**Loops** are used to execute a set of statements repeatedly until a particular condition is satisfied

**1. do-while loop**

the statements inside the loop execute and then the condition gets evaluated, if the condition returns true then the control gets transferred to the "do " else it jumps to the next statement after do while

Syntax:

do

{

statements;

}while (condition)

**2. while loop**

the condition is first evaluated and if it returns true then the statements inside while loop execute,if the condition returns false the control comes out of loop and jumps to the next statement after while loop

Syntax:

while( condition)

{

statements;

}

**Note:** The important point to when using **do-while loop** and **while loop** we **need to use increment or decrement statements inside the loop**, so that the loop variable gets changed on ach iteration and at some point condition returns false, this way we can end the execute of the loop otherwise the loop would execute indefinitely.

**3. for loop**

As program executes, the interpreter always keep track of which statement is about to be executes We call this the control flow of the program

Step 1 : In for loop, initialization happens first and only one time, initialization part of for loop only executes once

Step 2 : Condition in for loop is evaluated on each iteration, if condition is true then the statements inside the for body gets executes. once the condition returns false, the statements in for loop does not execute and control gets transferred to the next statement in the program after for loop.

Step 3 : After every execution of for loop's body, the increment/decrement part of for loop executes that updates the counter.

Step 4 : After third step, the control jumps to second step and condition is re-evaluated.

Syntax

for ( initialization; condition ; increment/decrement)

{

statements;

}

**4. Enhanced for loop**

enhanced for loop is useful when we want to iterate Array/Collections. it is easy to write and understand

Syntax

for ( data\_type : array\_name)

{

statements;

}

**Continue :** is mostly used inside loops. whenever it is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration skipping of the statements inside loops' body for the current iteration.

this particularly useful when we want to continue the loop but do not want the rest of the statements in loop body to execute for the particular iteration,

**break :**

a) use break to come out the loop instantly, whenever a break statement is encountered inside loop, the control come out of loop and loop gets terminated for rest of the iteration.

b) we use break in switch case control