```python
# 1) WAP to multiply two square matrices.
# Matrices are written in python as [[a,b],[c,d]] where [a,b] is a
single row .
# Note: This program works only for square matrices.

r=int(input("Enter the number of rows"))
c=int(input("Enter the number of columns"))
m1=[]
m2=[]
temp=[]
e=0
pro=[]
print("For matrix 1")
for i in range(0,r):
    for j in range(0,c):
        e=int(input("Enter an element."))
        temp.append(e)
    m1.append(temp)
    temp=[]
print("For matrix 2")
for i in range(0,r):
    for j in range(0,c):
        e=int(input("Enter an element."))
        temp.append(e)
    m2.append(temp)
    temp=[]
for i in range(0,r):
    for j in range(0,c):
        e=(m1[i][j])*(m2[j][i])
        temp.append(e)
    pro.append(temp)
    temp=[]
print(m1,"*",m2,"=",pro)
```

```
Enter the number of rows 2
Enter the number of columns 2

For matrix 1

Enter an element. 1
Enter an element. 2
Enter an element. 1
Enter an element. 2

For matrix 2

Enter an element. 3
Enter an element. 4
Enter an element. 3
Enter an element. 4
```

```
[[1, 2], [1, 2]] * [[3, 4], [3, 4]] = [[3, 6], [4, 8]]
```

```
# 2) WAP to generate all subsets of a given set of numbers.

r = int(input("Enter the number of elements"))
li = []
subset = [[]]
m1 = []
for i in range(r):
    e = int(input("Enter an element"))
    li.append(e)
for i in range(r):
    for j in range(i+1, r+1):
        m1.append(li[i:j])
        if m1 not in subset:
            subset.append(m1)
print(subset)
```

```
Enter the number of elements 5
Enter an element 1
Enter an element 2
Enter an element 3
Enter an element 4
Enter an element 5
```

```
[[], [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4], [1, 2, 3, 4, 5], [2], [2,
3], [2, 3, 4], [2, 3, 4, 5], [3], [3, 4], [3, 4, 5], [4], [4, 5],
[5]]]
```

```
# 3) WAP to print the following patterns:
# *
# **
# ***
# ****
# *****

a="*"
for i in range(5):
    print(a)
    a=a+"*"
```

```
*
**
***
****
*****
```

```
# 3) 1
#    1 2 1
#    1 2 3 2 1
#    1 2 3 4 3 2 1
```

```
#    1 2 3 4 5 4 3 2 1

for i in range(1,6):
    for j in range(1,i+1):
        print(j,"",end="")
    for k in range(i-1,0,-1):
        print(k,"",end="")
    print()
```

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

```
# 3) *
#    ***
#    *****
#    *******
#    *********
#    *******
#    *****
#    ***
#    *

a="*"
b=[]
for i in range(9):
    if i<=4:
        print(a)
        b.append(a)
        a=a+"*"+"*"
    else:
        print(b[8-i])
```

```
*
***
*****
*******
*********
*******
*****
***
*
```

```
# 3) 1 2 3 4 5 4 3 2 1
#     1 2 3 4 3 2 1
#       1 2 3 2 1
#         1 2 1
#           1
```

```
for i in range(5, 0, -1):
    for j in range(5 - i):
        print("  ", end="")
    for k in range(1, i + 1):
        print(k, "", end="")
    for l in range(i - 1, 0, -1):
        print(l, "", end="")
    print()
```

```
1 2 3 4 5 4 3 2 1
  1 2 3 4 3 2 1
    1 2 3 2 1
      1 2 1
        1
```

```
# 3) *
#    ***
#    *****
#    *******
#    *********

a="*"
for i in range(5):
    print(a)
    a=a+"*"+"*"
```

```
*
***
*****
*******
*********
```

```
# 4) WAP to print Pascal Traingle.

# importing "factorial" fucntion from the "math" module.
from math import factorial
n=int(input("Enter the number of iterations"))
for i in range(n):
    for j in range(n-i+1):
        print(end=" ")
    for k in range(i+1):
        print(factorial(i)//(factorial(k)*factorial(i-k)),"",end="")
    print()
```

```
Enter the number of iterations 6
```

```
       1
      1 1
     1 2 1
    1 3 3 1
```

```
   1 4 6 4 1
  1 5 10 10 5 1
```

```
# 5) WAP tp print a zig-zag pattern number pattern.

n=int(input("Enter the number upto which the pattern must be
printed"))
a=[]
k=0
l=0
for i in range(1,n+1):
    a.append(i)
for i in range(1,(n//5)+1):
    if i==1:
        for j in range(5):
            print(a[l+j],"",end="")
    elif i%2!=0:
        l=k-4
        for j in range(5):
            print(a[l+j],"",end="")
    else:
        for j in range(5):
            print(a[k-j],"",end="")
    print("")
    if(i==1):
        k=4
    k+=5
```

```
Enter the number upto which the pattern must be printed 50

1 2 3 4 5
10 9 8 7 6
11 12 13 14 15
20 19 18 17 16
21 22 23 24 25
30 29 28 27 26
31 32 33 34 35
40 39 38 37 36
41 42 43 44 45
50 49 48 47 46
```