In [1]:
```
for i in range (1,6):
    print('*')
```

```
*
*
*
*
*
```

Right Angle Triangle

In [2]:
```
for i in range (1,6):
    print('*' *i)
```

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

Inverted Right Angle Triangle

In [3]:
```
for i in range (5,0,-1):
    print('*' *i)
```

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

Full Star Pyramid

In [14]:
```
n=5
for i in range (1,n+1): #defining the range, n+1 is to print n, as n+1 is exclud

    for j in range(n-i): #defining the spaces from center top cone, then decreme
        print(' ',end='') #to print empty spaces

    for j in range(2*i-1): #to define the no of items
        print('*',end='')

    print()
```

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

Inverted Full Star Pyramid

In [7]:
```
n=5
for i in range (n,0,-1): #defining the range, n+1 is to print n, as n+1 is exclu

    for j in range(n-i): #defining the spaces from center top cone, then decreme
        print(' ',end='') #to print empty spaces

    for j in range(2*i-1): #to define the no of items
        print('*',end='')

    print()
```

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

Hallow Pyramid Pattern

In [11]:
```
n=5
for i in range (1,n+1): #defining the range
    for j in range (n-i): #defining the spaces from center
        print(' ',end='') #to print the spaces

    for j in range(2*i-1): #to define the number of items
        if j==0 or j==2*i-2 or i==5:
#j==0: Print * at the beginning of each row, j==2*i-2: Print * at the end of eac
            print('*', end='')
        else:
            print(' ',end='')

    print()
```

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

Step Pattern

In [13]:
```
n = 5
for i in range(1,n+1):
    print(''*(n-i)+' * '*(2*i-1))
```

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

- print(''*(n-i)+' * '*(2*i-1)) - For each iteration, this prints a line with: - ''*(n-i)` - Empty strings multiplied by (n-i), which creates leading spaces (though this doesn't actually create spaces as intended)

- `' * '*(2*i-1)` - The string ' * ' repeated (2*i-1) times, which creates an increasing number of asterisks with spaces between them
- The pattern would print 1, 3, 5, 7, and 9 asterisks (with spaces) on consecutive lines, with decreasing leading spaces, creating a triangular shape.

Inverted Step Pattern

In [15]:
```
n = 5
for i in range(n,0,-1):
    print(''*(n-i)+' * '*(2*i-1))
```

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

Full square

In [19]:
```
n = 3
for i in range(n):
    print(' * '*n) #if spaces are not provided it will look like a rectangle
    #*n is means where we are printing "n" no of values every time
```

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

In [20]:
```
n = 6
for i in range(n):
    print(' * '*n)
```

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

Hallow Square Pattern

In [29]:
```
n = 5
for i in range(n):
    for j in range(n):
        if i==0 or i==4 or j==0 or j==4: #need to mention i,j ==0 & 4.
            print('*',end='')
        else:
            print(' ',end='')
    print()
```

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

The `if` condition `i==0 or i==4 or j==0 or j==4` checks if the current position is on the border of the square:

- `i==0` checks if it's the first row
- `i==4` checks if it's the last row
- `j==0` checks if it's the first column
- `j==4` checks if it's the last column

Right angle Triangle with Number Pattern

In [32]:
```
n=5
for i in range(1,n+1):
    print(' '.join(str(x) for x in range(1,i+1)))
```

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

- `range(1,i+1)` creates a sequence of numbers from 1 to `i`
- `str(x) for x in range(1,i+1)` converts each number in that sequence to a string
- `' '.join(...)` joins these strings together with spaces between them
- `print(...)` outputs the resulting string

Floyds Triangle

In [33]:
```
num=1
for i in range(1,6):
    for j in range(1,i+1):
        print(num,end='')
        num+=1
    print()
```

```
1
23
456
78910
1112131415
```