Lecture14

March 13, 2024

[2]:
```
import numpy as np
```

1 Arrays

[22]:
```
# create array
L=[1,2,3,4]
a=np.array(L)  # from list
np.arange(5,30, .5)    # using arange
np.linspace(10,20, 10)
np.ones(10)
np.zeros(10)
```

[22]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[25]:
```
a=np.arange(10)
a.shape
```

[25]:
```
(10,)
```

[26]:
```
a.shape=(5,2)
```

[27]:
```
a
```

[27]:
```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

[28]:
```
a=a.reshape(2,5)
```

[28]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

[30]:
```
type(a)
```

```
[30]: numpy.ndarray
```

```
[31]: a.dtype
```

```
[31]: dtype('int64')
```

2 Lists VS np.array

```
[39]: L=[1, 2,3, 2.3, 'hasan']
```

```
[40]: np.array([1,2,3, 2.3]).dtype
```

```
[40]: dtype('float64')
```

```
[41]: np.array([1,2,3.2,'hasan'])
```

```
[41]: array(['1', '2', '3.2', 'hasan'], dtype='<U32')
```

```
[43]: a=np.array(L)
```

```
[44]: L[2]
```

```
[44]: 3
```

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

```
[45]: '3'
```

```
[ ]:
```

3 subscripting and slicing

```
[46]: a
```

```
[46]: array(['1', '2', '3', '2.3', 'hasan'], dtype='<U32')
```

```
[49]: a[0:2]
```

```
[49]: array(['1', '2'], dtype='<U32')
```

```
[50]: a=np.arange(10).reshape(2,5)
```

```
[52]: a
```

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

```
[55]: # a[row_id, col_id]
      # a[row_id][col_id]
      a[1,2]
      a[1][2]
```

```
[55]: 7
```

```
[57]: b=np.arange(25).reshape(5,5)
      b
```

```
[57]: array([[ 0,  1,  2,  3,  4],
              [ 5,  6,  7,  8,  9],
              [10, 11, 12, 13, 14],
              [15, 16, 17, 18, 19],
              [20, 21, 22, 23, 24]])
```

```
[59]: b[1, :4]
```

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

```
[60]: b[[0,-1],[0,-1]]
```

```
[60]: array([ 0, 24])
```

```
[61]: b[:, 2]
```

```
[61]: array([ 2,  7, 12, 17, 22])
```

```
[62]: b[:][2]
```

```
[62]: array([10, 11, 12, 13, 14])
```

3.0.1 Exercise-1

```
[68]: # create numpy 10x10 array of zeros and frame it with ones
      a=np.zeros((5,5))
      a[[0,-1],:]=1
      a[:,[0,-1]]=1
      a
```

```
[68]: array([[1., 1., 1., 1., 1.],
              [1., 0., 0., 0., 1.],
              [1., 0., 0., 0., 1.],
              [1., 0., 0., 0., 1.],
        [1., 1., 1., 1., 1.]])
```

```
[71]: a=np.ones((5,5))
      a[1:-1,1:-1]=0
      a
```

```
[71]: array([[1., 1., 1., 1., 1.],
             [1., 0., 0., 0., 1.],
             [1., 0., 0., 0., 1.],
             [1., 0., 0., 0., 1.],
             [1., 1., 1., 1., 1.]])
```

4 Operators

```
[74]: # Arithmetic
      # +,-,*,/, %, **
      a=np.arange(5)
      b=np.arange(5)
      print(a)
      print(b)
```

```
[0 1 2 3 4]
[0 1 2 3 4]
```

```
[77]: a+b
```

```
[77]: array([0, 2, 4, 6, 8])
```

```
[78]: L1=[1,2,3]
      L2=[1,2,3]
      L1+L2
```

```
[78]: [1, 2, 3, 1, 2, 3]
```

```
[79]: a*b
```

```
[79]: array([ 0,  1,  4,  9, 16])
```

```
[82]: %%timeit
      L1=list(range(100000))
      L2=list(range(100000))
      [L1[i]+L2[i] for i in range(100000)]
```

```
7.06 ms ± 452 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

[83]:
```
%%timeit
a=np.arange(100000)
b=np.arange(100000)
a+b
```

```
124 µs ± 8.98 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

[84]:
```
# Arithmetic broadcasting
# +,-,*,/,%,**
a
```

[84]:
```
array([0, 1, 2, 3, 4])
```

[85]:
```
a*2
```

[85]:
```
array([0, 2, 4, 6, 8])
```

[86]:
```
# Comparison Operators
# >, <,>=,<=,==, != (on arrays or as broadcasting)
a
```

[86]:
```
array([0, 1, 2, 3, 4])
```

[87]:
```
b
```

[87]:
```
array([0, 1, 2, 3, 4])
```

[89]:
```
a==b
```

[89]:
```
array([ True,  True,  True,  True,  True])
```

[90]:
```
a>3
```

[90]:
```
array([False, False, False, False,  True])
```

[92]:
```
# Logical Operators
# &, |, ~
c=np.array([True, True, False, True])
d=np.array([True, False, True, True])
```

[95]:
```
c&d
~c
```

[95]:
```
array([False, False,  True, False])
```

[98]:
```
# important methods
c.all()
```

```
c.any()
```

```
[98]: True
```

```
[99]: a
```

```
[99]: array([0, 1, 2, 3, 4])
```

```
[104]: k=np.array([10,0,5,6,7])
```

```
[105]: a&k
```

```
[105]: array([0, 0, 0, 2, 4])
```

```
[122]: # important functions
       # np.mean, np.median, np.sum, np.std, np.argmax, np.argmin, np.argsort, np.
        ↪unique(return_counts=True)
```

5 Fancy indexing

```
[123]: a
```

```
[123]: array([0, 1, 2, 3, 4])
```

```
[124]: a<3
```

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

```
[125]: a[np.array([True, True, False, False, True])]
```

```
[125]: array([0, 1, 4])
```

```
[126]: a[a<3]
```

```
[126]: array([0, 1, 2])
```

```
[127]: a
```

```
[127]: array([0, 1, 2, 3, 4])
```

```
[128]: a[a%2==0]
```

```
[128]: array([0, 2, 4])
```

```
[129]: a[a!=4]
```

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

```
[130]: a[(a!=4)&(a!=0)]
```

```
[130]: array([1, 2, 3])
```

```
[131]: x=np.arange(30).reshape(5,6)
       x
```

```
[131]: array([[ 0,  1,  2,  3,  4,  5],
              [ 6,  7,  8,  9, 10, 11],
              [12, 13, 14, 15, 16, 17],
              [18, 19, 20, 21, 22, 23],
              [24, 25, 26, 27, 28, 29]])
```

```
[132]: sum(x[x%2==0])
```

```
[132]: 210
```

```
[146]: # cat datafiles/attendance.txt
       a=np.loadtxt('datafiles/attendance.txt',dtype=str, delimiter='-')
       a[:,1:]
```

```
[146]: array([['111110111110100011111010110101010101111010'],
              ['010111111011111111111101111110111111111101111'],
              ['111110111111111111101111110111111101111110111'],
              ['110101010111111101111101110111111111111101111'],
              ['111111110000001100000011111000000111111110111'],
              ['101111110100011111010110101010101111111111111'],
              ['111011110111111111110111011111111111101110111'],
              ['111111111111111110111111111111111111111111101'],
              ['111111011110100000111111011111111101111111011'],
              ['111111111110111111111111111111110111111111110111'],
              ['101011111110111111011110111111111110010111111'],
              ['101111110111101111111111101011111110110111111'],
              ['101000111111010110101010101100000011111111111'],
              ['111011111111101111111011111111101111110111111'],
              ['111111100000011111111111111111111110111101111']], dtype='<U45')
```

```
[142]: # find how many times  each of these employees was absent
       # find the number of employees who ere absent at the first day of the month
       # find the id of the employee of the largest number absents
```

```
[ ]:
```

```
[ ]:
```

[ ]:
```

```
[ ]:
```

```
[ ]:
```

```