# NB 05 - Numpy Basics

August 17, 2021

## 1 Importing numpy library

[1]:
```
import numpy as np
```

## 2 Creating numpy arrays

## 3 One dimensional arrays

[2]:
```
L=[23,33,35,56,67,78]
arr=np.array(L)
print(arr)
```

```
[23 33 35 56 67 78]
```

[3]:
```
arr
```

[3]:
```
array([23, 33, 35, 56, 67, 78])
```

## 4 Two dimensional arrays

[4]:
```
L=[[12,22,23],[56,54,77]]
arr=np.array(L)
print(arr)
```

```
[[12 22 23]
 [56 54 77]]
```

[5]:
```
arr
```

[5]:
```
array([[12, 22, 23],
       [56, 54, 77]])
```

## 5 Array properties

[6]:
```
arr1=np.array([23,33,35,56,67,78])
arr2=np.array([[12,22,23],[56,54,77]])
```

[7]:
```
arr1
```

[7]:
```
array([23, 33, 35, 56, 67, 78])
```

[8]:
```
arr2
```

[8]:
```
array([[12, 22, 23],
       [56, 54, 77]])
```

## 6 Shape of the array

[9]:
```
print(arr1.shape)
print(arr2.shape)
```

```
(6,)
(2, 3)
```

## 7 Dimensions of the array

[10]:
```
print(arr1.ndim)
print(arr2.ndim)
```

```
1
2
```

## 8 Data type inside arrays

[11]:
```
print(arr1.dtype)
print(arr2.dtype)
```

```
int32
int32
```

## 9 Size of arrays

[12]:
```
print(arr1.size)
print(arr2.size)
```

```
6
6
```

# 10 Special arrays

# 11 Null array

[13]:
```
a=np.zeros(10)
print(a)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

[14]:
```
b=np.zeros((3,3))
print(b)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

# 12 Ones array

[15]:
```
a=np.ones(20)
print(a)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

[16]:
```
b=np.ones((5,5))
print(b)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

# 13 Identity matrix

[17]:
```
I3=np.eye(3)
print(I3)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

# 14 Simulating random integers

[18]:
```
r=np.random.randint(2,8) #One random integer from 2 to 8
```

[19]:
```
r
```

[19]: 4

[20]:
```
r=np.random.randint(2,8,10) #10 random integers from 2 to 8
```

[21]:
```
r
```

[21]: array([2, 6, 2, 3, 3, 7, 3, 4, 7, 3])

## 15 Simulating random floats from 0 to 1

[22]:
```
r=np.random.random(5)
```

[23]:
```
r
```

[23]: array([0.61137111, 0.67125064, 0.12821443, 0.51479995, 0.80780689])

[24]:
```
r=np.random.random((5,5))
```

[25]:
```
r
```

[25]:
```
array([[0.79180125, 0.24705266, 0.00990013, 0.89888192, 0.89751506],
       [0.86574371, 0.07442892, 0.16804242, 0.45352123, 0.13654964],
       [0.44112076, 0.2998447 , 0.21958022, 0.26185555, 0.85814004],
       [0.02016033, 0.82784135, 0.38094173, 0.82990084, 0.23007408],
       [0.69007785, 0.89695573, 0.9417007 , 0.748662  , 0.3674867 ]])
```

## 16 Simulating random floats from standard normal distribution

[26]:
```
r=np.random.randn(10)
```

[27]:
```
r
```

[27]:
```
array([ 0.08805111,  0.3963974 ,  0.31434471,  2.93454512,  0.74311956,
        0.26796766, -1.83129744,  0.6586675 , -0.27352593, -0.0383923 ])
```

[28]:
```
r=np.random.randn(3,3)
```

[29]:
```
r
```

[29]:
```
array([[-0.17034571, -0.65104964, -0.36328228],
       [ 1.24571524, -0.54127275,  1.55958281],
       [ 0.99025215,  0.17295969,  0.90552233]])
```

# 17 Simulating data from probability distributions

# 18 Normal distribution

[30]:
```
r=np.random.normal(50,5,4) #4 random values from a normal distribution with
↪mean 50 and standard deviation 5
```

[31]:
```
r
```

[31]:
```
array([54.57264439, 44.65264004, 50.87078731, 39.67120809])
```

[32]:
```
r=np.random.normal(50,5,(4,4))
```

[33]:
```
r
```

[33]:
```
array([[44.60543388, 40.31635676, 51.11305565, 43.42294919],
       [47.84170306, 49.54310321, 52.85899464, 42.48294551],
       [51.67990493, 55.46277233, 54.84861391, 57.49485868],
       [56.58284842, 49.22961379, 47.84663515, 57.97129371]])
```

# 19 Uniform distribution

[34]:
```
r=np.random.uniform(10,20,5)
```

[35]:
```
r
```

[35]:
```
array([19.00755161, 15.93920718, 13.1350319 , 18.41674919, 11.88593802])
```

[36]:
```
r=np.random.uniform(10,20,(5,5))
```

[37]:
```
r
```

[37]:
```
array([[18.30770765, 14.81656595, 13.5431284 , 18.8535427 , 16.6831832 ],
       [12.16937541, 16.86739071, 15.66947572, 17.75458259, 14.24981517],
       [12.52338178, 19.98040681, 18.30361661, 14.49774745, 18.93250109],
       [16.38957576, 16.97838255, 16.61034418, 15.87375434, 19.72826284],
       [19.14768313, 16.4781066 , 15.64857987, 15.83218007, 11.94067217]])
```

# 20 Generating sequence

# 21 arange function with single dimension

[38]:
```
arr=np.arange(10)
arr
```

[38]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[39]:
```
arr=np.arange(3,10)
```

[40]:
```
arr
```

[40]:
```
array([3, 4, 5, 6, 7, 8, 9])
```

## 22 Changing the shape of the sequence array

[41]:
```
arr=np.arange(20)
arr.shape=(4,5)
arr
```

[41]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

[42]:
```
arr=np.arange(20).reshape(4,5)
arr
```

[42]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

## 23 Indexing & slicing one dimensional numpy arrays

[43]:
```
a=np.array([12,22,34,56,54,56])
```

[44]:
```
a[0]
```

[44]:
```
12
```

[45]:
```
a[2]
```

[45]:
```
34
```

[46]:
```
a[-1]
```

[46]:
```
56
```

[47]:
```
a[2:5]
```

[47]:
```
array([34, 56, 54])
```

[48]:
```
a[3:]
```

[48]: array([56, 54, 56])

[49]:
```
a[:3]
```

[49]: array([12, 22, 34])

[50]:
```
a[::2]
```

[50]: array([12, 34, 54])

[51]:
```
a[::-1]
```

[51]: array([56, 54, 56, 34, 22, 12])

## 24 Indexing & slicing multi dimensional numpy arrays

[52]:
```
b=np.array([[23,33,34,45,56],[23,33,34,56,54],[78,65,67,64,78]])
b
```

[52]:
```
array([[23, 33, 34, 45, 56],
       [23, 33, 34, 56, 54],
       [78, 65, 67, 64, 78]])
```

[53]:
```
b[0,3]
```

[53]: 45

[54]:
```
b[:2,1:4]
```

[54]:
```
array([[33, 34, 45],
       [33, 34, 56]])
```

[55]:
```
b[1,[2,4]]
```

[55]: array([34, 54])

## 25 Numpy arrays are mutable

[56]:
```
a=np.array([23,33,45,56,54])
```

[57]:
```
a[0]=100
```

[58]:
```
a
```

[58]: array([100,  33,  45,  56,  54])

## 26 Array masking

[59]:
```
a=np.array([34,45,33,23,45,67,65,78,21])
```

[60]:
```
a>50
```

[60]:
```
array([False, False, False, False, False,  True,  True,  True, False])
```

[61]:
```
a[a>50]
```

[61]:
```
array([67, 65, 78])
```

## 27 Replacing existing values with new values

[62]:
```
a=np.array([-23,34,-15,-22,56])
```

[63]:
```
np.where(a>0)
```

[63]:
```
(array([1, 4], dtype=int64),)
```

[64]:
```
b=np.where(a>0,a,0)
```

[65]:
```
b
```

[65]:
```
array([ 0, 34,  0,  0, 56])
```

## 28 Basic numpy functions

## 29 numpy functions with scaler values

[66]:
```
np.sin(12) #Trigonometry
```

[66]:
```
-0.5365729180004349
```

[67]:
```
np.log(10) #Log
```

[67]:
```
2.302585092994046
```

[68]:
```
np.power(4,2) #Power
```

[68]:
```
16
```

[69]:
```
np.exp(2) #Exponential values
```

[69]:
```
7.38905609893065
```

[70]:
```
np.round(34.567) #Rounding
```

[70]: 35.0

## 30 numpy functions with arrays

[71]:
```
arr=np.array([34,33,21,25,56,32,44,78,12])
```

[72]:
```
arr.min()
```

[72]: 12

[73]:
```
arr.max()
```

[73]: 78

[74]:
```
arr.mean()
```

[74]: 37.22222222222222

[75]:
```
arr.sort()
arr
```

[75]: array([12, 21, 25, 32, 33, 34, 44, 56, 78])

[76]:
```
np.median(arr)
```

[76]: 33.0

## 31 Special numpy values

[77]:
```
np.pi
```

[77]: 3.141592653589793

[78]:
```
np.e
```

[78]: 2.718281828459045

[79]:
```
np.inf
```

[79]: inf

[80]:
```
np.NINF
```

[80]: -inf

[81]:
```
np.nan
```

[81]: nan

# 32 Stacking numpy arrays

# 33 Vertical stacking & horizontal stacking

[82]:
```
a=np.array([[3,4,5],[6,3,5]])
b=np.array([[2,3,7],[8,2,6]])
```

[83]:
```
c=np.vstack((a,b))
print(c)
```

```
[[3 4 5]
 [6 3 5]
 [2 3 7]
 [8 2 6]]
```

[84]:
```
d=np.hstack((a,b))
print(d)
```

```
[[3 4 5 2 3 7]
 [6 3 5 8 2 6]]
```

# 34 Appending

[85]:
```
a=np.array([[3,4,5],[6,3,5]])
b=np.array([[2,3,7],[8,2,6]])
```

[86]:
```
c=np.append(a,b,axis=0)
print(c)
```

```
[[3 4 5]
 [6 3 5]
 [2 3 7]
 [8 2 6]]
```

[87]:
```
d=np.append(a,b,axis=1)
print(d)
```

```
[[3 4 5 2 3 7]
 [6 3 5 8 2 6]]
```

# 35 Concatenation

[88]:
```
a=np.array([[3,4,5],[6,3,5]])
b=np.array([[2,3,7],[8,2,6]])
```

[89]:
```
c=np.concatenate((a,b),axis=0)
print(c)
```

```
[[3 4 5]
 [6 3 5]
 [2 3 7]
 [8 2 6]]
```

[90]:
```
d=np.concatenate((a,b),axis=0)
print(d)
```

```
[[3 4 5]
 [6 3 5]
 [2 3 7]
 [8 2 6]]
```

## 36 Tiling

[91]:
```
a=np.array([23,34])
b=np.tile(a,(1,2))
print(b)
```

```
[[23 34 23 34]]
```

[92]:
```
a=np.array([23,34])
b=np.tile(a,(2,2))
print(b)
```

```
[[23 34 23 34]
 [23 34 23 34]]
```

[93]:
```
a=np.array([[23,34],[56,54]])
b=np.tile(a,(1,2))
print(b)
```

```
[[23 34 23 34]
 [56 54 56 54]]
```

## 37 Splitting arrays

## 38 Horizontal splitting

[94]:
```
a=np.random.random((4,4))
a
```

[94]:
```
array([[0.07435012, 0.11670273, 0.47773019, 0.93682874],
       [0.23300694, 0.15293516, 0.64771157, 0.22359685],
       [0.0779587 , 0.96814964, 0.15262315, 0.0607329 ],
```

```
       [0.16087789, 0.57945626, 0.9672175 , 0.82908587]])
```

[95]:
```
b=np.hsplit(a,2)
b
```

[95]:
```
[array([[0.07435012, 0.11670273],
        [0.23300694, 0.15293516],
        [0.0779587 , 0.96814964],
        [0.16087789, 0.57945626]]),
 array([[0.47773019, 0.93682874],
        [0.64771157, 0.22359685],
        [0.15262315, 0.0607329 ],
        [0.9672175 , 0.82908587]])]
```

## 39 Vertical spliiting

[96]:
```
a=np.random.random((4,4))
a
```

[96]:
```
array([[0.49935641, 0.72657731, 0.64025663, 0.66155411],
       [0.07087192, 0.98248534, 0.00679808, 0.07583775],
       [0.18140862, 0.75472511, 0.71291313, 0.99019375],
       [0.48112791, 0.95371078, 0.55753499, 0.72748352]])
```

[97]:
```
b=np.vsplit(a,2)
b
```

[97]:
```
[array([[0.49935641, 0.72657731, 0.64025663, 0.66155411],
        [0.07087192, 0.98248534, 0.00679808, 0.07583775]]),
 array([[0.18140862, 0.75472511, 0.71291313, 0.99019375],
        [0.48112791, 0.95371078, 0.55753499, 0.72748352]])]
```

## 40 Rolling array elements

[98]:
```
arr=np.array([1,2,3,4])
np.roll(arr,1)
```

[98]:
```
array([4, 1, 2, 3])
```

[99]:
```
arr=np.array([1,2,3,4])
np.roll(arr,2)
```

[99]:
```
array([3, 4, 1, 2])
```

## 41 Changing the data type

[100]:
```
a=np.array([12.34,23.46,32.56])
b=a.astype(int)
b
```

[100]:
```
array([12, 23, 32])
```

[101]:
```
a=np.array(["12.34","23.46","32.56"])
b=a.astype(float)
b
```

[101]:
```
array([12.34, 23.46, 32.56])
```

## 42 Matrix operations

## 43 Scaler multiplication

[102]:
```
m1=np.array([[12,22,23],[45,43,33]])
3*m1
```

[102]:
```
array([[ 36,  66,  69],
       [135, 129,  99]])
```

## 44 Matrix addition & substraction

[103]:
```
m1=np.array([10,20,30,40])
m2=np.array([30,40,50,60])
```

[104]:
```
m1+m2
```

[104]:
```
array([ 40,  60,  80, 100])
```

[105]:
```
m1-m2
```

[105]:
```
array([-20, -20, -20, -20])
```

[106]:
```
m1=np.array([[12,22,23],[45,43,33]])
m2=np.array([[22,21,43],[46,23,13]])
```

[107]:
```
m1+m2
```

[107]:
```
array([[34, 43, 66],
       [91, 66, 46]])
```

[108]:
```
m1-m2
```

[108]:
```
array([[-10,   1, -20],
       [ -1,  20,  20]])
```

## 45 Broadcasting

[109]:
```
m1=np.array([10,20,30,40])
```

[110]:
```
m1=m1+100
m1
```

[110]:
```
array([110, 120, 130, 140])
```

## 46 Matrix multiplication

[111]:
```
m1=np.array([[12,23],[10,25]])
m2=np.array([[10,15],[5,15]])
```

[112]:
```
m1*m2 #Hadamard product
```

[112]:
```
array([[120, 345],
       [ 50, 375]])
```

[113]:
```
np.dot(m1,m2)
```

[113]:
```
array([[235, 525],
       [225, 525]])
```

[114]:
```
m1.dot(m2)
```

[114]:
```
array([[235, 525],
       [225, 525]])
```

[115]:
```
np.dot(m2,m1)
```

[115]:
```
array([[270, 605],
       [210, 490]])
```

[116]:
```
m2.dot(m1)
```

[116]:
```
array([[270, 605],
       [210, 490]])
```

## 47 Trace of a matrix

[117]:
```
m1=np.array([[12,23],[10,25]])
```

[118]:
```
np.trace(m1)
```

[118]:
```
37
```

## 48 Diagonal elements of a square matrix

[119]:
```
m1=np.array([[12,23],[10,25]])
```

[120]:
```
np.diag(m1)
```

[120]:
```
array([12, 25])
```

## 49 Matrix transpose

[121]:
```
m1=np.array([[12,23],[10,25]])
np.transpose(m1)
```

[121]:
```
array([[12, 10],
       [23, 25]])
```

## 50 Determinant of a matrix

[122]:
```
m1=np.array([[12,23],[10,25]])
np.linalg.det(m1)
```

[122]:
```
69.99999999999996
```

[123]:
```
m2=np.array([[12,23,45],[10,25,15],[10,20,30]])
np.linalg.det(m2)
```

[123]:
```
-299.99999999999943
```

## 51 Inverse of a matrix

[124]:
```
m1=np.array([[12,23],[10,25]])
np.linalg.inv(m1)
```

[124]:
```
array([[ 0.35714286, -0.32857143],
       [-0.14285714,  0.17142857]])
```

[125]:
```
m2=np.array([[12,23,45],[10,25,15],[10,20,30]])
np.linalg.inv(m2)
```

[125]:
```
array([[-1.5       , -0.7       ,  2.6       ],
       [ 0.5       ,  0.3       , -0.9       ],
```

```
        [ 0.16666667,  0.03333333, -0.23333333]])
```

## 52 Eigenvalues & normalized eigenvectors

```
[126]: m1=np.array([[12,23],[10,25]])
```

```
[127]: eigvals,eigvecs=np.linalg.eig(m1)
```

```
[128]: eigvals
```

```
[128]: array([ 2., 35.])
```

```
[129]: eigvecs
```

```
[129]: array([[-0.91707006, -0.70710678],
               [ 0.39872611, -0.70710678]])
```

```
[ ]:
```