The World of Numpy

Python Mauritius UserGroup (pymug)

More info: mscc.mu/python-mauritius-usergroup-pymug/

Where Are We? 🏠

Why	Where
codes	github.com/pymug
share events	twitter.com/pymugdotcom
ping professionals	linkedin.com/company/pymug
all info	pymug.com
discuss	facebook.com/groups/318161658897893
tell friends by like	facebook.com/pymug

Support Us

Please support us by subscribing to our mailing list:
https://mail.python.org/mailman3/lists/pymug.python.org/

Abdur-Rahmaan Janhangeer
(Just a Python programmer)

twitter: @osdotsystem
github: github.com/abdur-rahmaanj

www.pythonmembers.club

The World of Numpy

⚠️ Best to use ***Jupyter*** from ***Anaconda*** to try out the examples

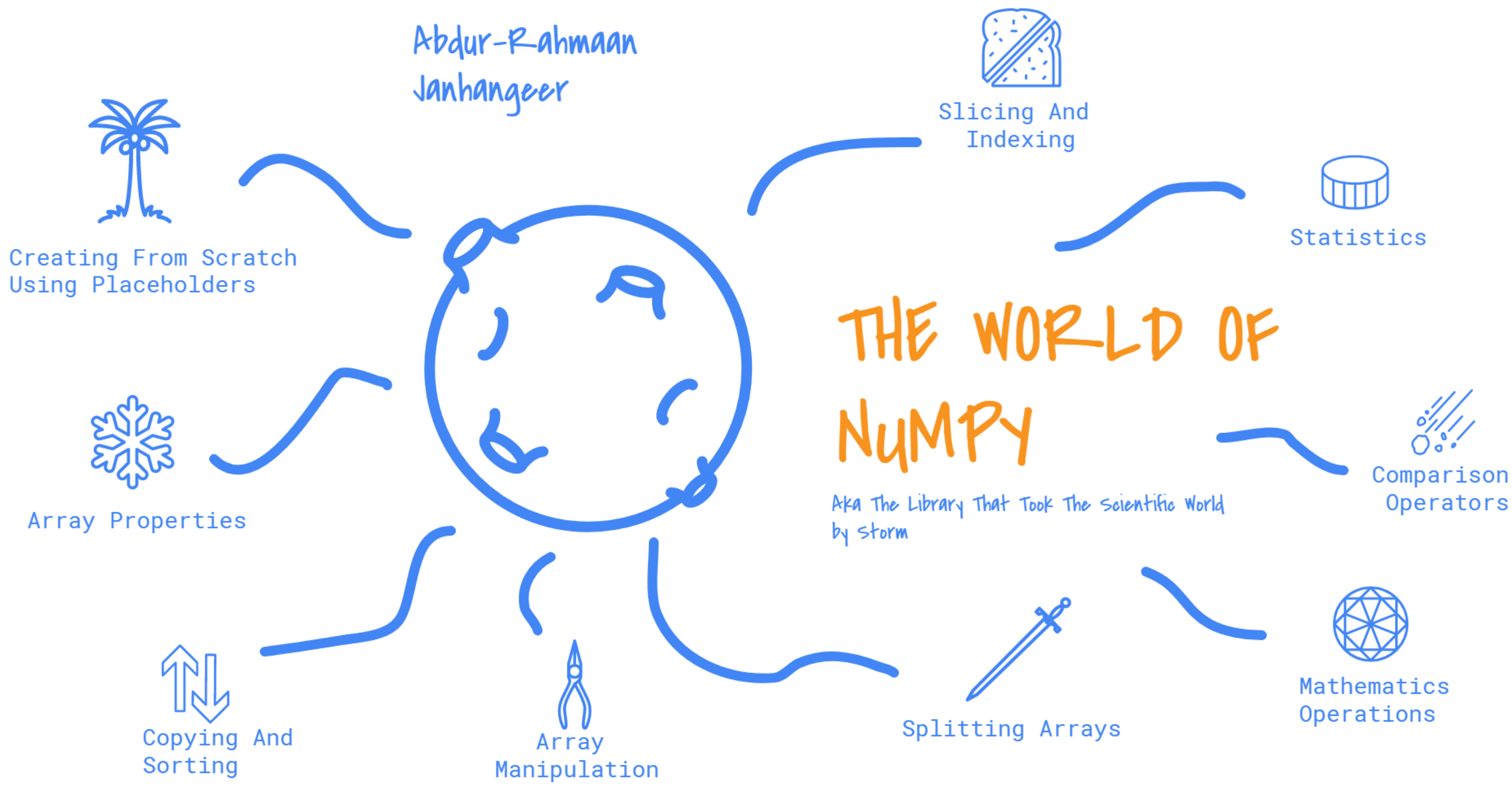
Jupyter Shortcuts

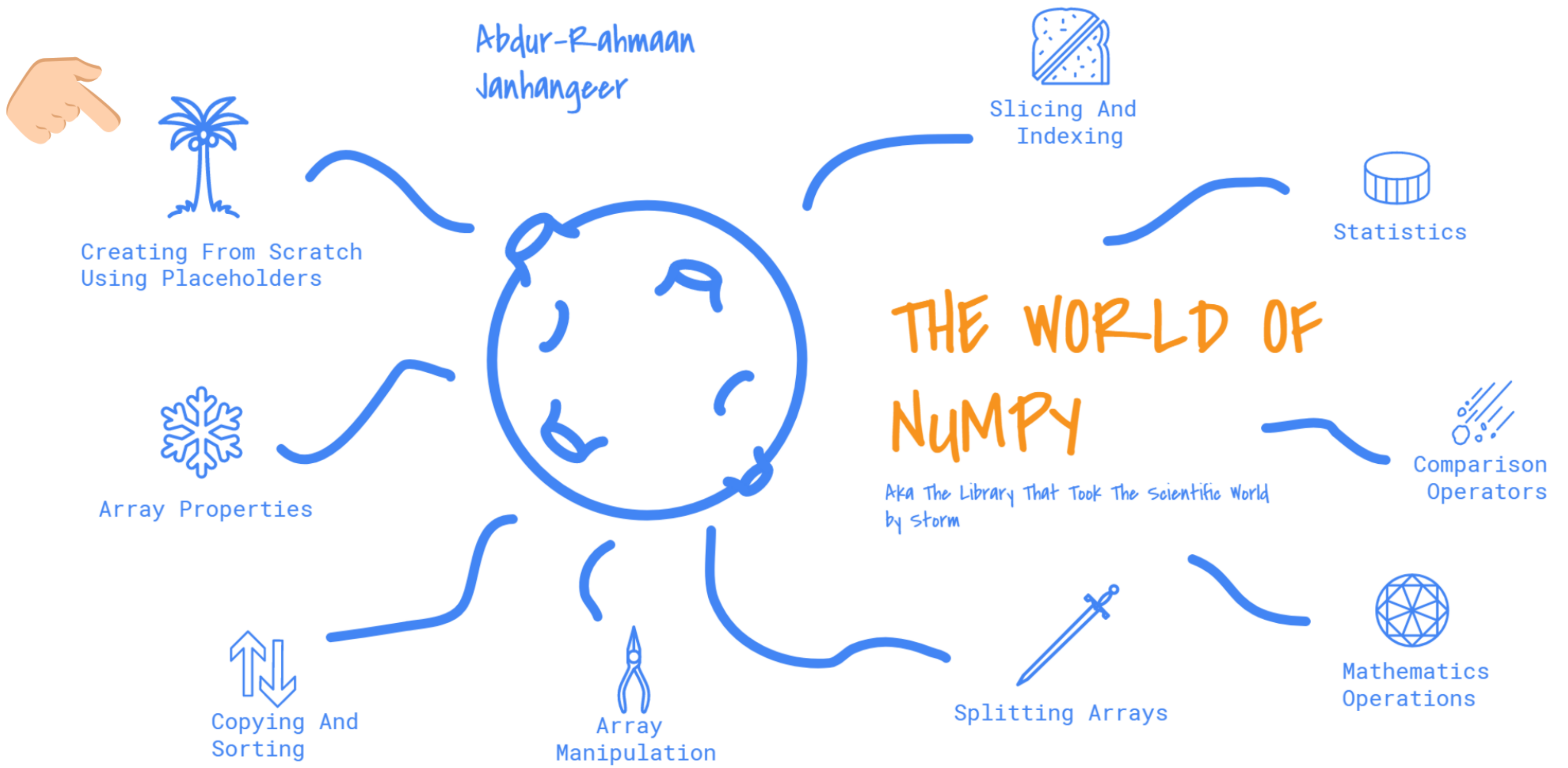
`alt + enter` : run with new cell

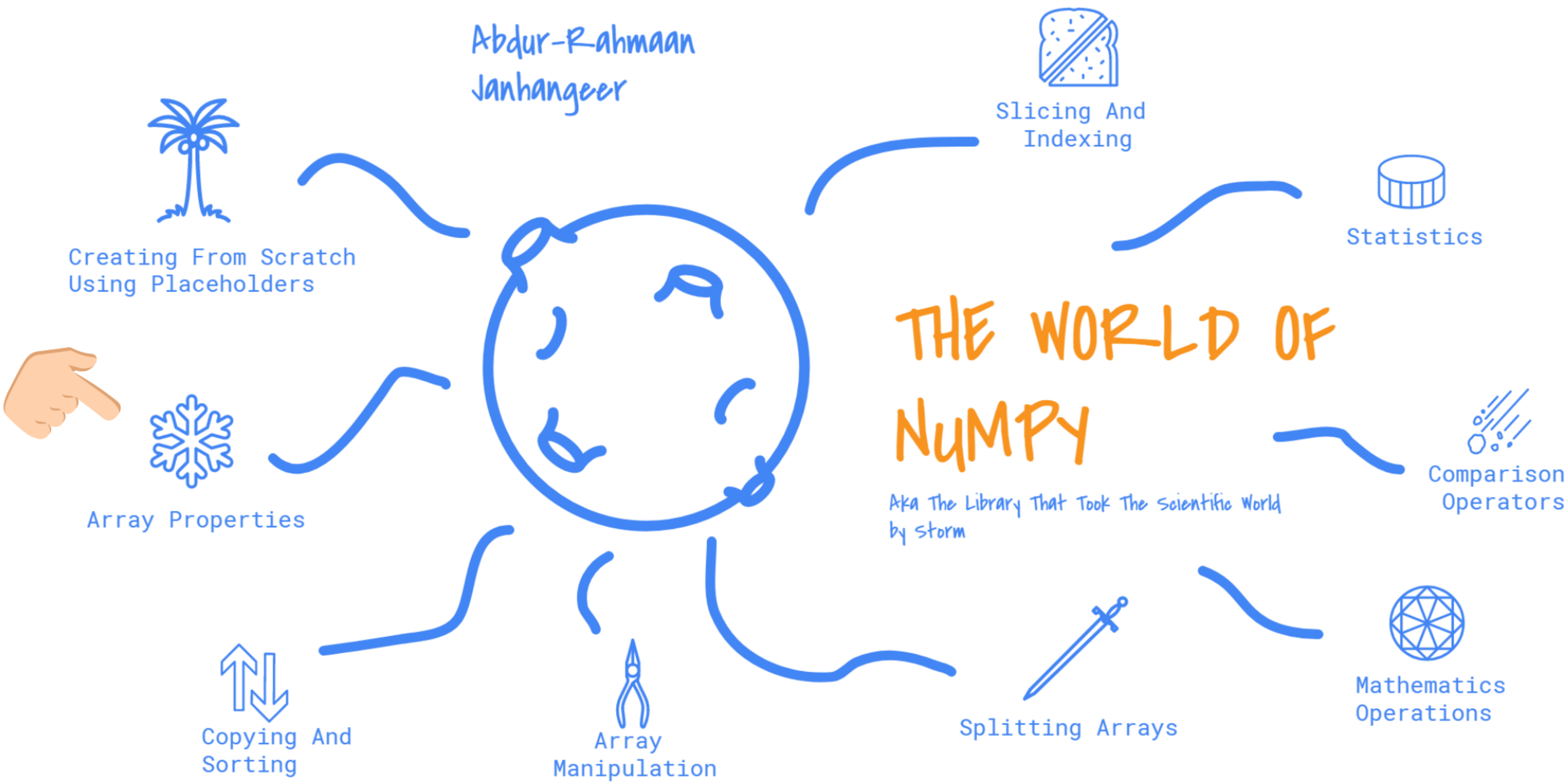
`ctrl + enter` : run

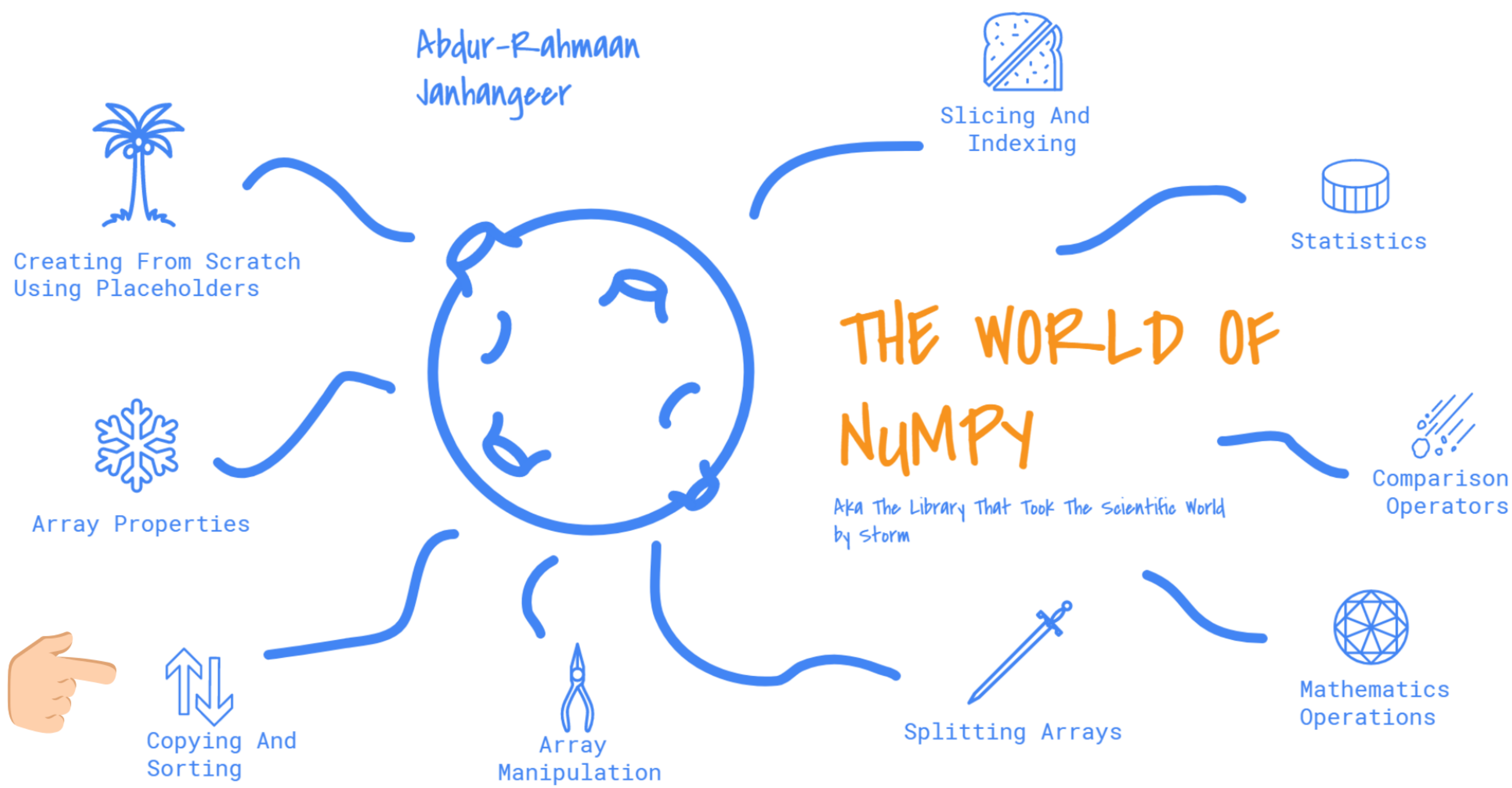
Numpy

- Top 10 imports, including std lib

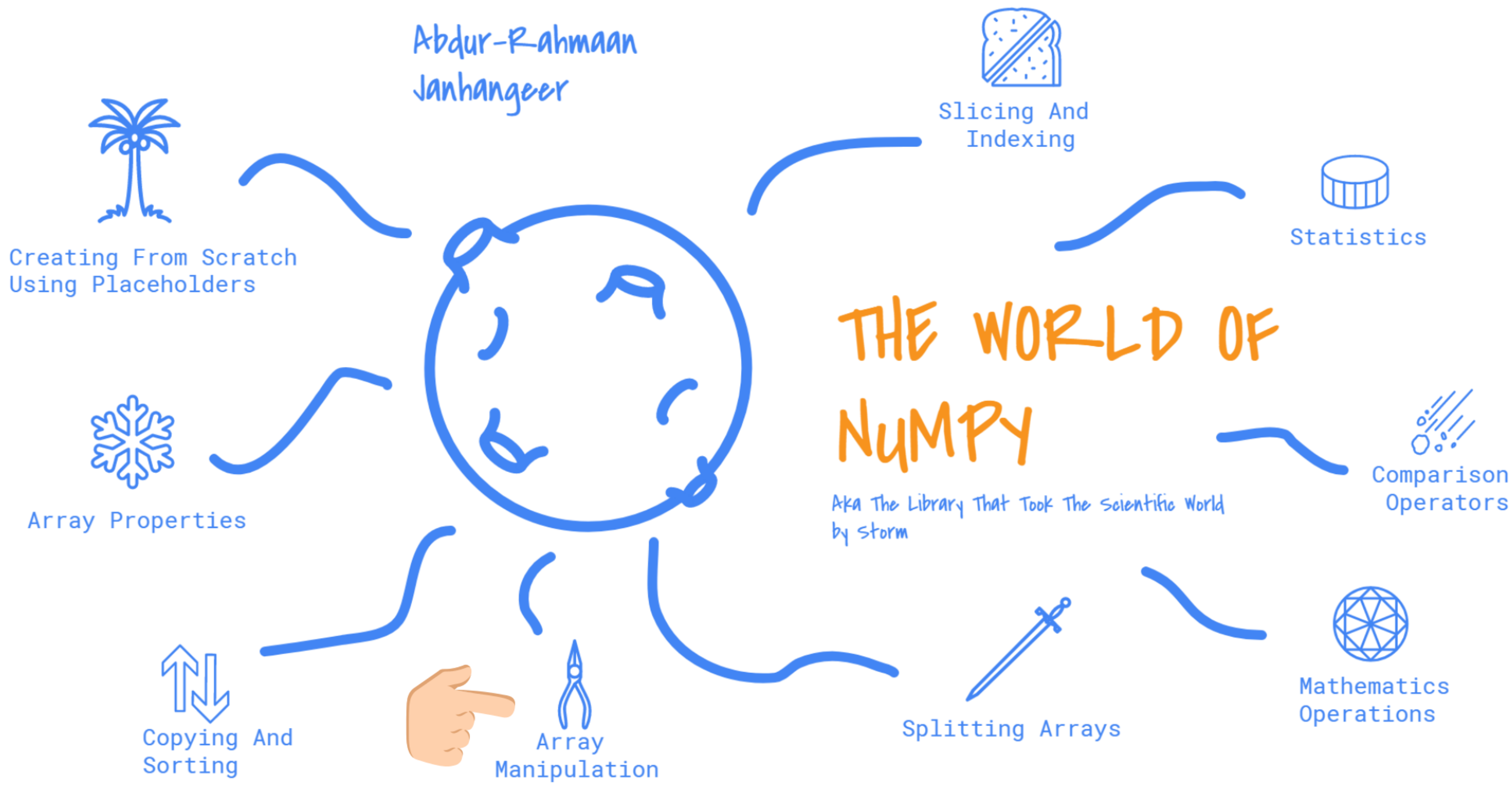

Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by storm
Creating From Scratch Using Placeholders
Array Properties
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations
Comparison Operators
Statistics
Slicing And Indexing


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Creating From Scratch Using Placeholders
Array Properties
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations
Comparison Operators
Statistics
Slicing And Indexing

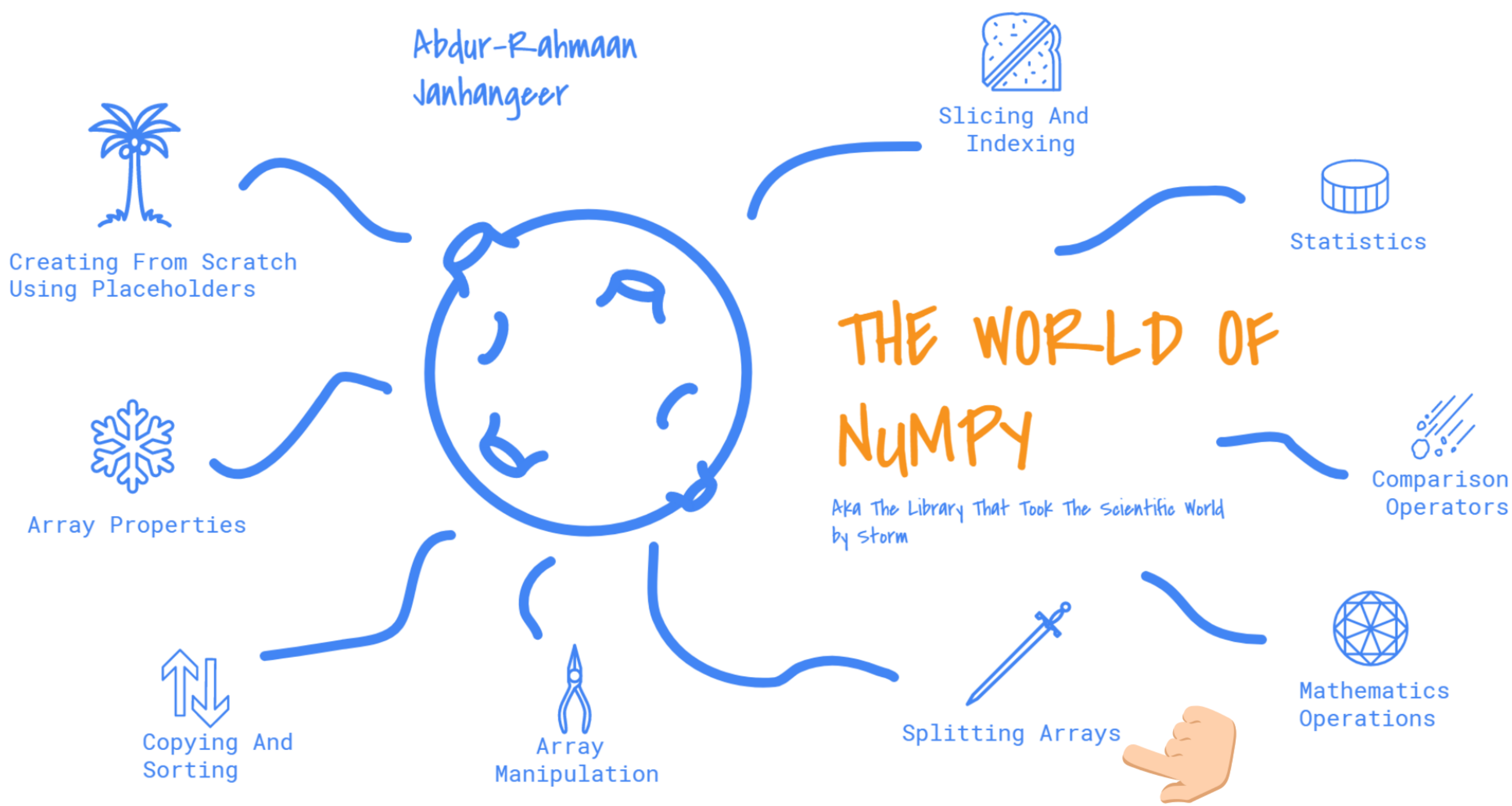

THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Abdur-Rahmaan Janhangeer
Creating From Scratch Using Placeholders
Slicing And Indexing
Statistics
Array Properties
Comparison Operators
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations

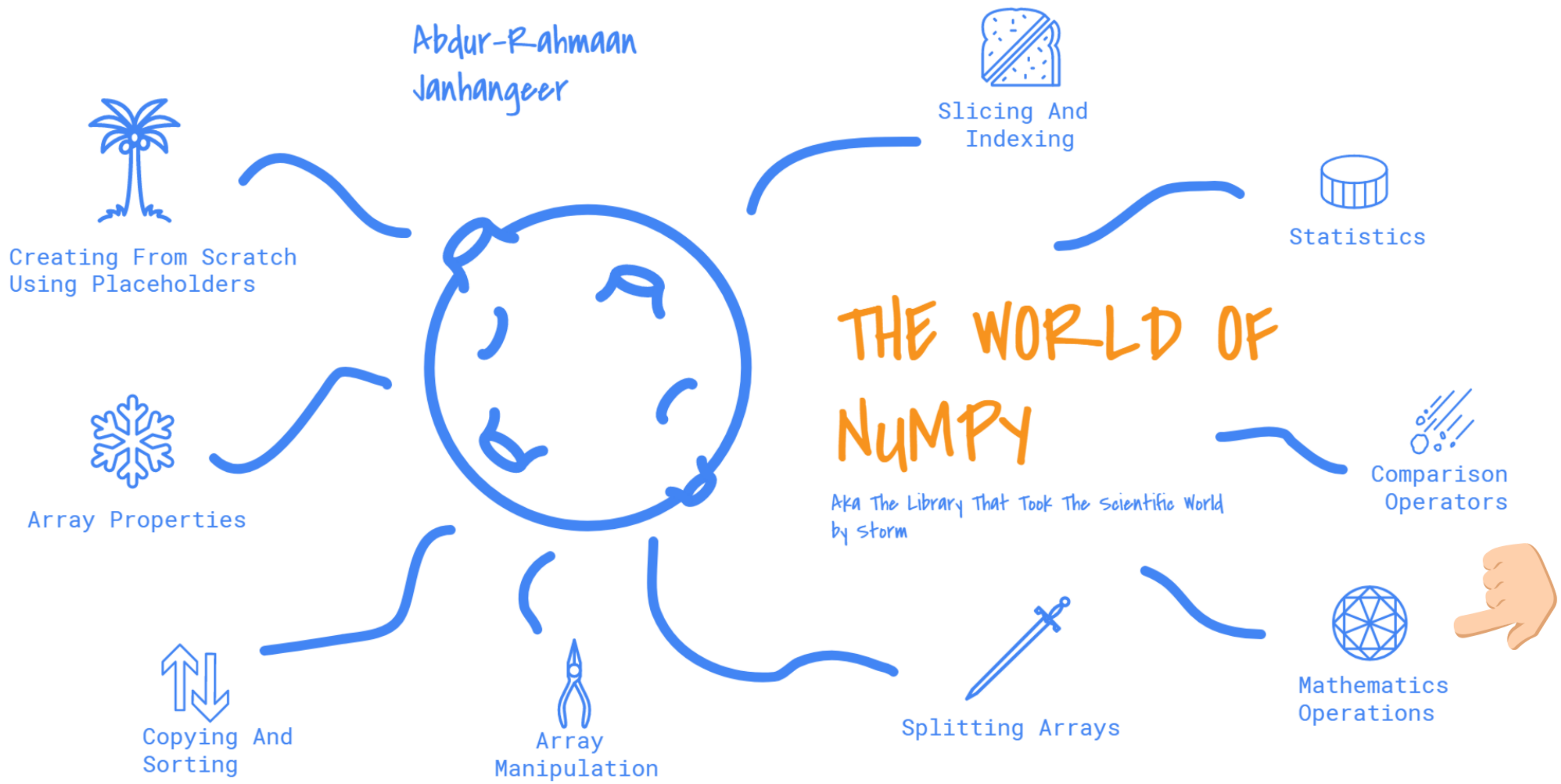


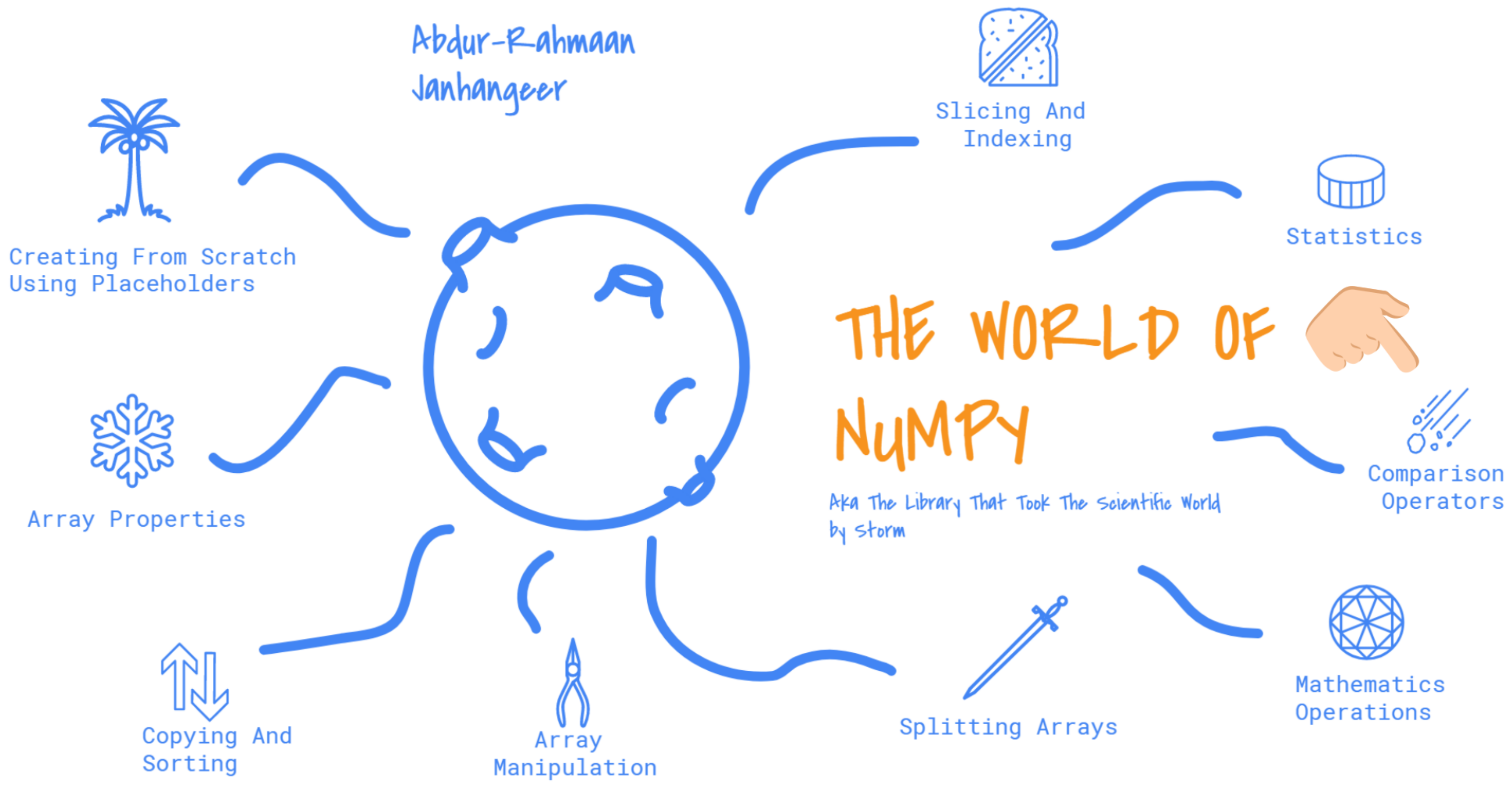
Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by storm
Creating From Scratch Using Placeholders
Slicing And Indexing
Statistics
Array Properties
Comparison Operators
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations

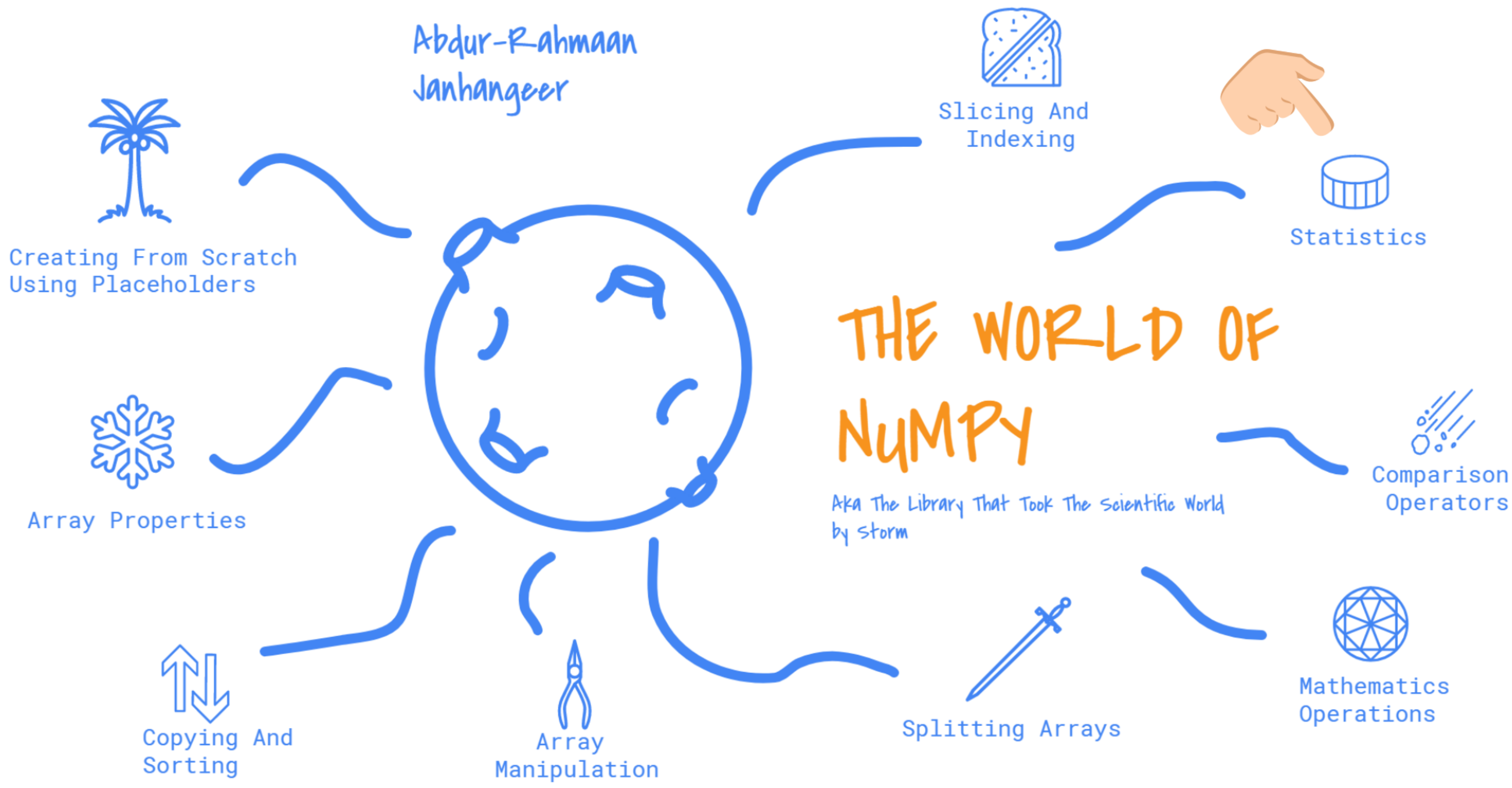


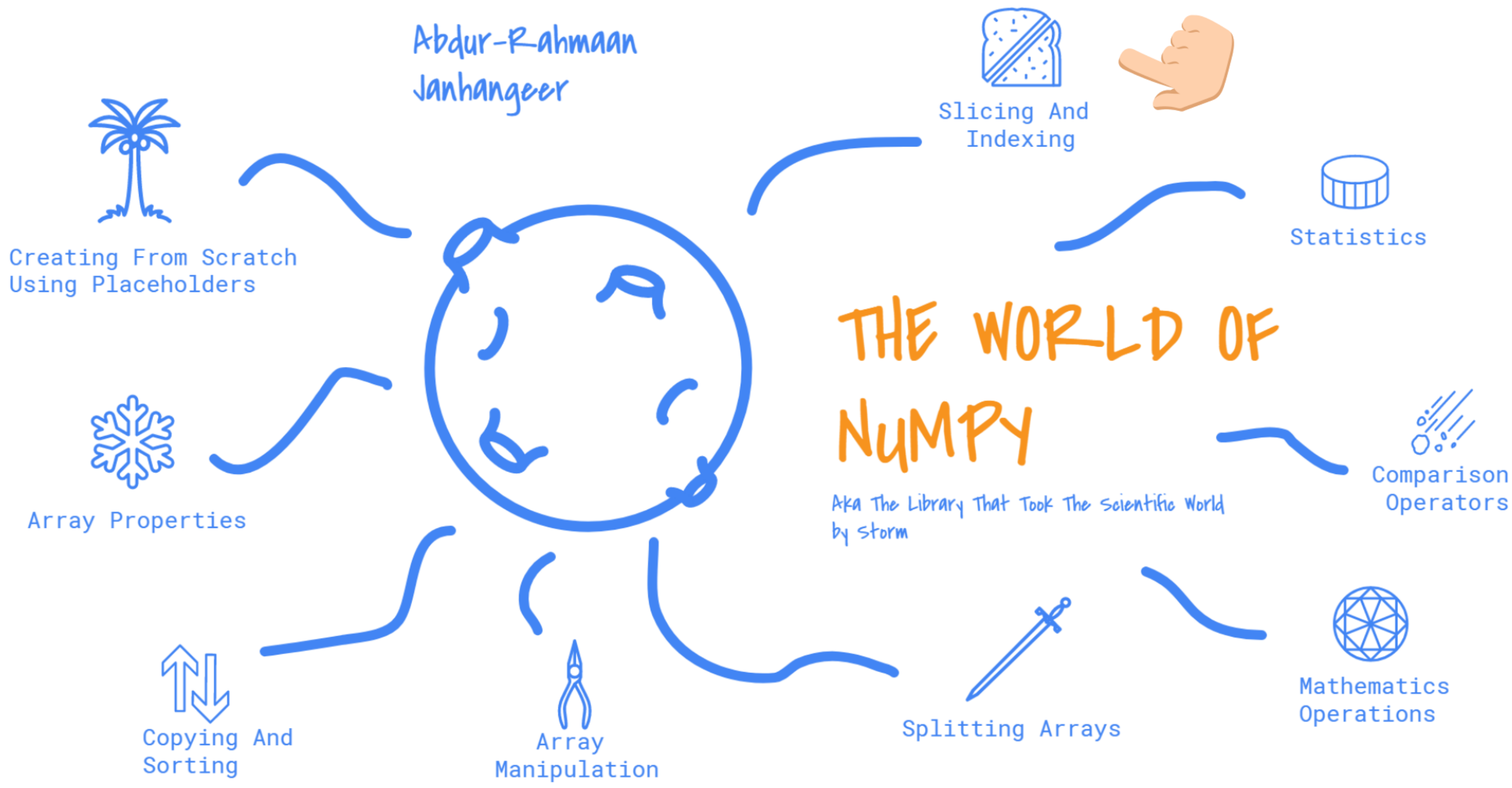
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Abdur-Rahmaan Janhangeer
Creating From Scratch Using Placeholders
Slicing And Indexing
Statistics
Array Properties
Comparison Operators
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by storm
Creating From Scratch Using Placeholders
Slicing And Indexing
Statistics
Array Properties
Comparison Operators
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Creating From Scratch Using Placeholders
Array Properties
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations
Comparison Operators
Statistics
Slicing And Indexing


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by storm
Creating From Scratch Using Placeholders
Array Properties
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations
Comparison Operators
Statistics
Slicing And Indexing


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Creating From Scratch Using Placeholders
Slicing And Indexing
Statistics
Array Properties
Comparison Operators
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations


Abdur-Rahmaan Janhangeer
THE WORLD OF NUMPY
Aka The Library That Took The Scientific World by Storm
Creating From Scratch Using Placeholders
Array Properties
Copying And Sorting
Array Manipulation
Splitting Arrays
Mathematics Operations
Comparison Operators
Statistics
Slicing And Indexing

What we need to cover more:

- data types
- loading and saving data from text file

Why Numpy?

n-dimentional array manipulation is just easy and coool

Importing Numpy

```
import numpy as np
```

About numpy arrays

Numpy arrays can only hold one type of data, making them less flexible but more powerful for computation

Numpy Data Types

- bool_ Boolean (True or False) stored as a byte
- int_ Default integer type (same as C long; normally either int64 or int32)
- intc Identical to C int (normally int32 or int64)
- intp Integer used for indexing (same as C ssize_t; normally either int32 or int64)
- int8 Byte (-128 to 127)
- int16 Integer (-32768 to 32767)
- int32 Integer (-2147483648 to 2147483647)

- int64 Integer (-9223372036854775808 to 9223372036854775807)
- uint8 Unsigned integer (0 to 255)
- uint16 Unsigned integer (0 to 65535)
- uint32 Unsigned integer (0 to 4294967295)
- uint64 Unsigned integer (0 to 18446744073709551615)
- float_ Shorthand for float64
- float16 Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
- float32 Single precision float: sign bit, 8 bits exponent, 23 bits mantissa

- float64 Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
- complex_ Shorthand for complex128
- complex64 Complex number, represented by two 32-bit floats (real and imaginary components)
- complex128 Complex number, represented by two 64-bit floats (real and imaginary components)

Creating arrays

From Existing Data

```
import numpy as np

x = [3, 4, 5, 6, 2, 4, 2]
y = [10, 30, 15, 16, 32, 76, 23]

np_x = np.array(x)
np_y = np.array(y)
```

We can create np arrays from scratch

```
# creates 3 by 4 matrix filled with zeros
>>> zeros = np.zeros([3, 4])
>>> zeros
```

similarly

```
np.ones([3, 3])
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

empty()

```
np.empty([2, 2])
```

```
array([[0., 0.],
       [0., 0.]])
```

random data

```
np.random.random([3, 4])
```

```
array([[0.60905384, 0.07042812, 0.34772305, 0.47484597],
       [0.63558091, 0.16576978, 0.72063246, 0.93037311],
       [0.18482108, 0.3001279 , 0.79063717, 0.66592421]])
```

we can also specify the data type when creating

```
np.ones([2, 2], dtype='int')
```

for a normal array we can use a number instead of a list

```
np.ones(5, dtype='int') # here 5
```

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

To check

- np.linspace(start, stop, steps)
- np.full
- np.arange(start, stop, steps)

Array Properties

Let's have an array, we can see some properties

```
x = np.array([[1, 2, 3, 4], [2, 4, 6, 7]])
print(x.shape)
print(x.ndim)
print(x.dtype)
print(x.size)
```

```
(2, 4)
2
int32
8
```

Numpy is Powerful

Numpy Maths Operations

with numpy you can do numerical operation with arrays simply!

```
x = np.array([1, 2, 3, 4])
x + 1
```

```
array([2, 3, 4, 5])
```

`+` above is same as `np.add(x)`. Similarly np.divide, etc

```python
x * 2, x / 2
```

```
(array([2, 4, 6, 8]), array([0.5, 1. , 1.5, 2. ]))
```

```python
np.cos(x)
```

```
array([ 0.54030231, -0.41614684, -0.9899925 , -0.65364362])
```

```python
np.sin(x + 1)
```

```
array([ 0.90929743,  0.14112001, -0.7568025 , -0.95892427])
```

Rounding

```
np.around(np.cos(x), decimals=1)
```

```
array([ 0.5, -0.4, -1. , -0.7])
```

`np.ceil()` rounds to the top, try it!

Stats and some tidbits

to find min we use

```
np.min(x)
```

```
1
```

```
np.max(x)
```

```
4
```

```
np.mean(x)
```

```
2.5
```

```python
np.median(x)
```

```
2.5
```

```python
np.std(x)
```

```
1.118033988749895
```

```python
np.var(x) # variance
```

```
1.25
```

```
np.sum(x)
```

```
10
```

more to look at: `np.percentile(array, percent))`, `a.corrcoef()`, `b.cumsum(axis=1)`

Numpy Concept: Axis

you might see in many examples axis, axis=0 means column and axis=1 means rows

More Fun: Comparing

To compare, just use operators!

```
ages = np.array([21, 22, 23, 26, 20, 30, 22, 21, 40])
```

finding ages greater than 25

```
ages > 25
```

```
array([False, False, False,  True, False,
True, False, False,  True])
```

```
ages == 22
```

```
array([False,  True, False, False, False,
False,  True, False, False])
```

normal operators work like !=, <= etc. You can also compare arrays

```
(x * 2) == (x * 3)
```

```
array([False, False, False, False])
```

all and any

```
np.all(ages > 20)
```

```
False
```

```
np.any(ages < 30)
```

```
True
```

New Arrays

```
ages[ages > 25]
```

```
array([26, 30, 40])
```

any comparison operator can be used

Indexing

Numpy's indexing works same as lists

```
ages[0]
```

```
21
```

```
ages[0:5:1] # start, stop, step
```

```
array([21, 22, 23, 26, 20])
```

```
ages[::] # means start from begining till end with step 1
```

```
array([21, 22, 23, 26, 20, 30, 22, 21, 40])
```

```python
ages[::-1] # step -1 reverses
```

```
array([40, 21, 22, 30, 20, 26, 23, 22, 21])
```

Now let us see 2D indexing and numpy's flexibility

```python
grid = np.array([[20, 22, 34],
                 [33, 32, 12],
                 [23, 93, 89]])
grid[0, 0] # row, colum
```

```
20
```

```python
grid[:2, :2] #row upto 2, column upto 2
```

```
array([[20, 22],
       [33, 32]])
```

```python
grid[:2, ::-1] # 2 rows, reverse columns
```

```
array([[34, 22, 20],
       [12, 32, 33]])
```

The Need For Clones (Copies)

let's extract a 2 by 2 grid

```
grid2 = grid[:2, :2]
```

```
grid2
```

```
array([[20, 22],
       [33, 32]])
```

changing some data

```
grid2[0][0] = 1
```

```
grid2
```

```
array([[ 1, 22],
       [33, 32]])
```

```
grid
```

```
array([[ 1, 22, 34],
       [33, 32, 12],
       [23, 93, 89]])
```

grid 1 also changed. that's why we need copies.

copies achieved by

```
grid2 = grid[:2, :2].copy()
```

Array Manipulation

splitting is done as

```
np.split(ages,3)
```

```
[array([21, 22, 23]),
array([26, 20, 30]),
array([22, 21, 40])]
```

```
ages.reshape((3, 3))
```

```
array([[21, 22, 23],
       [26, 20, 30],
       [22, 21, 40]])
```

```python
np.concatenate([grid, grid])
```

```
array([[ 1, 22, 34],
       [33, 32, 12],
       [23, 93, 89],
       [ 1, 22, 34],
       [33, 32, 12],
       [23, 93, 89]])
```

```python
np.sort(grid, axis=0)
# axis=0 means columns,
# axis=1 means rows
```

```
array([[ 1, 22, 12],
       [23, 32, 34],
       [33, 93, 89]])
```

```python
arr = np.array([
            [2, 34, 2],
            [32, 4, 2]])
arr.T # transpose, 2x3 -> 3x2
```

```
array([[ 2, 32],
       [34,  4],
       [ 2,  2]])
```

'vector' manipulations or linear algebra

```
np.linalg.inv(grid)
```

```
array([[ 0.07693674,  0.05348259, -0.0366027 ],
       [-0.11820362, -0.03078358,  0.04930704],
       [ 0.10363362,  0.01834577, -0.030828  ]])
```

```
# dot product

x = np.array([[23, 45, 78], [34, 94, 32]])
y = np.array([[92, 23, 90], [23, 67, 32], [23, 67, 32]])
np.dot(x, y)
```

```
array([[4945, 8770, 6006],
       [6026, 9224, 7092]])
```

Text Files With Numpy

Numpy has can load text files off the bat sing `np.loadtxt` but using pandas

Numpy Concept: broadcasting

Broadcasting simply means when doing operations, arrays are stretched so as to be able to do the operation.

Next: Go Deeper

Find a numpy tuto and see what we did not include XD