Loops In Python

- In Python, loops are used to execute a block of code repeatedly
- There are two main types of loops in Python: ### 1. 'For' loops ### 2. 'While' loops ### 3. Loops Controls Statement

What is the use of Loops?why we used that?? (interview Question)

Answer:

- Loops in Python are used to execute a block of code repeatedly.
- They are essential for automating repetitive tasks, processing collections of data (e.g., lists or strings), and iterating through sequences or until a specific condition is met.
- for loops are commonly used for iteration, while while loops are used when you need to repeat code based on a condition.
- Loop control statements like break and continue provide flexibility in managing loop execution, making Python a powerful language for tasks that involve repetition and iteration.

1. For loops

- A 'for loop' is used to iterate over a sequence (such as a list, tuple, string, or range) and execute a block of code for each item in the sequence.

In [2]:
```
# if we have to print 1 to 10 number then we use fo loop

for i in range(1,11):   # in for loop we have to sepcify the range or also mention
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [4]:
```
# for loop for datatype like tuple

for i in (1,2,3,4,5,6,7,8):
    print(i)

# for loop is very very flexible and powerful in python it used with all datatypes
```

```
1
2
3
4
5
6
7
8
```

In [7]:
```
# how to print 1 to 20 with differnce of 2

for i in range(1,21,2): # 2 is the
    print(i)
```

```
1
3
5
7
9
11
13
15
17
19
```

- The 2 inside the range() function is called the "step" or "stride." It determines the increment between each consecutive value in the range. In this case, the loop will iterate over the numbers from 1 to 20, with a step of 2.
- So, the loop will print the following numbers: 1, 3, 5, 7, 9, 11, 13, 15, 17, and 19. The loop starts at 1, increments by 2 in each iteration, and stops before reaching or exceeding 21.

In [8]:
```
# how to print 10 to 1 in reverse order

for i in range(10,0,-1):
    print(i)
```

```
10
9
8
7
6
5
4
3
2
1
```

Program - The current population of a town is 10000. The population of the town is increasing at the rate of 10% per year. You have to write a program to find out the population at the end of each of the last 10 years.

In [12]:
```
# for this we use for loop
curr_pop = 10000
for i in range(10,0,-1):
    print(i,curr_pop)
```

```
    curr_pop /= 1.1
```

```
10 10000
9 9090.90909090909
8 8264.462809917353
7 7513.148009015775
6 6830.134553650703
5 6209.213230591548
4 5644.739300537771
3 5131.5811823070635
2 4665.07380209733
1 4240.976183724845
```

2. While Loop

- A while loop is used to repeatedly execute a block of code as long as a certain condition is true.

In [14]:
```
# if we want to print 1 to 5 with the help of while
count = 1
while count <= 5:
    print(count)
    count += 1
```

```
1
2
3
4
5
```

In [16]:
```
# Program ->print 12 table

num = int(input('Enter the Number'))

i = 1

while i<11:
    print(num * i)
    i+=1
```

```
Enter the Number12
12
24
36
48
60
72
84
96
108
120
```

3. Loop Control Statements:

In addition to basic loops, Python provides loop control statements that allow you to customize loop behavior:

- 'break': Used to exit the loop prematurely based on a certain condition.

- 'continue': Skips the current iteration of the loop and proceeds to the next iteration.
- 'else' clause: Can be used with a for or while loop to specify a block of code that will be executed after the loop has finished normally (i.e., without hitting a break statement).

while loop with else

In [17]:
```
x = 1

while x < 3:
    print(x)
    x += 1

else:
    print('limit Crossed')
```

```
1
2
limit Crossed
```

Break and Continue

In [20]:
```
# Using break

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in numbers:
    if num == 6:
        break  # Exit the loop when num equals 6
    print(num)

print("Loop ended")
```

```
1
2
3
4
5
Loop ended
```

In this example, the loop will print numbers 1 through 5 and then exit the loop when num becomes 6 due to the break statement. The "Loop ended" message will be printed after the loop finishes.

In [21]:
```
# Using continue

numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num == 3:
        continue  # Skip printing 3 and continue with the next iteration
    print(num)

print("Loop ended")
```

```
1
2
4
5
Loop ended
```

In this example, when num equals 3, the continue statement is encountered, which skips the printing of 3 and moves on to the next iteration. As a result, the loop will print numbers 1, 2, 4, and 5. The "Loop ended" message will be printed after the loop finishes.

In [23]:
```
# Using Continue and Break
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num == 3:
        continue  # Skip printing 3
    elif num == 5:
        break     # Exit the loop when 5 is encountered
    print(num)

print("Loop finished normally")
```

```
1
2
4
Loop finished normally
```