Unit-II
Lecture: 6
for, foreach Loops, break and continue

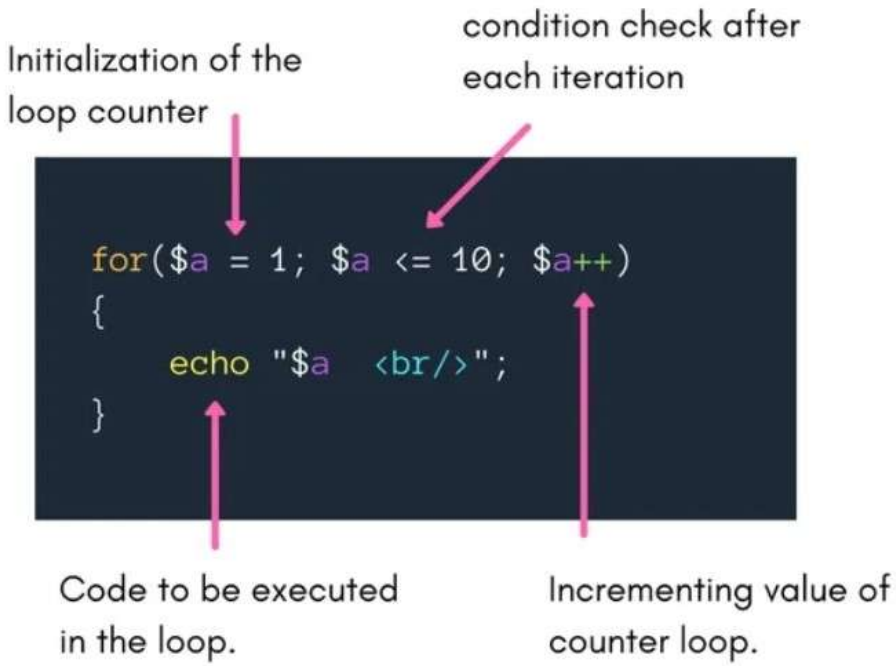
PHP for loop:
The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times.

Syntax:

```
for(initialization; condition; increment){
  //code to be executed
}
```

The parameters of for loop have following meaning:

- **initialization:** it is used to initialize the counter variables, 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 evaluated at the end of each iteration.



Nested for loops:

We can also use a for loop inside another for loop.

e.g.

```php
<?php

for($a = 0; $a <= 2; $a++)

{

    for($b = 0; $b <= 2; $b++)

    {

        echo "$b $a ";

    }

}

?>
```

Output will be:

0 0

1 0

2 0

0 1

1 1

2 1

0 2

1 2

2 2

PHP foreach Loop:

The foreach loop in PHP is used to access key-value pairs of an array. This loop only works with arrays and you do not have to initialize any loop counter or set any condition for exiting from the loop, everything is done implicitly by the loop.

Syntax:

```
foreach ($array as $value) {
        //code to be executed
}
```

Example:

```
<?php
$colors = array("Red", "Green", "Blue");
// Loop through colors array
foreach($colors as $value)
        {
                echo $value . "<br>";
        }
 ?>
```

There is one more syntax of foreach loop, which is extension of the first.

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

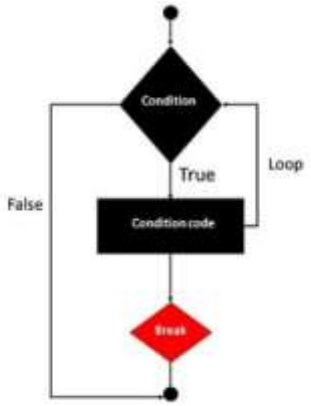
Example:

```
<?php
$superhero=array(
“name” => “Shaktiman”,
“email”=> shkatiman@gmail.com
“age” => 32 );
// loop through superhero array
foreach($superhero as $key =>$value){
echo $key . “ : ” .$value . “<br>”;
}
?>
```

The break Statement:

The PHP break keyword is **used to terminate the execution of a loop prematurely.**

The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.



e.g. in the example below, we want to find the first number divisible by 13, which is between 1762 and 1800, starting from 1762.

```php
<?php
$x = 13;
for($i = 1762; $i < 1800; $i++)
{
    if($i % $x == 0)
    {
        echo "The number is $i";
        break;
    }
}
?>
```

The output will be:
The number is 1768.

The continue statement:

The PHP **continue** keyword is used to **halt the current iteration of a loop but it does not terminate the loop.**

Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

Example:

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
?>
```

The above example will skip the value of 4.