Boolean Indexing

In [2]:

```
import numpy as np
```

In [16]:

```
# book example page 112
#two array one is name n one is number
names = np.array(['Bob','Joe','Will','Bob','Joe','Joe'])
print(names)
print(names == 'Bob')
print(names == 'Joe')
```

```
['Bob' 'Joe' 'Will' 'Bob' 'Joe' 'Joe']
[ True False False  True False False]
[False  True False False  True  True]
```

In [6]:

```
data = np.random.randn(7,4) # generate random numbers every time neww numbers
#having 7 rows and 4 columns
data
```

Out[6]:

```
array([[-0.36715055, -0.7851479 ,  0.6941056 , -1.75986987],
       [ 0.0071448 , -0.33168077, -0.26699984, -0.42817525],
       [-1.4437191 ,  2.02214902,  0.1094695 ,  0.44448232],
       [ 0.80384857, -1.52555777,  0.98783002,  0.22898578],
       [-1.34673211,  0.48681187,  2.24165706, -0.59268519],
       [ 0.11585403,  1.28924746, -0.25057666,  0.31658724],
       [-0.86011421,  1.9720727 ,  1.0531205 ,  0.135843  ]])
```

In [12]:

```
data[True,False,False,False,False,True,False] #jo row humko uthani thi us mei trye likl
#first true means first row # 7 timmes
```

Out[12]:

```
array([], shape=(0, 7, 4), dtype=float64)
```

In [11]:

```
bol = np.array([True,False,False,False,False,True,False])
data[bol]
```

Out[11]:

```
array([[-0.36715055, -0.7851479 ,  0.6941056 , -1.75986987],
       [ 0.11585403,  1.28924746, -0.25057666,  0.31658724]])
```

In [13]:

```
bol = np.array([True,False,False,False,False,True,False],dtype = "bool")
data[bol]  # sirf true wali row uthaega
```

Out[13]:

```
array([[-0.36715055, -0.7851479 ,  0.6941056 , -1.75986987],
       [ 0.11585403,  1.28924746, -0.25057666,  0.31658724]])
```

In [14]:

```
names == 'Bob' #isko b ye numbers ki form mei lega
```

Out[14]:

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

In [20]:

```
# condition
data>1
```

Out[20]:

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

In [22]:

```
data  # where value >1 shows true else false
```

Out[22]:

```
array([[-0.36715055, -0.7851479 ,  0.6941056 , -1.75986987],
       [ 0.0071448 , -0.33168077, -0.26699984, -0.42817525],
       [-1.4437191 ,  2.02214902,  0.1094695 ,  0.44448232],
       [ 0.80384857, -1.52555777,  0.98783002,  0.22898578],
       [-1.34673211,  0.48681187,  2.24165706, -0.59268519],
       [ 0.11585403,  1.28924746, -0.25057666,  0.31658724],
       [-0.86011421,  1.9720727 ,  1.0531205 ,  0.135843  ]])
```

In [24]:

```
data[data>1] # put this in data frame then tue wali values show krega
```

Out[24]:

```
array([2.02214902, 2.24165706, 1.28924746, 1.9720727 , 1.0531205 ])
```

#

In [26]:

```
names=="Bob" #now this answer we put in data tu row index 0 iska dat dipslay krado
```

Out[26]:

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

In [29]:

```
names[names=='Bob']
```

Out[29]:

```
array(['Bob', 'Bob'], dtype='<U4')
```

In [30]:

```
data1  = np.arange(64).reshape(8,8)
data1
```

Out[30]:

```
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, 30, 31],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47],
       [48, 49, 50, 51, 52, 53, 54, 55],
       [56, 57, 58, 59, 60, 61, 62, 63]])
```

In [31]:

```
bolarr = np.array([False, False, True, True, False, True, True, False],dtype="bool")
```

In [32]:

```
bolarr = np.array(["Ali", "Nehal", "Hamza", "Umair", "Ali", "Nehal","Nasir","Nasir"])
cond = bolarr=="Ali"
cond
```

Out[32]:

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

In [34]:

```
data1[cond]
```

Out[34]:

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [32, 33, 34, 35, 36, 37, 38, 39]])
```

In [35]:

```
1 data1
```

Out[35]:

```
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, 30, 31],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47],
       [48, 49, 50, 51, 52, 53, 54, 55],
       [56, 57, 58, 59, 60, 61, 62, 63]])
```

In [37]:

```
1 data1[[2,1,6,4,0]] # it will pick 2nd row then 1st row then 6th thrn 4th thrn 0th row
```

Out[37]:

```
array([[16, 17, 18, 19, 20, 21, 22, 23],
       [ 8,  9, 10, 11, 12, 13, 14, 15],
       [48, 49, 50, 51, 52, 53, 54, 55],
       [32, 33, 34, 35, 36, 37, 38, 39],
       [ 0,  1,  2,  3,  4,  5,  6,  7]])
```

In [38]:

```
1 cond  = (bolarr=="Ali") | (bolarr=="Nasir") #or
2 cond
```

Out[38]:

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

In [40]:

```
1 data1[~cond,3:6]
```

Out[40]:

```
array([[11, 12, 13],
       [19, 20, 21],
       [27, 28, 29],
       [43, 44, 45]])
```

In [45]:

```
1 ~False
```

Out[45]:

```
-1
```

In [46]:

```
~True
```

Out[46]:

```
-2
```

In [48]:

```
data1[~cond]  # un rows ko laega jahan value ttue thi means condition row wise true th
#ali jese 0th n 4th per he tu wo col n row same lega
```

Out[48]:

```
array([[ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29, 30, 31],
       [40, 41, 42, 43, 44, 45, 46, 47]])
```

Fancy Indexing

In [50]:

```
arrf = np.ones((8, 4),dtype = "int")
arrf
```

Out[50]:

```
array([[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, 1, 1, 1],
       [1, 1, 1, 1]])
```

In [62]:

```
for i in range(8):
    arrf[i]=i
arrf
```

Out[62]:

```
array([[0, 0, 0, 0],
       [1, 1, 1, 1],
       [2, 2, 2, 2],
       [3, 3, 3, 3],
       [4, 4, 4, 4],
       [5, 5, 5, 5],
       [6, 6, 6, 6],
       [7, 7, 7, 7]])
```

In [63]:

```
arrf[2]
```

Out[63]:

```
array([2, 2, 2, 2])
```

In [66]:

```
arrf[[7,3,2,0],1:3]
```

Out[66]:

```
array([[7, 7],
       [3, 3],
       [2, 2],
       [0, 0]])
```

In [68]:

```
arrf[[7,3,2,0]]  # 7th row 3rd then 2nd 0 row..
```

Out[68]:

```
array([[7, 7, 7, 7],
       [3, 3, 3, 3],
       [2, 2, 2, 2],
       [0, 0, 0, 0]])
```

In [69]:

```
arrf1 = np.arange(16).reshape(4,4)
```

In [70]:

```
arrf1
```

Out[70]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [72]:

```
arrf1[[2,3,0,1],[2,1,0,3]]
#(2,2),(3,1),(0,0),(1,3)
```

Out[72]:

```
array([10, 13,  0,  7])
```

In [74]:

```
arrf1[[1, 0, 3, 2]] [0:2, [0, 3, 1, 2]]

# 1[0, 3, 1, 2], 0[0, 3, 1, 2],3[0, 3, 1, 2],2[0, 3, 1, 2]
# (1,0),(1,3),(1,1),(1,2)

```

Out[74]:

```
array([[4, 7, 5, 6],
       [0, 3, 1, 2]])
```

In [78]:

```
arrf1[[1, 0, 3, 2]] [[0, 3, 1, 2]]
```

Out[78]:

```
array([[ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [ 0,  1,  2,  3],
       [12, 13, 14, 15]])
```

In [83]:

```
arrf1[[1, 0, 3, 2]] [0:3,[0, 3, 1, 2]]
```

Out[83]:

```
array([[ 4,  7,  5,  6],
       [ 0,  3,  1,  2],
       [12, 15, 13, 14]])
```

In [87]:

```
arrt= np.arange(12).reshape(4,3)
arrt
```

Out[87]:

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [89]:

```
arrt.T  #transpose (3,4)
```

Out[89]:

```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [90]:

```
1 arrt.transpose()
```

Out[90]:

```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [91]:

```
1 np.transpose(arrt)
```

Out[91]:

```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [92]:

```
1 arrt.swapaxes(1,0)
```

Out[92]:

```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [94]:

```
1 arrt.swapaxes(0,1)
```

Out[94]:

```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [95]:

```
1 a =[11,22,33,44,44,22,11]
2 a
```

Out[95]:

```
[11, 22, 33, 44, 44, 22, 11]
```

In [97]:

```
1 sett ={11,22,33,44}
2 sett
```

Out[97]:

```
{11, 22, 33, 44}
```

In [100]:

```
1 aset={1,1,1,55,55,88} # takes only one time the repeated value
2 aset
```

Out[100]:

```
{1, 55, 88}
```

In [ ]:

```
1
```