In [1]:
```
#manipulating attributes of array
#indexing of arrays
#slicing of arrays
#reshaping of arrays
#joining and spliting of arrays
```

In [15]:
```
import numpy as np
#size attribute
x=np.random.randint(10,size=(3,3))
```

In [17]:
```
#ndim is the number of dimensions of array,shape gives tuple of array,size gives total size of array
print("Dimensions of array is:",x.ndim)
print("Shape of array is:",x.shape)
print("size of array is:",x.size)
```

```
Dimensions of array is: 2
Shape of array is: (3, 3)
size of array is: 9
```

In [20]:
```
#dtype gives data type of array
print("data type of array is:",x.dtype)
#itemsize returns the size of each element of array
print("The size of each element of array is:",x.itemsize)
#nbytes returns the sum of bytes of all elements
print("total size of array in bytes is:",x.nbytes)
```

```
data type of array is: int32
The size of each element of array is: 4
total size of array in bytes is: 36
```

In [35]:
```
#indexing of array
a=np.arange(1,11)
print(a)
print("First element of array is:",a[0])#forward indexing
print("last element of array using backward indexing:",a[-1])#backward indexing
print("Fourth element of array is:",a[3])
#indexing of multi dimensional arrays
a1=np.random.randint(10,size=(3,3,3))
print(a1)
print("Element at blocksize 1 row 2 column 3 is:",a1[0,1,2])#comma seperated indices method
print("Element at blocksize 1 row 2 column 3 is:",a1[0][1][2])#giving each index seperately
```

```
[ 1  2  3  4  5  6  7  8  9 10]
First element of array is: 1
last element of array using backward indexing: 10
Fourth element of array is: 4
[[[1 1 6]
  [1 2 1]
  [4 6 2]]

 [[1 7 7]
  [5 8 3]
  [5 1 9]]

 [[4 9 5]
  [3 1 9]
  [4 1 9]]]
Element at blocksize 1 row 2 column 3 is: 1
Element at blocksize 1 row 2 column 3 is: 1
```

In [37]:
```
#modification of array values using indexes
print(a)
a[1]=-1#assigning value of second element as -1
print("Modified array:")
print(a)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
Modified array:
[ 1 -1  3  4  5  6  7  8  9 10]
```

In [38]:
```
#when float is inserted it is truncated in int array as arrays have fixed data types
print(a)
a[2]=11.2324553
print("Modified array:")
print(a)
```

```
[ 1 -1  3  4  5  6  7  8  9 10]
Modified array:
[ 1 -1 11  4  5  6  7  8  9 10]
```

In [43]:
```
#slicing of array
a=np.array([1,2,3,4,5,6])
print(a[0:3])#prints elements from index 0 to 2
print(a[:5])#prints element sfrom beginning till index 4
print(a[2:])#print elemnts from index 2 till end of array
print(a[0::2])#prints even index elements of array
print(a[::-1])#prints array in reverse
```

```
[1 2 3]
[1 2 3 4 5]
[3 4 5 6]
[1 3 5]
[6 5 4 3 2 1]
```

In [58]:
```
#slicing in multidimensional array
a=np.random.randint(0,10,(3,4,5))
print(a)
print("Columns only:")
print(a[:2,:3])#2 rows,3 column
print("First columns only:")
print(a[:,:,0])# : represents empty slicing
print("First rows only:")
print(a[:,0,:])
```

```
[[[4 5 3 9 5]
  [3 5 9 2 2]
  [8 1 9 2 9]
  [7 8 0 3 0]]

 [[9 0 9 0 5]
  [7 3 1 2 1]
  [5 8 3 2 7]
  [0 7 2 1 3]]

 [[7 5 7 5 2]
  [3 5 3 5 6]
  [5 8 2 1 6]
  [1 9 3 9 1]]]
Columns only:
[[[4 5 3 9 5]
  [3 5 9 2 2]
  [8 1 9 2 9]]

 [[9 0 9 0 5]
  [7 3 1 2 1]
  [5 8 3 2 7]]]
First column only:
[[4 3 8 7]
 [9 7 5 0]
 [7 3 5 1]]
First row only:
[[4 5 3 9 5]
 [9 0 9 0 5]
 [7 5 7 5 2]]
```

In [59]:
```
#unlike python lists changes in slices change the original list
#slice of numpy array is more of a view of an array rather than a copy
num_list=np.array([1,2,3,4,5,6,7,8,9,0])
py_list=[1,2,3,4,5,6,7,8,9,10]
slice_num_list=num_list[0:6]
slice_py_list=py_list[0:6]
slice_py_list[1]=-1#updating the second element of sliced regular list
slice_num_list[1]=-1#updating the second element of sliced numpy list
print(num_list)#prinitng numpy list
print(py_list)#prinitng regular list
```

```
[ 1 -1  3  4  5  6  7  8  9  0]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [68]:
```
#creating copies of array slices using copy function(deep copy)
arr=np.array([range(i,i+3) for i in [2,3,4,5,6,7,8]])
print(arr)
#creating a shallow copy of first column
slice_shallow=arr[:,0]
#creating a deep copy using copy function of first column
slice_deep=arr[:,0].copy()
#changing first element in shallow copy
slice_shallow[0]=0
#changing second element value in deep copy
slice_deep[1]=0
print("Modified array:")
print(arr)
#first element of first column is modifed in original array.Hwoever second element is not changed
#because the slice is properly copied
```

```
[[ 2  3  4]
 [ 3  4  5]
 [ 4  5  6]
 [ 5  6  7]
 [ 6  7  8]
 [ 7  8  9]
 [ 8  9 10]]
Modified array:
[[ 0  3  4]
 [ 3  4  5]
 [ 4  5  6]
 [ 5  6  7]
 [ 6  7  8]
 [ 7  8  9]
 [ 8  9 10]]
```

In [73]:
```
#reshape of arrays
ar=np.arange(1,10)#row array(1x9)
print("Original array:")
print(ar)
#reshaping 1x9 array to 3x3 array
ar1=ar.reshape((3,3))
print("Reshaped array:")
print(ar1)
```

```
Original array:
[1 2 3 4 5 6 7 8 9]
Reshaped array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [84]:
```
#using new axis-conversion to rwo and column matrix
ar=np.array([1,2,3,4,5,6])#row array(1x9)
print("Original array:")
print(ar)
print("Row matrix:")
print(ar[np.newaxis,:])
print("Column matrix:")#converted into column array(9x1)
print(ar[:,np.newaxis])
```

```
Original array:
[1 2 3 4 5 6]
Row matrix:
[[1 2 3 4 5 6]]
Column matrix:
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]]
```

In [89]:
```
#concatenation of arrays
x=np.array([1,2,3,4,5])
y=np.array([6,7,8,9,0])
z=np.concatenate([x,y])#concatennating arrays x and y using concatenate function
print(z)
c=np.random.randint(0,10,(2,2))
d=np.random.randint(0,10,(2,2))
print(c)
print("+")
print(d)
print("=")
e=np.concatenate([c,d])
print(e)
```

```
[1 2 3 4 5 6 7 8 9 0]
[[3 1]
 [3 1]]
+
[[1 2]
 [6 0]]
=
[[3 1]
 [3 1]
 [1 2]
 [6 0]]
```

In [95]:
```
#concatenating by axis-contenating vertically
c=np.random.randint(0,10,(2,2))
d=np.random.randint(0,10,(2,2))
print(c)
print("+")
print(d)
print("=")
e=np.concatenate([c,d],axis=1)
print(e)
#concatenating muliple arrays
print("concatenating three arrays:")
f=np.concatenate([c,d,e],axis=1)
print(f)
```

```
[[6 8]
 [8 0]]
+
[[9 9]
 [7 0]]
=
[[6 8 9 9]
 [8 0 7 0]]
concatenating three arrays:
[[6 8 9 9 6 8 9 9]
 [8 0 7 0 8 0 7 0]]
```

In [102]:
```
#stacking arrays using hstack and vstack
i=np.array([1,2,3,4,5,6])
j=np.array([[1,2,3,4,5,6],
            [7,8,9,10,11,12]])
#vertically stacking arrays
k=np.vstack([i,j])
print("Stacking array vertically:")
print(k)
#horizontally stacking arrays
print("Stacking array horizontally:")
l=np.array([[7,8,9],
            [13,14,15]])
m=np.hstack([j,l])
print(m)
#dstack-stacking along third axis
```

```
Stacking array vertically:
[[ 1  2  3  4  5  6]
 [ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
Stacking array horizontally:
[[ 1  2  3  4  5  6  7  8  9]
 [ 7  8  9 10 11 12 13 14 15]]
```

In [106]:
```
#splitting array
n=np.array([1,2,3,4,5,6,7,8,9])
#splitting array in two at 3rd element,3rd element is included at 2nd half
print("Original array:")
print(n)
n1,n2=np.split(n,[3])
print("First part of original array:")
print(n1)
print("Second part of original array:")
print(n2)
```

```
Original array:
[1 2 3 4 5 6 7 8 9]
First part of original array:
[1 2 3]
Second part of original array:
[4 5 6 7 8 9]
```

In [114]:
```
#horizontal split for 2 dimension array
a=np.array([[1,2,3,4],[5,6,7,8]])
print("Original array:")
print(a)
a1,a2=np.hsplit(n,[2])
print("First part of original array by horizontal splitting:")
print(a1)
print("Second part of original array by horizontal splitting:")
print(a2)
```

```
Original array:
[[1 2 3 4]
 [5 6 7 8]]
First part of original array by horizontal splitting:
[1 2]
Second part of original array by horizontal splitting:
[3 4 5 6 7 8 9]
```

In [120]:
```
#vertical split for 2 dimension array
a=np.array([[1,2,3,4],
            [5,6,7,8],
            [9,10,11,12],
            [13,14,15,16]])
print("Original array:")
print(a)
a1,a2=np.vsplit(a,[2])
print("First part of original array by vertical splitting:")
print(a1)
print("Second part of original array by vertical splitting:")
print(a2)
#np.dpsplit- splitting across third axis
```

```
Original array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
First part of original array by vertical splitting:
[[1 2 3 4]
 [5 6 7 8]]
Second part of original array by vertical splitting:
[[ 9 10 11 12]
 [13 14 15 16]]
```

In [ ]: