PYTHON SYNTAX

(List & String)


```
L1 = [10, 20, 30, 40, 50]
print(L1)
print(*L1)
print(*L1, sep="~")
```

```
[10, 20, 30, 40, 50]
10 20 30 40 50
10~20~30~40~50
```

```
str1 = "Python"
print(str1)
```

```
Python
```

PYTHON SYNTAX

(List & String)

```
L1 = [10, 20, 30, 40, 50]
print(L1[2])
```

30

```
str1 = "Python"
print(str1[2])
```

t

PYTHON SYNTAX

(List & String)

```
L1 = [ ]
str1 = ''''
print(L1)
print(str1)
```

```
[]
```

```
L = [0] * 5
print(*L)
str1 = "ABC" * 5
print(str1)
```

```
0 0 0 0 0
ABCABCABCABCABC
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
L1 = [10, 20, 30]
L2 = [40, 50]
L3 = L1 + L2
print(L3)
```

```
[10, 20, 30, 40, 50]
```

```
str1 = "Mathew"
str2 = "Kurian"
str3 = str1 + str2
print(str3)
```

```
MathewKurian
```

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

PYTHON SYNTAX

(List & String)

```
print(ord('A'), chr(65))  65 A
print(ord('Z'), chr(90))  90 Z
print(ord('a'), chr(97))  97 a
print(ord('z'), chr(122)) 122 z
print(ord('0'), chr(48))  48 0
print(ord('9'), chr(57))  57 9
```

```
L = [10, 20, 30]
str1 = "Mathew"
print(L[-1])
print(str1[-1])
print(str1[-2])
```

```
30
w
e
```

```
L = [10, 20, 30, 40, 50]
print(max(L))
print(min(L))
print(sum(L))
```

```
50
10
150
```

```
str1 = "AaZz01"
print(max(str1))
print(min(str1))
print(sum(str1))
```

```
z
0
---------
TypeError
```

PYTHON SYNTAX

(List & String)

```
#Slicing
L = [10, 20, 30, 40, 50]
print(L[1:4])
print(L[:4])
print(L[1:])
print(L[4:1:-1])
print(L[::])
L1 = L[::-1]
print(L)
print(L1)
```

```
[20, 30, 40]
[10, 20, 30, 40]
[20, 30, 40, 50]
[50, 40, 30]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
[50, 40, 30, 20, 10]
```

out of range slice indexes are handled gracefully when used for slicing

```
#Slicing
str1 = "Mathew"
print(str1[1:4])
print(str1[:4])
print(str1[1:])
print(str1[4:1:-1])
print(str1[::])
str2 = str1[::-1]
print(str1)
print(str2)
```

```
ath
Math
athew
eht
Mathew
Mathew
wehtaM
```

PYTHON SYNTAX

(List & String)


```
L = [10, 20, 30, 40, 50]
L[2] = 999
print(*L)
```

```
10 20 999 40 50
```

```
L = [10, 20, 30, 40, 50]
L.append(999)
print(*L)
```

```
10 20 30 40 50 999
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
L = [10, 20, 30, 40, 50]
L.insert(2, 999)
print(*L)
```

```
10 20 999 30 40 50
```

```
L = [10, 20, 30, 40, 50]
del L[2]
print(*L)
```

```
10 20 40 50
```

```
L = [10, 20, 30, 40, 50]
str1 = "Python"
print(len(L))
print(len(str1))
```

```
5
6
```

PYTHON SYNTAX

(List & String)

```
L = [10, 20, 30, 40, 50]
#x = del L[2]
x = L.pop()
print(x)
y = L.pop(2)
print(y)
print(L)
```

```
50
30
[10, 20, 40]
```

```
L = [10, 20, 30, 20, 40]
L.remove(20)
print(*L)
```

```
10 30 20 40
```

PYTHON SYNTAX

(List & String)

```
int_str = "12345"
alph_str = "A"
print(float(int_str))
print(int(int_str))
print(int(alph_str))
```

```
12345.0
12345

----------
ValueError
```

```
str1 = "123.50"
print(float(str1))
print(int(str1))
```

```
123.5

----------
ValueError
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
L = [10, 20, 30, 40, 50]
for i in range(len(L)):
    print(L[i], end= " ")
```

0 1 2 3 4

```
10 20 30 40 50
```

```
str1 = "I Love Python"
for i in range(len(str1)):
    print(str1[i], end=" ")
```

0 1 2 3 4 5 6 7 8 9 10 11 12

```
I   L o v e   P y t h o n
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
L = [10, 20, 30, 40, 50]
for ele in L:
    print(ele, end=" ")
```

```
10 20 30 40 50
```

```
str1 = "I Love Python"
for ele in str1:
    print(ele, end=" ")
```

```
I  L o v e  P y t h o n
```

PYTHON SYNTAX

(List & String)


```
#join can be applied only on strings and list of strings
#Output will be always a string
str1 = "I love Python"
x = "###".join(str1)
print(x, type(x))
```

```
I### ###l###o###v###e### ###P###y###t###h###o###n <class 'str'>
```

```
L = [10, 20, 30, 40, 50]
x = " ".join(map(str, L))
print(x, type(x))
```

```
10 20 30 40 50 <class 'str'>
```

```
inp_str = "MATHEW"
str_list = list(inp_str)
print(str_list, type(str_list))
str1 = str(str_list)
print(str1, type(str1))
str2 = "".join(str_list)
print(str2, type(str2))
```

```
['M', 'A', 'T', 'H', 'E', 'W'] <class 'list'>
['M', 'A', 'T', 'H', 'E', 'W'] <class 'str'>
MATHEW <class 'str'>
```

PYTHON SYNTAX

(List & String)

```
#split can be applied only on strings
#Output of split is always a list of strings
inp_str = "Mathew kurian"
str_list = inp_str.split()
print(str_list)
```

```
['Mathew', 'kurian']
```

PYTHON SYNTAX

(List & String)

```
L1 = [10, 20, 30, 30, 40, 50]
L2 = L1.reverse()
print(L1)
print(L2)
```

```
[50, 40, 30, 30, 20, 10]
None
```

```
L1 = [10, 20, 30, 30, 40, 50]
L2 = L1[: :-1]
print(L1)
print(L2)
```

```
[10, 20, 30, 30, 40, 50]
[50, 40, 30, 30, 20, 10]
```

```
str1 = "Mathew"
str1 = str1[: :-1]
print(str1)
```

```
wehtaM
```

PYTHON SYNTAX

(List & String)

```
L = [30, 10, 50, 20, 40]
L.sort()
print(L)

L.sort(reverse=True)
print(L)
```

```
[10, 20, 30, 40, 50]
[50, 40, 30, 20, 10]
```

```
#output of 'sorted' is always a list
L1 = [30, 10, 50, 20, 40]
L2 = sorted(L1)
print(L1)
print(L2)

L3 = sorted(L1, reverse=True)
print(L3)
```

```
[30, 10, 50, 20, 40]
[10, 20, 30, 40, 50]
[50, 40, 30, 20, 10]
```

PYTHON SYNTAX

(List & String)

```
#output of 'sorted' is always a list
str1 = "EBACGDHF"
L1 = sorted(str1)
print(L1)

L2 = sorted(str1, reverse=True)
print(L2)

str2 = "".join(L1)
print(str2)

str3 = "".join(L2)
print(str3)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
['H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']
ABCDEFGH
HGFEDCBA
```

```
L = [30, 555, 30, 40, 30, 555]
str1 = "Python Python"
print(L.count(30))
print(str1.count("th"))
print(L.index(555))
print(str1.index("th"))
```

```
3
2
1
2
```

PYTHON SYNTAX

(List & String)

```
str1 = "I love programming"
str2 = " "
print(str1.lower())
print(str1.upper())
print(str1.islower())
print(str1.isupper())
print(str2.isspace())
```

```
i love programming
I LOVE PROGRAMMING
False
False
True
```

```
str1 = "12345"
print(str1.isdecimal())
print(str1.isdigit())
print(str1.isalpha())
print(str1.isalnum())
print(str1.isnumeric())
```

```
True
True
False
True
True
```

PYTHON SYNTAX

(List & String)

```
str1 = "How are you How are you How are you"
print(str1.replace("are", "about"))
print(str1.replace("are", "about", 2))
print(str1.count("How", 0, len(str1)))
print(str1.find("you", 0, len(str1)))
```

```
How about you How about you How about you
How about you How about you How are you
3
8
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
str1 = "I Love Python"
if 'Lov' in str1:
    print("YES, its a substring")
else:
    print("NO, its not a substring")
```

```
YES, its a substring
```

```
str1 = "I Love Python"
if 'Lov' not in str1:
    print("YES, its a substring")
else:
    print("NO, its not a substring")
```

```
NO, its not a substring
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
x = "PYTHON"
for val in x:
    if(val=="T"):
        print("I am T")
```

```
I am T
```

```
L1 = ['A', 'B', 'C', 'C', 'D', 'E']
L2 = [1, 2, 3, 3, 10]
L3 = ["One", "Two", "Three", "Three"]

for ele1, ele2, ele3 in zip(L1, L2, L3):
    print(ele1, ele2, ele3)
```

```
A 1 One
B 2 Two
C 3 Three
C 3 Three
```

PYTHON SYNTAX

(List & String)

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
#For taking line separated array input
n = int(input())
L = [int(input()) for i in range(n)]
print(L)
```

```
3
10
20
30
[10, 20, 30]
```

```
#For taking space separated array input
L = [int(x) for x in input().split()]
print(L)
```

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

PYTHON SYNTAX

(Two Dimensional List & String [Matrix/Grid])


```
M = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]
print(M)
print(*M)
print(M[1])
print(M[1][2])
```

	j=0	j=1	j=2
i=0	10	20	30
i=1	40	50	60
i=2	70	80	90

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

PYTHON SYNTAX

(Two Dimensional List & String [Matrix/Grid])

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
M = ["ABCD", "EFGH", "IJKL"]
print(M)
print(*M)
print(M[1])
print(M[1][2])
```

	j=0	j=1	j=2	j=3
i=0	'A'	'B'	'C'	'D'
i=1	'E'	'F'	'G'	'H'
i=2	'I'	'J'	'K'	'L'

```
['ABCD', 'EFGH', 'IJKL']
ABCD EFGH IJKL
EFGH
G
```

PYTHON SYNTAX

(Two Dimensional List & String [Matrix/Grid])

MANIFOLD
INSTITUTE OF TECHNICAL EDUCATION
AN IIT-NIT ALUMNI INITIATIVE

```
M = [[10, 20, 30], [60, 40, 50], [70, 80, 90]]
M[1].sort()
M[2].reverse()
print(M)
```

	j=0	j=1	j=2
i=0	10	20	30
i=1	60	40	50
i=2	70	80	90

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

PYTHON SYNTAX

(Two Dimensional List & String [Matrix/Grid])


```
#2 DIMENSIONAL ARRAY - LINE SEPARATED INPUT
m, n = int(input()), int(input())
p, q = int(input()), int(input())

A = [[int(input()) for j in range(n)] for i in range(m)]
B = [[int(input()) for j in range(q)] for i in range(p)]
C = [[0 for j in range(n)] for i in range(m)]

if(m==p and n==q):
    for i in range(m):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
            print(C[i][j], end=" ")
        print()
```

```
3
2
3
2
5
5
5
5
5
5
6
4
6
4
6
4
11 9
11 9
11 9
```

```
#2 DIMENSIONAL ARRAY - SPACE SEPARATED INPUT
m, n = int(input()), int(input())
p, q = int(input()), int(input())

A = [[int(x) for x in input().split()] for i in range(m)]
B = [[int(x) for x in input().split()] for i in range(p)]
C = [[0 for j in range(n)] for i in range(m)]

if(m==p and n==q):
    for i in range(m):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
            print(C[i][j], end=" ")
        print()
```

```
3
2
3
2
5 5
5 5
5 5
6 4
6 4
6 4
11 9
11 9
11 9
```

NO. CONVERSIONS

```
#input: String
#output: Decimal
n = "100"
print(int(n, 2))
print(int(n, 8))
print(int(n, 16))
```

```
4
64
256
```

```
#input: Decimal
#output: String
n = 100
print(bin(n))
print(oct(n))
print(hex(n))
```

```
0b1100100
0o144
0x64
```

```
n = 100
print(bin(n)[2:])
```

```
1100100
```