In [1]:
```
import numpy as np
```

In [2]:
```
x= np.array(5)#scalar
x
```

Out[2]:
```
array(5)
```

In [4]:
```
type(x)
```

Out[4]:
```
numpy.ndarray
```

In [5]:
```
x.ndim
```

Out[5]:
```
0
```

In [6]:
```
x1=np.array([5])#vector
```

In [7]:
```
x1.ndim
```

Out[7]:
```
1
```

In [8]:
```
x1=np.array([5,6,7,8,9,10])
```

In [9]:
```
x1.ndim
```

Out[9]:
```
1
```

In [12]:
```
x1=np.array([[5,6,7,8],[2,3,4,5]],dtype=int,ndmin=5)
print(x1)
print(x1.ndim
print(x1.shape)#doubt
```

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

In [9]:
```
x1=np.array([[[5,6,7],[2,3,4],[5,6,7],[3,4,5]]])
print(x1)
print(x1.ndim)
```

```
[[[5 6 7]
  [2 3 4]
  [5 6 7]
  [3 4 5]]]
3
```

In [11]:
```
x1=np.array([[[[[5,6,7],[2,3,4],[5,6,7],[3,4,5]]]]])
print(x1)
print(x1.ndim)
```

```
[[[[[5 6 7]
    [2 3 4]
    [5 6 7]
    [3 4 5]]]]]
5
```

In [12]:
```
x1=np.array([[5,6,7],[8,9,10],[1,2,3]])
```

In [13]:
```
x1
```

Out[13]:
```
array([[ 5,  6,  7],
       [ 8,  9, 10],
       [ 1,  2,  3]])
```

In [197]:
```
x1.ndim
```

Out[197]:
```
2
```

In [198]:
```
x1=np.array([[[5,6,7],[8,9,10]]])
```

In [199]:
```
x1
```

Out[199]:
```
array([[[ 5,  6,  7],
        [ 8,  9, 10]]])
```

In [200]:
```
x1.ndim
```

Out[200]:
```
3
```

In [201]:
```
#
```

In [202]:
```
x1=np.array([5,6,7,8,9,10],dtype=str)
```

In [203]:
```
x1
```

Out[203]:
```
array(['5', '6', '7', '8', '9', '10'], dtype='<U2')
```

In [204]:
```
x1=np.array([5,6,7,8,9,10],dtype=complex)
```

In [205]:
```
x1
```

Out[205]:
```
array([ 5.+0.j,  6.+0.j,  7.+0.j,  8.+0.j,  9.+0.j, 10.+0.j])
```

In [206]:
```
x1=np.array([5,6,7,8,9,10],dtype=float)
```

In [207]:
```
x1
```

Out[207]:
```
array([ 5.,  6.,  7.,  8.,  9., 10.])
```

In [208]:
```
x1=np.array([5,6,7,8,9,10],dtype=np.int8)
```

In [209]:
```
x1
```

Out[209]:
```
array([ 5,  6,  7,  8,  9, 10], dtype=int8)
```

In [210]:
```
x1=np.array([127,6,7,8,9,10],dtype=np.int8)
```

In [211]:
```
x1
```

Out[211]:
```
array([127,   6,   7,   8,   9,  10], dtype=int8)
```

In [212]:
```
x1=np.array([128,6,7,8,9,10],dtype=np.int8)
```

In [213]:
```
x1
```

Out[213]:
```
array([-128,    6,    7,    8,    9,   10], dtype=int8)
```

In [214]:
```
x1=np.array([129,6,7,8,9,10],dtype=np.int8)
```

In [215]:
```
x1
```

Out[215]:
```
array([-127,    6,    7,    8,    9,   10], dtype=int8)
```

In [216]:
```
x1=np.array([129,6,7,8,9,10],dtype=np.bool_)
```

In [217]:
```
x1
```

Out[217]:
```
array([ True,  True,  True,  True,  True,  True])
```

In [69]:
```
x1=np.array([129,6,7,8,9,10],dtype=int,ndmin=31)
```

In [219]:
```
x1
```

Out[219]:
```
array([[[[[[[[[[[[[[[[[[[[[[[[[[[[[[129,
                                    6,
                                    7,
                                    8,
                                    9,
                                   10]]]]]]]]]]]]]]]]]]]]]]]]]]]]])
```

In [220]:
```
x1=np.array([129,6,7,8,9,10],dtype=int,ndmin=33)
x1
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_3040\3748579729.py in <module>
----> 1 x1=np.array([129,6,7,8,9,10],dtype=int,ndmin=33)
      2 x1

ValueError: ndmin bigger than allowable number of dimensions NPY_MAXDIMS (=32)
```

In [144]:
```
np.array([1,6,0,8,9])
```

Out[144]:
```
array([1, 6, 0, 8, 9])
```

In [145]:
```
np.array([1,6,0,8.0,9])
```

Out[145]:
```
array([1., 6., 0., 8., 9.])
```

In [146]:
```
np.array([1,6,0,8.0,9.5+0j])
```

Out[146]:
```
array([1. +0.j, 6. +0.j, 0. +0.j, 8. +0.j, 9.5+0.j])
```

In [147]:
```
np.array([1,"6",0,8.0,9.5+0j])
```

Out[147]:
```
array(['1', '6', '0', '8.0', '(9.5+0j)'], dtype='<U64')
```

In [148]:
```
np.array([[1],"6",0,8.0,9.5+0j])
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3040\1975447580.py:1: VisibleDeprecationWarning: Creating an ndarra
y from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  np.array([[1],"6",0,8.0,9.5+0j])
```

Out[148]:
```
array([list([1]), '6', 0, 8.0, (9.5+0j)], dtype=object)
```

In [149]:
```
np.array([(1),"6",0,8.0,9.5+0j])
```

Out[149]:
```
array(['1', '6', '0', '8.0', '(9.5+0j)'], dtype='<U64')
```

In [150]:
```
np.array([1,"6",0,8.0,9.5+0j])
```

Out[150]:
```
array(['1', '6', '0', '8.0', '(9.5+0j)'], dtype='<U64')
```

In [151]:
```
#shape
```

In [152]:
```
e=np.array(5)
```

In [153]:
```
e.shape
```

Out[153]:
```
()
```

In [154]:
```
e=np.array([5,6,7,8])
```

In [155]:
```
e.shape
```

Out[155]: (4,)

In [156]:
```
e=np.array([[5,6,7,8],[6,7,8,9]])
```

In [157]:
```
e.shape
```

Out[157]: (2, 4)

In [158]:
```
e=np.array([[5,6,7,8],[6,7,8,9],[3,4,5,6]])
```

In [159]:
```
e.shape
```

Out[159]: (3, 4)

In [160]:
```
e=np.array([[5,6,7,8],[6,7,8,9],[3,4,5,6],[3,4,5,6]])
```

In [161]:
```
e.shape
```

Out[161]: (4, 4)

In [162]:
```
e=np.array([5,6,7,6,7,8,9])
```

In [163]:
```
e[0]
```

Out[163]: 5

In [164]:
```
e[-1]
```

Out[164]: 9

In [165]:
```
e[:3]
```

Out[165]: array([5, 6, 7])

In [166]:
```
e=np.array('i',[5,6])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_3040\1630738349.py in <module>
----> 1 e=np.array('i',[5,6])

TypeError: Field elements must be 2- or 3-tuples, got '5'
```

In [105]:
```
import array as np
e=np.array('i',[5,6])
```

In [45]:
```
e
```

Out[45]:
```
array('i', [5, 6])
```

In [46]:
```
e.insert(0,2)
e
```

Out[46]:
```
array('i', [2, 5, 6])
```

In [47]:
```
del e[0]
e
```

Out[47]:
```
array('i', [5, 6])
```

In [48]:
```
#practice
```

In [52]:
```
x1=np.array([3,4,5,6,7],dtype=str)
```

In [53]:
```
x1
```

Out[53]:
```
array(['3', '4', '5', '6', '7'], dtype='<U1')
```

In [59]:
```
x1=np.array([4,5,6,7,8,9,10],dtype=int,ndmin=23)
x1
```

Out[59]:
```
array([[[[[[[[[[[[[[[[[[[[[[[ 4,  5,  6,  7,  8,  9,
                              10]]]]]]]]]]]]]]]]]]]]]]])
```

In [75]:
```
x1=np.array([[4,5,6],[7,8,9],[2,4,10]],dtype=int,ndmin=3)
print(x1)
print(x1.ndim)
```

```
[[[ 4  5  6]
  [ 7  8  9]
  [ 2  4 10]]]
3
```

In [76]:
```
x1=np.array([5,6,7,8,9,10],dtype=np.int8)
```

In [77]:
```
x1
```

Out[77]:
```
array([ 5,  6,  7,  8,  9, 10], dtype=int8)
```

In [79]:
```
x1=np.array([3,4,5,6.0,6,7])
print(x1)
print
```

Out[79]:
```
array([3., 4., 5., 6., 6., 7.])
```

In [86]:
```
x1=np.array([[5,6,68],[3,4,5]])
print(x1)
print(x1.shape)
```

```
[[ 5  6 68]
 [ 3  4  5]]
(2, 3)
```

In [107]:
```
e=np.array("i",[1,2,4,6])
```

In [108]:
```
e.insert(1,9)
e
```

Out[108]:
```
array('i', [1, 9, 2, 4, 6])
```

In [ ]:
```
del e[0]
```