Numpy Freshers

note before running any code, run `pip install matplotlib numpy` in your terminal

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

```
import numpy as np # the library to learn in this module
import matplotlib.pyplot as plt # just used for showing images and graphs here
```

```
a = np.zeros(3) # an array filled with zeroes having 3 elements
a
```

```
array([0., 0., 0.])
```

```
a.shape
```

```
(3,)
```

```
b = np.ones(10) # an array filled with ones having 10 elements
b
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
c = np.linspace(2, 10, 5) # a range from 2 to 10 w 5 elements
c
```

```
array([ 2.,  4.,  6.,  8., 10.])
```

```
d = np.array([10, 20])
d
```

```
array([10, 20])
```

```
a_list = [1,2,3,4,5]
e = np.array([a_list])
e
```

```
array([[1, 2, 3, 4, 5]])
```

```
b_list = [[1,2,3],[4,5,6],[7,8,9]]
f = np.array(b_list)
f
```

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

```
np.random.seed(60)
g = np.random.randint(10, size=6)
g
```

```
array([1, 6, 3, 8, 9, 1], dtype=int32)
```

```
# can use normal python indexing with numpy arrays
g[0:2]
```

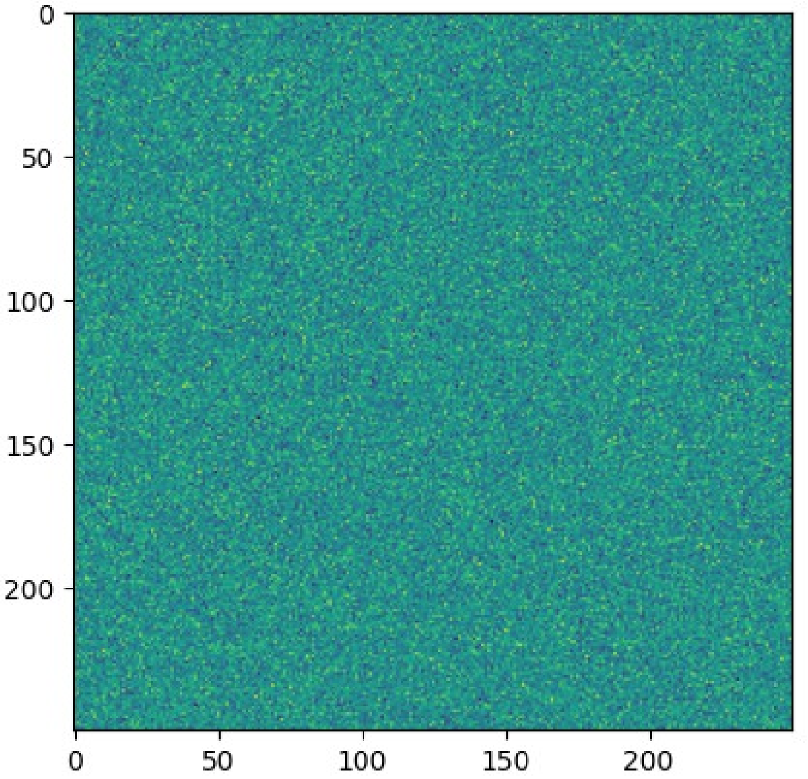
```
array([6, 9], dtype=int32)
```

```
img = np.random.randn(250, 250) # creates a 250x250 matrix filled with
random numbers from a uniform gaussian/normal distribution of mean 0
and variance 1
# the curve of the distribution (prolly) looks like a bell with peak
at x = 0
img
```

```
array([[ 0.20642859, -0.45203477, -0.18171225, ...,  0.94531576,
         0.12604099,  0.65215058],
       [ 0.69072997,  0.12053134,  0.51675033, ..., -1.18998873,
         1.32110756, -0.08590738],
       [ 0.47942739, -0.16077646, -1.67645517, ..., -0.16506097,
        -1.07005751, -1.04490365],
       ...,
       [ 0.91236342,  1.99226847,  0.50530997, ...,  0.97436956,
        -0.26542829, -0.33257631],
       [ 0.09769181,  0.53948442,  1.33183853, ...,  1.18646449,
        -0.24834789, -0.49382488],
       [-1.60903608, -1.44302205, -1.537846  , ..., -0.02507422,
        -0.83271507,  1.4782132 ]], shape=(250, 250))
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1e4bfd82c30>
```

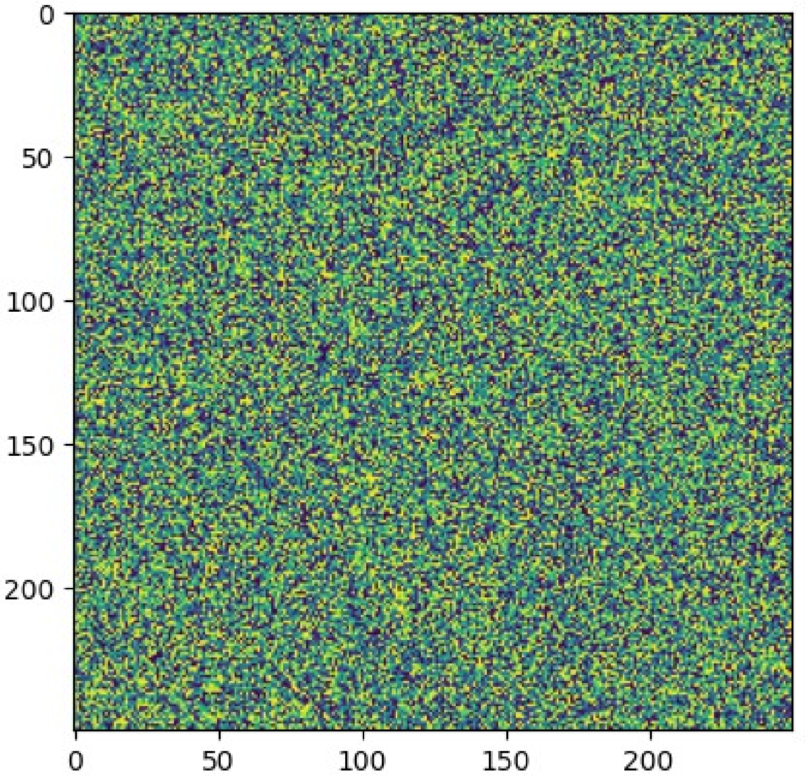


```
x = np.sin(img) # applies sine function on each element of the array
x

array([[-0.42630038, -0.65852821, -0.93130601, ..., -0.25792241,
         0.40224801,  0.1180299 ],
       [-0.99561262, -0.25010321,  0.10784151, ..., -0.03865291,
         0.65106588, -0.85613297],
       [ 0.29799137, -0.60369943,  0.09782824, ...,  0.89928848,
         0.48341168,  0.99985402],
       ...,
       [ 0.1756496 ,  0.75544068, -0.7063979 , ...,  0.2667522 ,
        -0.09004359,  0.52966233],
       [ 0.99998933,  0.90955048, -0.46957395, ..., -0.94319745,
         0.98444593, -0.11252258],
       [-0.08971117, -0.3203463 , -0.93758733, ...,  0.31607456,
         0.27870136,  0.4733032 ]], shape=(250, 250))

plt.imshow(x)

<matplotlib.image.AxesImage at 0x1e4bc8dba70>
```



```
np.sum(x) # sum of all elements in the array

np.float64(-52.03467951064626)

np.max(x) # maximum element of the array

np.float64(0.999999998212617)

np.min(x) # minimum element of the array

np.float64(-0.9999999999998166)

np.var(x) # variance (sigma) of the array

np.float64(0.4321620453792374)

np.mean(x) # mean/avg of the array

np.float64(-0.0008325548721703401)

np.std(x) # standard deviation of the array

np.float64(0.6573903295449648)

np.prod(x) # product of all elements

np.float64(0.0)
np.argmin(x) # index of smallest element
```

```
np.int64(51175)
```

```
np.argmax(x) # index of largest element
```

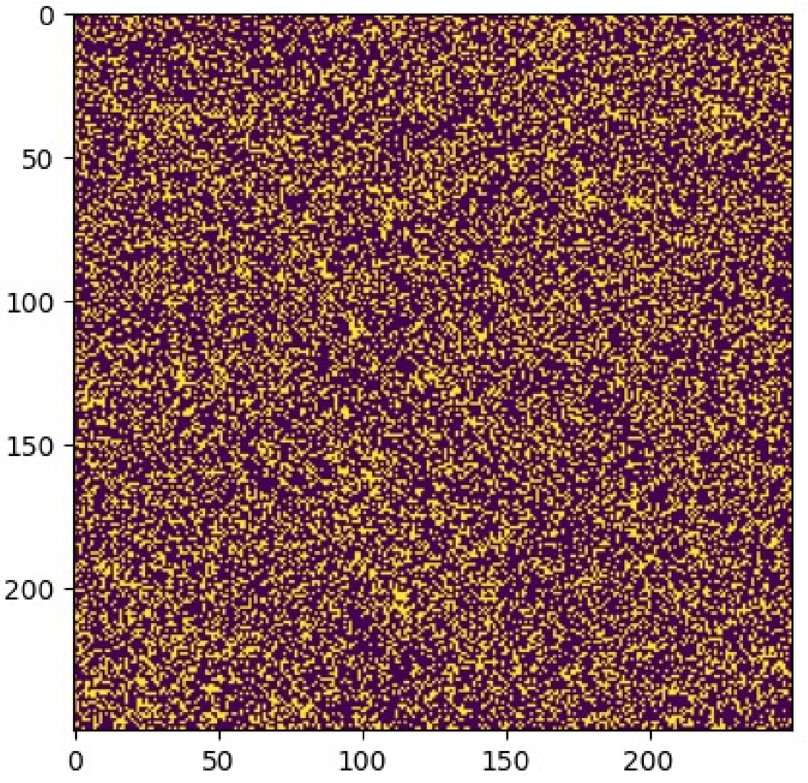
```
np.int64(47162)
```

```
w = np.array([1,2,3,4,5])
print(w < 3) # a binary array which runs the comparision per element
of the array
print(w > 3)
```

```
[ True  True False False False]
[False False False  True  True]
```

```
mask = np.where(img > 0.5, 255, 122) # in the image if any value is
greater than 0.5 replace it with 255 else replace it with 122
plt.imshow(mask)
```

```
<matplotlib.image.AxesImage at 0x1e4beb84e00>
```



```
a_1 = np.array([1,2,4,5])
a_2 = np.array([0,1,2,3])
a_1 + a_2
```

```
array([1, 3, 6, 8])
```

```
a_1 + 30
```

```
array([31, 32, 34, 35])
```

```
a_1 * 10
```

```
array([10, 20, 40, 50])
```

```
a_1 @ a_2 # multiplies them like matrices 25 = 1*0 + 2*1 + 4*2 + 5*3
(dot product of the two vectors)
```

```
np.int64(25)
```

```
box = np.array([[1,2,3],[4,5,6]])
print(box)
print('transpose')
print(box.T)
```

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

For other functions check here

some cool flashy stuff which i found useful are given below

```
# 2x + 3y = 10
# 5x + 7y = 20
# to be solved
A = np.array([[2, 3],[5, 7]])
B = np.array([10, 20])
x, y = np.linalg.solve(A, B)
print('x = ', round(x))
print('y = ', round(y))
```

```
x =  -10
y =  10
```