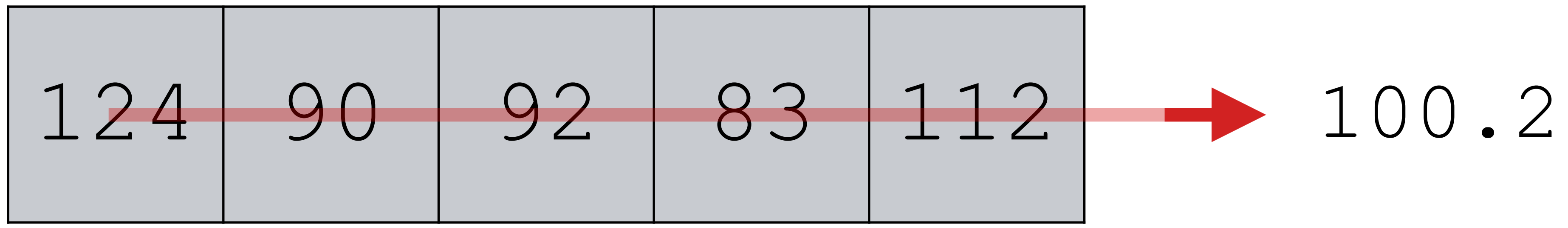
```
np.sum(array_evens)
```



NUMPY MEAN COMPUTES THE MEAN OF THE VALUES OF A NUMPY ARRAY

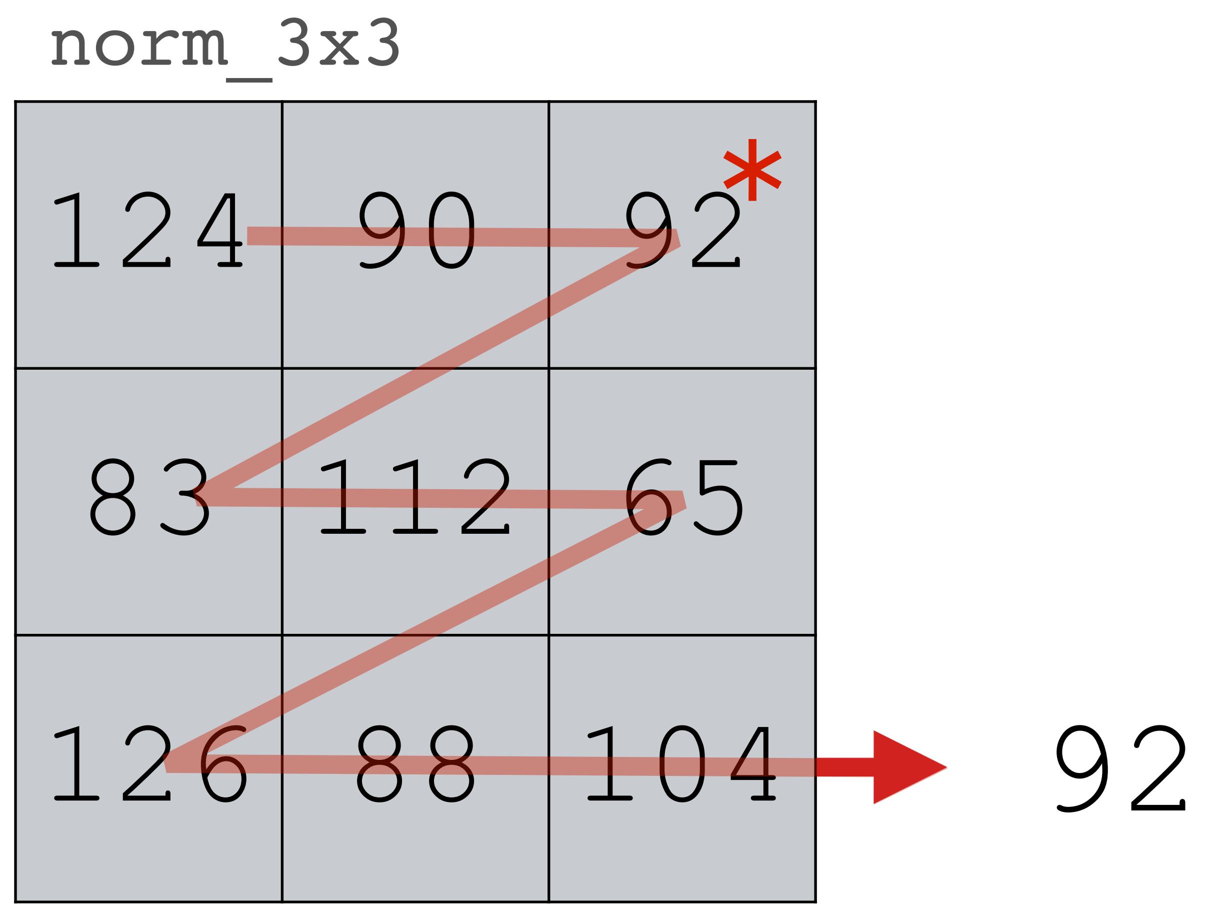
```
np.mean(norm_5)
```

`np.mean()` calculates the mean of all of the values



NUMPY MEDIAN COMPUTES THE MEDIAN

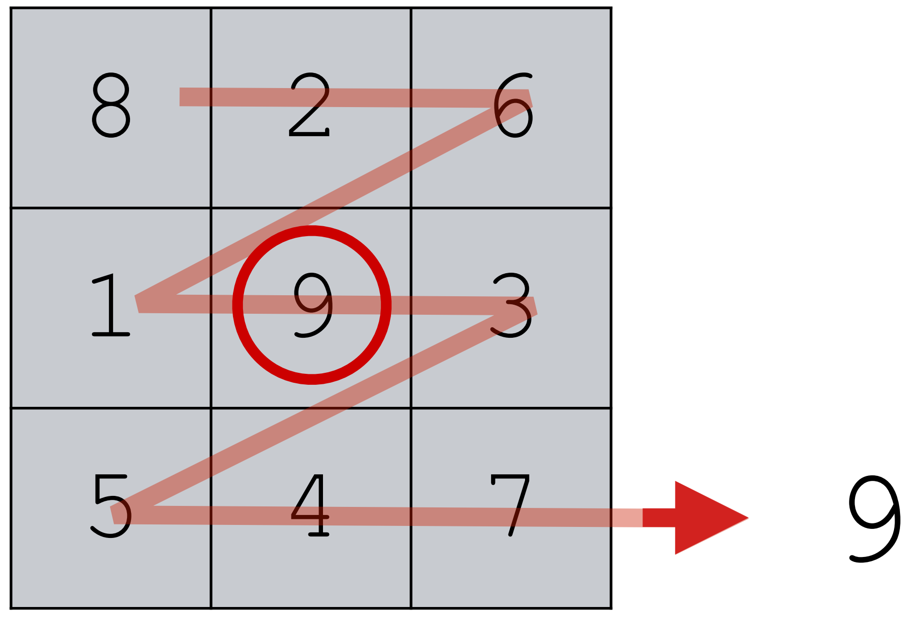
```
np.random.seed(1)
norm_3x3 = np.random.normal(size = (3,3), loc = 100, scale = 15).astype(int)


np.median(norm_3x3)
```



If we use `np.median()` on a 2D array *without* specifying an axis, the function calculates the median of all of the values

NUMPY MAX IDENTIFIES THE MAXIMUM

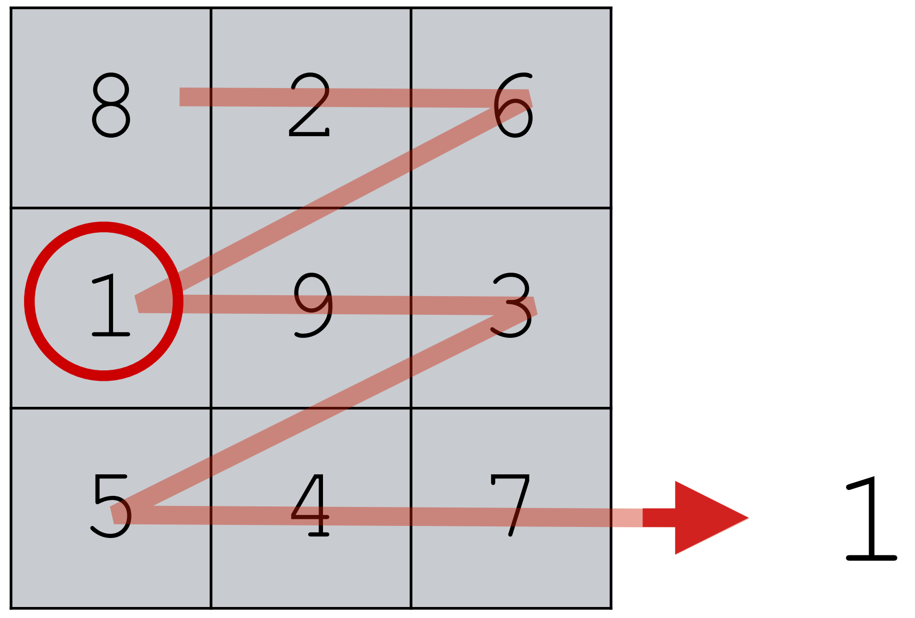
```
integers_1to9 = range(1,10)
np.random.seed(42)
random_3x3 = np.random.choice(a = integers_1to9, size = (3,3),replace = False)

np.max(random_3x3)
```



If we use `np.max()` on a 2D array *without* specifying an axis, the function calculates the maximum of all of the values

NUMPY MIN FINDS THE MINIMUM VALUE

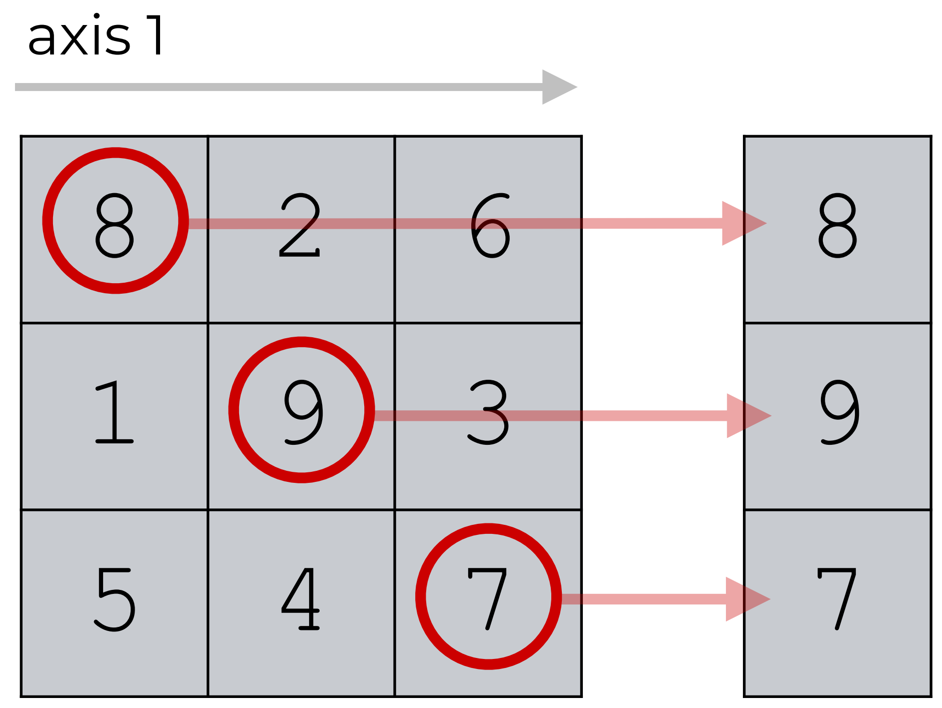
```
integers_1to9 = range(1,10)
np.random.seed(42)
random_3x3 = np.random.choice(a = integers_1to9, size = (3,3),replace = False)

np.min(random_3x3)
```



If we use `np.min()` on a 2D array *without* specifying an axis, the function calculates the minimum of all of the values

Note that we can use these aggregation functions along an axis

```
integers_1to9 = range(1,10)
np.random.seed(42)
random_3x3 = np.random.choice(a = integers_1to9, size = (3,3),replace = False)

np.max(random_3x3, axis = 1)
```



If we use `np.max()` on a 2D array with `axis = 1`, the function calculates the maxima of the rows

(i.e., calculates maxima in the axis-1 direction)

7: LEARN NUMPY ARRAY MATH

SHARP SIGHT

We often need to perform math operations on entire matrices

- At minimum, you should know how to add and subtract arrays (using np.add and np.subtract).

- You should also probably know some simple math functions to operate on the individual elements of the array
  - np.power, np.log, and np.exp.

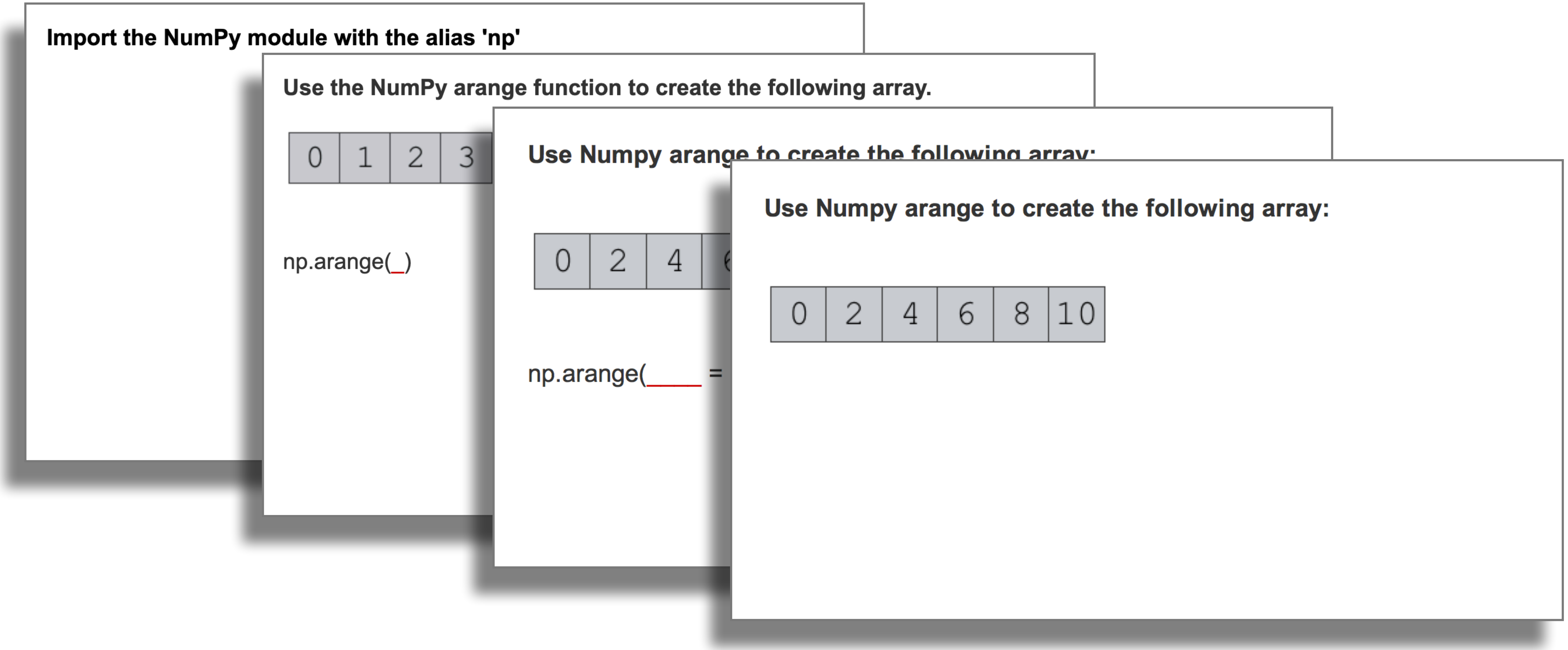
- linear algebra operations. These can be useful for machine learning and deep learning
  - np.power, np.log, and np.exp.

One extra secret ....

SHARP SIGHT

YOU NEED TO PRACTICE

Find a way to practice an drill syntax



Import the NumPy module with the alias 'np'

Use the NumPy arange function to create the following array.

0	1	2	3

np.arange(_)

Use Numpy arange to create the following array:

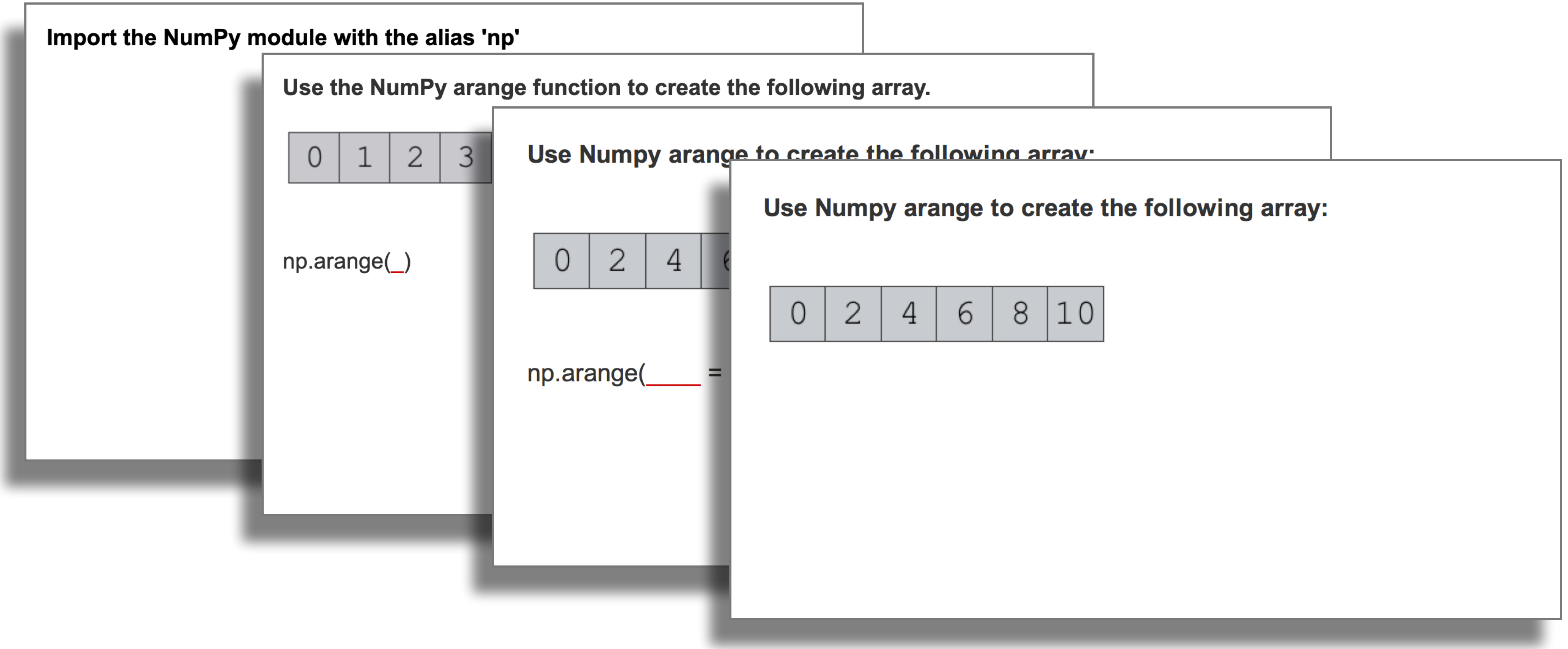
0	2	4	6

np.arange(____ =

Use Numpy arange to create the following array:

0	2	4	6	8	10

Practice is the secret to learning any skill, and it's the secret to mastering NumPy



Import the NumPy module with the alias 'np'

Use the NumPy arange function to create the following array.

0	1	2	3

np.arange(_)

Use Numpy arange to create the following array:

0	2	4

np.arange(____ =

Use Numpy arange to create the following array:

0	2	4	6	8	10

If you use these secrets, you'll be able to master Numpy fast

SHARP SIGHT

If you use these secrets, you'll be able to master Numpy fast

1. Use the 80/20 rule
2. Learn how arrays are structured
3. Learn how axes work
4. Learn the important array creation tools
5. Learn how to reshape arrays
6. Learn array aggregation techniques
7. Learn array math

- And practice

If you use these secrets, you'll be able to master Numpy fast

- You should be able to master Numpy in 2-4 weeks

- You need to focus on the right tools (and ignore the rest) you should be able to learn everything quickly.

- But, this also requires drilling and practice.

Thanks for reading. If you liked this tutorial, give it a like and subscribe to my account for more advice on data science and AI.

SHARP SIGHT