In [1]:
```
#fancy indexing- passing an array of multiple indices to access multiple elements
```

In [2]:
```
import numpy as np
a=np.arange(0,10)
print(a)
#for example we are accessing three elements of the list
print("First element=",a[0],"Fourth element=",a[3],"Seventh element=",a[8])
```

```
[0 1 2 3 4 5 6 7 8 9]
First element= 0 Fourth element= 3 Seventh element= 8
```

In [3]:
```
#using fancy indexing
idx_array=np.array([0,3,8])
print("Elements at first,fourth and seventh element:")
print(a[idx_array])
```

```
Elements at first,fourth and seventh element:
[0 3 8]
```

In [4]:
```
#while fancy indexig shape of the index array is followed rather than original array
print("Original array:")
print(a)
idx_array=np.array([0,3,8,4])#standard fanncy indexing
print("1 dimesional display:")
print(a[idx_array])
idx_array=np.array([0,3,8,4]).reshape((2,2))#here the index is 2 dimesional hence output is also 2 dimensional
print("2 dimesional display:")
print(a[idx_array])
```

```
Original array:
[0 1 2 3 4 5 6 7 8 9]
1 dimesional display:
[0 3 8 4]
2 dimesional display:
[[0 3]
 [8 4]]
```

In [7]:
```
#fancy indexing for multidimesional arrays
x= np.arange(12).reshape((3, 4))
print("x=")
print(x)
row_idx=[0,1,2]#array with index used for row
column_idx=[0,1,2]#array with index used for column
print("Fancy indexing on multi dimensional arrays=")#element x[0,0],x[1,1] and x[2,2]
print(x[row_idx,column_idx])
```

```
x=
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Fancy indexing on multi dimensional arrays=
[ 0  5 10]
```

In [12]:
```
#combination of fancy indexing with other indexing methods
x=np.arange(12).reshape((3, 4))
print("x=")
print(x)
print("Combining fancy indexing with slicing")
print(x[2,[1,2,3]])#returns second,third and fourth element in row 3
```

```
x=
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Combining fancy indexing with slicing
[ 9 10 11]
```

In [21]:
```
#modifying values with fancy indexing
a=np.arange(0,10)
print("a=")
print(a)
indices=[0,1,2,5,7]
a[indices]=0 #replaces values at the indieces array with 0
print("After replacing element 0,1,2,5,7 with 0=")
print(a)
indices1=[1,3,5]
a[indices1]-=1
print("Decrementing element 1,3,5 by 1=")
print(a)
```

```
a=
[0 1 2 3 4 5 6 7 8 9]
After replacing element 0,1,2,5,7 with 0=
[0 0 0 3 4 0 6 0 8 9]
Decrementing element 1,3,5 by 1=
[ 0 -1  0  2  4 -1  6  0  8  9]
```

In [30]:
```
#replacing value in array muliptle times
x=np.zeros(10)
print("x=")
print(x)
x[[0,0]]=[4,6]#assigning element at index 0 with 4 and then reassigning with 6
print("After assigning=")
print(x)#the value which is assigned last is the final value
```

```
x=
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
After assigning=
[6. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [35]:
```
#non repitition of operations in numpy array
x=np.zeros(10)
print("x=")
print(x)
index=[0,1,1,2,2,2,3,3,3,3]
x[index]+=1
print(x)
#here we expect one to be added one time to index 0 two times to index 1 and three times to index 2 and so on
#but the operation is not repeated more than once because of multiple times assignments
```

```
x=
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 0. 0. 0. 0. 0. 0.]
```

In [38]:
```
#ways to obtain repeated operations
x=np.zeros(10)
print("x=")
print(x)
index=[0,1,1,2,2,2,3,3,3,3]
#using at() function
np.add.at(x,index,1)
print(x)
```

```
x=
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 2. 3. 4. 0. 0. 0. 0. 0. 0.]
```