```
import numpy as np # importing numpy module.
```

```
np.__version__ #numpy version
```

```
'2.0.2'
```

```
list_new = [1,2,3,4,5] #declaring a list and assigning to numpy
numpy_array_new = np.array(list_new) # declaring as array in numpy

print(array_new)
```

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

```
#creating 2D array
array_2D = np.array([[1,2,3],[4,5,6]])
print(array_2D)
```

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

```
# creating 3D array

array_3D = np.array([[[1,2,3],[4,5,6],[7,8,9]]])

print(array_3D)
```

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

```
# dtype

array_dtype_int = np.array([1,2,3], dtype=int)
print(array_dtype_int)

array_dtype_float = np.array([1,2,3], dtype=float)
print(array_dtype_float)

array_dtype_complex = np.array([1,2,3], dtype=complex)
print(array_dtype_complex)

array_dtype_bool = np.array([1,2,4,5,0], dtype=bool)
print(array_dtype_bool)
```

```
[1 2 3]
[1. 2. 3.]
[1.+0.j 2.+0.j 3.+0.j]
[ True  True  True  True False]
```

Start coding or generate with AI.

```
# example for arange in numpy
numpy_arange = np.arange(7)
numpy_array_new
```

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

```
numpy_arange_2 = np.arange(7,2)
numpy_arange_2
```

```
array([], dtype=int64)
```

```
np.arange(1,25,2)
```

```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23])
```

```
np.arange(1,11).reshape(5,2) # prints a 2D matrices
```

```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

```
np.zeros((4,5)) # printing a 4X5 matrices with float zeros
```

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

```
np.zeros([4,5], dtype='int32')
```

```
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0]], dtype=int32)
```

```
# np.ones and np.zeros
np.ones((5,5), dtype=float) # prints ones
```

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

```
np.ones((5,5), dtype=int) # prints ones as int
```

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

```
np.ones((5,5), dtype=bool) # prints ones as bool
```

```
array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

```
np.arange(1,26).reshape(5,5) # reshape (should be a correct multiples for forming matrix)
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
# Another Type --> random()
```

```
np.random.random((4,3))
```

```
array([[0.49779078, 0.48233881, 0.8649055 ],
       [0.24197386, 0.4710522 , 0.42753976],
       [0.0461131 , 0.8895131 , 0.41924359],
       [0.31095083, 0.14695276, 0.8225554 ]])
```

```
np.ndarray
```

```
numpy.ndarray
```

```
b = np.random.randint(10,20,(5,4))
b
```

```
array([[17, 14, 13, 19],
       [16, 13, 12, 12],
       [18, 19, 11, 13],
       [13, 15, 11, 10],
       [19, 15, 11, 16]])
```

```
b[1:3]
```

```
array([[16, 13, 12, 12],
       [18, 19, 11, 13]])
```

```
b[1, 0]
```

```
np.int64(16)
```

```
b[0:-2]
```

```
array([[17, 14, 13, 19],
       [16, 13, 12, 12],
       [18, 19, 11, 13]])
```

```
# trying with 10 X 10 matrix

arr2 = np.random.randint(0,100,(10,10))
arr2
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [78, 94, 21, 33, 81, 24,  6, 10,  6, 56],
       [16, 32,  8, 62, 79, 70, 24, 26,  3, 58],
       [39, 68,  7, 84, 49,  5, 41, 46, 89, 66],
       [40, 33, 66,  2, 36, 19, 35, 74,  6, 79],
       [89, 64, 59, 86, 13, 72,  3, 49,  5, 49],
       [25, 17, 55, 50,  9, 31, 48, 59, 61, 22],
       [81,  8, 95, 77, 79, 23, 98, 16, 70, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

```
print(arr2[::-1])
```

```
[[96 15 46 53  9 39 99 10 74 26]
 [94 32  5 34 58 88 22  1 50 77]
 [81  8 95 77 79 23 98 16 70 79]
 [25 17 55 50  9 31 48 59 61 22]
 [89 64 59 86 13 72  3 49  5 49]
 [40 33 66  2 36 19 35 74  6 79]
 [39 68  7 84 49  5 41 46 89 66]
 [16 32  8 62 79 70 24 26  3 58]
 [78 94 21 33 81 24  6 10  6 56]
 [20 86 61  9 77 63 11 20  0 33]]
```

```
print(arr2[::-2])
```

```
[[96 15 46 53  9 39 99 10 74 26]
 [81  8 95 77 79 23 98 16 70 79]
 [89 64 59 86 13 72  3 49  5 49]
 [39 68  7 84 49  5 41 46 89 66]
 [78 94 21 33 81 24  6 10  6 56]]
```

```
print(arr2[::-3])
```

```
[[96 15 46 53  9 39 99 10 74 26]
 [25 17 55 50  9 31 48 59 61 22]
 [39 68  7 84 49  5 41 46 89 66]
 [20 86 61  9 77 63 11 20  0 33]]
```

```
#arr2[0:10:3]
# This slices rows from index 0 to 9, taking 3rd row
arr2
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [78, 94, 21, 33, 81, 24,  6, 10,  6, 56],
       [16, 32,  8, 62, 79, 70, 24, 26,  3, 58],
       [39, 68,  7, 84, 49,  5, 41, 46, 89, 66],
       [40, 33, 66,  2, 36, 19, 35, 74,  6, 79],
       [89, 64, 59, 86, 13, 72,  3, 49,  5, 49],
       [25, 17, 55, 50,  9, 31, 48, 59, 61, 22],
       [81,  8, 95, 77, 79, 23, 98, 16, 70, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

```
arr2[0:10:3]
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [39, 68,  7, 84, 49,  5, 41, 46, 89, 66],
       [25, 17, 55, 50,  9, 31, 48, 59, 61, 22],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

```
arr2[0:10:4]
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [40, 33, 66,  2, 36, 19, 35, 74,  6, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77]])
```

```
arr2
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [78, 94, 21, 33, 81, 24,  6, 10,  6, 56],
       [16, 32,  8, 62, 79, 70, 24, 26,  3, 58],
       [39, 68,  7, 84, 49,  5, 41, 46, 89, 66],
       [40, 33, 66,  2, 36, 19, 35, 74,  6, 79],
       [89, 64, 59, 86, 13, 72,  3, 49,  5, 49],
       [25, 17, 55, 50,  9, 31, 48, 59, 61, 22],
       [81,  8, 95, 77, 79, 23, 98, 16, 70, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

Numpy Array Functions

```
print(arr2.max())
```

```
99
```

```
print(arr2.min())
```

```
0
```

```
print(arr2.mean())
```

```
44.71
```

```
print(arr2.mode())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-85-e65b1ba001bc> in <cell line: 0>()
----> 1 print(arr2.mode())

AttributeError: 'numpy.ndarray' object has no attribute 'mode'
```

Next steps: Explain error

```
from numpy import *
a = median(arr2)
a
```

```
np.float64(43.5)
```

```
arr2
```

```
array([[20, 86, 61,  9, 77, 63, 11, 20,  0, 33],
       [78, 94, 21, 33, 81, 24,  6, 10,  6, 56],
       [16, 32,  8, 62, 79, 70, 24, 26,  3, 58],
       [39, 68,  7, 84, 49,  5, 41, 46, 89, 66],
       [40, 33, 66,  2, 36, 19, 35, 74,  6, 79],
       [89, 64, 59, 86, 13, 72,  3, 49,  5, 49],
       [25, 17, 55, 50,  9, 31, 48, 59, 61, 22],
       [81,  8, 95, 77, 79, 23, 98, 16, 70, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

```
row =5
col = 6

print(arr2[row,col])
```

```
3
```

```
print(arr2[6,5])
```

```
31
```

```
print(arr2[5:6, 2:3])
```

```
[[59]]
```

```
arr2[7:10]
```

```
array([[81,  8, 95, 77, 79, 23, 98, 16, 70, 79],
       [94, 32,  5, 34, 58, 88, 22,  1, 50, 77],
       [96, 15, 46, 53,  9, 39, 99, 10, 74, 26]])
```

```
arr2[:,col] # prints the column as blow is the result of column
```

```
array([11,  6, 24, 41, 35,  3, 48, 98, 22, 99])
```

```
print(row)
print(col)
```

```
5
6
```

```
print(arr2[row,:])
```

```
[89 64 59 86 13 72  3 49  5 49]
```

```
print(arr2[:,-1])
```

```
[33 56 58 66 79 49 22 79 77 26]
```

```
print(arr2[:])
```

```
[[20 86 61  9 77 63 11 20  0 33]
 [78 94 21 33 81 24  6 10  6 56]
 [16 32  8 62 79 70 24 26  3 58]
 [39 68  7 84 49  5 41 46 89 66]
 [40 33 66  2 36 19 35 74  6 79]
 [89 64 59 86 13 72  3 49  5 49]
 [25 17 55 50  9 31 48 59 61 22]
 [81  8 95 77 79 23 98 16 70 79]
 [94 32  5 34 58 88 22  1 50 77]
 [96 15 46 53  9 39 99 10 74 26]]
```

Masking:

Masking means applying a condition to a numpy arry to filter/select elements based on True/False values.

```
arr2 > 50 # This will just set the values greater than 50 to True and 50 and less than to False
```

```
array([[False,  True,  True, False,  True,  True, False, False, False,
        False],
       [ True,  True, False, False,  True, False, False, False, False,
         True],
       [False, False, False,  True,  True,  True, False, False, False,
         True],
       [False,  True, False,  True, False, False, False, False,  True,
         True],
       [False, False,  True, False, False, False, False,  True, False,
         True],
       [ True,  True,  True,  True, False,  True, False, False, False,
        False],
       [False, False,  True, False, False, False, False,  True,  True,
        False],
       [ True, False,  True,  True,  True, False,  True, False,  True,
         True],
       [ True, False, False, False,  True,  True, False, False, False,
         True],
       [ True, False, False,  True, False, False,  True, False,  True,
        False]])
```

```
print(arr2[arr2 > 50]) # this would give the values greater than 50
```

```
[86 61 77 63 78 94 81 56 62 79 70 58 68 84 89 66 66 74 79 89 64 59 86 72
 55 59 61 81 95 77 79 98 70 79 94 58 88 77 96 53 99 74]
```

Start coding or generate with AI.