**What is NumPy?**

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

**Why Use NumPy?**

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

**Data Science**: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

**Why is NumPy Faster Than Lists?**

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

**Which Language is NumPy written in?**

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++.

Where is the NumPy Codebase?

The source code for NumPy is located at this github repository https://github.com/numpy/numpy

github: enables many people to work on the same codebase.

In [1]:

pip

install

numpy

Requirement already satisfied: numpy in c:\users\prafu\anaconda3\lib\sitepackages (1.21.5) Note: you may need to restart the kernel to use updated packages.

In [2]:

**import**

numpy

**as**

np

In [4]:

*#array -->similar kind data -->int-int,float-float*

In [11]:

*#create 1D Array*

In [120]:

a

**=**

np

.

array

([

23

,

646

,

785

,

34

,

7856

,

35

,

75

])

In [121]:

a

Out[121]:

array([ 23, 646, 785, 34, 7856, 35, 75]) In [7]:

type

(

a

)

Out[7]:

numpy.ndarray In [123]:

a

.

shape

Out[123]:

(7,) In [8]:

a

[

0

]

Out[8]:

23

[9]: a[4]

Out[9]:

7856

In [10]:

*#CREATE 2D Array*

In [14]:

b

**=**

np

.

array

([[

1

,

2

,

3

,

4

]

,

[

5

,

6

,

7

,

8

]])

In [15]:

b

Out[15]:

array([[1, 2, 3, 4],

[5, 6, 7, 8]]) In [ ]:

*#shape structure(row,col)*

In [19]:

b

.

shape

Out[19]:

(2, 4)

In [ ]:

*#DTYPE-Memory size*

In [20]:

b

.

dtype

Out[20]: dtype('int32') In [21]:

b

.

itemsize

Out[21]:

4

[22]:

a**=**np.array([23,646,785,34,7856,35,75], dtype**=**'int64') In [23]:

a

.

dtype

Out[23]: dtype('int64') In [24]:

*#byte*

In [25]:

a

.

itemsize

Out[25]:

8 In [27]:

a

.

size

Out[27]:

7 In [26]:

a

.

itemsize

**\***

a

.

size

Out[26]:

56

**byte count on list (28)**

In [28]:

**import**

sys

In [29]:

temp

**=**

[

23

,

646

,

785

,

34

,

7856

,

35

,

75

]

[30]: temp

Out[30]:

[23, 646, 785, 34, 7856, 35, 75] In [31]:

sys

.

getsizeof

(

23

)

Out[31]:

28 In [32]:

sys

.

getsizeof

(

785

)

Out[32]:

28 In [34]:

sys

.

getsizeof

(

785

)

**\***

len

(

temp

)

Out[34]:

196 In [35]:

sys

.

getsizeof

(

23

)

**\***

len

(

temp

)

Out[35]:

196 In [36]:

b

Out[36]:

array([[1, 2, 3, 4],

[5, 6, 7, 8]]) In [37]:

*#check dimenional*

In [38]:

b

.

ndim

Out[38]:

2

[39]: b[1,:]

Out[39]: array([5, 6, 7, 8]) In [40]:

*# b[row,col]*

In [41]:

*# b[row, :]-->b[row , :all col]*

In [42]:

b

[

0

,:]

Out[42]: array([1, 2, 3, 4]) In [43]:

b

Out[43]:

array([[1, 2, 3, 4],

[5, 6, 7, 8]]) In [44]:

b

[

0

,

0

]

Out[44]:

1

In [45]:

b

[

1

,

1

]

Out[45]:

6

In [46]:

b

[

0

,

2

]

Out[46]:

3

[48]: b[0,2:]

Out[48]: array([3, 4]) In [51]:

b

[

0

,

1

:]

Out[51]: array([2, 3, 4]) In [124]:

a

**=**

np

.

array

([[[

1

,

2

,

3

,],[

4

,

5

,

6

]]

,

[[

11

,

22

,

33

]

,

[

7

,

8

,

9

]]])

In [125]:

a

Out[125]:

array([[[ 1, 2, 3],

[ 4, 5, 6]],

[[11, 22, 33],

[ 7, 8, 9]]]) In [57]:

a

.

ndim

Out[57]:

3

In [59]:

*#matrix 3d - like cube*

In [60]:

*# a[matrix,row,col]*

In [61]:

a

[

1

,

1

,

0

]

Out[61]:

7

[128]: a[1,1,0:2]

Out[128]: array([7, 8]) In [62]:

a

[

0

,

1

,

0

]

Out[62]:

4

In [65]:

a

**=**

np

.

ones

((

2

,

2

))

In [66]:

a

Out[66]:

array([[1., 1.], [1., 1.]]) In [67]:

*# by default it's in float*

In [70]:

*#2D*

a

**=**

np

.

ones

((

2

,

2

)

,

dtype

**=**

'int'

)

In [71]:

a

Out[71]:

array([[1, 1], [1, 1]]) In [72]:

*#3D*

a

**=**

np

.

ones

((

2

,

2

,

2

)

,

dtype

**=**

'int'

)

[73]:

a

Out[73]:

array([[[1, 1],

[1, 1]],

[[1, 1],

[1, 1]]]) In [74]:

*#3D*

a

**=**

np

.

ones

((

3

,

2

,

2

)

,

dtype

**=**

'int'

)

In [75]:

a

Out[75]:

array([[[1, 1],

[1, 1]],

[[1, 1],

[1, 1]],

[[1, 1],

[1, 1]]]) In [77]:

a

**=**

np

.

ones

((

4

,

4

)

,

dtype

**=**

'int'

)

In [78]:

a

Out[78]:

array([[1, 1, 1, 1], [1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]]) In [79]:

b

**=**

np

.

zeros

((

4

,

4

)

,

dtype

**=**

'int'

)

[80]: b

Out[80]:

array([[0, 0, 0, 0], [0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0]]) In [81]:

np

.

full

((

3

,

3

)

,

20

)

Out[81]:

array([[20, 20, 20], [20, 20, 20],

[20, 20, 20]]) In [82]:

*#random int*

In [85]:

np

.

random

.

rand

(

3

,

3

)

Out[85]:

array([[0.49487747, 0.71933476, 0.13562713], [0.47321862, 0.88645229, 0.4794912 ],

[0.72102862, 0.92443424, 0.25760914]]) In [129]:

temp

**=**

np

.

random

.

randint

(

200

,

500

,

size

**=**

(

3

,

3

))

In [130]:

temp

Out[130]:

array([[260, 259, 426], [262, 385, 328],

[300, 271, 493]]) In [88]:

np

.

identity

(

4

)

Out[88]:

array([[1., 0., 0., 0.], [0., 1., 0., 0.],

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

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

[89]: a

Out[89]:

array([[1, 1, 1, 1], [1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]]) In [90]:

a

**\***

2

Out[90]:

array([[2, 2, 2, 2], [2, 2, 2, 2],

[2, 2, 2, 2],

[2, 2, 2, 2]]) In [91]:

a

**-**

1

Out[91]:

array([[0, 0, 0, 0], [0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0]]) In [92]:

a

**/**

5

Out[92]:

array([[0.2, 0.2, 0.2, 0.2], [0.2, 0.2, 0.2, 0.2],

[0.2, 0.2, 0.2, 0.2],

[0.2, 0.2, 0.2, 0.2]]) In [94]:

temp

Out[94]:

array([[320, 356, 395], [291, 342, 273],

[225, 202, 249]])

[95]: np.max(temp)

Out[95]:

395 In [96]:

np

.

min

(

temp

)

Out[96]:

202 In [97]:

np

.

sum

(

temp

)

Out[97]:

2653 In [100]:

*# position*

In [98]:

np

.

argmin

(

temp

)

Out[98]:

7 In [99]:

np

.

argmax

(

temp

)

Out[99]:

2 In [101]:

temp

.

reshape

(

1

,

9

)

Out[101]:

array([[320, 356, 395, 291, 342, 273, 225, 202, 249]]) [102]:

temp.reshape(9,1)

Out[102]:

array([[320], [356],

[395],

[291],

[342],

[273],

[225],

[202],

[249]]) In [103]:

a

**=**

np

.

arange

(

10

)

In [104]:

a

Out[104]:

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]) In [ ]:

*#index no. in result*

In [105]:

np

.

where

(

a

**>**

4

)

Out[105]: (array([5, 6, 7, 8, 9], dtype=int64),) In [106]:

a

**=**

np

.

array

([[

320

,

356

,

395

,

291

,

342

,

273

,

225

,

202

,

249

]])

In [107]:

a

Out[107]:

array([[320, 356, 395, 291, 342, 273, 225, 202, 249]]) [108]:

np.where(a**>**291)

Out[108]: (array([0, 0, 0, 0], dtype=int64), array([0, 1, 2, 4], dtype=int64)) In [109]:

**import**

pandas

**as**

pd

In [112]:

df

**=**

pd

.

DataFrame

({

'name'

:[

'abc'

,

'def'

,

'ghi'

]

,

'id'

:[

101

,

102

,

103

]})

In [113]:

df

Out[113]:

**name id**

1. abc 101
2. def 102
3. ghi 103

In [ ]:

*#numpy array*

In [114]:

temp

**=**

np

.

array

([

45

,

754

,

35

])

In [115]:

temp

Out[115]:

array([ 45, 754, 35])

In [118]:

*#convert in series to concat with pandas dataframe*

[116]:

pd.Series(temp,name**=**'test')

Out[116]:

1. 45
2. 754
3. 35

Name: test, dtype: int32 In [119]:

*#concat*

In [117]:

pd

.

concat

([

df

,

pd

.

Series

(

temp

,

name

**=**

'test'

)]

,

axis

**=**

1

)

Out[117]: **name id test**

|  |  |  |
| --- | --- | --- |
| **0** abc | 101 | 45 |
| **1** def | 102 | 754 |
| **2** ghi | 103 | 35 |

In [ ]: