Numpy ### From basic to advance

Why Numpy?

- Effecient use of storage; numpy allocates less space for the same numerical value compared to python
- Faster computations
- Functionality
  - numpy array are mutable

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

Create rank 1 numpy array

In [2]:
```
an_array=np.array([1,2,3,4,5,6])
```

In [3]:
```
an_array.dtype
```

Out[3]:
```
dtype('int32')
```

In [4]:
```
type(an_array)
```

Out[4]:
```
numpy.ndarray
```

In [6]:
```
an_array.shape
```

Out[6]:
```
(6,)
```

In [7]:
```
print(an_array.shape)
```

```
(6,)
```

In [8]:
```
#arrays are mutiple
an_array[0]=888
an_array
```

Out[8]:
```
array([888,   2,   3,   4,   5,   6])
```

In [9]:
```
an_array.size
```

Out[9]:
```
6
```

In [10]:
```
twoD= np.array([[1,2,3,4],[5,6,7,8]])
```

In [11]:
```
twoD.shape
```

Out[11]:
```
(2, 4)
```

In [12]:
```
twoD[1]
```

Out[12]:
```
array([5, 6, 7, 8])
```

Special kinds of numpy arrays

- Zero matrix np.zeros(shape of ndarray)
- Array of ones np.ones(shape of ndarray)
- Full of a specific number np.full(shape, desired number)
- Diagonal np.eye()
- Random foats between 0 and 1 np.random.random(shape) shape is of the form (rows, columns)

In [15]:
```
# make sure you put the dimension of the array in tuple, otherwise it will giv
e only 1D array of zeros
zeros= np.zeros((3,3))
```

In [83]:
```
zeros
```

Out[83]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [18]:
```
np.zeros(2)
```

Out[18]:
```
array([0., 0.])
```

In [23]:
```
nines = np.full((4,4), 9)
```

In [24]:
```
nines
```

Out[24]:
```
array([[9, 9, 9, 9],
       [9, 9, 9, 9],
       [9, 9, 9, 9],
       [9, 9, 9, 9]])
```

In [21]:
```
np.full(3,4)
```

Out[21]:
```
array([9, 9, 9])
```

In [25]:
```
diag = np.eye(4,4)
```

In [26]:
```
diag
```

Out[26]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [27]:
```
rand=np.random.random((4,5))
```

In [28]:
```
rand
```

Out[28]:
```
array([[0.62846509, 0.15311307, 0.78382488, 0.50252415, 0.61031993],
       [0.15988735, 0.02167429, 0.93329793, 0.56364851, 0.24344136],
       [0.38286741, 0.74509738, 0.31641274, 0.80937076, 0.10393275],
       [0.72948173, 0.37106967, 0.16314785, 0.59350391, 0.11193887]])
```

Numpy Random Numbers Generators

1. **np.random.rand** or np.random.random: uniform </U> distribution betweein 0 and 1
2. **np.random.randn**: from standard normal distribution -3 to 3, mean =0, std =1
3. **np.random.randint**(low, high, shape): generates a random int within a specific range
4. **np.random.uniform(low,high,size)**: generates random numbers number from a uniform distribution within a specified range
5. **np.random.normal(mean, std, size)**: generates random numbers from a normal distribution with specified mean and std
6. **np.random.choice**: picks a random from a given 1D array
7. **np.random.shuffle**: shuffles the elelments of an array in place
8. ***np.random.permutation()**

In [29]:
```
np.random.rand(2,3)
```

Out[29]:
```
array([[0.28407029, 0.52377717, 0.6211543 ],
       [0.35909238, 0.60912432, 0.00144858]])
```

In [32]:
```
np.random.randn(2,3)
```

Out[32]:
```
array([[-0.05456948, -2.47635119, -0.32095651],
       [-0.77294637,  0.6017935 , -2.01598825]])
```

In [37]:
```
np.random.randint(0,50,3)
```

Out[37]:
```
array([47, 19, 39])
```

In [38]:
```
np.random.randint(1,7,(3,3))
```

Out[38]:
```
array([[4, 1, 2],
       [2, 1, 4],
       [2, 3, 6]])
```

In [39]:
```
np.random.uniform(1,7,(3,3))
```

Out[39]:
```
array([[3.13064547, 4.58887848, 2.04410522],
       [6.00838127, 1.12004857, 4.19955574],
       [3.15636952, 4.16583232, 3.08835395]])
```

In [40]:
```
np.random.normal(34,2.5,(3,3))
```

Out[40]:
```
array([[36.75160641, 35.37859239, 36.79095631],
       [34.65084341, 35.23104313, 35.20395724],
       [34.51542855, 33.96079292, 27.52431136]])
```

In [56]:
```
np.random.choice([1,2,3])
```

Out[56]:
```
3
```

In [49]:
```
x=[]
for i in range(20):
    x.append(np.random.choice(np.arange(1,7)))
```

In [50]:
```
x
```

Out[50]:
```
[3, 6, 3, 5, 3, 1, 3, 5, 2, 4, 5, 3, 1, 3, 5, 3, 6, 6, 3, 1]
```

In [57]:
```
np.random.choice(np.arange(100))
```

Out[57]:
```
68
```

In [71]:
```
x2=np.arange(10)
```

In [72]:
```
x2
```

Out[72]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [73]:
```
np.random.shuffle(x2)
```

In [74]:
```
x2
```

Out[74]:
```
array([0, 4, 2, 6, 9, 8, 1, 3, 5, 7])
```

In [75]:
```
x2
```

Out[75]:
```
array([0, 4, 2, 6, 9, 8, 1, 3, 5, 7])
```

In [77]:
```
x3=['a','b','c','d']
```

In [78]:
```
np.random.shuffle(x3)
```

In [79]:
```
x3
```

Out[79]:
```
['c', 'b', 'd', 'a']
```

```
In [80]: x3
Out[80]: ['c', 'b', 'd', 'a']

In [81]: np.random.shuffle(x3)

In [82]: x3
Out[82]: ['a', 'c', 'd', 'b']
```

Indexing Arrays

Things to remember:

- indexing starts with 0
- output is -1
- :x means everything before to x-1
- slicing creates reference to the original array, not a new array. Any changes made to either the original or the slice is reflected in both.
- If we need to create a new array we need to place the slice inside np.array() ex(np.array(array[:,3])

```
In [106]: My_array=np.array([['a','b','c'],['h','i','j'
          ],['x','y','z']])

In [107]: My_array
Out[107]: array([['a', 'b', 'c'],
                 ['h', 'i', 'j'],
                 ['x', 'y', 'z']], dtype='<U1')

In [90]: My_array[0]
Out[90]: array(['a', 'b', 'c'], dtype='<U1')

In [91]: My_array[:,1]
Out[91]: array(['b', 'i', 'y'], dtype='<U1')

In [94]: My_array[1:3,1:3]
Out[94]: array([['i', 'j'],
                ['y', 'z']], dtype='<U1')

In [95]: My_array
Out[95]: array([['a', 'b', 'c'],
                ['h', 'i', 'j'],
                ['x', 'y', 'z']], dtype='<U1')
```

In [110]:
```
an_array=My_array[1:3,1:3]
```

In [111]:
```
an_array
```

Out[111]:
```
array([['i', 'j'],
       ['y', 'z']], dtype='<U1')
```

In [112]:
```
new_array=np.array(My_array[1:3,1:3])
```

In [113]:
```
new_array
```

Out[113]:
```
array([['i', 'j'],
       ['y', 'z']], dtype='<U1')
```

Numpy array are mutable

In [114]:
```
# we will see that changing element [2,2] in My_array will also be relflected in an_array
My_array[2,2]='m'
```

In [115]:
```
My_array
```

Out[115]:
```
array([['a', 'b', 'c'],
       ['h', 'i', 'j'],
       ['x', 'y', 'm']], dtype='<U1')
```

In [116]:
```
an_array
```

Out[116]:
```
array([['i', 'j'],
       ['y', 'm']], dtype='<U1')
```

In [117]:
```
# new_array is not affected
new_array
```

Out[117]:
```
array([['i', 'j'],
       ['y', 'z']], dtype='<U1')
```

In [118]:
```
new_array[0,0]='n'
```

In [119]:
```
new_array
```

Out[119]:
```
array([['n', 'j'],
       ['y', 'z']], dtype='<U1')
```

In [120]:
```
My_array
```

Out[120]:
```
array([['a', 'b', 'c'],
       ['h', 'i', 'j'],
       ['x', 'y', 'm']], dtype='<U1')
```

In [121]:
```
an_array[0,0]='1'
```

In [122]:
```
My_array
```

Out[122]:
```
array([['a', 'b', 'c'],
       ['h', '1', 'j'],
       ['x', 'y', 'm']], dtype='<U1')
```

Boolean Indexing

- Filter creates an array of true and false
- array[filter] returns the values where the index was true

In [126]:
```
big_array=np.array([[1,2,3,4],[11,22,33,44], [111,222,333,444]])
```

In [127]:
```
big_array
```

Out[127]:
```
array([[  1,   2,   3,   4],
       [ 11,  22,  33,  44],
       [111, 222, 333, 444]])
```

In [133]:
```
filter = (big_array > 33)
```

In [134]:
```
filter
```

Out[134]:
```
array([[False, False, False, False],
       [False, False, False,  True],
       [ True,  True,  True,  True]])
```

In [137]:
```
print(big_array[filter])
```

```
[ 44 111 222 333 444]
```

In [138]:
```
indices = np.where(filter)
```

In [139]:
```
indices
```

Out[139]:
```
(array([1, 2, 2, 2, 2], dtype=int64), array([3, 0, 1, 2, 3], dtype=int64))
```

'any()' and 'all()'

In [226]:
```
filter.any()
```

Out[226]:
```
True
```

In [227]:
```
filter.all()
```

Out[227]:
```
False
```

Arithmetic and Statistical Operations

In [145]:
```
big_array+0.5
```

Out[145]:
```
array([[  1.5,   2.5,   3.5,   4.5],
       [ 11.5,  22.5,  33.5,  44.5],
       [111.5, 222.5, 333.5, 444.5]])
```

In [148]:
```
x=np.array([[1,2,3],[1,2,3]])
y = np.array([[1,1,1],[0,0,0]])
```

In [149]:
```
x+y
```

Out[149]:
```
array([[2, 3, 4],
       [1, 2, 3]])
```

In [150]:
```
x-y
```

Out[150]:
```
array([[0, 1, 2],
       [1, 2, 3]])
```

In [151]:
```
x*y
```

Out[151]:
```
array([[1, 2, 3],
       [0, 0, 0]])
```

In [153]:
```
np.sqrt(16)
```

Out[153]:
```
4.0
```

In [154]:
```
x.sum()
```

Out[154]:
```
12
```

In [155]:
```
y.mean()
```

Out[155]:
```
0.5
```

In [156]:
```
big_array.max()
```

Out[156]:
```
444
```

In [157]:
```
big_array.min()
```

Out[157]:
```
1
```

```
In [160]: big_array.mean(axis=0)
Out[160]: array([ 41.,  82., 123., 164.])

In [161]: big_array.mean(axis=1)
Out[161]: array([  2.5,  27.5, 277.5])

In [163]: big_array.std()
Out[163]: 143.53135545935598
```

Sorting

array **.sort()**

```
In [182]: unsorted = np.random.randint(0,15,5)

In [183]: unsorted
Out[183]: array([13,  0, 13, 11,  1])

In [190]: sorted.sort()

In [191]: sorted
Out[191]: array([ 0,  1, 11, 13, 13])

In [192]: unsorted
Out[192]: array([13,  0, 13, 11,  1])

In [193]: ex = np.array(['x', 'y', 'a', 'd', 'c'])

In [194]: ex.sort()

In [195]: ex
Out[195]: array(['a', 'c', 'd', 'x', 'y'], dtype='<U1')
```

Unique

np.unique(array)

```
In [196]: redun = np.array([1,1,1,1,1,333,2,2,2])
```

In [199]:
```
np.unique(redun)
```

Out[199]:
```
array([  1,   2, 333])
```

Set Operations

np.intersection1d
np.union1d
np.setdif1d
np.inld

In [200]:
```
s1 = np.array(['desk', 'chair', 'bulb'])
s2 = np.array(['lamp', 'bulb', 'chair'])
```

In [201]:
```
np.intersect1d(s1,s2)
```

Out[201]:
```
array(['bulb', 'chair'], dtype='<U5')
```

In [202]:
```
np.union1d(s1,s2)
```

Out[202]:
```
array(['bulb', 'chair', 'desk', 'lamp'], dtype='<U5')
```

In [204]:
```
np.setdiff1d(s1,s2)
```

Out[204]:
```
array(['desk'], dtype='<U5')
```

In [205]:
```
np.in1d(s1,s2)
```

Out[205]:
```
array([False,  True,  True])
```

Broadcasting

In [207]:
```
start = np.zeros((4,3))
```

In [208]:
```
add_rows = np.array([1,0,1])
```

In [209]:
```
start
```

Out[209]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [210]:
```
add_rows
```

Out[210]:
```
array([1, 0, 1])
```

In [211]:
```
y = start + add_rows
print(y)
```

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

In [212]:
```
add_col = np.array([[0,1,2,3]])
```

In [215]:
```
add_col = add_col.T
```

In [216]:
```
x = start + add_col
```

In [217]:
```
print(x)
```

```
[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]
 [3. 3. 3.]]
```

In [223]:
```
ar=np.arange(20)
```

In [224]:
```
ar.reshape(4,5)
```

Out[224]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [ ]:
```
_
```