In [1]:
```
import sys
sys.version
```

Out[1]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
9 64 bit (AMD64)]'
```

In [4]:
```
import numpy as np
```

In [6]:
```
import importlib.metadata as metadata

np_version = metadata.version("numpy")

print("My numpy version is: ", np_version)
```

```
My numpy version is:  1.26.4
```

In [5]:
```
my_list = [0,1,2,3,4,5]
my_list
```

Out[5]:
```
[0, 1, 2, 3, 4, 5]
```

In [6]:
```
type(my_list)
```

Out[6]:
```
list
```

In [7]:
```
arr = np.array(my_list)    #changing list to array
arr
```

Out[7]:
```
array([0, 1, 2, 3, 4, 5])
```

In [7]:
```
np.arange(10)       #arange prints the values of array with 0 index to n-1 index
```

Out[7]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [8]:
```
np.arange(10,20)      #arange prints the values from starting index to n-1 index
```

Out[8]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [10]:
```
n= np.arange(10,30,3)       #step 3 values prints
n
```

Out[10]:
```
array([10, 13, 16, 19, 22, 25, 28])
```

In [11]:
```
np.arange(20,8)        # in range first arg should be grater than 2nd arg
                       #it gives o/p but with out any values
```

Out[11]:
```
array([], dtype=int32)
```

In [12]:
```
np.range(10,20,30,4)     # we can not pass 4 arg
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[12], line 1
----> 1 np.range(10,20,30,4)

File ~\anaconda3\Lib\site-packages\numpy\__init__.py:333, in __getattr__(attr)
    330     "Removed in NumPy 1.25.0"
    331     raise RuntimeError("Tester was removed in NumPy 1.25.")
--> 333 raise AttributeError("module {!r} has no attribute "
    334                      "{!r}".format(__name__, attr))

AttributeError: module 'numpy' has no attribute 'range'
```

In [13]:
```
n1= np.arange(-10,9)
```

In [14]:
```
n1
```

Out[14]:
```
array([-10,  -9,  -8,  -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,
         3,   4,   5,   6,   7,   8])
```

Array is collection of similar data types

3 types of Arrays are there

1 dimensional array,2 dimensional array and n dimensional array(nd)

1. ! Dimensional Array

In [11]:
```
np.zeros(3)     # in numpy array prints values in float
```

Out[11]:
```
array([0., 0., 0.])
```

In [12]:
```
np.zeros(2)
```

Out[12]:
```
array([0., 0.])
```

In [13]:
```
np.zeros(8)
```

Out[13]:
```
array([0., 0., 0., 0., 0., 0., 0., 0.])
```

In [14]:
```
np.zeros(8,dtype= int)      # datatype changing
```

Out[14]:
```
array([0, 0, 0, 0, 0, 0, 0, 0])
```

In [15]:
```
n_int = np.zeros(9,dtype = int)
n_int
```

Out[15]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [16]:
```
n1 = np.zeros(4,dtype = 'str')
n1
```

Out[16]:
```
array(['', '', '', ''], dtype='<U1')
```

In [17]:
```
n1 = np.zeros(4,dtype=int)
n1
```

Out[17]:
```
array([0, 0, 0, 0])
```

In [20]:
```
n1 = np.ones(1)
n1
```

Out[20]:
```
array([1.])
```

In [21]:
```
np.ones(3)
```

Out[21]:
```
array([1., 1., 1.])
```

In [22]:
```
np.ones(3,dtype = int)
```

Out[22]:
```
array([1, 1, 1])
```

2 Dimensional Array should ends with 2 open braces & 2 closed braces

in tis 2d array we can print matrix form

and size of the array we denotes as 2*2 or 2/2 matrix which are rows and coloumns of the array

In [25]:
```
n2 = np.zeros((3,3))
n2
```

Out[25]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [26]:
```
n2 = np.zeros((2,2))
n2
```

Out[26]:
```
array([[0., 0.],
       [0., 0.]])
```

In [27]:
```
np.zeros((3,3),dtype=int)
```

Out[27]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [28]:
```
n2 = np.zeros((2,2),dtype= int)
n2
```

Out[28]:
```
array([[0, 0],
       [0, 0]])
```

In [29]:
```
n2 = np.ones((3,2),dtype = int)      #it is 3/2 matrix form
n2
```

Out[29]:
```
array([[1, 1],
       [1, 1],
       [1, 1]])
```

In [31]:
```
n3 = np.ones((2,5))
n3
```

Out[31]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [32]:
```
n3 = np.ones((2,5), dtype = int)
n3
```

Out[32]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [ ]:
```
# here we can give n dimensions then it will be ndimesional array
```

In [34]:
```
n4 = np.ones((10,10), dtype =int)
n4
```

Out[34]:
```
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, 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],
       [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, 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]])
```

In [36]:
```
n4 = np.ones((20,20), dtype = int)
n4
```

```
Out[36]: 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, 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, 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, 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],
                [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, 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, 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, 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],
                [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, 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, 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, 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],
                [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, 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, 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, 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]])
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```