LOOPS

LOOPS

- Loops are used to execute the same block of code again and again, as long as a certain condition is met.

- The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort.

 - **while** — loops through a block of code as long as the condition specified evaluates to true.

 - **do…while** — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.

 - **for** — loops through a block of code until the counter reaches a specified number.

 - **foreach** — loops through a block of code for each element in an array.

PHP while Loop

- The while statement will loops through a block of code as long as the condition specified in the while statement evaluate to true.

```
<?php
$i = 1;
while($i <= 3){
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

OUTPUT:

```
The number is 2
The number is 3
The number is 4
```

PHP do…while Loop

- The do-while loop is a variant of while loop, which evaluates the condition at the end of each loop iteration.

- With a do-while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

PHP do…while Loop(contd.)

```
<?php
$i = 1;
do{
    $i++;
    echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

OUTPUT:

```
The number is 2
The number is 3
The number is 4
```

Difference between while and do-while loop

while Loop	do-while loop
The while loop is also named as entry control loop.	The do-while loop is also named as exit control loop.
The body of the loop does not execute if the condition is false.	The body of the loop executes at least once, even if the condition is false.
Condition checks first, and then block of statements executes.	Block of statements executes first and then condition checks.
This loop does not use a semicolon to terminate the loop.	Do-while loop use semicolon to terminate the loop.

PHP for Loop

- The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times
- The parameters of for loop have following meanings:

 - **initialization** — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.

 - **condition** — in the beginning of each iteration, condition is evaluated. If it evaluates to true, the loop continues and the nested statements are executed. If it evaluates to false, the execution of the loop ends.

 - **increment** — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

PHP for Loop(contd.)

```php
<?php
for($i=1; $i<=3; $i++){
    echo "The number is " . $i . "<br>";
}
?>
```

OUTPUT:

```
The number is 1
The number is 2
The number is 3
```

PHP foreach Loop

- The foreach loop is used to iterate over arrays.
- It is used to loop through each key/value pair in an array.

```
<?php
$colors = array("Red", "Green", "Blue");

// Loop through colors array
foreach($colors as $value){
    echo $value . "<br>";
}
?>
```

OUTPUT:

```
Red
Green
Blue
```

PHP foreach Loop(contd.)

```
<?php
$superhero = array(
    "name" => "Peter Parker",
    "email" => "peterparker@mail.com",
    "age" => 18
);

// Loop through superhero array
foreach($superhero as $key => $value){
    echo $key . " : " . $value . "<br>";
}
?>
```

OUTPUT:

```
name : Peter Parker
email : peterparker@mail.com
age : 18
```

PHP Break

- The break statement can also be used to jump out of a loop.
- PHP break statement breaks the execution of the current for, while, do-while, switch, and for-each loop.
- If you use break inside inner loop, it breaks the execution of inner loop only.

PHP Break: inside loop

```php
<?php
for($i=1;$i<=10;$i++){
echo "$i <br/>";
if($i==5){
break;
}
}
?>
```

OUTPUT:

```
1
2
3
4
5
```

PHP Break: inside loop(contd.)

- The PHP break statement breaks the execution of inner loop only.

```
<?php
for($i=1;$i<=3;$i++){
 for($j=1;$j<=3;$j++){
  echo "$i   $j<br/>";
  if($i==2 && $j==2){
   break;
  }
 }
}
?>
```

OUTPUT:

```
1 1
1 2
1 3
2 1
2 2
3 1
3 2
3 3
```

PHP Break: inside switch statement

```
<?php
$num=200;
switch($num){
case 100:
echo("number is equals to 100");
break;
case 200:
echo("number is equal to 200");
break;
case 50:
echo("number is equal to 300");
break;
default:
echo("number is not equal to 100, 200 or 500");
}
?>
```

OUTPUT:

```
number is equal to 200
```

PHP Break: with array of string

```
<?php
//declare an array of string
$number = array ("One", "Two", "Three", "Stop", "Four");
foreach ($number as $element) {
if ($element == "Stop") {
break;
}
echo "$element </br>";
}
?>
```

OUTPUT:

```
One
Two
Three
```

PHP Continue

- The PHP continue statement is used to continue the loop.
- It continues the current flow of the program and skips the remaining code at the specified condition.
- The continue statement is used within looping and switch control structure when you immediately jump to the next iteration.
- The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop.
- The continue statement allows the user to skip the execution of the code for the specified condition.

PHP Continue Example with for loop

```
<?php
    //outer loop
    for ($i =1; $i<=3; $i++) {
        //inner loop
        for ($j=1; $j<=3; $j++) {
            if (!($i == $j) ) {
                continue;       //skip when i and j does not have same values
            }
            echo $i.$j;
            echo "</br>";
        }
    }
?>
```

OUTPUT:

```
11
22
33
```

PHP continue Example in while loop

```
<?php
   //php program to demonstrate the use of continue statement

   echo "Even numbers between 1 to 20: </br>";
   $i = 1;
   while ($i<=20) {
      if ($i %2 == 1) {
         $i++;
         continue;   //here it will skip rest of statements
      }
      echo $i;
      echo "</br>";
      $i++;
   }
?>
```

```
OUTPUT:
Even numbers between 1 to 20:
2
4
6
8
10
12
14
16
18
20
```

PHP continue Example with array of string

```
<?php
    $number = array ("One", "Two", "Three", "Stop", "Four");
    foreach ($number as $element) {
        if ($element == "Stop") {
            continue;
        }
        echo "$element </br>";
    }
?>
```

OUTPUT:

```
One
Two
Three
Four
```