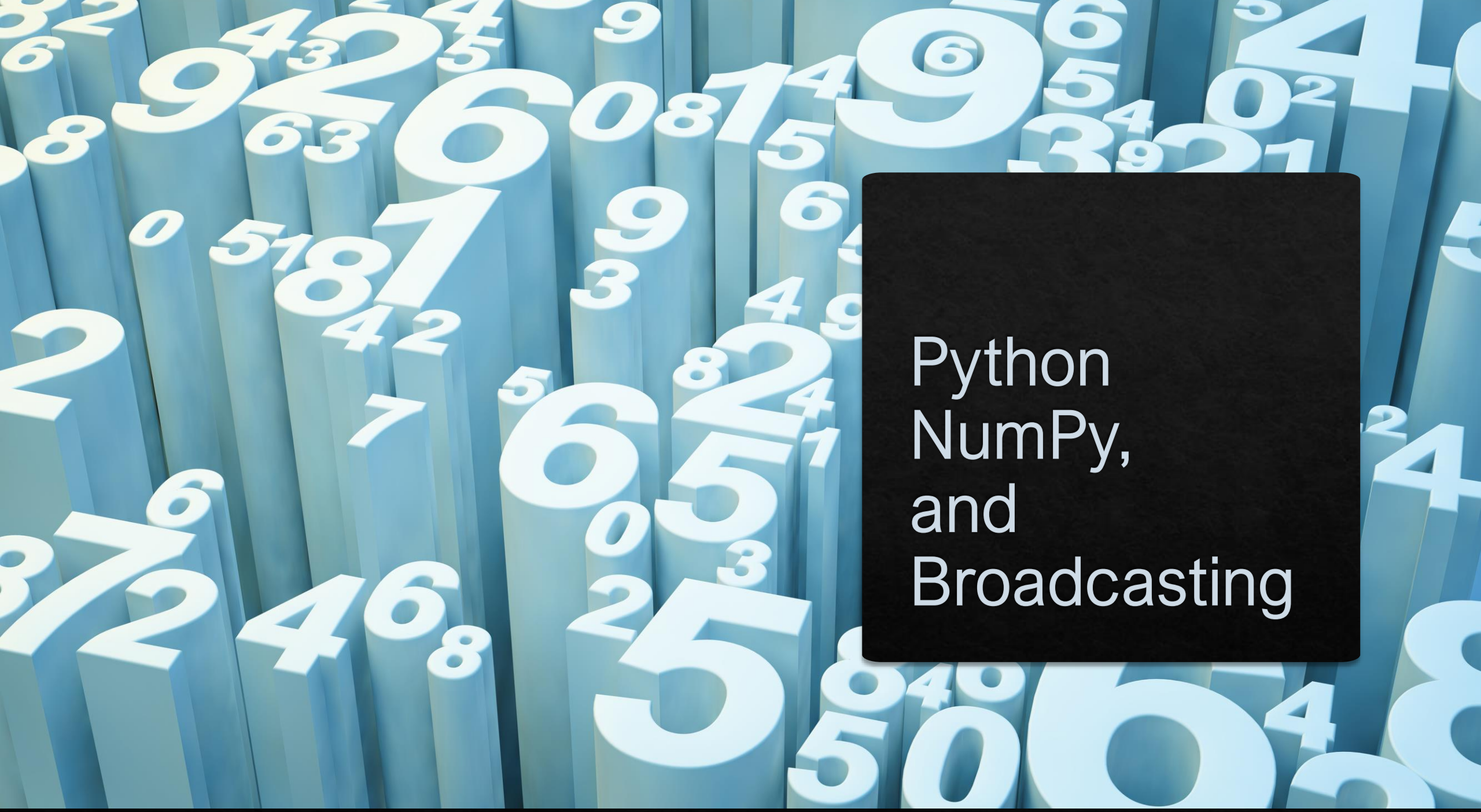


Python NumPy, and Broadcasting

What is NumPy?

Definition

Numerical Python

A Module

Open source and used in almost every field of science and engineering.

Used With

Pandas, SciPy, Matplotlib,

Scikit-learn, scikit-image and other data science/science packages.

A universal standard

What it does

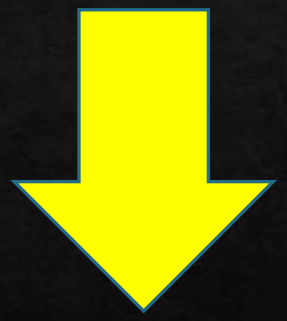
- It contains a multidimensional array and matrix data structures
- Provides a *nd-array* (a homogeneous n-dimensional array object) with methods to efficiently operate on it
- Performs a wide variety of mathematical operations on arrays
- Adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices
- Supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices

All about Arrays

- Usually a fixed-size container of items of ***the same type and size***
- The number of dimensions and items in an array is defined by its shape
- The **shape** is a tuple of non-negative integers that specify the sizes of each dimension
- The **rank** of the array is the number of dimensions.

- A central data structure of the NumPy library
- A grid of values and it contains information about raw data, how to locate an element and how to interpret it
- The grid has elements that can be indexed in various ways
- The elements are all the same, referred to as the array ***dtype***

Biggest difference between Python lists and NumPy arrays:
NumPy arrays are homogeneous



Of the same or a similar kind or nature

Open VSCode
make sure you are in the right directory

Initialize jupyter notebook in the VS Code CLI, or work directly with it in VS Code

One way to initialize NumPy arrays

arr = np.array([1, 2, 3, 4, 5, 6])

Or

a = np.array([[1,2,3,4], [5,6,7,8],[9,10,11,12]])

Access elements by starting with element “0”

print(a[0])

[1 2 3 4]

Useful keywords

- Ndarray = N-dimensional array
- 1-D = one dimensional array
- 2-D = two-dimensional array
- Vector = an array with a single dimension
- Matrix = an array with two dimensions
- Tensor = an array with 3+ dimensions
- Dimensions = axes

[[0., 0., 0.], [1., 1., 1.]]

- 2-D array
- 2 axes
- First axis: length of 2
- Second axis: length of 3

You need to install it

In your terminal, type

pip install numpy

You can import 2 ways

import numpy

import numpy as np

Depends on what you need to do.

Create a file in vscode & name it: week5

Creating arrays

General

```
import numpy

a = numpy.array([1, 2, 3, 4, 5])
print(a)
```

zeros

```
import numpy as np

b = np.zeros(4)
print(b)
```

ones

```
import numpy as np

c = np.ones(2)
print(c)
```

empty

```
import numpy as np

d = np.empty(3)

print(d)
```

**Creates an array whole initial content is random and depends on the state of memory.

** Speed.

range

```
import numpy as np

e = np.arange(5)

print(e)
```

Contains a range of evenly spaced intervals

```
import numpy as np

f = np.arange(2, 9, 2)

print(f)
```

With values that are spaced linearly in a specified interval

```
import numpy as np


g = np.linspace(0, 10, num=5)
print(g)
```

Contains a range of evenly spaced intervals

```
import numpy as np


f = np.arange(2, 9, 2)
print(f)
```

Sorting

```
import numpy as np
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
a = np.sort(arr)
print(a)
```

Other commands

argsort = an indirect sort along a specified axis

lexsort = an indirect stable sort on multiple keys

searchsorted = will find elements in a sorted array

partition = partial sort

Concatenation

```
import numpy as np

a = np.array([1, 2, 3, 4])

b = np.array([5, 6, 7, 8])

arr = np.concatenate((a, b))

print(arr)
```

```
import numpy as np

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

y = np.array([[5, 6]])

arr = np.concatenate((x, y), axis=0)

print(arr)
```

Knowing shape and size

```
array_example = np.array([[[0, 1, 2, 3],
                           [4, 5, 6, 7]],

                          [[0, 1, 2, 3],
                           [4, 5, 6, 7]],

                          [[0 ,1 ,2, 3],
                           [4, 5, 6, 7]]])
```

Knowing shape and size - continued

```
t = array_example.ndim
print(t)        # 3 (dimensions)

u = array_example.size
print(u)        # 2, 4 (2 axis w/ length of 4)

v = array_example.shape
print(v)        # 3, 2, 4
```

Reshaping

Start with this:

```
import numpy as np
a = np.arange(6)
print(a)
```

Change it:

```
b = a.reshape(3, 2)
print(b)
```

Indexing and Slicing

```
import numpy as np
data = np.array([1, 2, 3])

a = data[1]
b = data[0:2]
c = data[1:]
d = data[-2:]

print(“a”, a)
print(“b”, b)
print(“c”, c)
print(“d”, d)
```

```
import numpy as np
a = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

print(a[a < 5])

five_up = (a >= 5)
print(a[five_up])

divisible_by_2 = a[a%2==0]
print(divisible_by_2)
```

Other functions

- Can use “&” and “|” to get Booleans
- np.nonzero() = select elements or indices from an array
- np.vstack() = stack vertically
- np.hstack() = stack horizontally
- np.hsplit() = split arrays
- np.view() = create new array object that looks at same data as the original array
- copy() = complete copy of the array and its data.
- Addition/Subtraction
- Multiplication/Division
- Max/min
- Sum/mean
- Product
- Standard deviation

Broadcasting

A mechanism for performing math operations on arrays of *unequal* shapes

- To determine if two arrays are broadcast-compatible, align the entries of their shapes such that their trailing dimensions are aligned, and then check that each pair of aligned dimensions satisfy either of the following conditions:
  - the aligned dimensions have the same size
  - one of the dimensions has a size of 1

- The two arrays are broadcast-compatible if either of these conditions are satisfied for each pair of aligned dimensions.

```
import numpy as np

# a shape-(3, 4) array
x = np.array([ [-0. , -0.1, -0.2, -0.3],
               [-0.4, -0.5, -0.6, -0.7],
               [-0.8, -0.9, -1. 0, -1.1] ])

# a shape-(4,) array
y = np.array([1, 2, 3, 4])

# multiplying a shape-(4,) array with a shape-(3, 4) array
# `y` is multiplied by each row of `x`
x * y
array([ [-0. , -0.2, -0.6, -1.2],
        [-0.4, -1. , -1.8, -2.8],
        [-0.8, -1.8, -3. , -4.4] ])
```