WEEK-4
Control Flow: Loops
While loop: general format; examples For loop: general format, examples. Range();nesting loops and conditional statements; Controlling loop execution: Break, continue, pass statements;

while loop

- The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.
- **Syntax:**
  while Boolean_Expression:
      statement(s)
- The while loop starts with the while keyword and ends with a colon.
- With a while statement, first the Boolean expression is evaluated before the statements in the while loop block is executed.
- If the Boolean expression evaluates to False, then the statements in the while loop block are never executed.
- If the Boolean expression evaluates to True, then the while loop block is executed.
- After each iteration of the loop block, the Boolean expression is again checked, and ifit is True, the loop is iterated again.
- Each repetition of the loop block is called an iteration of the loop.
- This process continues until the Boolean expression evaluates to False and at this point the while statement exits.
- Execution then continues with the first statement after the while loop.
- Flowchart of while is given below

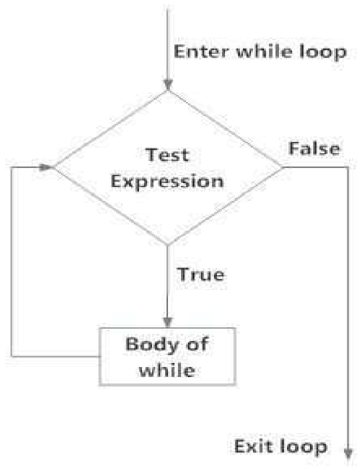


Fig: operation of while loop

Example 1:Program to Display First 10 Numbers Using while Loop Starting from 0

```
i = 0
while i < 10:
    print(f"Current value of i is {i}")

    i = i + 1
```

Example 2: Program to Find the sum and Average of n Natural Numbers Where n Is the Input from the User

```
number = int(input("Enter a number up to which you want to find the average"))

i = 1

sum = 0
while i <=number:

    sum = sum + i

    i+=1

average = sum/number
print(f”The sum of {number} natural numbers is {sum}”)

print(f"The average of {number} natural numbers is {average}")
```

Example 3: Program to Find the GCD of Two Positive Numbers by using repeatative subtraction

```
m = int(input("Enter first positive number"))

n = int(input("Enter second positive number"))

if m == 0 and n == 0:

    print("Invalid Input")

if m == 0:

    print(f"GCD is {n}")

if n == 0:

    print(f"GCD is {m}")
```

```
while m != n:
        if m > n:
              m = m – nif
        n > m:
              n = n – m
print(f"GCD of two numbers is {m}")
```

Example 4: Python Program to Find the Sum of Digits in a Number

```
number = int(input('Enter a number'))
sum = 0
remainder = 0

while number !=0:
        remainder = number % 10
        result = result + remainder
        number = int(number //10)
print(f"The sum of all digits is {result}")
```

While loop with else

- while loops can have an optional else block.
- The else part is executed if the condition in the while loop evaluates to False.

'''Example to illustrate the use of else statementwith the while loop'''

```
counter = 0
while counter < 3:
   print("Inside loop")
   counter = counter + 1
else:
   print("Inside else")
```

for loops in python

- The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects.
- Iterating over a sequence is called traversal.

  Syntax: for iteration_variable in sequence:

  loop body

- The for loop starts with for keyword and ends with a colon.
- The body of for loop is separated from the rest of the code using indentation.
- The first item in the sequence gets assigned to the iteration variable iteration_variable.Here, iteration_variable can be any valid variable name.
- Then the statement block is executed.
- This process of assigning items from the sequence to the iteration_variable and then executing the statement continues until all the items in the sequence are completed.
- Loop continues until we reach the last item in the sequence.

range() function:

- The range() function is used in conjuction with for loop that generates a sequence of numbers which can be iterated through using for loop.
- The syntax for range() function is,
- *Syntax: range([start ,] stop [, step])*
  - Both start and step arguments are optional and the range argument value should always be an integer.
  - start → value indicates the beginning of the sequence. If the start argument is not specified, then the sequence of numbers start from zero by default.
  - stop → Generates numbers up to this value but not including the number itself.
  - step → indicates the difference between every two consecutive numbers in thesequence. The step value can be both negative and positive but not zero.

Example 1:Program to Demonstrate for Loop Using range() Function

```
print("Only "stop" argument value specified in range function")

for i in range(3):
                print(f"{i}")
print("Both "start" and "stop"argument values specified in range function")

for i in range(2, 5):
        print(f"{i}")
print("All three arguments "start", "stop" and "step" specified in range function")

        for i in range(1, 6, 3):
           print(f"{i}")
```

Example 2:Program to Iterate through Each Character in the String Using for Loop

```
for each_character in "Athani":
            print(f"Iterate through character {each_character} in the string 'Athani' ")
```

```
OUTPUT:
 Iterate through character A in the string 'Athani'
 Iterate through character t in the string 'Athani' Iterate
 through character h in the string 'Athani' Iterate
 through character a in the string 'Athani' Iterate
 through character n in the string 'Athani' Iterate
 through character i in the string 'Athani'
```

Example 3: Program to Find the Sum of All Odd and Even Numbers up to a Number Specifiedby the User.

```
number = int(input("Enter a number"))

even = 0
odd = 0
for i in range(number):
if i % 2 == 0:
        even = even + i
    else:
        odd = odd + i
print(f"  Even numbers are {even} and Odd numbers are {odd}")
```

Example 4: Program to Find the Factorial of a Number

```
number = int(input('Enter a number'))

factorial = 1
if number < 0:
    print("Factorial doesn't exist for negative numbers")
```

```
elif number == 0:
      print('The factorial of 0 is 1')
else:
      for i in range(1, number + 1):
          factorial = factorial * i
      print(f"The factorial of number {number} is {factorial}")
```

for loop with else

- A for loop can have an optional else block as well.
- The else part is executed if the items in the sequence used in for loop exhausts.
- Example:

```
for i in range(3):
   print(i)
else:
  print("Inside else")
```

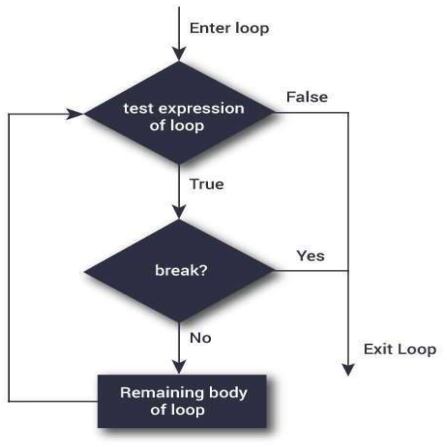
Controlling loop execution: Break, continue statements:

- In Python, break and continue statements can alter the flow of a normal loop.
- Loops iterate over a block of code until the test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression. The break and continue statements are used in these cases.
- The break and continue statements provide greater control over the execution of code in a loop.

break statement

- The break statement terminates the loop containing it. Control of the program flows tothe statement immediately after the body of the loop.
- If the break statement is inside a nested loop (loop inside another loop), the breakstatement will terminate the innermost loop.
- Syntax of break
  Break

Flowchart



- Example:

```
# Use of break statement inside the
loopfor val in "laxmi":
    if val == "x":
        break
    print(val)
print("The end")
```

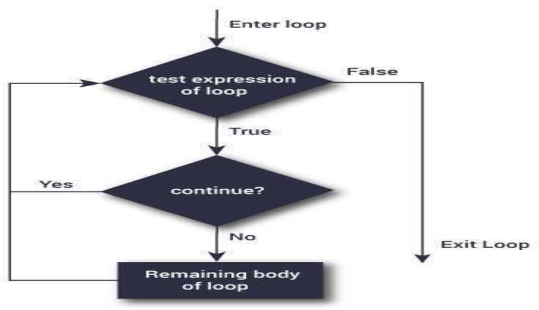
```
Output:
  l
  a
The end
```

continue statement

- The continue statement is used to skip the rest of the code inside a loop for thecurrent iteration only.
- Loop does not terminate but continues on with the next iteration.
- Syntax:

```
Continue
```

- Flowchart:



Example :

```
for val in "laxmi":if
   val == "m":
      continue #skip the current iteration
   print(val)
print("The end")
```

```
Output:
 l
 a
 x
 i
The end
```

Nested loops:

 Python programming language allows the use of one loop inside another loop.

 Syntax for a nested for loop statement in Python programming language is as follows

```
for var in sequence:
    for var in sequence:
        statements(s)
    statements(s)
```

 Syntax for a nested while loop statement in Python programming language is as follows

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Ex:

```
for i in range(1,10):
    for j in range(1,10):
        print (i*j)
```

pass statement:

- The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.
- It is a null operation, nothing happens when it is executed.
- It acts as a placeholder with empty body.
- It is used in both loops and functions.
- Syntax for a pass statement in Python is as follows

```
pass
```

Ex1:

```
for letter in 'python':
    pass
print (letter)
```