Unit 1 :

Introduction to PHP: Introduction to PHP, History and Features of PHP, Installation & Configuration of PHP, Embedding PHP code in Your Web Pages, Understanding PHP, HTML and White Space, Comments in PHP, Sending Data to the Web Browser, Data types, Keywords, Variables, Constants in PHP, Expressions in PHP, Operators in PHP. **Conditional statements:** if, if-else, switch, The? Operator**,** **Looping statements:** while Loop, do-while Loop, for Loop, foreach loop, break, continue.

FLOW CONTROL FUNCTIONS IN PHP

PHP supports a number of traditional programming constructs for controlling the flow of execution of a program. Control statements are conditional statements that execute a block of statements if the condition is correct. The statement inside the conditional block will not execute until the condition is satisfied.

Control sequence allows user to execute block of code depending on conditions. It control the flow of code, order in which the various statement are execute how many times a particular code should execute when the code block will ended.

There are 3 types of control structure, they are:

- Conditional / decision making statements
- Looping statements
- Jumping statements

Conditional statements: it is used to perform different actions based on different conditions it breaks based on different condition it breaks the sequential flow of programs and jumps to the different code based on the condition of input value.

- If statement
- If else statement
- If else ladder statement
- Switch statement

Looping statements

They executes a block of code for specific times until the condition becomes false. It helps to execute the code again and again. It avoids rewriting the code.

There are 4 loop statements:

- While
- Do-while
- For
- Foreach

Jumping statements: it is used to transfer the control from one point to another point in the program.

- Break
- Continue
- Goto

CONDITIONAL STATEMENTS: PHP provides us with five conditional statements:

1. if statement
2. if…else statement
3. if…elseif…else statement
4. switch statement

1. if Statement

The if statement is the simplest form of decision making. It executes a block of code if the specified condition evaluates to true. If the condition evaluates to false, the block of code is skipped.

Syntax:

```
if (condition){
   // if TRUE then execute this code
}
```

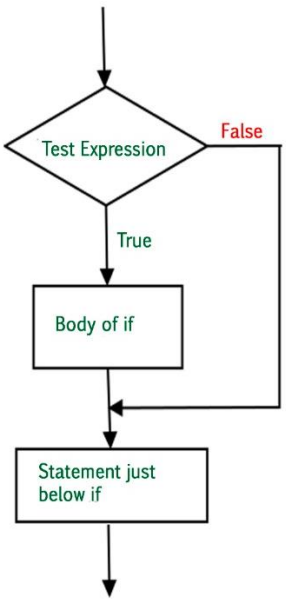
Example:

```
<?php
$x = 12;
if ($x > 0) {
   echo "The number is positive";
}
?>
```

Output

The number is positive

Flowchart



2. if...else Statement

The if-else statement is an extension of the if statement. It allows you to specify an alternative block of code that is executed when the condition is false. This is useful when there are two mutually exclusive conditions.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

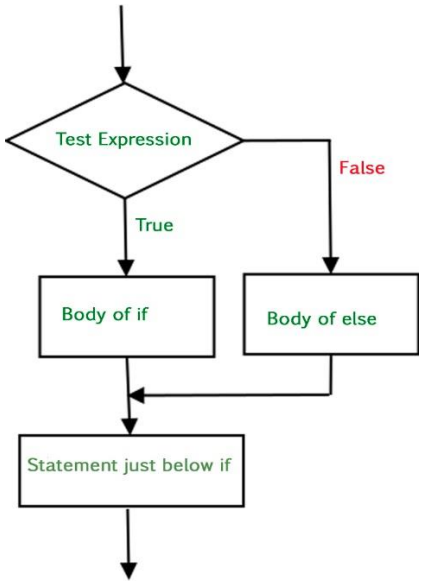
Example:

```
<?php
$x = -12;
if ($x > 0) {
   echo "The number is positive";
}
else{
   echo "The number is negative";
}
?>
```

Output

The number is negative

Flowchart



3. if...elseif...else Statement

In scenarios where you need to evaluate multiple conditions, you can use the if-elseif-else ladder. This construct allows you to check multiple conditions in sequence. The first condition that evaluates to true will execute its corresponding block of code, and all other conditions will be skipped.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
else {
   // if FALSE then execute this code
}
```

Example:

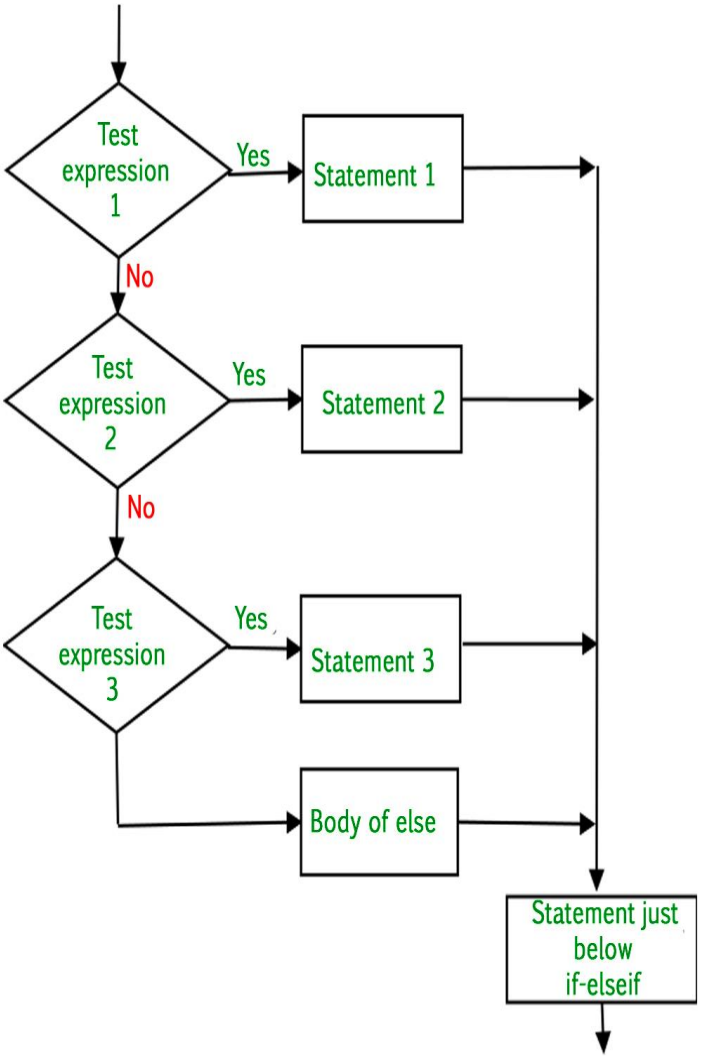
```
<?php
$x = "August";

if ($x == "January") {
   echo "Happy Republic Day";
}
elseif ($x == "August") {
   echo "Happy Independence Day!!!";
}
else{
   echo "Nothing to show";
}
?>
```

Output

Happy Independence Day!!!

Flowchart



4. switch Statement

The "switch" performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, **break** and **default**.

- The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
- The **default** statement contains the code that would execute if none of the cases match.

Syntax

```
switch (variable) {
  case value1:
    // Code to be executed if the variable equals value1
    break;
  case value2:
    // Code to be executed if the variable equals value2
    break;
  default:
```

```
    // Code to be executed if no cases match
}
```

Example:

```
<?php
$day = "Monday";
switch ($day) {
  case "Monday":
    echo "Start of the week!";
    break;
  case "Friday":
    echo "End of the week!";
    break;
  case "Saturday":
  case "Sunday":
    echo "Weekend!";
    break;
  default:
    echo "Mid-week!";
}
?>
```

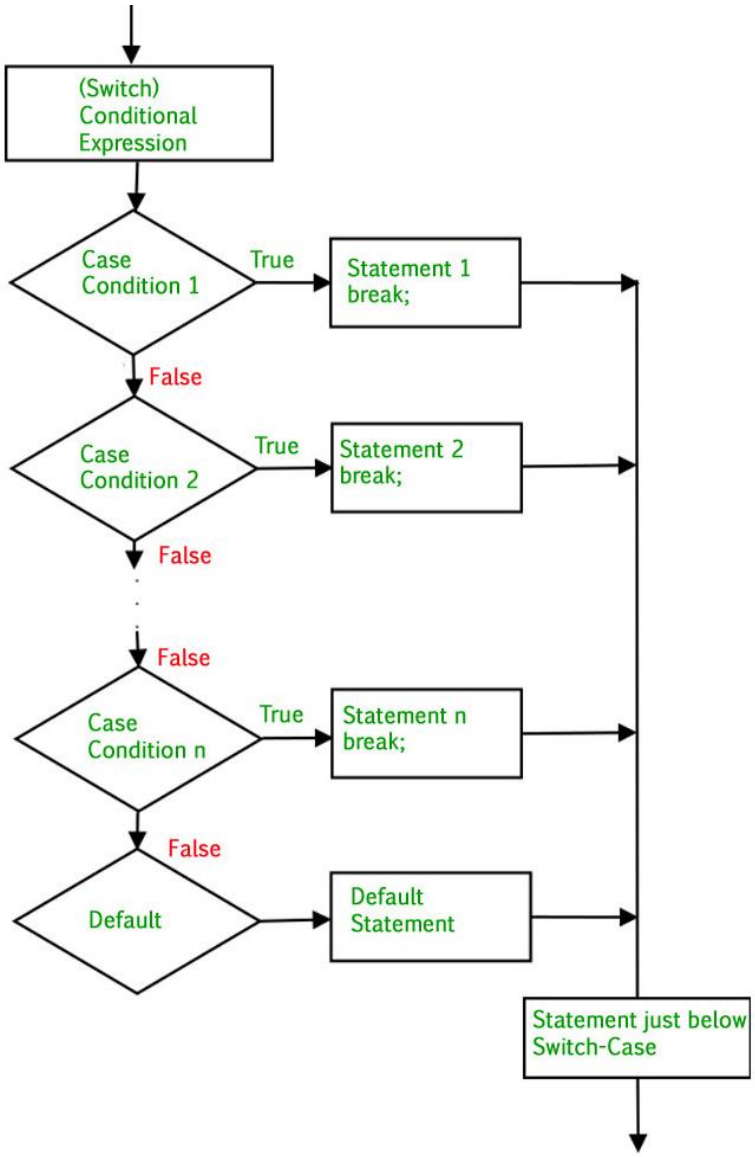
Output

Start of the week!

Note:

- *The switch statement compares the variable's value strictly (both type and value).*
- *If no case matches, the default block is executed (if provided).*
- *The break statement is crucial; without it, the program will continue to execute subsequent case blocks (a behavior known as "fall-through").*

Flowchart



Important points to be noticed about switch case:

1. The default is an optional statement. Even it is not important, that default must always be the last statement.

2. There can be only one default in a switch statement. More than one default may lead to a Fatal error.

3. Each case can have a break statement, which is used to terminate the sequence of statement.

4. The break statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.

5. PHP allows you to use number, character, string, as well as functions in switch expression.

6. Nesting of switch statements is allowed, but it makes the program more complex and less readable.

7. You can use semicolon (;) instead of colon (:). It will not generate any error.

5. Ternary Operators

PHP also provides a shorthand way to perform conditional checks using the ternary operator (?:). This operator allows you to write simple if-else conditions in a compact form.

Syntax

(condition) ? if TRUE execute this : otherwise execute this;

Example:

```
<?php
$age = 20;
echo ($age >= 18) ? "You are an adult." : "You are a minor.";
?>
```

Output

You are an adult.

LOOPING STATEMENTS:

In PHP, Loops are used to repeat a block of code multiple times based on a given condition. PHP provides several types of loops to handle different scenarios, including while loops, for loops, do...while loops, and foreach loops.

In this article, we will discuss the different types of loops in PHP, their syntax, and examples.

Types of Loops in PHP

Below are the following types of loops in PHP:

- for loop
- while loop
- do-while loop
- foreach loop

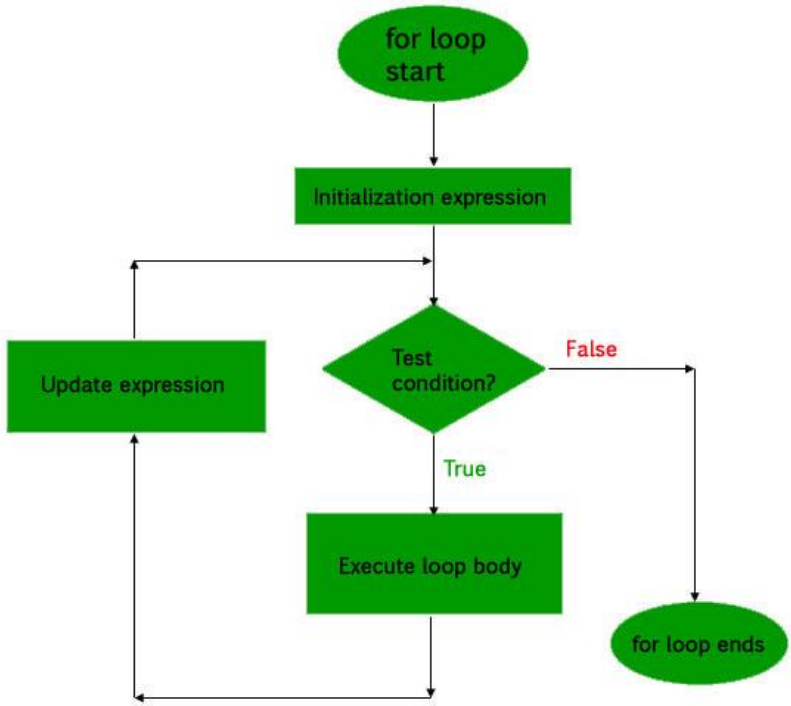
1. PHP for Loop

PHP for loop is used when you know exactly how many times you want to iterate through a block of code. It consists of three expressions:

- **Initialization:** Sets the initial value of the loop variable.
- **Condition:** Checks if the loop should continue.
- **Increment/Decrement:** Changes the loop variable after each iteration.

Syntax

```
for ( Initialization; Condition; Increment/Decrement ) {
   // Code to be executed
}
```



Example: Printing numbers from 1 to 5 using a for loop.

```
<?php


// Code to illustrate for loop
for ($num = 1; $num <= 5; $num += 1) {
   echo $num . "\n";
}
?>
```

Output

1

2

3

4

5

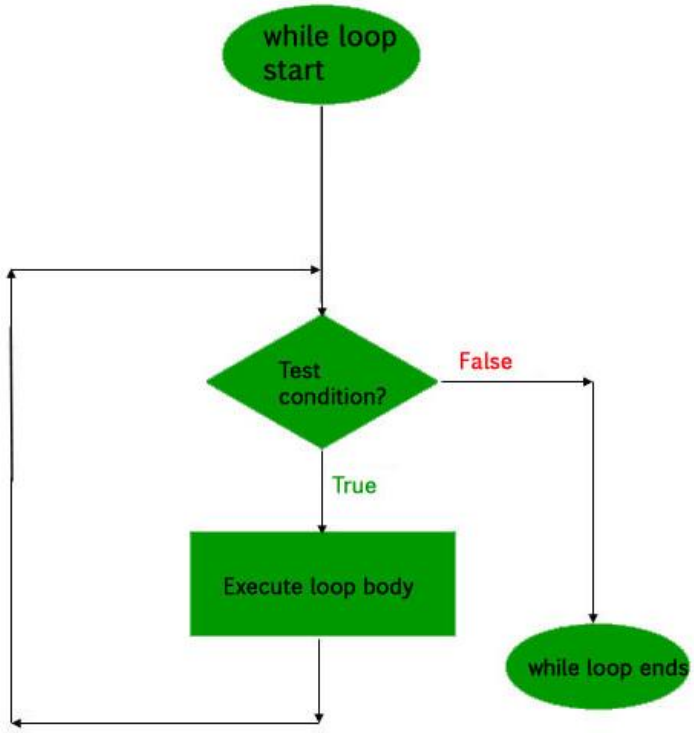
2. PHP while Loop

The while loop is also an entry control loop like for loops. It first checks the condition at the start of the loop, and if it's true then it enters into the loop and executes the block of statements and goes on executing it as long as the condition holds true.

Syntax

```
while ( condition ) {
```

```
// Code is executed
                                                    }
```



Example: Printing numbers from 1 to 5.

```
<?php
$num = 1;
while ($num <= 5) {
   echo $num . "\n";
   $num++;
}
?>
```

Output

1

2

3

4

5

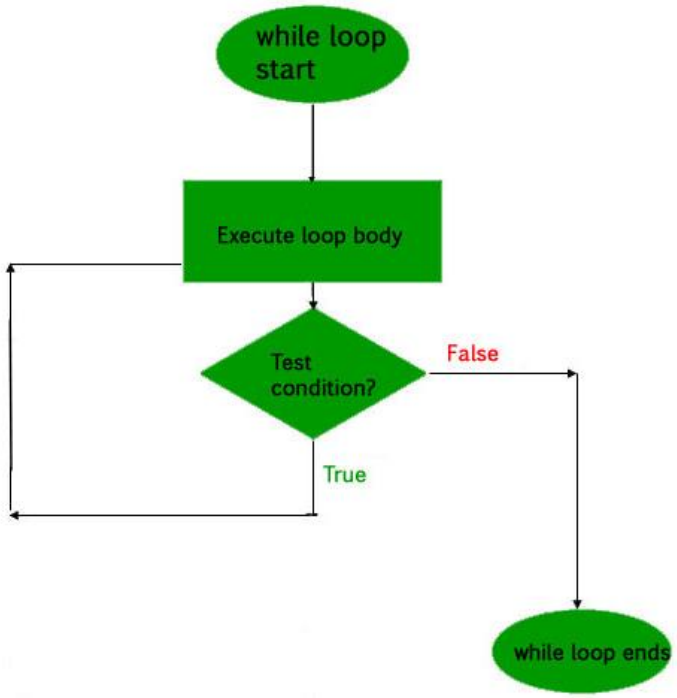
3. PHP do-while Loop

The do-while loop is an exit control loop, which means, it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once using

the do...while loop. After executing once, the program is executed as long as the condition holds true.

Syntax

```
do {
   // Code is executed
} while ( condition );
```



Example: Printing numbers from 1 to 5.

```
<?php
$num = 1;
do {
   echo $num . "\n";
   $num++;
} while ($num <= 5);
?>
```

Output

1

2

3

4

5

4. PHP foreach Loop

This foreach loop is used to iterate over arrays. For every counter of loop, an array element is assigned, and the next counter is shifted to the next element. It simplifies working with arrays and objects by automatically iterating through each element.

Syntax:

```
foreach ( $array as $value ) {
   // Code to be executed
}
or
foreach ($array as $key => $value) {
   // Code to be executed
}
```

- **$array:** The array to iterate over.
- **$value:** The current value of the array element during each iteration.

Example: Iterating through an array

```
<?php
// foreach loop over an array
$arr = array (10, 20, 30, 40, 50, 60);
foreach ($arr as $val) {
   echo $val . " ";
}
echo "\n";
// foreach loop over an array with keys
$ages = array(
   "Anjali" => 25,
   "Kriti" => 30,
   "Ayushi" => 22
);
foreach ($ages as $name => $age) {
   echo $name . " => " . $age . "\n";
}
?>
```

Output

```
10 20 30 40 50 60
```

Anjali => 25

Kriti => 30

Ayushi => 22

Why Use Loops?

Loops allow you to execute a block of code multiple times without rewriting the code. This is useful when working with repetitive tasks, such as:

- Iterating through arrays or data structures
- Acting a specific number of times
- Waiting for a condition to be met before proceeding

Jumping statements: it is used to transfer the control from one point to another point in the program.

- Break
- Continue
- Goto

Break: it is used to terminate the execution of a loop at that point itself. It immediately coming out of a loop.
Eg:

```
<?php
$i=0; while($i<5)
{
$i++; if($i==3) break;
echo $i;
}
?>
```

o/p:

1

2

Continue: it is used to stop the current iteration and starts the next iteration until condition becomes false.

Note: Always break and continue keywords are preceded by a conditional statement.

Ex:

```
<?php
$i=0; while($i<=5)
{
```

```
$i++; if($i==3) continue; echo $i;
}
?>
```

o/p:

1

2

4

5

Goto: it allows making an absolute jump to another point in the program.

goto transfer the control from one statement to other within the program by ignoring any type of nesting limitations.

A label must be a valid identifier followed by colon (:).

Syntax: Statement 1;

Goto labelname;

Statement2;

Labelname: statements;

Ex:

```
<?php
$i=0; while($i<=10)
{
$i++; if($i==10) goto stop; else
echo $i;
}
stop: echo " value is greater than 10 hence stopped" ;
?>
```

o/p;

1

2

3

4

5

6

7

8

9

Value is greater than 10 hence stopped