```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import numpy as np
>>> np.zeros((2,3))
array([[0., 0., 0.],
       [0., 0., 0.]])
>>> # Array of zeros of size 2,3
>>> np.ones((2,4))
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.]])
>>> # Same thing with one
>>> np.ones((2,4)) * 10
array([[10., 10., 10., 10.],
       [10., 10., 10., 10.]])
>>> # Filling with 10
>>> np.eye(4) # Identity matrix
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
>>> np.random.random() #random numers
0.2807962700586123
>>> np.random.random((2,3)) #random numers
array([[0.2103353 , 0.15051973, 0.36443085],
       [0.57512776, 0.25316415, 0.40860195]])
>>> np.random.randn(2,3)
array([[ 0.04860901,  1.58491027, -0.95645219],
       [ 0.07867425,  0.05462022, -0.21664653]])
>>> # We get a matrix with normal numbers obtained with a gaussian distribution
>>> R = np.random.randn(10000)
>>> R.mean
<built-in method mean of numpy.ndarray object at 0x0000022AFC8F3800>
>>> np.mean(R)
-0.010457345240715375
>>> R.mean() # Another way to get the mean of an array of values
-0.010457345240715375
>>> R.var() # Tells us the variance
1.0043456598282816
>>> R.std() # square root of the variance
1.0021704744345055
>>> R = np.random.randn(10000,5)
>>> R
array([[ 0.86705099, -0.18286406,  1.02432197, -1.58843635,  0.93077412],
       [ 0.96344453,  0.66756744,  0.20195855,  0.11103069,  0.22457488],
       [-1.9116706 ,  0.70335346, -1.63798217, -1.19292066,  0.42447724],
       ...,
       [ 0.63160694,  0.27342206,  0.93186911,  0.23309635,  0.4720166 ],
       [ 1.18635746,  1.16782448,  1.35163805,  0.45655696,  1.48101392],
       [ 0.78488121,  1.86024648,  2.55521617, -0.25832961, -0.09109353]])
>>> R.mean() # Calculates the mean of all elements
0.003967339318012176
>>> R.mean(axis = 0) # Calculates the mean of the first line
array([-0.01390319,  0.00935167, -0.00168198,  0.00878638,  0.01728382])
```

```
>>> # Each row ****
>>> # Each COLUMN ^*****
>>> R.mean(axis = 1) # Calculates the mean of each row
array([ 0.21016933,  0.43371522, -0.72294855, ...,  0.50840221,
        1.12867818,  0.97018415])
>>> R.mean(axis = 0).shape
(5,)
>>>  R.mean(axis = 1).shape

SyntaxError: unexpected indent
>>> R.mean(axis = 1).shape
(10000,)
>>> np.cov(R) # Covariance
array([[ 1.25076673,  0.12426178, -0.10356901, ...,  0.24674993,
         0.41670047,  0.56594044],
       [ 0.12426178,  0.13421904, -0.04310304, ...,  0.00215497,
         0.03280798,  0.09025482],
       [-0.10356901, -0.04310304,  1.45553594, ..., -0.19536475,
         0.12585285, -0.1502391 ],
       ...,
       [ 0.24674993,  0.00215497, -0.19536475, ...,  0.08170913,
         0.06613571,  0.20401592],
       [ 0.41670047,  0.03280798,  0.12585285, ...,  0.06613571,
         0.15761446,  0.20733429],
       [ 0.56594044,  0.09025482, -0.1502391 , ...,  0.20401592,
         0.20733429,  1.49360776]])
>>> np.cov(R).shape
(10000, 10000)
>>> # Numpy interprets each column as a vector observation
>>> np.cov(R, rowvar = False) # We'd have to do it that way
array([[ 9.91839350e-01, -3.79763535e-03,  2.42975887e-03,
        -1.12495794e-04,  1.03851887e-02],
       [-3.79763535e-03,  9.86312118e-01,  8.88441430e-03,
        -1.09105142e-02,  6.92004378e-03],
       [ 2.42975887e-03,  8.88441430e-03,  9.92610478e-01,
         1.05627648e-02, -4.73313735e-03],
       [-1.12495794e-04, -1.09105142e-02,  1.05627648e-02,
         9.96510224e-01, -2.56389276e-03],
       [ 1.03851887e-02,  6.92004378e-03, -4.73313735e-03,
        -2.56389276e-03,  1.00025124e+00]])
>>> # Random Integers #
>>> np.random.randint(0,10,size=(3,3))
array([[8, 7, 4],
       [3, 2, 6],
       [4, 9, 6]])
>>> np.random.choice(10, size = (3,3))
array([[2, 0, 3],
       [8, 1, 9],
       [8, 9, 6]])
>>>
```