1. If you create one numpy array, from another numpy array, then both the arrays share same memory, ie changes on one array will be reflected to another one as well.

Ex:

a = np.arange(0,10)

print(a)

b = a[:5]

print(b)

b[0] = 10;

print(b, a)

np.shares\_memory(a,b) #True

If you want them to be on different memory then use **copy()** method, so any changes on one array will not be reflected to another one:

c= a[:5].copy()

np.shares\_memory(a,c) #False

1. If you are masking, then while masking deep copy happens, that means the newly created array will be on different memory than older array.

Ex:

a = np.random.randint(0, 20, 25);

mask = (a % 2 == 0)

new\_a = a[mask];

print(new\_a)

np.shares\_memory(a, new\_a)

1. np.eye(10,10). It creates a 2-D numpy array of size 10X10 and all the diagnol elements will be 1 and rest all of the elements are zero. np.eye(3,3,1). It creates a 2-D numpy array of size 10X10 and all the elements that are just 1 row above the diagnol elements(in same column) will be 1 and rest all of the elements are zero. np.eye(3,3,-1). It creates a 2-D numpy array of size 10X10 and all the elements that are just 1 row below the diagnol elements(in same column) will be 1 and rest all of the elements are zero.

Ex:

a = np.eye(5, 5, 1)

print(a)

a = np.eye(5, 5, -1)

print(a)

Output:

[[0. 1. 0. 0. 0.]

[0. 0. 1. 0. 0.]

[0. 0. 0. 1. 0.]

[0. 0. 0. 0. 1.]

[0. 0. 0. 0. 0.]]

[[0. 0. 0. 0. 0.]

[1. 0. 0. 0. 0.]

[0. 1. 0. 0. 0.]

[0. 0. 1. 0. 0.]

[0. 0. 0. 1. 0.]]