Loops

The Loops are used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming. Or Loops are very useful when the number of lines needs to execute more than one time.

There are 4 types of loops in PHP Language

1. **For** loop
2. **While** loop
3. **do while** loop
4. **For** each loop

1. for Loop

for loop is used to execute a set of statements repeatedly until **for – loops** through a block of code a specified number of times. We can say it an open ended loop**.** It is usually preferable when the number of iterations are known as **for** loop.

Syntax:

```
for( initialization;  test-expression;  update-expression)
{
        Statement(s);
}
```

Flowchart

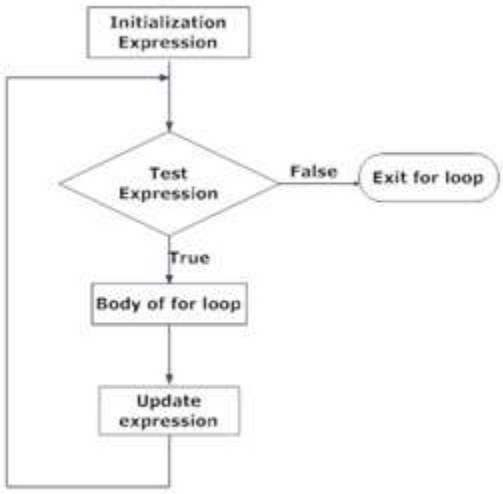


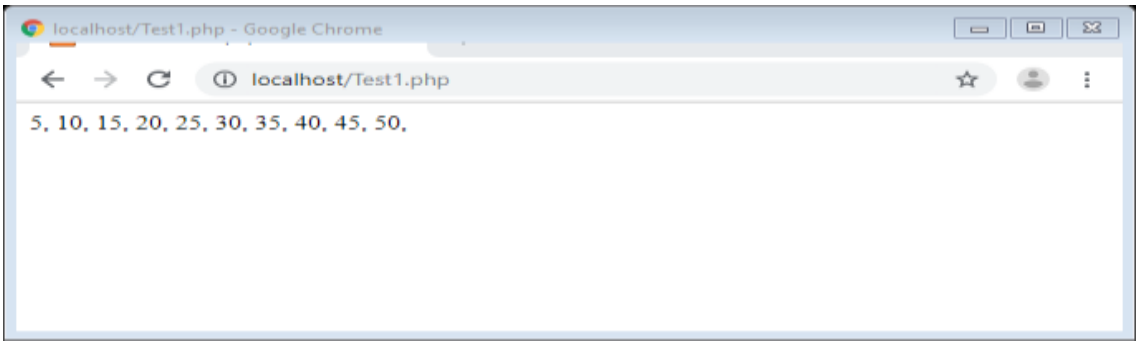
Figure: Flowchart of for loop

Explanation:

The initial expression is initialized only once at the beginning of the 'for' loop. Then, the condition is checked by the program. If the condition is false, **for** loop is terminated. But, if condition is true then, the codes are executed and update expression is updated. Again, the condition is checked. If it is false, loop is terminated and if it is true, the same process repeats until condition is false.

Example

```
<?php
   for ( $a=5; $a<=50; $a+=5)          // if a is less than and equals to 50
        echo "$a,  ";                  // then it will print value of a, then update value by 5
?>
```



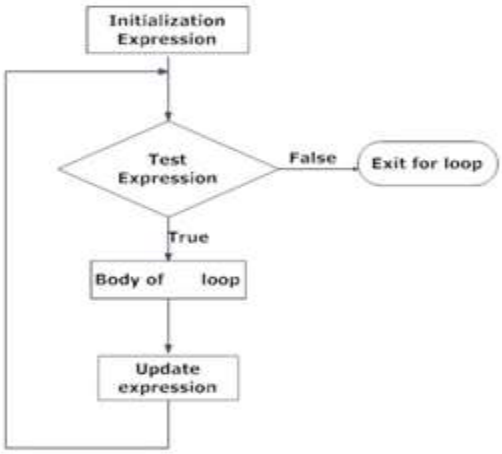
2. while loop

while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The condition is evaluated, and if the condition is true, the code within the block is executed first. This repeats until the condition becomes false. Because **while** loop checks the condition before the block is executed, it is also known as a **pre-test loop**. Compare with the do **while** loop, which tests the condition after the loop has executed.

Syntax:

```
Initialization_exp;
while (condition_Exp)
  {
     Statement(s);
     Update_Exp;
   }
```

Flow Chart

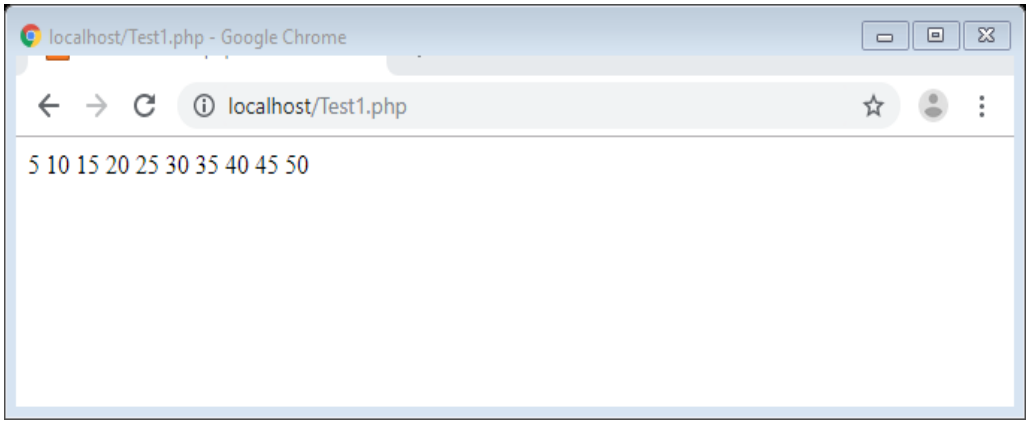


Explanation:

first checks whether A is TRUE, which it is, so then the {loop body} is entered, where it executes statement(s) that are in the body of loop, after execution of every statement once it checks the condition again if the condition remains TRUE it will execute the body again . Process repeating until the A remains true.

Example:

```
<?php
    $a=5;
    while ( $a<=50)          // if a is less than and equals to 50
    {    echo " $a  ";       // then it will print value of a, then update value by 5
         $a += 5;
    }
?>
```



Do...while loop

In PHP, do...**while** loop is very similar to **while** loop. Only difference between these two loops is that, in **while** loops, condition is checked at first but, in do...**while** loop code is executed at first then the condition is checked. So, the code is executed at least once in do...**while** loops that's why it is also called ***Post Test*** loop.

Syntax:

```
Initialization;
do
{
    Statement(s);
    Update value;
} while (condition);
```

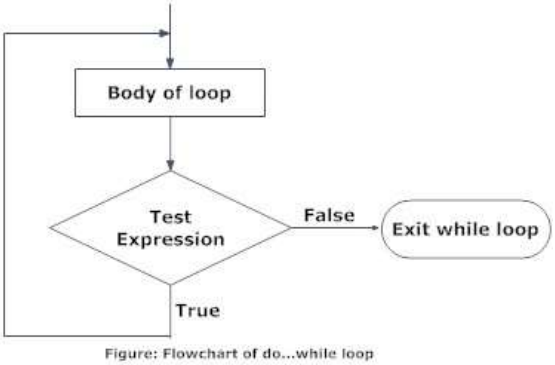


Figure: Flowchart of do...while loop

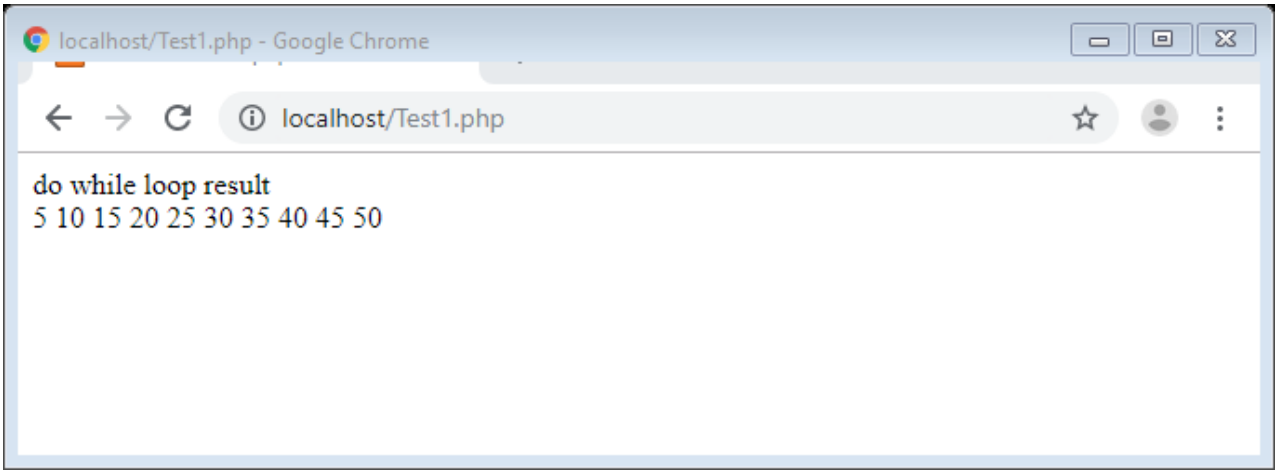
Notice, there is semicolon in the end of **while** (); in do...**while** loop.

Explanation:

At first codes inside body of do is executed. Then, the test expression is checked. If it is true, code/s inside body of do are executed again and the process continues until test expression becomes false (zero).

Example:

```
<?php
  echo "dowhile loop result <br>";
   $a=5;
    do
   {    echo  " $a ";       // then it will print value of a, then update value by 5
        $a  +=5;            //  increment or update by 5 in variable $a
    } while ($a<=50);    // if a is less than and equals to 50
?>
```



3. *foreach* loop

The *foreach* construct provides an easy way to iterate over arrays. *foreach* works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable.

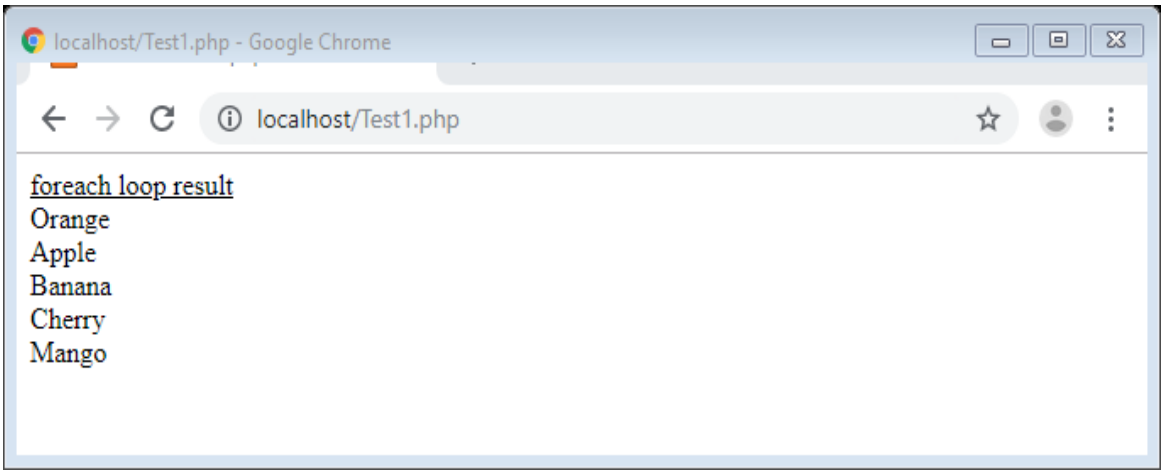
Syntax-1

```
foreach (array_expression as $value)
    statement;
```

The first form loops over the array given by array expression

Example-1

```
<?php
    echo "<u>foreach loop result </u><br>";
    $FruitsList = array ("Orange", "Apple", "Banana",   "Cherry", " Mango");
    foreach ( $FruitsList as $value )
   {
     echo  "$value<br />";
   }
?>
```



Syntax-2:

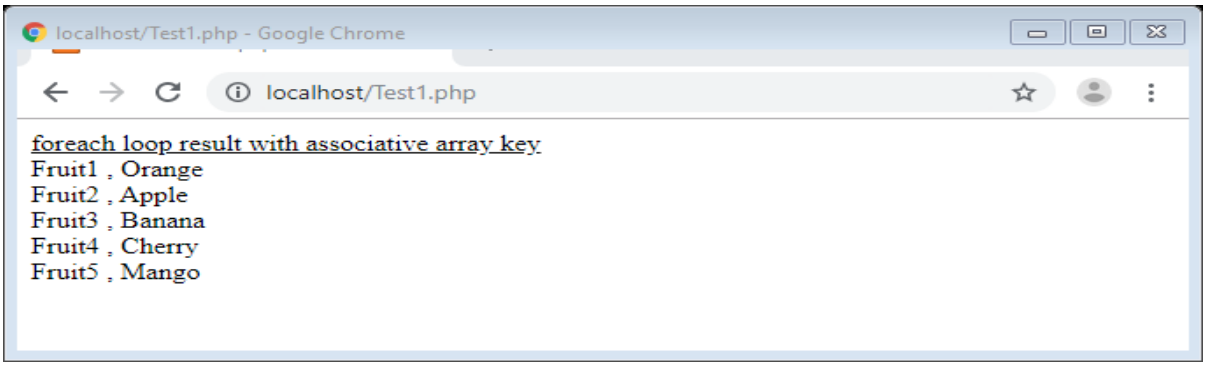
```
foreach (array_expression as $key => $value)
    statement;
```

Associative array

The second form will additionally assign the current element's key to the $key variable on each iteration. $key and $values variables can be named anything, $key will be equal to array's key, and $value will be equal to that key's value.

Example

```
<?php
   echo "<u>foreach loop result with associative array key </u><br>";
   $FruitsList = array ("Fruit1" => "Orange","Fruit2"=>"Apple", "Fruit3"=>"Banana", "Fruit4"=>
                        "Cherry", "Fruit5"=>"Mango");
  foreach ( $FruitsList as $key=> $value )
  {
   echo "$key , $value<br />";
  }
?>
```



➢ Nested Loop

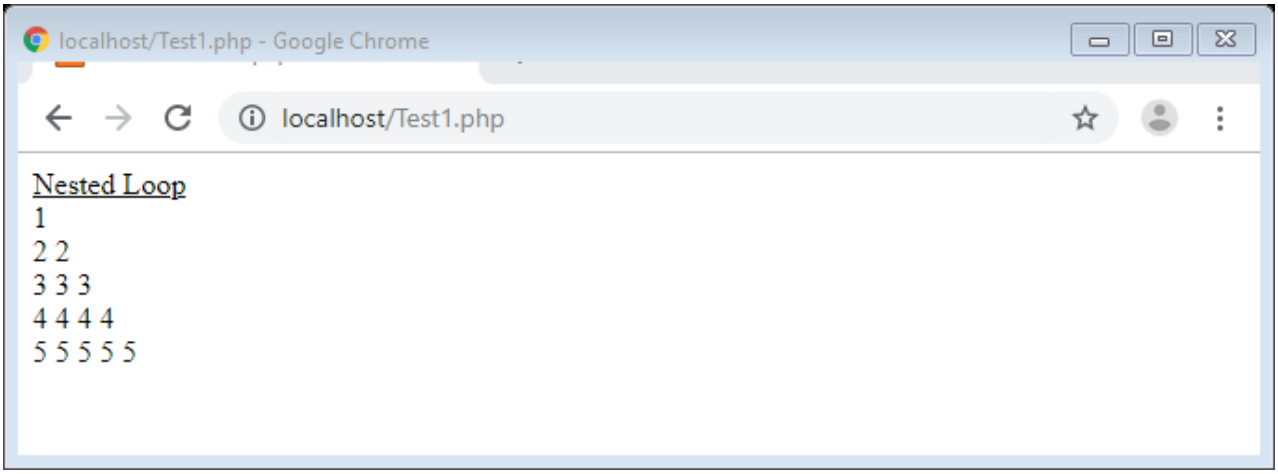
Like nested if statement, if loop uses within a loop then this terminology is called **"Nested loop"**

Syntax:

```
for (initialization; condition; increment/decrement)      // Outer Loop
{                                                         // Outer loop start here
    for (initialization; condition; increment/decrement)  //  Inner Loop
    {                                                     // Inner loop start here
        statement;                                        // execute this statement
    }
}
```

Example:

```
<?php
        echo "<u>Nested Loop</u> <br> ";
        for ($a=1; $a<=5;  $a++) {
                for ($b=1; $b<=$a; $b++) {
                        echo "$a ";
                }
              echo "<br>";
        }
?>
```

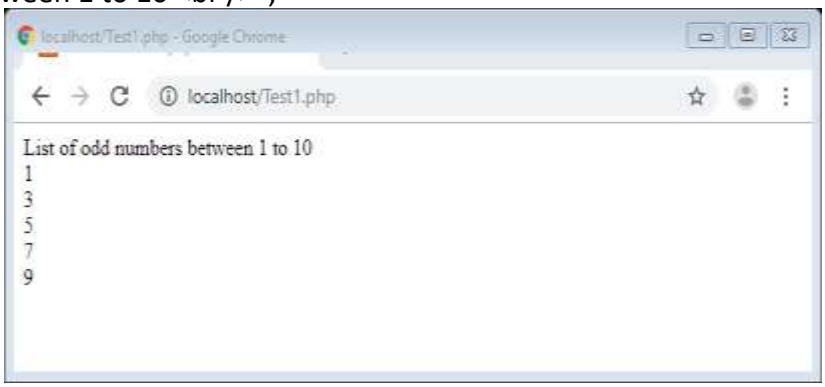


The ***Continue*** keyword.

Sometimes a situation arises where we want to take the control to the beginning of the loop (**for** example **for**, **while**, do **while** etc.) skipping the rest statements inside the loop which have not yet been executed. The keywords ***continue*** allow us to do this. When the keywords ***continue*** executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.

In the following example, the lists of odd numbers between 1 to 10 have printed. In the **while** loop we test the remainder (here $x % 2) of every number, if the remainder is 0 then it becomes an even number and to avoid printing of even numbers continue statement is immediately used and the control passes to the beginning of the loop.

```
<?php
        $x=1;
        echo "List of odd numbers between 1 to 10 <br />";
        while ($x<=10) {
            if (($x % 2)==0) {
                $x++;
                continue;
            }else{
                echo $x. "<br />";
              $x++;
            }
    } ?>
```



Exercise

Theory Questions

1. Write down the syntax of Nested **while** and Nested do **while** loops.
2. Describe the purpose of a "counter" variable when executing a loop.
3. Which type of post-test loop and pre-test loop?
4. Which type of looping structure is used to iterate through elements of array?

Practical Questions

1. Write a program to print table of a number entered by user using ***while*** loop and do **while** loop.
2. Write a program to print following output using nested **for** loop

```
1
12
123
1234
12345
```

3. Write down the code to display series from 0 to 5 and their sum numbers, using loop of your choice.

Count	Total
0	0
1	1
2	3
3	6
4	10
5	15

4. Write down the code to display following output

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

5. Write a program series of prime number from 1 to 25.

Objective MCQ's:

1. Each repetition of a looping statement is called a(n) _____
   a) Recurrence
   b) Iteration
   c) Duplication
   d) Re-execution

2. Which of the following is the correct syntax for a **for** loop statement.
   a)
```
for($i=0; $i=<10; $i++)
   Echo "display from a for statement ";
```
   b)
```
for($i=0, $i=<10, $i++)
   echo "display from a for statement ";
```
   c)
```
for($i=0; $i=<10);
   echo "display from a for statement ";
```
   d)
```
for
{
   echo "display from a for statement ";
}while ($i=0; $i<10; $i++);
```

3. Counter variable _____.
   a) Can only be increment.
   b) Can only be decrement
   c) Can be incremented or decremented
   d) Do not change

4. Which of the following is the correct syntax for **while** statement?
   a)
```
while ($i <10 , $i++){
   echo "<b> $i </b> ";
}
```
   b)
```
while ($i <10){
   echo "<b> $i </b> ";
  $i++;
}
```
   c)
```
while ($i <10 , $i++)
   echo "<b> $i </b> ";
  $i++;
```
   d)
```
while ($i <10 , $i++); {
   echo "<b> $i </b> ";
}
```

5. These keywords are used in the loops, if depend on requirement.
   a) continue
   b) break;
   c) a and b, both
   d) case