Flow of Control

Flow of Control refers to the order in which statements are executed in a program.

Sequential Flow

The default control flow in a program is sequential flow, in which statements are executed line-by-line one after the other in a sequence in which they are written.

Example

```
x = 6
y = 7
z = y - x
print(z)
```

Conditional Flow

Conditional flow refers to execution of certain statements only if a specific condition is met. This is accomplished by the use of conditional statements such as if, if-else, and if-elif-else.

Example

```
num = int(input("Enter a number : "))
if(num>5):
     print("Number is greater than 5")
else:
     print("Number is less than 5")
```

Output
Enter a number : 55
Number is greater than 5

Iterative Flow

Iteration means 'repetition'.
Iterative flow repeats statements in a block of code. Repetition or looping can be performed a fixed number of times or until a certain condition is met. This is accomplished through the use of iterative statements: **for** and **while**.

Example-1

```
name = input("Enter your name : ")
for x in range(5): # range function creates a sequence of integers from 0 to 4
    print("Hello", name)
```

Example-2

```
name = input("Enter your name : ")
i=1
while i<=5:
     print("Hello", name)
     i +=1
```

Conditional Statements

Conditional statements execute specific blocks of code based on certain conditions. In Python, conditional statements are implemented using the keywords if, if-else, and if-elif-else.

Terminology

Indentation

Indentation refers to the spaces at the beginning of a line.

Example

```
if a<5:
     print(a)
     print('Inner Block')
print('Outside block')
```

Block of code

A block of code is a set of statements that are grouped together and executed as a single unit. A block of code is identified by the indentation of the lines of code.

if statement

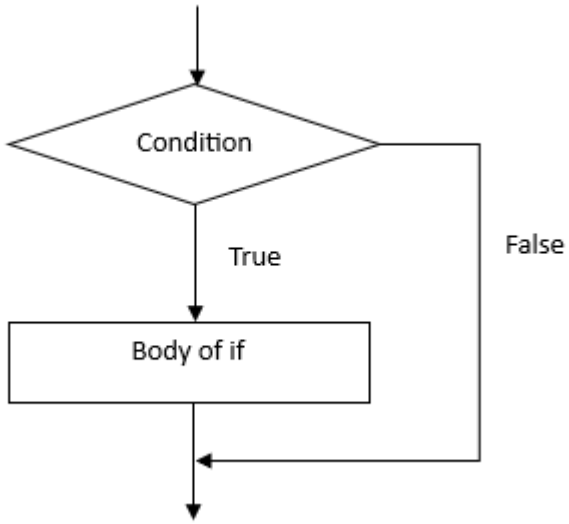
The if statement is used to execute a block of code only if a certain condition is true.

Syntax:

if <condition>:

    Set of Statements

Flow Chart of if-statement



Example-

```
age = int(input("Enter your age : "))
if age>=18:
     print('Congratulations')
     print('You are allowed to vote')
```

Program

Write a python Program to find the absolute value of the number entered by the user.

```
num = int(input("Enter any number : "))
if num>=0:
     abs_num = num
else:
     abs_num= -num
print(abs_num)
```

Output

Enter any number : -5
5

Program

Write a python Program to sort three numbers entered by a user.

```
a = float(input("Enter the first number : "))
b = float(input("Enter the second number : "))
c = float(input("Enter the third number : "))
if a > b:
     a,b = b,a
if a > c:
     a,c = c,a
if b > c:
     b,c = c,b
print (a, "<", b, "<", c)
```

Output

Enter the first number : 30
Enter the second number : 25
Enter the third number : 20
20.0 < 25.0 < 30.0

if-else statement

The if-else statement is used to execute one block of code if a certain condition is true, and another block of code if the condition is false.

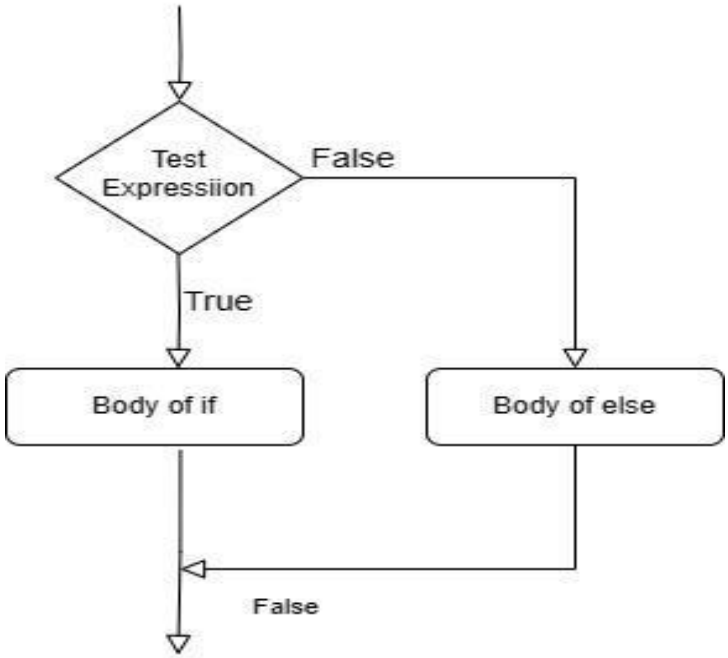
Syntax of if-else

if <condition>:
    set of statements
else:
    set of statements

Example

```
age = int(input("Enter your age : "))
if age>=18:
     print('Congratulations')
     print('You are allowed to vote')
else:
     print('You are not allowed to vote')
print('Thanks')
```

Flowchart of if-else statement



Program

Write a python Program to check if a number entered by a user is divisible by 3.

```
num=int(input("Enter a number : "))
if num % 3 == 0:
     print(num,"is divisible by 3")
else:
     print(num,"is not divisible by 3")
```

Output

Enter a number : 601
601 is not divisible by 3

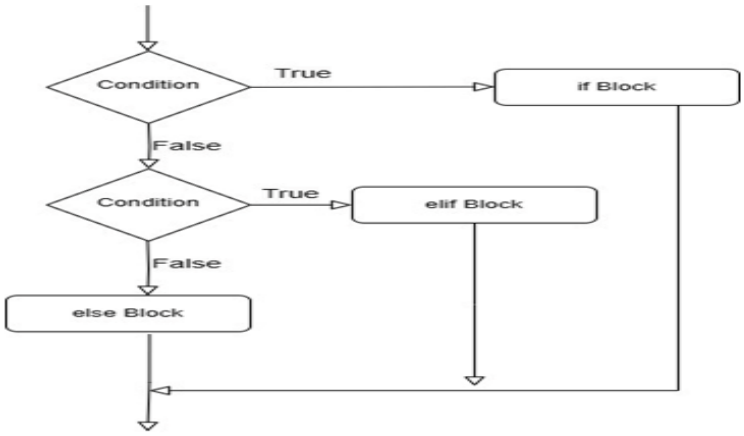
if-elif-else statement

if-elif-else statement is used to check multiple conditions.

Program

```
per = int(input("Enter Percentage : "))
if per >= 75:
     print("Distinction")
elif per >= 60:
     print("Grade-A")
elif per >= 50:
     print("Grade-B")
elif per >= 40:
     print("Grade-C")
else:
     print("Grade-D")
```

Flowchart of if-elif-else statement



Iterative Statement

Iterative statements execute a set of instructions multiple times. '**for**' and '**while**' loops are the iterative statements in Python.

while Loop

The while loop repeatedly executes a block of code as long as the specified condition is true.

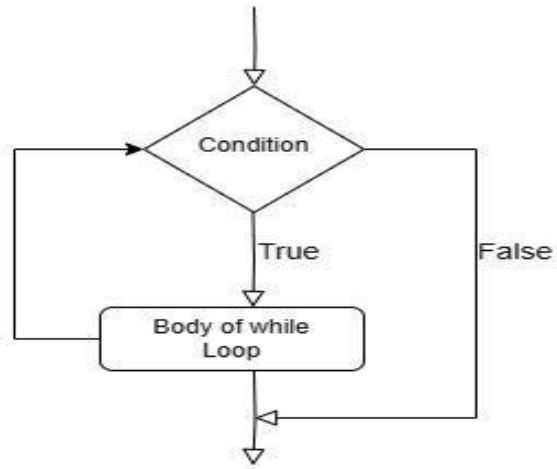
The while loop is an exit controlled loop, i.e. the user is responsible for exiting the loop by changing the value of the condition to False. While loop may run infinitely if the condition remains true.

Syntax:

while (condition):

    block of statements

Flowchart of while loop



Example

Write a Python Program (using a while loop) to Find the Sum of First 10 Natural Numbers.

```
num=1
sum=0
while (num <= 10):
      sum = sum + num
      num = num+1
print ("Sum of Natural Numbers : ", sum)
```

Example

Write a Python Program (using the while loop) to Find the Sum of First N Natural Numbers where N is entered by the user.

```
num=1
sum=0
n = int(input("Enter the value of n : "))
while (num <= n):
      sum = sum + num
      num = num+1
print ("Sum of Natural Numbers : ", sum)
```

range( ) Function

range( ) is a built-in function that returns a sequence of numbers.

Syntax

range(start, stop, step)

start: The starting value of the sequence (inclusive). If not specified, it defaults to 0.

stop: The ending value of the sequence (exclusive).

step: The difference between two consecutive elements. Its default value is 1.

The range( ) function can be used in for loops to iterate over a sequence of numbers.

Case-1 x = list(range(5)) print(x) **Output** [0, 1, 2, 3, 4]	**Case-2** x = list(range(3,6)) print(x) **Output** [3, 4, 5]	**Case-3** x = list(range(3,20,2)) print(x) **Output** [3, 5, 7, 9, 11, 13, 15, 17, 19]	**Case-4** x = list(range(0, -9, -1)) print(x) **Output** [0, -1, -2, -3, -4, -5, -6, -7, -8]

for Loop

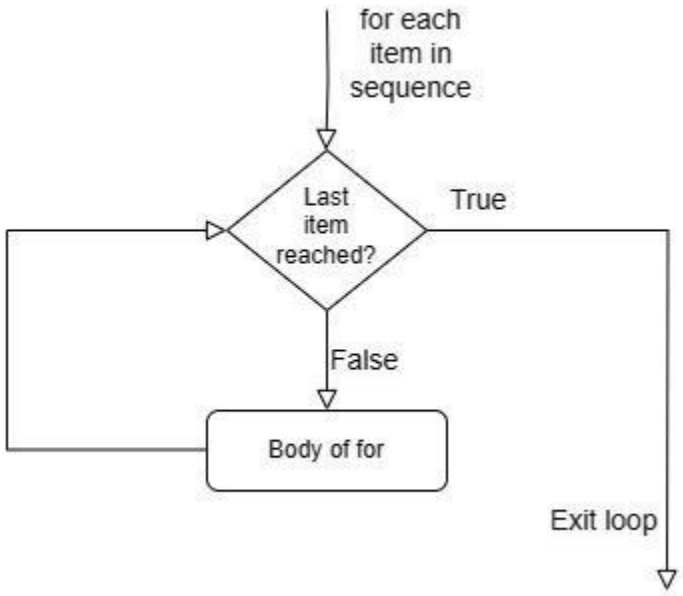
The for loop is used to iterate over a sequence (such as a list, tuple, string, etc.) and execute a block of code for each item in the sequence.

Syntax

for <control_variable> in <sequence>:

    Block of code

Flowchart of for loop



Example-1

```
list1 = [1,2,3,4,5,6,7,8,9,10]
for var1 in list1:
      print(var1)
```

Example-2

```
str1 = 'India'
for each_character in str1:
      print(each_character)
```

Program-1

Write a Python Program (using for loop) to Find the Sum of First 10 Natural Numbers.

```
sum=0
for num in range(1,11):
      sum = sum + num
print ("Sum of Natural Numbers : ", sum)
```

Output:

Sum of Natural Numbers : 55

Program-2

Write a Python Program (using for loop) to Find the Sum of First N Natural Numbers where N is entered by the user.

```
sum=0
n = int(input("Enter the value of n : "))
for num in range(1, n+1):
      sum = sum + num
print ("Sum of Natural Numbers : ", sum)
```

Output:

Enter the value of n : 20

Sum of Natural Numbers : 210

Program-3

Write a Python Program (using for loop) to find the factorial of a number.

```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
      print("Sorry, factorial does not exist for negative numbers")
else:
      for i in range(1, num + 1):
factorial *= i
print("The factorial of", num, "is", factorial)
```

Output

Enter a number: 5

The factorial of 5 is 120

Break and Continue Statement

Break Statement: The break statement is used to terminate a loop immediately. It is typically used with conditional statements.

Continue Statement: The continue statement skips all the remaining statements in the current iteration of the loop and moves the control to the beginning of the next iteration.

Example :

```
fruits = ['apple', 'banana', 'cherry']
for x in fruits:
      if x == 'banana':
            break
      print(x)
```

Example :

Write a program in Python to check if a number entered by a user is a prime number or not.

```
num = int(input("Enter a number: "))
flag=False
if num > 1:
      for i in range(2, num):
      if (num % i) == 0:
            flag=True
            break
      if flag==True:
            print(num, "is not a prime number")
      else:
            print(num, "is a prime number")
```

Example :

Write a program in Python to print all natural numbers from 1 to 10 except 7.

```
for i in range(1,11):
      if i==7:
            continue
      print(i)
```

Example :

Write a program in Python to print all natural numbers between 1 and 50 (both inclusive) which are not multiple of 3.

```
for x in range(1, 51):
      if x % 3 ==0 :
            continue
      print(x)
```

Nested loops

Nested loops refers to a loop within a loop.

Example: Generating a pattern

```
n = int(input("Enter the number of rows: "))
for i in range(n):
      for j in range(i+1):
            print("*", end="")
      print()
```

Output

Enter the number of rows: 5

*

**

***

****

*****