

We'll cover the following

- Direct and indirect access
 - Indexing and Fancy Indexing
 - Ravel and Flatten
- Temporary copy

Direct and indirect access

maxxing and minxing, maxxing

```
3 Z = np.zeros(9)
4
5 Z_view = Z[1:3]
6
7 Z_view[...] = 1 #Z_view modifies the base array 'Z'
8 print(Z)
9
10 Z = np.zeros(9)
11
12 Z_copy = Z[[0,1,2]]
13
14 Z_copy[...] = 1 #Z_copy does not modify the base array 'Z'
15 print(Z)
```

RUN

SAVE

RESET

Output

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

```
1 import numpy as np
2 Z = np.zeros(9)
3 index = [0,1,2]
4 Z[index] = 1 # store 1 at index 0,1,2
5 print(Z)
```

RUN **SAVE** **RESET**

Output 1.721s

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

```
1 import numpy as np
2 Z = np.random.uniform(0,1,(5,5)) #draws sample from a uniform distribution
3 Z1 = Z[:,3:]
4 #print(Z1,Z)
5 Z2 = Z[:,0,1,2, :]
6 #print(Z1,Z2)
7 print(np.allclose(Z1,Z2)) #returns True if two arrays are element-wise equal within a tolerance.
8 print(Z1.base is Z) #return true if memory of Z1 is shared with Z and false otherwise
9 print(Z2.base is Z) #return true if memory of Z2 is shared with Z and false otherwise
10 print(Z2.base is None) #return true if memory of Z2 is not shared
```

RUN **SAVE** **RESET**

Output 0.928s

```
True
True
False
True
```

[illegible]

Temporary copy

```
1 import numpy as np
2 X = np.ones(10, dtype=np.int) #create an array X of size 10 containing ones
3 Y = np.ones(10, dtype=np.int) #create an array Y of size 10 containing ones
4 A = 2*X + 2*Y #store 2*X + 2*Y in A
5 print("X:",X)
6 print("Y:",Y)
7 print("A=2*X + 2*Y :\nA:",A)
```

RUN **SAVE** **RESET**

Output 0.7065

```
X: [1 1 1 1 1 1 1 1 1 1]
Y: [1 1 1 1 1 1 1 1 1 1]
A=2*X + 2*Y :
A: [4 4 4 4 4 4 4 4 4 4]
```

```
1 import numpy as np
2 X = np.ones(10, dtype=np.int) #create an array X of size 10 containing ones
3 Y = np.ones(10, dtype=np.int) #create an array Y of size 10 containing ones
4 print("X:",X,"Y:",Y,"\n np.multiply(X, 2, out=X)")
5 np.multiply(X, 2, out=X) # multiply X with 2 and store the result in X
6 print("X:",X,"Y:",Y,"\n np.multiply(Y, 2, out=Y)")
7 np.multiply(Y, 2, out=Y) # multiply Y with 2 and store the result in Y
8 print("X:",X,"Y:",Y,"\n np.add(X, Y, out=X)")
9 np.add(X, Y, out=X) # add X and Y and store the result in X
10 print("X:",X,"Y:",Y)
```

```
1 import numpy as np
2 from tools import timeit
3
4 def test(X,Y):
5     timeit("Z=X + 2.0*Y", globals()) #time taken by Z=X + 2.0*Y
6     timeit("Z = X + 2*Y", globals()) #time taken by Z=X + 2*Y
7     timeit("np.add(X, Y, out=X); np.add(X, Y, out=X)", globals())
8     #time taken by np.add(X, Y, out=X); np.add(X, Y, out=X)
9
10 X = np.ones(100000000, dtype=np.int)
11 Y = np.ones(100000000, dtype=np.int)
12 test(X,Y)
```

Output

```
1 loops, best of 3: 1.08 sec per loop
1 loops, best of 3: 928 msec per loop
1 loops, best of 3: 414 msec per loop
```

Solve this Quiz!

1 Does the following code return a view or a copy?

```
Z = np.zeros(9)
Z_1 = Z[:4]
Z_1[...] = 1
```

☐ A) View

☐ B) Copy

COMPLETED 0%

1 of 5

< >

Now that we have learned views and copies, let's move on to a coding challenge.