In [1]:

```
!pip install numpy
```

```
Requirement already satisfied: numpy in /Users/mohit/opt/anaconda3/li
b/python3.8/site-packages (1.20.1)
```

In [5]:

```
import numpy as np
```

In [ ]:

```

```

In [6]:

```
marks = [10, 20, 30]
```

In [7]:

```
marks*2
```

Out[7]:

```
[10, 20, 30, 10, 20, 30]
```

In [9]:

```
marks+[2]
```

Out[9]:

```
[10, 20, 30, 2]
```

In [12]:

```
type(marks)
```

Out[12]:

```
list
```

In [11]:

```
arr = np.array(marks)
arr
```

Out[11]:

```
array([10, 20, 30])
```

In [13]:

```
type(arr)
```

Out[13]:

```
numpy.ndarray
```

In [14]:

```
arr*2
```

Out[14]:

```
array([20, 40, 60])
```

In [15]:

```
arr+2
```

Out[15]:

```
array([12, 22, 32])
```

In [ ]:

```

```

In [18]:

```
np.arange(5)
```

Out[18]:

```
array([0, 1, 2, 3, 4])
```

In [23]:

```
np.arange(10, 20, 2, dtype='float')
```

Out[23]:

```
array([10., 12., 14., 16., 18.])
```

In [19]:

```
np.arange(10, 20, 1.5)
```

Out[19]:

```
array([10. , 11.5, 13. , 14.5, 16. , 17.5, 19. ])
```

In [ ]:

```

```

In [26]:

```
np.linspace(1, 10, 5)
```

Out[26]:

```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

In [ ]:

```

```

In [27]:

```
type(arr)
```

Out[27]:

```
numpy.ndarray
```

In [ ]:

```

```

In [28]:

```
lst = list(range(0, 10))
lst
```

Out[28]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [33]:

```
%timeit [i**2 for i in lst]
```

```
2.24 µs ± 43 ns per loop (mean ± std. dev. of 7 runs, 100000 loops eac
h)
```

In [30]:

```
arr = np.array(lst)
arr
```

Out[30]:

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

In [34]:

```
%timeit arr**2
```

```
449 ns ± 7.09 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops ea
ch)
```

In [ ]:

```

```

In [37]:

```
arr = np.arange(10, 25)
arr
```

Out[37]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24])
```

In [38]:

```
arr[0]
```

Out[38]:

```
10
```

In [39]:

```
arr[-1]
```

Out[39]:

```
24
```

In [41]:

```
arr[3:8]
```

Out[41]:

```
array([13, 14, 15, 16, 17])
```

In [42]:

```
np.array([10,20,30,40.0])
```

Out[42]:

```
array([10., 20., 30., 40.])
```

In [ ]:

```

```

In [44]:

```
arr[1] = 110
```

In [45]:

```
print(arr)
```

```
[ 10 110  12  13  14  15  16  17  18  19  20  21  22  23  24]
```

In [47]:

```
arr
```

Out[47]:

```
array([ 10, 110,  12,  13,  14,  15,  16,  17,  18,  19,  20,  21,  2
2,
        23,  24])
```

In [ ]:

```

```

In [48]:

```
lst2d = [ [4,5,6], [7,8,9] ]
lst2d
```

Out[48]:

```
[[4, 5, 6], [7, 8, 9]]
```

In [50]:

```
arr2d = np.array(lst2d)
arr2d
```

Out[50]:

```
array([[4, 5, 6],
       [7, 8, 9]])
```

In [51]:

```
arr2d*2
```

Out[51]:

```
array([[ 8, 10, 12],
       [14, 16, 18]])
```

In [52]:

```
arr2d + 2
```

Out[52]:

```
array([[ 6,  7,  8],
       [ 9, 10, 11]])
```

In [53]:

```
arr2d
```

Out[53]:

```
array([[4, 5, 6],
       [7, 8, 9]])
```

In [55]:

```
len(arr2d)
```

Out[55]:

```
2
```

In [56]:

```
arr2d.size
```

Out[56]:

```
6
```

In [59]:

```
arr2d.shape
```

Out[59]:

```
(2, 3)
```

In [60]:

```
arr2d.ndim
```

Out[60]:

```
2
```

In [62]:

```
arr2d[1][1]
```

Out[62]:

```
8
```

In [65]:

```
arr2d[1, 1]
```

Out[65]:

```
8
```

In [66]:

```
arr
```

Out[66]:

```
array([ 10, 110,  12,  13,  14,  15,  16,  17,  18,  19,  20,  21,  2
2,
        23,  24])
```

In [67]:

```
arr.ndim
```

Out[67]:

```
1
```

In [68]:

```
arr.shape
```

Out[68]:

```
(15,)
```

In [ ]:

```

```

In [69]:

```
np.array([1,2])
```

Out[69]:

```
array([1, 2])
```

In [81]:

```
temp = np.array([[1,2]])
temp
```

Out[81]:

```
array([[1, 2]])
```

In [82]:

```
temp.shape
```

Out[82]:

```
(1, 2)
```

Q- What will be output for the following code?

```
a = np.array([1, 2, 3,4,5], ndmin = 2)
print(a)
```

A : [[1, 2, 3, 4, 5]]

B : [1, 2, 3, 4, 5]

C : Error

D : Null

In [87]:

```
np.array([1, 2, 3,4,5], ndmin=2)
```

Out[87]:

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

Reshape

In [94]:

```
m1 = np.arange(10, 30, 1)
m1
```

Out[94]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26,
       27, 28, 29])
```

In [95]:

```
m1.shape
```

Out[95]:

```
(20,)
```

In [96]:

```
m1.ndim
```

Out[96]:

```
1
```

In [99]:

```
m1= m1.reshape((5,4))
m1
```

Out[99]:

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21],
       [22, 23, 24, 25],
       [26, 27, 28, 29]])
```

In [100]:

```
m1.shape
```

Out[100]:

```
(5, 4)
```

In [101]:

```
m1.reshape((4,5))
```

Out[101]:

```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29]])
```

In [102]:

```
m1.reshape((2, 10))
```

Out[102]:

```
array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])
```

In [103]:

```
m1.reshape((4,4))
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
 last)
<ipython-input-103-20867944c50b> in <module>
----> 1 m1.reshape((4,4))

ValueError: cannot reshape array of size 20 into shape (4,4)
```

In [106]:

```
m1
```

Out[106]:

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21],
       [22, 23, 24, 25],
       [26, 27, 28, 29]])
```

In [107]:

```
m1.reshape(2,10)
```

Out[107]:

```
array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])
```

In [ ]:

```

```

In [118]:

```
np.arange(16).reshape(-1, 1)
```

Out[118]:

```
array([[ 0],
       [ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12],
       [13],
       [14],
       [15]])
```

In [ ]:

```

```

In [109]:

```
# arr = np.linspace([12,12,13,14.0],18,3)
# arr
```

In [ ]:

```
# np.linspace([1,2],[10,20],10,axis=1)
```

In [ ]:

```

```

In [119]:

```
a = np.array([[1,2,3],[0,1,4]])
print(a.ndim)
print(a.shape)
```

```
2
(2, 3)
```

In [ ]:

```

```

Special Matrix

In [128]:

```
np.zeros(10)
```

Out[128]:

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

In [129]:

```
np.zeros((4,4))
```

Out[129]:

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

In [130]:

```
np.ones((3,3))
```

Out[130]:

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

In [131]:

```
np.ones((3,3))*5
```

Out[131]:

```
array([[5., 5., 5.],
       [5., 5., 5.],
       [5., 5., 5.]])
```

In [136]:

```
np.identity(4, )
```

Out[136]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [135]:

```
np.eye(4)
```

Out[135]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [ ]:

```

```

Slicing

In [138]:

```
arr = np.arange(10, 30).reshape(4,5)
arr
```

Out[138]:

```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29]])
```

In [140]:

```
arr[-1,-1]
```

Out[140]:

```
29
```

In [144]:

```
arr[  1: , 2:4 ]
```

Out[144]:

```
array([[17, 18],
       [22, 23],
       [27, 28]])
```

In [143]:

```
arr[  : , :  ]
```

Out[143]:

```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29]])
```

In [ ]:

```

```

In [150]:

```
arr[1:3 , :]
```

Out[150]:

```
array([[15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

In [ ]:

```

```

np.split()

np.hsplit()

np.vsplit()

In [151]:

```
# np.split()
```

In [152]:

```
x = np.arange(10,100,10)
x
```

Out[152]:

```
array([10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [153]:

```
np.split(x, 3)
```

Out[153]:

```
[array([10, 20, 30]), array([40, 50, 60]), array([70, 80, 90])]
```

In [154]:

```
np.split(x, [4,5,6])
```

Out[154]:

```
[array([10, 20, 30, 40]), array([50]), array([60]), array([70, 80, 9
0])]
```

In [ ]:

```

```

In [ ]:

```
np.hsplit()
```

In [157]:

```
arr
```

Out[157]:

```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29]])
```

In [ ]:

```

```

In [160]:

```
arr
```

Out[160]:

```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29]])
```

In [158]:

```
np.mean(arr)
```

Out[158]:

```
19.5
```

In [161]:

```
np.sum(arr)/20
```

Out[161]:

```
19.5
```

Operations

In [163]:

```
arr + 2
```

Out[163]:

```
array([[12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21],
       [22, 23, 24, 25, 26],
       [27, 28, 29, 30, 31]])
```

In [164]:

```
arr*2
```

Out[164]:

```
array([[20, 22, 24, 26, 28],
       [30, 32, 34, 36, 38],
       [40, 42, 44, 46, 48],
       [50, 52, 54, 56, 58]])
```

In [ ]:

```

```

What will be printed?

```
a = np.array([1,2,3,5,8])
b = np.array([0,3,4,2,1])
c = a + b
c = c*a
print (c[2])
```

A : 7

B : 12

C : 10

D : 21

In [167]:

```
a = np.array([1,2,3,5,8])
b = np.array([0,3,4,2,1])
c = a + b
c = c*a
c
```

Out[167]:

```
array([ 1, 10, 21, 35, 72])
```

In [169]:

```
arr1 = np.arange(9).reshape(3,3)
arr2= np.arange(10, 19).reshape(3,3)
```

In [170]:

```
arr1
```

Out[170]:

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

In [171]:

```
arr2
```

Out[171]:

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

In [173]:

```
arr1 + arr2
```

Out[173]:

```
array([[10, 12, 14],
       [16, 18, 20],
       [22, 24, 26]])
```

In [ ]:

```

```

Masking

In [182]:

```
m1 = np.arange(10,22).reshape(3,4)
m1
```

Out[182]:

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

In [183]:

```
mask = m1 < 16
mask
```

Out[183]:

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

In [185]:

```
np.sum(mask)
```

Out[185]:

```
6
```

In [186]:

```
m1
```

Out[186]:

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

In [187]:

```
mask
```

Out[187]:

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

In [190]:

```
m1[mask]
```

Out[190]:

```
array([10, 11, 12, 13, 14, 15])
```

In [191]:

```
m1
```

Out[191]:

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

In [192]:

```
m1[m1%2==0]
```

Out[192]:

```
array([10, 12, 14, 16, 18, 20])
```

In [ ]:

```

```

Vectorisation

In [193]:

```
import math
```

In [198]:

```
math.log(134)
```

Out[198]:

```
4.897839799950911
```

In [199]:

```
x = np.arange(1, 10)
x
```

Out[199]:

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

In [200]:

```
np.log(x)
```

Out[200]:

```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791,
       1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

In [201]:

```
math.log(x)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-201-907a0dd0c3e1> in <module>
----> 1 math.log(x)

TypeError: only size-1 arrays can be converted to Python scalars
```

In [ ]:

```

```

In [203]:

```
vec_log = np.vectorize(math.log)
```

In [204]:

```
vec_log(x)
```

Out[204]:

```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791,
       1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

In [ ]:

```

```

In [205]:

```
def custom_loss(a,b):
    return (a**2 + b**2)**0.5
```

In [207]:

```
vec_custom_loss = np.vectorize(custom_loss)
```

In [213]:

```
x = np.arange(10)
y = np.arange(10, 20)
```

In [214]:

```
vec_custom_loss(x, y)
```

Out[214]:

```
array([10.        , 11.04536102, 12.16552506, 13.34166406, 14.5602197
8,
       15.8113883 , 17.08800749, 18.38477631, 19.6977156 , 21.0237960
4])
```

In [ ]:

```

```

Matrix Multiplication

In [215]:

```
A = np.arange(12).reshape(3,4)
A
```

Out[215]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [216]:

```
B = np.arange(12).reshape(3,4)
B
```

Out[216]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [217]:

```
A*B
```

Out[217]:

```
array([[  0,   1,   4,   9],
       [ 16,  25,  36,  49],
       [ 64,  81, 100, 121]])
```

In [218]:

```
np.matmul(A, B)
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
 last)
<ipython-input-218-04c68bb92949> in <module>
----> 1 np.matmul(A, B)

ValueError: matmul: Input operand 1 has a mismatch in its core dimensi
on 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is differen
t from 4)
```

In [221]:

```
B = B.reshape(4,3)
```

In [222]:

```
np.matmul(A, B)
```

Out[222]:

```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

In [223]:

```
A@B
```

Out[223]:

```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

In [224]:

```
np.dot(A,B)
```

Out[224]:

```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

In [ ]:

```

```

In [225]:

```
A
```

Out[225]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [226]:

```
np.transpose(A)
```

Out[226]:

```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [227]:

```
A.T
```

Out[227]:

```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [230]:

```
A.reshape(-1)
```

Out[230]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [231]:

```
A.flatten()
```

Out[231]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [232]:

```
np.ravel(A)
```

Out[232]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [ ]:

```
```

In [ ]:

```
```

Merging of arrays

np.hstack()

np.vstack()

In [234]:

```
# from numpy import array
```

In [ ]:

```
np.hstack()
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```