Syntax of for

used to iterate over any iterable data type tuple, list, string...

for i in range(start_index, end_index, steps): ¶

    |
    |
    |

else:

    |
    |

```
start_index --> default is 0      (optional)
end_index --> ending index, if n given g
steps --> increment by number, default is 1 (optional)
else block will be executed at last or if loop is breaked
```

In [3]:
```
# for i in range (10):                    # 0 1 2 3 4 5 6 7 8 9
# for i in range (1,10):                  # 1 2 3 4 5 6 7 8 9
# for i in range (1,11):                  # 1 2 3 4 5 6 7 8 9 10
# for i in range (0,11,2):                # 0 2 4 6 8 10
# for i in range (11,0,-1):               # 11 10 9 8 7 6 5 4 3 2 1
# for i in [10, 20, 30, 50, 60]:          # 10 20 30 50 60
# for i in "Tej Patel":                   # T,e,j, ,P,a,t,e,l,
# for i in (10,5,2):                      # 10 5 2
# for i in range(10,5.5,2):               # invalid as only integers are all
    print(i, end=" ")


# print(list(range(10)))                  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
10 5 2
```

Syntax of while

used to iterate while given condition is true

while condition:

    |

else:

    |

```
else block will be executed at last or if loop is breaked
```

In [8]:
```
i = 0
while True :
    print(i, end=" ")
    i += 1
    if i > 10:
        break
else :
    print("This will not be executed at last")

print()


i = 0
while i < 10 :
    print(i, end=" ")
    i += 1
else :
    print("This will be executed at last")
```

```
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 This will be executed at last
```

for and while loop

Find the most optimal solution (minimum iteration same result)

01) WAP to print 1 to 10

In [1]:
```

```

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

02) WAP to print 1 to n

In [2]:
```

```

```
Enter n :  10

0 1 2 3 4 5 6 7 8 9 10
```

03) WAP to print odd numbers between 1 to n

In [4]:
```

```

```
Enter n :  50

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

04) WAP to print numbers between two given numbers which is

In [3]:

```
Enter n :  50

2 4 8 10 14 16 20 22 26 28 32 34 38 40 44 46 50
```

05) WAP to print sum of 1 to n numbers

In [6]:

```
Enter n :  8

36
```

06) WAP to print sum of series 1 + 4 + 9 + 16 + 25 + 36 + ...n

In [7]:

```
Enter n :  5

55
```

07) WAP to print sum of series 1 – 2 + 3 – 4 + 5 – 6 + 7 ... n

In [8]:

```
Enter n :  5

3
```

08) WAP to print multiplication table of given number.

In [9]:

```
Enter n :  5

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

09) WAP to find factorial of the given number

In [10]:

```
Number :  6

Factorial of 6 is : 720
```

10) WAP to find factors of the given number

[Basic Task : only n/2 iterations]
[Advanced Task : only n^2 iterations]

In [1]:

```
Number : 60
Factors of 60 are : 1 2 3 4 5 6 10 12 15 20 30
```

11) WAP to find whether the given number is prime or not.

In [11]:

```
Number :  103

103 is Prime
```

12) WAP to print sum of digits of given number

In [12]:

```
Number :  35421

Sum of digits of 35421 is 15
```

13) WAP to check whether the given number is palindrome or not

In [13]:

```
Number :  123321

123321 is palindrome
```

01) WAP to check whether the given number is Armstrong or not.

In [14]:

```
Number :  153

153 : No is armstrong
```

02) WAP to find out prime numbers between given two numbers.

In [17]:

```
Number 1 :  1
Number 2 :  50

1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

03) WAP to calculate x^y without using any function.

[Basic Task : Using for loops only no use of ** operators]
[Advance Task : without using *]

In [15]:

```
base (x) :  5
power (y) :  3

5 ^ 3 = 125
```

04) WAP to check whether the given number is perfect or not.

[Sum of factors including 1 excluding number itself]

In [22]:

```
Number :  128

128 is perfect no.
```

05) WAP to find the sum of 1 + (1+2) + (1+2+3) + (1+2+3+4)+...+ (1+2+3+4+....+n)

In [18]:

```
Number :  5

56 : is the sum
```

06) WAP to print Multiplication Table up to n

In [21]:

```
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 2   4   6   8  10  12  14  16  18  20  22  24  26  28  30
 3   6   9  12  15  18  21  24  27  30  33  36  39  42  45
 4   8  12  16  20  24  28  32  36  40  44  48  52  56  60
 5  10  15  20  25  30  35  40  45  50  55  60  65  70  75
 6  12  18  24  30  36  42  48  54  60  66  72  78  84  90
 7  14  21  28  35  42  49  56  63  70  77  84  91  98 105
 8  16  24  32  40  48  56  64  72  80  88  96 104 112 120
 9  18  27  36  45  54  63  72  81  90  99 108 117 126 135
10  20  30  40  50  60  70  80  90 100 110 120 130 140 150
11  22  33  44  55  66  77  88  99 110 121 132 143 154 165
12  24  36  48  60  72  84  96 108 120 132 144 156 168 180
13  26  39  52  65  78  91 104 117 130 143 156 169 182 195
14  28  42  56  70  84  98 112 126 140 154 168 182 196 210
15  30  45  60  75  90 105 120 135 150 165 180 195 210 225
```