Unit-III
Lecture-II
Processing Arrays with Loops and Iterators

One can iterate over an array using:

1. For Loop:

```
<?php

//define array

$cities = array('Hamirpur', 'Dharmshala', 'Shimla', 'Kinnaur', 'Chandigarh', 'Delhi');

// iterate over array

// print each value

for ($i=0; $i<count($cities); $i++)

{

echo $cities[$i] . "\r\n";

}

?>
```

In the above example, a for loop iterates over $cities array, printing each value found. The loop runs as many times as there are elements in the array; this information is quickly ascertained by a call to the count () function.

2. The foreach Loop: With a foreach loop, each time the loop runs, the current array element is assigned to a temporary variable, which can then by processed in any way you please-printed, copied to another variable, used in calculation and so on.

Unlike a for loop, a foreach loop doesn't use a counter; it automatically knows where it is in the array, and it moves forward continuously until it reaches the end of the array, at which point it automatically halts.

```
<?php

//define array
```

```
$cities = array('Hamirpur', 'Dharmshala', 'Shimla', 'Kinnaur', 'Chandigarh', 'Delhi');

//iterate over array

// print each value

foreach($cities as $c) {

\echo "$c \r\n";

}

?>
```

3. The Array Iterator: The third way is to used an ArrayIterator which was introduced in PHP5.0, which provides a ready-made, extensible tool to loop over array elements.

```
<?php

// define array

$cities = array(

 "United Kingdom" => "London",

 "United States" => "Washington",

 "France" => "Paris",

 "India" => "Delhi"

);

// create an ArrayIterator object

$iterator = new ArrayIterator($cities);

// rewind to beginning of array

$iterator->rewind();

// iterate over array

// print each value

while($iterator->valid()) {

 print $iterator->current() . " is in " . $iterator->key() . ". \r\n";
```

```
 $iterator->next();
}
?>
```

In the above example, an ArrayIterator object is initialized with an array variable, and the object's rewind() method is used to reset the internal array pointer to the first element of the array.

A while loop, which runs so long as a valid() element exists, can then be used to iterate over the array.

Individual array keys are retrieved with the key() method, and their corresponding values are retrieved with the current() method.

The next () method moves the internal array pointer forward to the next array element.

Project: Averaging the Grade of a class:

```
<html>
 <head>
 <title>Grade Averaging</title>
 </head>
 <body>
 <h2>Grade Averaging</h2>
<?php
// define array of grades
// ranging between 1 and 100
$grades = array(
 25, 64, 23, 87, 56, 38, 78, 57, 98, 95,
 81, 67, 75, 76, 74, 82, 36, 39,
 54, 43, 49, 65, 69, 69, 78, 17, 91
```

```
);
// get number of grades
$count = count($grades);
// iterate over grades
// calculate total and top/bottom 20%
$total = $top = $bottom = 0;
foreach ($grades as $g) {
 $total += $g;
 if ($g <= 20) {
 $bottom++;
 }
 if ($g >= 80) {
 $top++;
 }
}
// calculate average
$avg = round($total / $count);
// print statistics
echo "Class average: $avg <br />";
echo "Number of students in bottom 20%: $bottom <br />";
echo "Number of students in top 20%: $top <br />";
?>
 </body>
</html>
```