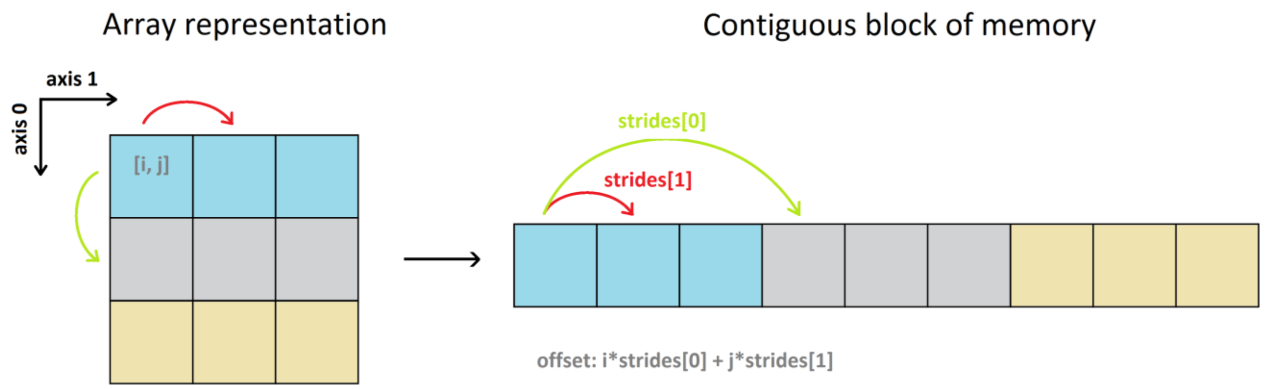
NUMPY LIBRARY

Introduction

- NumPy is an open-source Python library used for working with arrays
- It also has functions for working in domain of linear algebra, fourier transform, and matrices
- NumPy stands for Numerical Python
- The NumPy array is a data structure that efficiently stores and accesses multidimensional arrays
- It consists of a pointer to memory, along with metadata used to interpret the data stored there, notably ‘data type’, ‘shape’ and ‘strides’
- NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++

Why NumPy?

- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists
- Numpy array is a contiguous block of memory used to store the same type of data. When the type of data you store is determined, your memory stride is determined. Its essence is an array, so you can only store elements of the same type.
- Strides are necessary to interpret computer memory, which stores elements linearly, as multidimensional arrays. They describe the number of bytes to move forward in memory to jump from row to row, column to column, and so forth.


Array representation
Contiguous block of memory
axis 1
axis 0
[i, j]
strides[0]
strides[1]
offset: i*strides[0] + j*strides[1]

https://i.stack.imgur.com/oQQVl.png

Installing Numpy Module

- You may use Command Prompt/Terminal
- You need pip/conda to install various libraries

```
pip install numpy

conda install numpy
```

Note: It is pre-installed if Anaconda Software is used

Practicals on Numpy

- Creating Numpy Array of different dimensions (Scalars vs Vectors vs Matrices)
- Random Generator Number
- Importance of Random Seed
- Matrix Multiplication in Numpy
- Descriptive Statistics using Numpy
- Interquartile Range
- Broadcasting in Numpy
- How to sort Numpy Arrays?

NumPy Statistical Functions

- np.amin()- Minimum value of the element along a specified axis.
- np.amax()- Maximum value of the element along a specified axis.
- np.mean()- Mean value of the data set.
- np.median()- Median value of the data set.
- np.ptp()- Range of values along an axis(peak to peak).
- np.std()- Standard deviation
- np.var() – Variance.
- np.average()- Weighted average
- np.percentile()- nth percentile of data along the specified axis.

Statistics – Mean, Median and Range

- Mean - Compute the arithmetic mean along the specified axis.

np.mean([1,2,3,4,5])

- Median - Compute the median along the specified axis.

np.median([1,5,2,3,4])

- Range - Compute the median along the specified axis.

np.ptp([1,5,2,3,4])

*Ptp – Point to Point

Statistics – Standard Deviation and Variance

1. Standard deviation is the square root of the average of squared deviations from mean. The function used for this is np.std().

The formula for standard deviation is as follows −

std = sqrt(mean(abs(x - x.mean())2))**

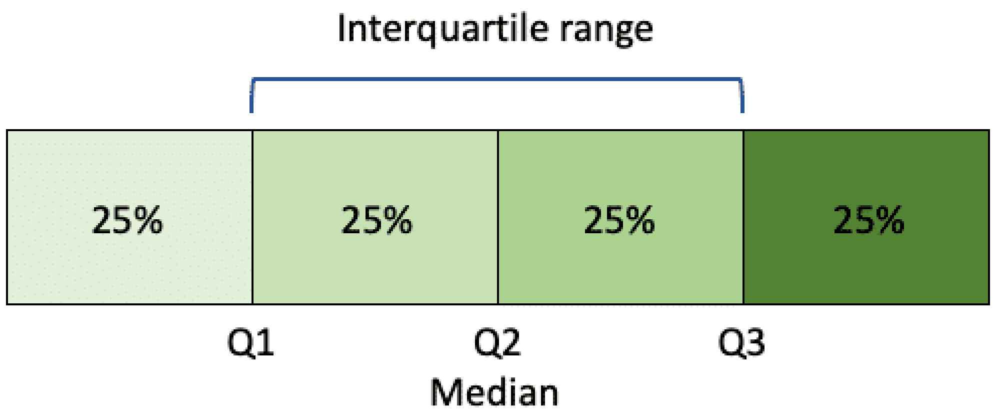
np.std([1,2,3,4])

2. Variance is the average of squared deviations, i.e., **mean(abs(x - x.mean())**2).**

Or, standard deviation is the square root of variance.

np.var([1,2,3,4])

Statistics – Interquartile Range



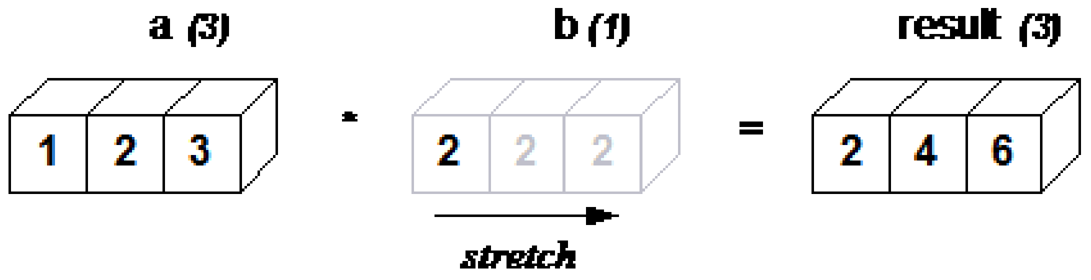
The **first quartile (Q1)**, is defined as the middle number between the smallest number and the median of the data set, the **second quartile (Q2) – median** of the given data set while the **third quartile (Q3)**, is the middle number between the median and the largest value of the data set.

In numpy, use np.percentile()

https://cdn.scribbr.com/wp-content/uploads/2020/09/iqr_quartiles.png

Broadcasting of an Array

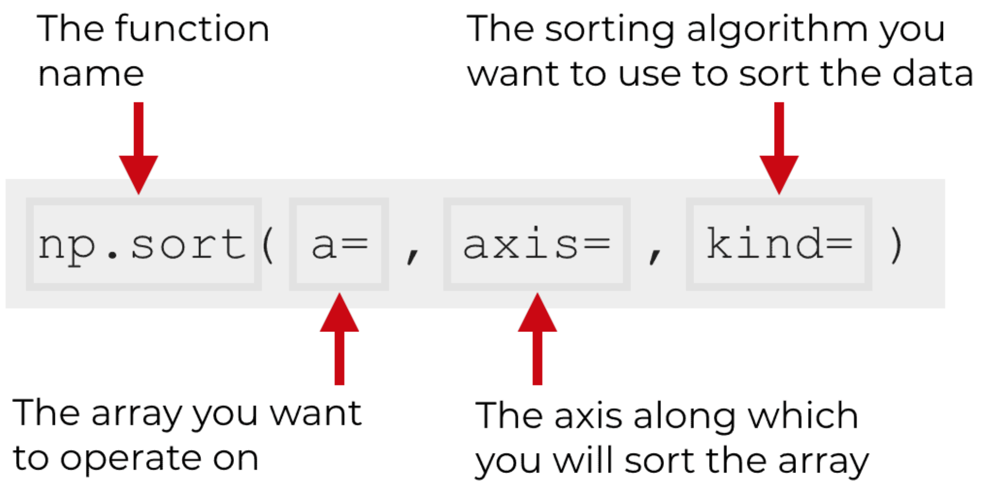
- Broadcasting describes how numpy treats arrays with different shapes during arithmetic operations.
- The smaller array is “broadcast” across the larger array so that they have compatible shapes.

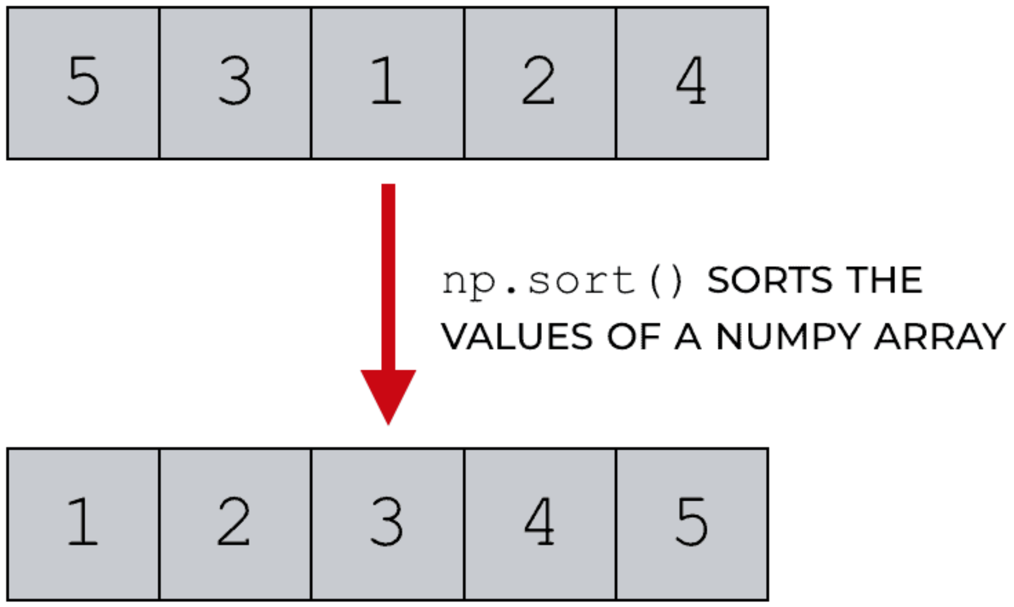


https://numpy.org/doc/stable/_images/theory.broadcast_1.gif

Sorting an Array

- Return a sorted copy of an array.

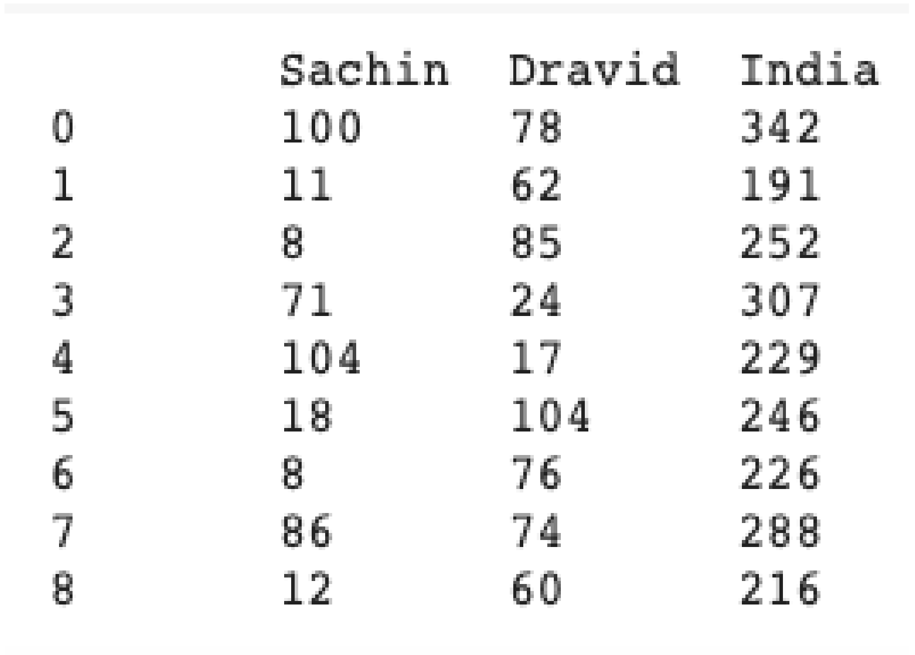

The function name
The sorting algorithm you want to use to sort the data
np.sort(a= , axis= , kind=)
The array you want to operate on
The axis along which you will sort the array



https://vrzkj25a871bpq7t1ugcgmn9-wpengine.netdna-ssl.com/wp-content/uploads/2019/05/numpy-sort-simple-example.png

Case Study

- Let us consider a dataset consisting of runs scored by Sachin and Dravid and team India across a subset of matches. The data set covers 15 matches. Let us try and answer a few questions



```
        Sachin  Dravid  India
0       100     78      342
1       11      62      191
2       8       85      252
3       71      24      307
4       104     17      229
5       18      104     246
6       8       76      226
7       86      74      288
8       12      60      216
```

Data Set Source - https://github.com/goradbj/MachineLearning/blob/main/cric.tsv

Case Study

- Firstly let us load the dataset using the command —

```
cric_data = np.loadtxt("cric_data.tsv", skiprows=1)
cric_data.shape()
Output: (225,4) Indicates 225 matches.
```

Case Study

- Let us first divide the n-d array into individual components.

```
Sachin = cric_data[:,1]
Dravid = cric_data[:,2]
India  = cric_data[:,3]

(Note - ':' indicates all values along that dimension)
```

Example Output:

```
Sachin = [100, 11, 8, 71.......]
```

Case Study

1. Find the Mean and Median of Sachin, Dravid and India.

We can use NumPy library functions **np.mean()** and **np.median()** to achieve the desired results.

```
def stats(col):
    print('Mean', np.mean(col))
    print('Median', np.median(col))

stats(Sachin)
Output: Mean 39.87  Median 27.0

stats(Dravid)
Output: Mean 32.06  Median 22.0

stats(India)
Output: Mean 220.79  Median 216.0
```

Case Study

2. How many centuries did Sachin and Dravid Score?

Using the **count_nonzero method(condition)** we can find a count of elements that satisfy a given condition.

```
np.count_nonzero(Sachin>99) Output: 23

np.count_nonzero(Dravid>99) Output: 8
```

Case Study

3) How many matches did Sachin and Dravid take to reach 1000 runs?

We can use the concept of cumulative sum to solve this question. The method **np.cumsum(array)** would perform the cumulative sum of all elements in the array.
Example: a=[1,2,3,4].
np.cumsum(a) will return [1,3,6,10]

```
sachin_cumscores = np.cumsum(sachin)
np.searchsorted(sachin_cumscores, 1000)
Output: 29
```

REFERENCES

1. https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
2. https://docs.python.org/3/library/
3. https://www.tutorialspoint.com/numpy
4. https://numpy.org/
5. https://towardsdatascience.com/
6. https://realpython.com/

THANK YOU