CS677 Lesson3 - Numpy Indexing

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

In [2]:
```
import matplotlib.pyplot as plt
```

Indexing and Slicing

In [3]:
```
arr = np.arange(10, 20)
arr
```

Out[3]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [4]:
```
arr.size
```

Out[4]:
```
10
```

In [5]:
```
# 5th element

arr[4]
```

Out[5]:
```
14
```

In [6]:
```
# 9 th element

arr[8]
```

Out[6]:
```
18
```

In [7]:
```
arr[4:8]
```

Out[7]:
```
array([14, 15, 16, 17])
```

In [8]:
```
arr[4:]
```

Out[8]:
```
array([14, 15, 16, 17, 18, 19])
```

In [9]:
```
arr[:4]
```

Out[9]:
```
array([10, 11, 12, 13])
```

In [10]:
```
arr[[4,8]]
```

Out[10]:
```
array([14, 18])
```

In [11]:
```
indices = [4, 8]
arr[indices]
```

Out[11]:
```
array([14, 18])
```

In [12]:
```
slice = arr[4:8]
slice
```

Out[12]:
```
array([14, 15, 16, 17])
```

In [13]:
```
slice[0] = 140
arr
```

Out[13]:
```
array([ 10,  11,  12,  13, 140,  15,  16,  17,  18,  19])
```

In [14]:
```
slice[-1] = 170
arr
```

Out[14]:
```
array([ 10,  11,  12,  13, 140,  15,  16, 170,  18,  19])
```

In [15]:
```
slice[:] = 0
arr
```

Out[15]:
```
array([10, 11, 12, 13,  0,  0,  0,  0, 18, 19])
```

In [16]:
```
arr[4] = 14
slice
```

Out[16]:
```
array([14,  0,  0,  0])
```

In [17]:
```
arr
```

Out[17]:
```
array([10, 11, 12, 13, 14,  0,  0,  0, 18, 19])
```

In [ ]:

In [ ]:

2-D indexing

In [18]:
```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr2d
```

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

In [19]:
```
arr2d.size
```

Out[19]:
```
9
```

In [20]:
```
arr2d.shape
```

Out[20]:
```
(3, 3)
```

In [21]:
```
arr2d[0]
```

Out[21]:
```
array([1, 2, 3])
```

In [22]:
```
arr2d[[1,2]]
```

Out[22]:
```
array([[4, 5, 6],
       [7, 8, 9]])
```

In [23]:
```
arr2d[1][2]
```

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

In [24]:
```
arr2d[1,2]
```

Out[24]:
```
6
```

In [25]:
```
slice = arr2d[:]
slice
```

Out[25]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [26]:
```
slice[1,2] = 60
slice
```

Out[26]:
```
array([[ 1,  2,  3],
       [ 4,  5, 60],
       [ 7,  8,  9]])
```

In [27]:
```
arr2d
```

Out[27]:
```
array([[ 1,  2,  3],
       [ 4,  5, 60],
       [ 7,  8,  9]])
```

In [28]:
```
arr2d[[1,2]]
```

Out[28]:
```
array([[ 4,  5, 60],
       [ 7,  8,  9]])
```

In [29]:
```
arr2d[:2]
```

Out[29]:
```
array([[ 1,  2,  3],
       [ 4,  5, 60]])
```

In [30]:
```
arr2d[:2, 1:]
```

Out[30]:
```
array([[ 2,  3],
       [ 5, 60]])
```

In [31]:
```
arr2d[:2, 2]
```

Out[31]:
```
array([ 3, 60])
```

In [32]:
```
arr2d[:, :1]
```

Out[32]:
```
array([[1],
       [4],
       [7]])
```

In [33]:
```
arr2d[:2, 1:] = 0
arr2d
```

Out[33]:
```
array([[1, 0, 0],
       [4, 0, 0],
       [7, 8, 9]])
```

3-D Indexing

In [34]:
```
arr3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
arr3d
```

Out[34]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

In [35]:
```
arr3d.size
```

Out[35]:
```
12
```

In [36]:
```
arr3d.shape
```

Out[36]:
```
(2, 2, 3)
```

In [37]:
```
arr3d[1]
```

Out[37]:
```
array([[ 7,  8,  9],
       [10, 11, 12]])
```

In [38]:
```
arr3d[1][1]
```

Out[38]:
```
array([10, 11, 12])
```

In [39]:
```
arr3d[1,1]
```

Out[39]:
```
array([10, 11, 12])
```

In [40]:
```
arr3d[1][1][1]
```

Out[40]:
```
11
```

In [41]:
```
arr3d[1,1,1]
```

Out[41]:
```
11
```

In [ ]:
```

```

Boolean Indexing

In [42]:
```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
print(names)
```

```
['Bob' 'Joe' 'Will' 'Bob' 'Will' 'Joe' 'Joe']
```

In [43]:
```
data = np.arange(28).reshape(7, 4)
data
```

Out[43]:
```
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]])
```

In [44]:
```
names == 'Joe'
```

Out[44]:
```
array([False,  True, False, False, False,  True,  True])
```

In [45]:
```
data[names == 'Joe']
```

Out[45]:
```
array([[ 4,  5,  6,  7],
       [20, 21, 22, 23],
       [24, 25, 26, 27]])
```

In [46]:
```
data[names == 'Joe', 0]
```

Out[46]:
```
array([ 4, 20, 24])
```

In [47]:
```
data[names == 'Joe', -2:]
```

Out[47]:
```
array([[ 6,  7],
       [22, 23],
       [26, 27]])
```

In [48]:
```
names != 'Joe'
```

Out[48]:
```
array([ True, False,  True,  True,  True, False, False])
```

In [49]:
```
~(names == 'Joe')
```

Out[49]:
```
array([ True, False,  True,  True,  True, False, False])
```

In [50]:
```
data[names != 'Joe']
```

Out[50]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

In [51]:
```
data[~(names == 'Joe') ]
```

Out[51]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

In [52]:
```
mask = (names == 'Bob') | (names == 'Will')
mask
```

Out[52]:
```
array([ True, False,  True,  True,  True, False, False])
```

In [53]:
```
slice = data[mask]
slice
```

Out[53]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

Notes:

Selecting data from an array by boolean indexing always creates a copy of the data.

The Python keywords *and* and *or* do not work with boolean arrays. Use & (and) and | (or) instead.

In [54]:
```
# Setting values
# set all even values to 0

slice[slice % 2 == 0] = 0
slice
```

Out[54]:
```
array([[ 0,  1,  0,  3],
       [ 0,  9,  0, 11],
       [ 0, 13,  0, 15],
       [ 0, 17,  0, 19]])
```

In [55]:
```
data
```

Out[55]:
```
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]])
```

In [56]:
```
data[names != 'Bob'] = -1
data
```

Out[56]:
```
array([[ 0,  1,  2,  3],
       [-1, -1, -1, -1],
       [-1, -1, -1, -1],
       [12, 13, 14, 15],
       [-1, -1, -1, -1],
       [-1, -1, -1, -1],
       [-1, -1, -1, -1]])
```

In [57]:
```
slice
```

Out[57]:
```
array([[ 0,  1,  0,  3],
       [ 0,  9,  0, 11],
       [ 0, 13,  0, 15],
       [ 0, 17,  0, 19]])
```

In [ ]:

In [ ]:

Case Study

In [58]:
```
np.random.seed(321)

scores = np.random.normal(60, 10, (100,2))
scores = np.around(scores)
scores.shape
```

Out[58]:
```
(100, 2)
```

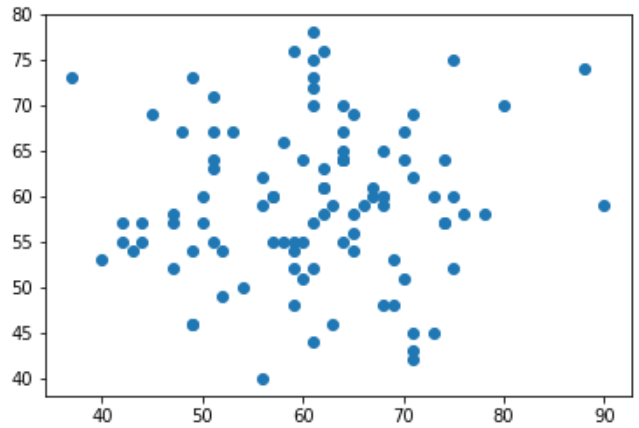
In [59]:
```
scores[:, 0]
```

Out[59]:
```
array([62., 60., 49., 47., 59., 57., 61., 44., 51., 59., 49., 58., 68.,
       64., 51., 50., 40., 74., 56., 64., 68., 61., 56., 63., 62., 61.,
       71., 61., 70., 78., 70., 51., 37., 64., 65., 47., 67., 69., 63.,
       71., 57., 69., 52., 60., 52., 68., 67., 90., 73., 42., 65., 48.,
       71., 75., 42., 76., 53., 56., 71., 74., 49., 45., 51., 61., 65.,
       65., 62., 51., 74., 71., 64., 66., 80., 43., 75., 68., 61., 62.,
       49., 59., 75., 88., 62., 61., 60., 68., 61., 59., 64., 47., 44.,
       58., 59., 57., 64., 54., 50., 64., 70., 73.])
```

In [60]:
```
scores[:, 1]
```

Out[60]:
```
array([76., 51., 54., 52., 76., 55., 75., 57., 55., 55., 46., 66., 60.,
       64., 64., 57., 53., 57., 59., 70., 59., 44., 62., 46., 58., 78.,
       45., 73., 64., 58., 67., 63., 73., 65., 56., 58., 61., 48., 59.,
       69., 60., 53., 49., 55., 54., 48., 60., 59., 60., 55., 69., 67.,
       42., 60., 57., 58., 67., 40., 62., 57., 46., 69., 71., 57., 58.,
       54., 61., 67., 64., 43., 64., 59., 70., 54., 52., 65., 70., 63.,
       73., 54., 75., 74., 61., 52., 64., 60., 72., 52., 55., 57., 55.,
       55., 48., 60., 67., 50., 60., 64., 51., 45.])
```

In [61]:
```
plt.scatter(scores[:, 0], scores[:, 1]);
```



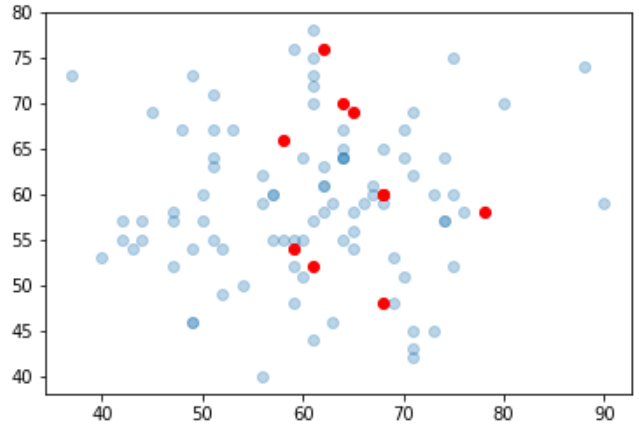
In [62]:
```
# randomly select 10 indices
indices = np.random.choice(scores.shape[0], 10, replace=False)
indices
```

Out[62]:
```
array([12, 79, 45, 29,  0, 83, 19, 11, 50, 85])
```

In [63]:
```
selection = scores[indices]
selection
```

Out[63]:
```
array([[68., 60.],
       [59., 54.],
       [68., 48.],
       [78., 58.],
       [62., 76.],
       [61., 52.],
       [64., 70.],
       [58., 66.],
       [65., 69.],
       [68., 60.]])
```

In [64]:
```
plt.scatter(scores[:, 0], scores[:, 1], alpha=0.3)
plt.scatter(selection[:, 0], selection[:, 1],
            facecolor='red');
```



Case Study - ix_() function

In [65]:
```
# combine different vectors to obtain result for each n-uplet
# to compute a * b for all tuples from the vectors a and b
```

In [66]:
```
a = np.array([2,3])
b = np.array([10, 15, 20])
ax, bx = np.ix_(a,b)
```

In [67]:
```
ax
```

Out[67]:
```
array([[2],
       [3]])
```

In [68]:
```
ax.shape
```

Out[68]:
```
(2, 1)
```

In [69]:
```
bx
```

Out[69]:
```
array([[10, 15, 20]])
```

In [70]:
```
bx.shape
```

Out[70]:
```
(1, 3)
```

In [71]:
```
result = ax * bx
```

In [72]:
```
result
```

Out[72]:
```
array([[20, 30, 40],
       [30, 45, 60]])
```

In [73]:
```
result[1,2]
```

Out[73]:
```
60
```

In [74]:
```
a[1] * b[2]
```

Out[74]:
```
60
```

In [75]:
```
# to compute a + b * c for all triplets from the vectors a, b, and c
```

In [76]:
```
a = np.array([2,3,4,5])
b = np.array([8,5,4])
c = np.array([5,4,6,8,3])
ax,bx,cx = np.ix_(a,b,c)
```

In [77]:
```
ax
```

Out[77]:
```
array([[[2]],

       [[3]],

       [[4]],

       [[5]]])
```

In [78]:
```
ax.shape
```

Out[78]:
```
(4, 1, 1)
```

In [79]:
```
bx
```

Out[79]:
```
array([[[8],
        [5],
        [4]]])
```

In [80]:
```
bx.shape
```

Out[80]:
```
(1, 3, 1)
```

In [81]:
```
cx
```

Out[81]:
```
array([[[5, 4, 6, 8, 3]]])
```

In [82]:
```
cx.shape
```

Out[82]:
```
(1, 1, 5)
```

In [83]:
```
result = ax + bx * cx
result
```

Out[83]:
```
array([[[42, 34, 50, 66, 26],
        [27, 22, 32, 42, 17],
        [22, 18, 26, 34, 14]],

       [[43, 35, 51, 67, 27],
        [28, 23, 33, 43, 18],
        [23, 19, 27, 35, 15]],

       [[44, 36, 52, 68, 28],
        [29, 24, 34, 44, 19],
        [24, 20, 28, 36, 16]],

       [[45, 37, 53, 69, 29],
        [30, 25, 35, 45, 20],
        [25, 21, 29, 37, 17]]])
```

In [84]:
```
result[3, 2, 4]
```

Out[84]:
```
17
```

In [85]:
```
a[3] + b[2] * c[4]
```

Out[85]:
```
17
```