Agenda

- **2-D arrays (Matrices)**✅
  - `reshape()` ✅
  - Transpose✅
  - Converting Matrix back to Vector - `flatten()` ✅
- **Introduction to use case**
- **Indexing and Slicing on 2D**
  - Indexing✅
  - Slicing✅
  - Masking (Fancy Indexing)✅
- **Universal Functions (ufunc) on 2D**
  - Aggregate Function/ Reduction functions - `sum()`, `mean()`, `min()`, `max()` ✅
  - Axis argument✅
  - Logical Operations✅
  - Sorting function - `sort()`, `argsort()` ✅
- **Use Case: Fitness Data analysis**
  - Loading data set and EDA using numpy
  - `np.argmax()`

2-D arrays (Matrices)

Transpose

In [4]:
```
a=np.arange(1,13)
a=a.reshape((3,4))
a
```

Out[4]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [5]:
```
a.transpose()
```

Out[5]:
```
array([[ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11],
       [ 4,  8, 12]])
```

In [6]:
```
a.T
```

Out[6]:
```
array([[ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11],
       [ 4,  8, 12]])
```

In [10]:
```
b=np.arange(1,13)
c=b.reshape((1,12))
```

In [11]:
```
b
```

Out[11]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [12]:
```
c
```

Out[12]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]])
```

In [14]:
```
b.shape
```

Out[14]:
```
(12,)
```

In [15]:
```
c.shape
```

Out[15]:
```
(1, 12)
```

In [16]:
```
c.T
```

Out[16]:
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12]])
```

In [17]:
```
b.T
```

Out[17]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [ ]:
```

```

`flatten()`

In [18]:
```
a
```

Out[18]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [ ]:
```
# nd array to 1d array
```

In [19]:
```
a.flatten()
```

Out[19]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [20]:
```
a.reshape((-1,))
```

Out[20]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [ ]:

Indexing and Slicing on 2D

Indexing

In [21]:
```
a
```

Out[21]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [22]:
```
# method1

a[1][1]
```

Out[22]:
```
6
```

In [23]:
```
# method2  # recommended
a[1,1]
```

Out[23]:
```
6
```

In [24]:
```
a[-1,-2]
```

Out[24]:
```
11
```

In [25]:
```
a[2,-2]
```

Out[25]:
```
11
```

In [26]:
```
a[[1,2,2],[1,2,3]]
```

Out[26]:
```
array([ 6, 11, 12])
```

In [28]:
```
a[[0,2,2],[2,3,0]]
```

Out[28]:
```
array([ 3, 12,  9])
```

In [29]:
```
a[[-2,-3,-1,-2],[-3,-1,-3,-1]]
```

Out[29]:
```
array([ 6,  4, 10,  8])
```

Slicing

In [30]:
```
a
```

Out[30]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [31]:
```
a[::,::]
```

Out[31]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [32]:
```
a[:2:1,::]
```

Out[32]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [33]:
```
a[:2:1,1::2]
```

Out[33]:
```
array([[2, 4],
       [6, 8]])
```

In [34]:
```
a[1::-1,2::1]
```

Out[34]:
```
array([[7, 8],
       [3, 4]])
```

In [35]:
```
a[1::-1,2::-1]
```

Out[35]:
```
array([[7, 6, 5],
       [3, 2, 1]])
```

In [36]:
```
a[0:2:2,-2::-2]
```

Out[36]:
```
array([[3, 1]])
```

In [39]:
```
# a[0:2:2,-2:-1:-2]
```

In [40]:
```
a[::-2,:-4:-3]
```
Out[40]:
```
array([[12],
       [ 4]])
```

Masking

In [41]:
```
a
```
Out[41]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [42]:
```
a+10
```
Out[42]:
```
array([[11, 12, 13, 14],
       [15, 16, 17, 18],
       [19, 20, 21, 22]])
```

In [43]:
```
a<10
```
Out[43]:
```
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True, False, False, False]])
```

In [44]:
```
a[a<10]
```
Out[44]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [45]:
```
a[a>100]
```
Out[45]:
```
array([], dtype=int64)
```

In [46]:
```
a[a<100]
```
Out[46]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [47]:
```
a[(a%2==0)|(a%5==0)]
```
Out[47]:
```
array([ 2,  4,  5,  6,  8, 10, 12])
```

In [48]:
```
a[(a%2==0)&(a%5==0)]
```
Out[48]:
```
array([10])
```

Universal Functions (ufunc) on 2D

`sum()`, `mean()`, `min()`, `max()`

In [49]:
```
a
```
Out[49]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [50]:
```
np.mean(a)
```
Out[50]:
```
6.5
```

In [52]:
```
np.max(a)
```
Out[52]:
```
12
```

In [ ]:
```

```

In [53]:
```
np.min(a)
```
Out[53]:
```
1
```

In [54]:
```
np.sum(a)
```
Out[54]:
```
78
```

Axis argument

In [55]:
```
a
```
Out[55]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [56]:
```
np.sum(a,axis=0)
```
Out[56]:
```
array([15, 18, 21, 24])
```

In [57]:
```
np.sum(a,axis=1)
```
Out[57]:
```
array([10, 26, 42])
```

In [58]:
```
np.sum(a,axis=-1)
```
Out[58]:
```
array([10, 26, 42])
```

In [59]:
```
np.sum(a,axis=-2)
```
Out[59]:
```
array([15, 18, 21, 24])
```

In [60]:
```
a
```
Out[60]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [61]:
```
np.min(a,axis=0)
```
Out[61]:
```
array([1, 2, 3, 4])
```

In [62]:
```
np.min(a,axis=-2)
```
Out[62]:
```
array([1, 2, 3, 4])
```

In [63]:
```
np.min(a,axis=1)
```
Out[63]:
```
array([1, 5, 9])
```

In [64]:
```
np.min(a,axis=-1)
```
Out[64]:
```
array([1, 5, 9])
```

In [74]:
```
a=np.arange(1,13)
a
```
Out[74]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [78]:
```
# np.sum(a,axis=1)
```

In [79]:
```
np.sum(a,axis=0)
```
Out[79]:
```
78
```

In [80]:
```
np.sum(a)
```
Out[80]:
```
78
```

Logical Operations

In [84]:
```
a
```
Out[84]:
```
array([[ 10,   2,  99],
       [100,   4,  21],
       [ 32,   5,  12],
       [  9,  57,  45]])
```

In [85]:
```
a<10
```
Out[85]:
```
array([[False,  True, False],
       [False,  True, False],
       [False,  True, False],
       [ True, False, False]])
```

In [86]:
```
a[a<10]
```
Out[86]:
```
array([2, 4, 5, 9])
```

In [87]:
```
b=np.arange(1,13).reshape((4,3))
b
```
Out[87]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

In [88]:
```
a
```
Out[88]:
```
array([[ 10,   2,  99],
       [100,   4,  21],
       [ 32,   5,  12],
       [  9,  57,  45]])
```

In [89]:
```
a<b
```
Out[89]:
```
array([[False, False, False],
       [False,  True, False],
       [False,  True, False],
       [ True, False, False]])
```

In [90]:
```
np.all(a<b)
```
Out[90]:
```
False
```

In [91]:
```
np.any(a<b)
```
Out[91]:
```
True
```

Sorting function - `sort()`, `argsort()`

In [81]:
```
a=np.array([[ 10,   2,  99],
       [100,   4,  21],
       [ 32,   5,  12],
       [  9,  57,  45]])
```

In [66]:
```
a
```
Out[66]:
```
array([[ 10,   2,  99],
       [100,   4,  21],
       [ 32,   5,  12],
       [  9,  57,  45]])
```

In [67]:
```
np.sort(a,axis=0)
```

Out[67]:
```
array([[  9,   2,  12],
       [ 10,   4,  21],
       [ 32,   5,  45],
       [100,  57,  99]])
```

In [69]:
```
# np.sort(a,axis=-2)
```

In [70]:
```
np.sort(a,axis=1)
```

Out[70]:
```
array([[  2,  10,  99],
       [  4,  21, 100],
       [  5,  12,  32],
       [  9,  45,  57]])
```

In [72]:
```
# np.sort(a,axis=-1)
```

In [82]:
```
a
```

Out[82]:
```
array([[ 10,   2,  99],
       [100,   4,  21],
       [ 32,   5,  12],
       [  9,  57,  45]])
```

In [83]:
```
np.sort(a)
```

Out[83]:
```
array([[  2,  10,  99],
       [  4,  21, 100],
       [  5,  12,  32],
       [  9,  45,  57]])
```

In [94]:
```
# 1D np array
```

In [93]:
```
a=np.array([2,7,3,9,4,1,6])
a
```

Out[93]:
```
array([2, 7, 3, 9, 4, 1, 6, 7])
```

In [95]:
```
#method 1
np.sort(a)
```

Out[95]:
```
array([1, 2, 3, 4, 6, 7, 7, 9])
```

In [96]:
```
a
```

Out[96]:
```
array([2, 7, 3, 9, 4, 1, 6, 7])
```

In [97]:
```
# method2
a.sort()
```

In [98]:
```
a
```

Out[98]:
```
array([1, 2, 3, 4, 6, 7, 7, 9])
```

In [99]:
```
# np.argsort()
```

In [100…
```
a=np.array([2,7,3,9,4,1,6])
a
```

Out[100…
```
array([2, 7, 3, 9, 4, 1, 6])
```

In [101…
```
np.sort(a)
```

Out[101…
```
array([1, 2, 3, 4, 6, 7, 9])
```

In [103…
```
np.argsort(a)
```

Out[103…
```
array([5, 0, 2, 4, 6, 1, 3])
```

In [104…
```
np.argmax(a)
```

Out[104…
```
3
```

In [105…
```
np.argmin(a)
```

Out[105…
```
5
```

In [ ]:

In [ ]:

`np.unique()`

In [ ]:

In [ ]:

In [ ]:

Use Case: Fitness Data analysis

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: