```
# NumPy -

#numeric Python

marks = [10,20,20,40]
marks

[10, 20, 20, 40]

marks + 10

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_19988/2309573683.py in <module>
----> 1 marks + 10

TypeError: can only concatenate list (not "int") to list

[  m+10    for m in marks]

[20, 30, 30, 50]

# how to create a numpy array

import numpy

marks = [10,20,20,40]

array_1 = numpy.array(marks)
array_1

array([10, 20, 20, 40])

array_1 + 10

array([20, 30, 30, 50])

### properties of numpy arrays
# What is numpy array - numpy array is a data structure in python
# which is used to store and retrieve the data

#how to install numpy array

pip install numpy

Requirement already satisfied: numpy in c:\users\maanz\anaconda3\lib\site-packages (1.20.3)
Note: you may need to restart the kernel to use updated packages.

# import the package
import numpy as np
```

```
# creation of numpy array
a = np.array(marks)
a
```

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

```
# we can do elementwise operation in numpy array
a + 10
```

```
array([20, 30, 30, 50])
```

```
a * 2
```

```
array([20, 40, 40, 80])
```

```
stud_details = [  ['john', 'NY City', 'Undergrad'], ['Roy', 'NY City',
'Master']   ]
a2 = np.array(stud_details)
a2
```

```
array([['john', 'NY City', 'Undergrad'],
       ['Roy', 'NY City', 'Master']], dtype='<U9')
```

```
# how to find the dimension of the array?
a.ndim
```

```
1
```

```
a2.ndim
```

```
2
```

```
# checking the shape and size of the array
a.shape
```

```
(4,)
```

```
a2.shape
```

```
(2, 3)
```

```
# ndim - to find number of dimensions of array
# shape - gives number of rows and columns
# np.array() - to create array
```

```
a.size
```

```
4
```

```
a2.size
```

```
6
```

```python
# Numpy array stores same data type
a3 = np.array( [10,20, 'python'] )
a3
```

```
array(['10', '20', 'python'], dtype='<U11')
```

```python
# indexing and slicing on arrays
a[0]
```

```
10
```

```python
a[-1]
```

```
40
```

```python
a[1] = 200
```

```python
a
```

```
array([ 10, 200,  20,  40])
```

```python
a[ 0 : 3]
```

```
array([ 10, 200,  20])
```

```python
# indexing and slicing on 2d array -
a3 = np.array([  [10,2,3,5], [16,9,8,7] , [19,20,3,6] ])
a3
```

```
array([[10,  2,  3,  5],
       [16,  9,  8,  7],
       [19, 20,  3,  6]])
```

```python
a3[1, 0]
```

```
16
```

```python
a3[1, 3]
```

```
7
```

```python
a3[1,-1]
```

```
7
```

```python
a3[2,-4]
```

```
19
```

```python
a3
```

```
array([[10,  2,  3,  5],
       [16,  9,  8,  7],
       [19, 20,  3,  6]])
```

```
a3[0 : 2]
```

```
array([[10,  2,  3,  5],
       [16,  9,  8,  7]])
```

```
a3[0 : 2, 0 : 2]
```

```
array([[10,  2],
       [16,  9]])
```

```
a3[  :   ,  0 : 2]
```

```
array([[10,  2],
       [16,  9],
       [19, 20]])
```

```
a3
```

```
array([[10,  2,  3,  5],
       [16,  9,  8,  7],
       [19, 20,  3,  6]])
```

```
a3[ : : 2]
```

```
array([[10,  2,  3,  5],
       [19, 20,  3,  6]])
```

```
a3[ : : 2 , : : 2]
```

```
array([[10,  3],
       [19,  3]])
```

```
#######

# create a array from the below list and store it in arr1
prices = [
    [10,20,30],
    [5,8,9],
    [10,15,25],
    [16,8,7]]
# find the number of rows and columns in the arr1
# check the size of arr1
# find the dimension of arr1
# slice the first 2 rows only of arr1,
# slice last 2 columns of arr1
# slice first 2 rows and last 2 columns of arr1

import numpy as np
arr1 = np.array(prices)
arr1
```

```
array([[10, 20, 30],
       [ 5,  8,  9],
       [10, 15, 25],
       [16,  8,  7]])
```

```
arr1.shape
```

```
(4, 3)
```

```
arr1.size
```

```
12
```

```
arr1.ndim
```

```
2
```

```
# slice the first 2 rows only of arr1,
arr1[0 : 2]
```

```
array([[10, 20, 30],
       [ 5,  8,  9]])
```

```
# slice last 2 columns of arr1
arr1[ : , 1 :  ]
```

```
array([[20, 30],
       [ 8,  9],
       [15, 25],
       [ 8,  7]])
```

```
# slice first 2 rows and last 2 columns of arr1
arr1[ : 2 , 1 : ]
```

```
array([[20, 30],
       [ 8,  9]])
```

```
arr1[1,1]  = 80
arr1
```

```
array([[10, 20, 30],
       [ 5, 80,  9],
       [10, 15, 25],
       [16,  8,  7]])
```

```
arr1[ [0,-1] ]
```

```
array([[10, 20, 30],
       [16,  8,  7]])
```

```
arr1[ [ 2, 1]]
```

```
array([[10, 15, 25],
       [ 5, 80,  9]])
```

```
arr1[ : ,   [-1,0] ]
```

```
array([[30, 10],
       [ 9,  5],
       [25, 10],
       [ 7, 16]])
```

```
arr1
```

```
array([[10, 20, 30],
       [ 5, 80,  9],
       [10, 15, 25],
       [16,  8,  7]])
```

```
arr1[  [0,-1]    ,   [0,-1]    ]
```

```
array([10,  7])
```

```
arr1[ [3 , 0 ]  ,  [0, 2] ]
```

```
array([16, 30])
```

```
arr1[1 , 1]
```

```
80
```

```
arr1[2,2]
```

```
25
```

```
arr1[  [1 , 2] ,  [1 , 2 ] ]
```

```
array([80, 25])
```

```
arr1
```

```
array([[10, 20, 30],
       [ 5, 80,  9],
       [10, 15, 25],
       [16,  8,  7]])
```

```
[20,8]
```

```
arr1[ 0 , 1]
```

```
20
```

```
arr1[ 3 , 1]
```

```
8
```

```
arr1[ [0,3]  ,  [1,1]  ]
```

```
array([20,  8])
```

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

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

```
np.arange(10, 20, 2)
```

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

```
np.arange(10, 20, 0.5)
```

```
array([10. , 10.5, 11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. 
,
       15.5, 16. , 16.5, 17. , 17.5, 18. , 18.5, 19. , 19.5])
```

```
#range(10,20,0.5)
```

```
arr2 = np.arange(0,25)
arr2
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 
16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

```
arr2.ndim
```

```
1
```

```
arr2.size
```

```
25
```

```
arr2.reshape(1,25)
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 
15,
        16, 17, 18, 19, 20, 21, 22, 23, 24]])
```

```
arr2.reshape(25,1)
```

```
array([[ 0],
       [ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12],
       [13],
       [14],
       [15],
       [16],
       [17],
       [18],
       [19],
       [20],
       [21],
       [22],
       [23],
       [24]])
```

```
arr2.reshape(5,5)
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
arr2.reshape(5,4)
```

```
----------------------------------------------------------------------
-----
ValueError                               Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_19988/3946076692.py in <module>
----> 1 arr2.reshape(5,4)

ValueError: cannot reshape array of size 25 into shape (5,4)
```

```
arr3 = np.arange(0,20)
arr3
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19])
```

```
# 1,20
# 20, 1
# 5,4
# 4,5
# 2,10
# 10, 2
# np.ones, np.zeros, np.identity, np.eye, np.random.randint
np.ones(10)

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

np.zeros(5)

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

np.ones( (2,2) )

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

np.zeros( (2,2) )

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

np.identity( 4 )

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

np.eye( 4, k = 1 )

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

np.random.randint(10, 20, 6)

array([19, 10, 13, 12, 13, 13])

## array operations
b = np.random.randint(0, 50, 25)
arr4 = b.reshape(5,5)
arr4

array([[27,  6, 37, 47, 22],
       [47, 25, 47, 46, 25],
       [40, 45, 14, 28,  1],
       [23, 49, 18, 43, 18],
       [38, 49, 25,  3, 42]])

# filtering the data
arr4 > 20
```

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

```
arr4[  arr4 > 20  ]
```

```
array([27, 37, 47, 22, 47, 25, 47, 46, 25, 40, 45, 28, 23, 49, 43, 38,
49,
       25, 42])
```

```
# find the values <20

# find the values between 20 to 30

arr4 > 20
```

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

```
arr4 < 30
```

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

```
arr4[(arr4 > 20)  &  (arr4 < 30)]
```

```
array([27, 22, 25, 25, 28, 23, 25])
```

```
# find the values between 15 to 25

arr4[(arr4 > 15) & (arr4<25)]
```

```
array([22, 23, 18, 18])
```

```
# np.where
np.where( arr4>20  )
```

```
(array([0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4],
       dtype=int64),
 array([0, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 3, 0, 1, 3, 0, 1, 2, 4],
       dtype=int64))
```

```
# np.where
np.where( arr4>20 , 'pass', 'fail' )

array([['pass', 'fail', 'pass', 'pass', 'pass'],
       ['pass', 'pass', 'pass', 'pass', 'pass'],
       ['pass', 'pass', 'fail', 'pass', 'fail'],
       ['pass', 'pass', 'fail', 'pass', 'fail'],
       ['pass', 'pass', 'pass', 'fail', 'pass']], dtype='<U4')

# numpy universal functions -
arr4.sum()

765

arr4

array([[27,  6, 37, 47, 22],
       [47, 25, 47, 46, 25],
       [40, 45, 14, 28,  1],
       [23, 49, 18, 43, 18],
       [38, 49, 25,  3, 42]])

arr4.sum(axis = 0)

array([175, 174, 141, 167, 108])

arr4.sum(axis = 1)

array([139, 190, 128, 151, 157])

arr4.min()

1

arr4.min(axis = 1)

array([ 6, 25,  1, 18,  3])

arr4.min(axis = 0)

array([23,  6, 14,  3,  1])

arr4.max()

49

arr4.mean()

30.6

arr4.mean(axis = 0)

array([35. , 34.8, 28.2, 33.4, 21.6])
```

```
arr4.mean(axis = 1)

array([27.8, 38. , 25.6, 30.2, 31.4])

arr4.std()

14.802702455970667

arr4.var()

219.12

np.sqrt(219.12)

14.802702455970667

np.sqrt(arr4)

array([[5.19615242, 2.44948974, 6.08276253, 6.8556546 , 4.69041576],
       [6.8556546 , 5.        , 6.8556546 , 6.78232998, 5.        ],
       [6.32455532, 6.70820393, 3.74165739, 5.29150262, 1.        ],
       [4.79583152, 7.        , 4.24264069, 6.55743852, 4.24264069],
       [6.164414  , 7.        , 5.        , 1.73205081, 6.4807407 ]])

np.log(arr4)

array([[3.29583687, 1.79175947, 3.61091791, 3.8501476 , 3.09104245],
       [3.8501476 , 3.21887582, 3.8501476 , 3.8286414 , 3.21887582],
       [3.68887945, 3.80666249, 2.63905733, 3.33220451, 0.        ],
       [3.13549422, 3.8918203 , 2.89037176, 3.76120012, 2.89037176],
       [3.63758616, 3.8918203 , 3.21887582, 1.09861229, 3.73766962]])

# array arithmatic
a + 20

array([ 30, 220,  40,  60])

a

array([ 10, 200,  20,  40])

b = np.array([1,2,3,4])
b

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

a + b

array([ 11, 202,  23,  44])

b = np.array([1,2])
a+b
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_19988/1186421745.py in <module>
      1 b = np.array([1,2])
----> 2 a+b

ValueError: operands could not be broadcast together with shapes (4,) (2,)
```

```
ar1 = np.array([ [10,20,30], [1,2,3], [101,102,103] ])
ar2 = np.array( [ [5,6,7], [8,9,10], [22,23,24]   ])
```

```
ar1
```

```
array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])
```

```
ar2
```

```
array([[ 5,  6,  7],
       [ 8,  9, 10],
       [22, 23, 24]])
```

```
ar1 * ar2
```

```
array([[  50,  120,  210],
       [   8,   18,   30],
       [2222, 2346, 2472]])
```

```
ar1 = np.array([ [10,20,30], [1,2,3], [101,102,103] ])
ar2 = np.array( [ [5,6,7], [8,9,10]  ])
ar1
```

```
array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])
```

```
ar2
```

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

```
ar1 + ar2
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_19988/1793376629.py in <module>
```

```
----> 1 ar1 + ar2

ValueError: operands could not be broadcast together with shapes (3,3)
(2,3)



ar1 = np.array([ [10,20,30], [1,2,3], [101,102,103] ])
ar2 = np.array( [ [5,6,7], [8,9,10], [22,23,24]   ])

np.matmul(ar1,ar2)

array([[ 870,  930,  990],
       [  87,   93,   99],
       [3587, 3893, 4199]])

ar1

array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])

ar2

array([[ 5,  6,  7],
       [ 8,  9, 10],
       [22, 23, 24]])

ar1

array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])

np.concatenate( [ar1,ar2] )

array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103],
       [  5,   6,   7],
       [  8,   9,  10],
       [ 22,  23,  24]])



ar1 = np.array([ [10,20,30], [1,2,3], [101,102,103] ])
ar2 = np.array( [ [5,6], [8,9], [22,23]   ])
ar1

array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])
```

```
ar2

array([[ 5,  6],
       [ 8,  9],
       [22, 23]])

np.concatenate([ar1,ar2])

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_19988/189908310.py in <module>
----> 1 np.concatenate([ar1,ar2])

<__array_function__ internals> in concatenate(*args, **kwargs)

ValueError: all the input array dimensions for the concatenation axis
must match exactly, but along dimension 1, the array at index 0 has
size 3 and the array at index 1 has size 2

np.concatenate([ar1,ar2], axis = 1)

array([[ 10,  20,  30,   5,   6],
       [  1,   2,   3,   8,   9],
       [101, 102, 103,  22,  23]])

ar1 = np.array([ [10,20,30], [1,2,3], [101,102,103] ])
ar2 = np.array( [ [5,6], [8,9], [22,23], [10,9]   ])
ar1

array([[ 10,  20,  30],
       [  1,   2,   3],
       [101, 102, 103]])

ar2

array([[ 5,  6],
       [ 8,  9],
       [22, 23],
       [10,  9]])

np.concatenate( [ar1,ar2], axis = 1 )

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_19988/1657275679.py in <module>
----> 1 np.concatenate( [ar1,ar2], axis = 1 )

<__array_function__ internals> in concatenate(*args, **kwargs)
```

```
ValueError: all the input array dimensions for the concatenation axis
must match exactly, but along dimension 0, the array at index 0 has
size 3 and the array at index 1 has size 4
```

```python
for i in a:
    print(i)
```

```
10
200
20
40
```

```python
a
```

```
array([ 10, 200,  20,  40])
```

```python
a2
```

```
array([['john', 'NY City', 'Undergrad'],
       ['Roy', 'NY City', 'Master']], dtype='<U9')
```

```python
for i in a2:
    print(i)
```

```
['john' 'NY City' 'Undergrad']
['Roy' 'NY City' 'Master']
```

```python
for i in np.nditer(a2):
    print(i)
```

```
john
NY City
Undergrad
Roy
NY City
Master
```

```python
# np.array()
# arr.shape()
# arr.size
# arr.ndim
# arr[index]
# arr[slicing rows, slicing cols]
# arr[ arr>4 ]
# np.where(cond, value if true, value if false)
# np.ones(), zeros(), identity, eye, np.arange(), reshape
# np.random.randint(start, end, number of values)
# np.mean(), np.var(), np.std(), np.min, np.max, np.sum(), np.sqrt,
np.log,
# np.concatenate, np.nditer



# Write a program to print all numbers from 1 to 10 using a for loop.
# Use a while loop to print the numbers from 5 down to 1.
# Write a program to print the sum of all numbers from 1 to 100 using
a loop.
# Write a program to print all the elements of a list using a for
loop.
# Create a program that prints all odd numbers between 1 and 20 using
a for loop.
# Write a program that prints the Fibonacci sequence up to 50 using a
while loop.
# Write a program to print a multiplication table (1 to 10) for a
given number using a for loop.
# Write a program that prints the prime numbers from 1 to 50 using a
for loop.
# Write a program that finds all the common elements between two lists
using a loop.
# Implement a program to print all perfect numbers between 1 and 1000.
# Create a program that finds all the numbers divisible by both 3 and
5 from 1 to 100.
# Write a program that uses nested loops to print a multiplication
table up to a given number (like a 10x10 table).
```

```
a = 5
while a>0:
    print(a)
    a = a - 1
```

```
5
4
3
2
1
```

```
# Write a program to print the sum of all numbers from 1 to 100 using
a loop.
```

```
s = 0
for i in range(1,101):
    s = s+i  # s = 1 , s = 3
print(s)
```

```
5050
```

```
# Write a program to print all the elements of a list using a for
loop.
```

```
my_list = [10,9,8,'python']

for i in my_list:
    print(i)
```

```
10
9
8
python
```

```
# Create a program that prints all odd numbers between 1 and 20 using
a for loop.

for i in range(1,21):
    if i%2 != 0:
        print(i)
```

```
1
3
5
7
9
11
13
15
17
19
```

```
# Write a program that prints the Fibonacci sequence up to 50 using a
while loop.

fibonacci_s = [0, 1]

for i in range(50):
    s = sum(fibonacci_s[-2 : ])
    fibonacci_s.append(s)
print(fibonacci_s)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393,
196418, 317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887,
9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141,
267914296, 433494437, 701408733, 1134903170, 1836311903, 2971215073,
4807526976, 7778742049, 12586269025, 20365011074]
```

```
# Write a program to print a multiplication table (1 to 10) for a
given number using a for loop.

num = int(input("Enter the num to print the table - "))
```

```
for i in range(1,11):
    print(f"{num} X {i} = {num*i} ")
```

```
Enter the num to print the table - 6
6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30
6 X 6 = 36
6 X 7 = 42
6 X 8 = 48
6 X 9 = 54
6 X 10 = 60
```

```
# Write a program that prints the prime numbers from 1 to 50 using a for loop.

for num in range(1,51):
    for i in range(2, 4):
        if num%i == 0:
            break
    else:
        print(num)
```

```
1
5
7
11
13
17
19
23
25
29
31
35
37
41
43
47
49
```

```
# num = 6
# for i in range(2,num):
#     if num%i == 0:
#         print('not a prime number')
#         break
# else:
#     print('Prime')


# Write a program that finds all the common elements between two lists
using a loop.

list1 = [10,20,8,9,6]
list2 = [1,2,6,10]

for i in list1:
    if i in list2:
        print(i)

10
6

# Implement a program to print all perfect numbers between 1 and 1000.

for num in range(1,1001):

    divisors = []
    for i in range(1,num):
        if num%i == 0:
            divisors.append(i)

    if sum(divisors) == num:
        print(num)


6
28
496



#Create a program that finds all the numbers divisible by both 3 and 5
from 1 to 100.

for num in range(1,101):
    if num%3 == 0 and num%5==0:
        print(num)

15
30
45
60
75
90
```

```python
#numpy array
# import numpy as np
# np.array()
# .shape - (ros,cols)
# .ndim - (number of dimensions)
# .size - no of elements/items in array
# - how to perform indexing and slicing on numpy array
# 2d array - array[ rows  ,   cols]
# np.arange(1,12, 0.5)
# np.zeros -
# np.ones -
# np.identity
# np.eye
# reshape -
# how to perform arithmatic
# array.min, axis = 0, 1
# max, .var, .mean, .sum, .std, np.sqrt, np.log, np.concatenate
# np.nditer, np.where(condition, if true output, output for false),
```