range
range is sequence data type
range is an iterable, which generate a sequence integers
The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.
“range class” is used to represent range object

Syntax1: range(stop)
Syntax2: range(start,stop,[step])

Range object is having 3 attributes

1. Start
2. Stop
3. Step

All these values must be integer type and step should not be 0
Start: the beginning value of the range, this value is included
Stop: end of value of range, this value excluded
Step: increment/decrement value (OR) the difference between values with range

Syntax-1: range(stop)
This syntax is used to generate sequence of +ve integers
This syntax accept stop value and default value of start=0,step=+1

r=range(10) ➔ start=0,stop=10,step=+1
0 1 2 3 4 5 6 7 8 9
r=range(-10) ➔ start=0,stop=-10,step=+1
r=range(5) ➔ start=0,stop=5,step=+1
0 1 2 3 4
r=range(10.0) ➔ SyntaxError
Example:

```
>>> r=range(10)
>>> print(r)
range(0, 10)
>>> for value in r:
        print(value,end=' ')
```

0 1 2 3 4 5 6 7 8 9

```
>>> r=range(5)
>>> for x in r:
        print("Python")

Python
Python
Python
Python
Python

>>> for value in range(5):
        print(value,end=' ')

0 1 2 3 4
>>>
```

Rules:

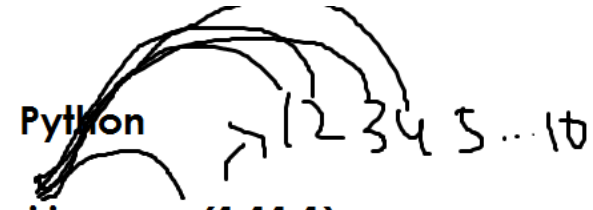
1. If the step +value, start<stop
2. If the step –value, start>stop

Syntax2: range(start,stop,[step])
This syntax allows to input three values
Start and stop is required arguments, step is default +1
r=range(1,11) ➔ start=1,stop=11,step=+1
1 2 3 4 5 6 7 8 9 10
r=range(1,11,2) ➔ start=1,stop=11,step=+2
1 3 5 7 9
r=range(2,11,2) ➔ start=2,stop=11,step=+2
2 4 6 8 10
r=range(1,10,-1) ➔ start=1,stop=10,step=-1
r=range(10,0,-1) ➔ start=10,stop=0,step=-1
10 9 8 7 6 5 4 3 2 1
r=range(-1,-11,-1) ➔ start=-1,stop=-11,step=-1
-1 -2 -3 -4 -5 -6 -7 -8 -9 -10
r=range(-11,0) ➔ start=-11,stop=0,step=+1
-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

C,C++,Java

```
for(i=1;i<=10;i++)
  printf("%d",i)
```

Python




```
for i in range(1,11,1):
  print("%d%i)
```

Example:

```
# write a program to generate math table of input number

num=int(input("Enter any number"))
for i in range(1,11): # 1 2 3 4 5 6 7 8 9 10
    print(f'{num}x{i}={num*i}')
```

Output:

```
Enter any number5
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50
>>>
```

Example:

```
# write a program to find factorial of input number
# 4 ==> 1x2x3x4=24
num=int(input("Enter any number"))
fact=1
for i in range(1,num+1):
    fact=fact*i

print(f'factorial {fact}')
```

Output:

```
Enter any number0
```

```
factorial 1
>>>
```

Nested looping statements

- ⇨ Nested while
- ⇨ Nested for

A looping statement inside a looping statement nested looping statement

Nested for
for loop inside for loop is called nested for
Syntax:

```
for variable in iterable: # outer loop
        for variable in iterable: # inner loop
                statement-1
                statement-2
        statement-3
```

Example:

```
# generate prime numbers between given range
m=int(input("Enter start value")) # 3
n=int(input("Enter stop value")) # 11
for num in range(m,n): # 3 4 5 6 7 8 9 10
    c=0
    for i in range(1,num+1): # 1 2 3
        if num%i==0:
            c+=1
    if c==2:
        print(num)
```

Output:

```
Enter start value2
Enter stop value20
2
3
5
7
11
13
17
19
```

>>>

Example:
 # write a program to generate tables from 1 to 10

```
for num in range(1,11): # 1 2 3 4 5 6 7 8 9 10
    for i in range(1,11): # 1 2 3 4 5 6 7 8 9 10
        print(f'{num}x{i}={num*i}')
    input("Press any key")
```

Output:

```
1x1=1
1x2=2
1x3=3
1x4=4
1x5=5
1x6=6
1x7=7
1x8=8
1x9=9
1x10=10
Press any key
2x1=2
2x2=4
2x3=6
2x4=8
2x5=10
2x6=12
2x7=14
2x8=16
2x9=18
2x10=20
Press any key
```