Patterns

Some Advanced Patterns

Pattern 2.1 - Inverted Triangle

```
# N = 3
* * *
* *
*
```

Approach:

From the above pattern, **we can observe:**

- ➔ **Number of Rows:** The pattern has 3 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** The number of columns in any row is equal to `N-rowNumber+1`.1st row has 3 columns `(3-1+1)`, 2nd row has 2 columns `(3-2+1)`, and so on. Thus, in a pattern of N rows, the `i`th row will have `N-i+1` columns.
- ➔ **What to print:** All the entries in any row are `"*"`.

Python Implementation:

```
N=int(input()) #Take user input, N= Number of Rows
row=1; #The loop starts with the 1st row
while row<=N: #Loop will on for N rows
    col=1; #The loop starts with the first column in the current row
    while col<=N-row+1: #Number of columns = N-rowNumber+1
        print("*",end="") #Printing a (*) in all columns
        col=col+1 #Increment the current column (Inner Loop)
    row=row+1 #Increment the current row (Outer Loop)
    print() #Add a new Line after each row is printed
```

Pattern 2.2 - Reversed Pattern

```
# N = 3
    *
  * *
* * *
```

Approach:

From the above pattern, **we can observe:**

- ➔ **Number of Rows:** The pattern has 3 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** The number of columns in any row is equal to `N`.
- ➔ **What to print:** In the 1st row, while `columnNumber <= 2(3-1)`, we print a `" "` in every column. Beyond the 2nd column, we print a `"*"`. Similarly, in the 2nd row, we print a `" "` till `columnNumber <=1(3-2)` and beyond the 1st column, we print a `"*"`. We can easily notice that if `col <= N-rowNumber`, we are printing a `" " (Space)`. And if `col > N-rowNumber`, we are printing a `"*"`.

Python Implementation:

```
N=int(input()) #Take user input, N= Number of Rows
row=1; #The loop starts with the 1st row
while row<=N: #Loop will on for N rows
    col=1; #The loop starts with the first column in the current row
    while col<=N:#The loop will go on for N columns
        if(col<=N-row):
            print(" ",end="") #Printing a (" ")
        else:
            print("*",end="") #Printing a (*)
        col=col+1 #Increment the current column (Inner Loop)
    row=row+1 #Increment the current row (Outer Loop)
    print() #Add a new Line after each row is printed
```

Pattern 2.3 - Isosceles Pattern

```
# N = 4
   1
  121
 12321
1234321
```

Approach:

From the above pattern **we can observe:**

- ➔ **Number of Rows:** The pattern has 3 rows. We have to print the pattern for N rows.
- ➔ **Number of Columns:** Similar to Pattern 2.2, we first have `N-rowNumber` columns of spaces. Following this, we have `2*rowNumber-1` columns of numbers.
- ➔ **What to print:** We can notice that if `col <= N-rowNumber`, we are printing a `" " (Space)`. Further, the pattern has two parts. First is the increasing part and second is the decreasing part. For the increasing part, we will initialise a variable `num=1`. In each row we will keep printing `num` till its value becomes equal to the `rowNumber` . We will increment `num` by 1 after printing it; ;this will account for the first part of the pattern. We have `num = rowNumber` at this stage. Now, for the decreasing part, we will again start printing `num` till `num>=1`. After printing `num` we will decrement it by 1.

Python Implementation:

```
N=int(input()) #Take user input, N= Number of Rows
row=1; #The loop starts with the 1st row
while row<=N: #Loop will on for N rows
    spaces =1 # Printing spaces
    while spaces<= N- row:
        print(" ",end="")
        spaces=spaces+1

    num=1 #Variable to print the numbers
    while num<=row: #Increasing Pattern
        print(num,end="")
        num=num+1;

    num=row-1 # We have to start printing the decreasing part from
one less than the rowNumber
    while num>=1: #Decreasing Pattern
        print(num,end="")
        num=num-1
    print()#New Line
    row=row+1
```

Practice Problems

Here are a few similar patterns problems for your practice. All the patterns have been drawn for N=4.

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

```
   1
  121
 12321
1234321
 12321
  121
   1
```

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

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