```
#Attributes
.shape
.size
.ndim
.dtype
```

```
import numpy as np
```

```
x = np.array([5,7,8,2,9,15])
print(x)
```

```
[ 5  7  8  2  9 15]
```

```
#shape
#no of rows and columns denchhaa shape le
print(x.shape)
```

```
(6,)
```

```
mat = np.array([[5, 4, 3],[ 9, 7, 6]])
print(mat)
```

```
[[5 4 3]
 [9 7 6]]
```

```
print(mat.shape)
```

```
(2, 3)
```

```
#size
#size le no of elements denchhaa 1d ma
#2d ma size le no of elements denchhaa
print(x.size)
```

```
6
```

```
print(mat.shape)
```

```
(2, 3)
```

```
print(x.size)
```

```
6
```

```
#ndim
#no of dimension denchhaa ndim le
print(mat.ndim)
```

```
2
```

```
#dtype
#elements ko type vandenaaa(data type vandencha)
print(x.dtype)
```

```
int64
```

```
# int 64 ko meaning suppose 5 vaye
# 5 lai binary ma convert gardaaa 101 hunchhaa ra yo 101 ta 3 bit ko vayo now yesko aagadi 62 ottaa 0 add garchhaa
```

```
#3 dimension ma
import numpy as np
matrix = np.array([
    [[2,4],[5,2]],
    [[7,5],[1,9]]
])
print(matrix)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 2
      1 #3 dimension ma
----> 2 matrix = np.array([[[2,4],[5,2],[[7,5],[1,9]]]])
      3 print(matrix)

ValueError: setting an array element with a sequence. The requested array has an inhomogeneous shape after 3 dimensions. The detected shape was (1, 3, 2) + inhomogeneous part.
```

#Slicing/Indexing

```
#indexing(starts from 0)
#particular element matraa chaiyo bhane
x = np.array([5,7,8,2,9,15])
print(x)
```

```
[ 5  7  8  2  9 15]
```

```
print(x[2])
```

```
8
```

```
#last ko element ko lagi -1 bata start
print(x[-1])
```

```
15
```

Start coding or generate with AI.

```
#slicing
#sabei elements chaoyenaa array ko kunnei part matraa chaiyo bahne slicing garna sakenchhaa

#syntax:x[start index: stop index]
x =  np.array([5,7,8,2,9,15])
print(x[0:3])
```

```
[5 7 8]
```

```
print(x[1:5])
```

```
[7 8 2 9]
```

```
print(x[0:])
# #stop index deyena bhane 7829
```

```
  Cell In[30], line 1
    print(x(1:))
            ^
SyntaxError: invalid syntax
```

```
#syntax : x[startindex : stopindex :step]# by default step is 1
```

Start coding or generate with AI.

```
x = np.array([5,7,8,2,9,15])
print(x[:5 :2])
```

```
[5 8 9]
```

```
x[::]
```

```
array([ 5,  7,  8,  2,  9, 15])
```

```
x[::-1]
```

```
array([15,  9,  2,  8,  7,  5])
```

```
x[1: -1]
```

```
array([7, 8, 2, 9])
```

```
x[-2:1: -1] #by defaukt +1 ho step science
```

```
array([9, 2, 8])
```

```
x[-2:1]
```

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

```
y =x[0:3]
print(y)
```

```
[5 7 8]
```

```
#Modification
print(x)
```

```
[ 5  7  8  2  9 15]
```

```
# yo amthi ko 2 lai 12 banauna paryo bhane
x[3] =12
print(x)
```

```
[ 5  7  8 12  9 15]
```

```
#copy
y =x[0:3]
print(y)
```

```
[5 7 8]
```

```
y[0] = 15
print(y)
```

```
[15  7  8]
```

```
print(x)
```

```
[15  7  8 12  9 15]
```

```
#copy view
#banako y copy hainaaa view ho, so inndepenet vayenaa
```

```
x =  np.array([5,7,8,2,9,15])
y =x[0: 3].copy()
print(y)
```

```
[5 7 8]
```

```
y[0] = 15
print(y)
```

```
[15  7  8]
```

```
print(x)
```

```
[ 5  7  8  2  9 15]
```

```
#copy ley chhutei memory location use garchhaa bhane view ley same memory location use garchhaa
```

Start coding or generate with AI.

```
# #Functions:
1.np.sum()
2.np.prod()

3. np.min()
4.np.max()

5.np.mean()
6.np.std()
 np.median()
 np.sort()
 np.unique()


x =  np.array([5,7,8,2,9,15])



sum = np.sum(x)
print(sum)
```

```
46
```

```
sum = np.sum(x)
print(f'sum={sum}')
```

```
sum=46
```

```
Start coding or generate with AI.


#x.shape #numpy ko array le call garne


prod = np.prod(x)
print(prod)
```

```
75600
```

```
min= np.min(x)
print( f'min={min}')
```

```
min=2
```

```
max= np.max(x)
print( f'max={max}')
```

```
max=15
```

```
mean= np.mean(x)
print( f'mean={mean}')
```

```
mean=7.666666666666667
```

```
#point pachhadi 6666666667 naaaikana .67 sammaa aaune garna ko lagi
mean= np.mean(x)
print( f'mean={mean:.2f}') #.2f ko maning point pachhdi ko 2 ottaa dene


```

```
mean=7.67
```

```
std= np.std(x)  #std=>standard deviation
print( f'std={std}')
```

```
std=3.986086914367133
```

```
#if sab value 5 vako vaye std = 0 kinaki value scatter vako chhaina


median= np.median(x)
print( f'median={median}')
```

```
median=7.5
```

```
sort= np.sort(x)  #by default ascending
print( f'sort={sort}')
```

```
sort=[ 2  5  7  8  9 15]
```

```
#for decending
sort= np.sort(x)[::-1]  #sort in ascending then reversse it
print( f'sort={sort}')
```

```
sort=[15  9  8  7  5  5  2]
```

```
#2 ottaaa 5 vako vayeni yeutaa matraa dentyo unique le gardaa
unique= np.unique(x)
print( f'unique={unique}')
```

```
unique=[ 2  5  7  8  9 15]
```

```
#for documentation
numpy documentation
```

```
help(np.sort)#sorting ko documentation
```

```
#Array arithmetic
np.add()
np.multiply()
np.divide()
np.floor_divide()
np.mod()
np.power()
np.sqrt()
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])
#corresponfing element ko sum
z = np.add(x,y)
print(z)
```

```
[7 5 9 5]
```

```
z= x + y
print(z)
```

```
[7 5 9 5]
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4,5])
#corresponfing element ko sum
z = np.add(x,y)
print(z)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[80], line 4
      2 y = np.array([2,1,3,4,5])
      3 #corresponfing element ko sum
----> 4 z = np.add(x,y)
      5 print(z)

ValueError: operands could not be broadcast together with shapes (4,) (5,)
```

```
#error aayo because k=array ko size same hunu paryo
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])

z = x/y
print(z)
```

```
[2.5  4.   2.   0.25]
```

```
#floor_divide ley point pachhadi ko dedena
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])

z = np.multiply(x,y)
print(z)
```

```
[10  4 18  4]
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])

z = np.mod(x,y)# mod ley remainder denchhaa
print(z)
```

```
[1 0 0 1]
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])
```

```
z = np.floor_divide(x,y) #point pachhadi ko dedemaa
print(z)
```

```
[2 4 2 0]
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])

z = np.power(x,y)
print(z)
```

```
[ 25   4 216   1]
```

```
x = np.array([5,4,6,1])
y = np.array([2,1,3,4])

z = np.sqrt(x) #sqrt ma(x,y) deda dedena individually ki ta x ko ki y ko sqrt garnu parchhaa.
print(z)
```

```
[2.23606798 2.         2.44948974 1.        ]
```

```
#cube root lagauna ko lagi .sqrt jasto specific function chhaina so k garne vandaa power ma 1/3 type ko hannee


#braodcasting
- np.zeros()
- np.ones()
- np.full()
- np.arange()
-np.linspsace()
-np.random.random()
-np.random.randint()
-np.random.uniform()

#random generation
```