Agenda:

• PHP Arrays
• PHP Loops

PHP Arrays

• A normal declaration of a variable will hold only one value at a time.

• If you have a list of items (a list of car names, for example), storing the cars in single variables would look like this:

```php
$courses1 = “HDIT”;
$courses2 = “HDCS”;
$courses3 = “BIT”;
$courses4 = “BCS”;
```
PHP Arrays

• What if you want to go through the courses and find a specific one?
• What if you had 300 items or more?
• The best solution here is to use an array.

PHP Arrays

• An array is a special variable, which can store multiple values in one single variable.
• An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
• Each element in the array has its own index so that it can be easily accessed.

PHP Arrays

• In PHP there are three kinds of arrays;  
  – Numeric Array
  – Associative Array
  – Multidimensional Array
• In this lecture, however, only numeric array will be covered.
PHP Numeric Array

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array in PHP;
  - Automatically assigning indexes
  - Manually assigning indexes

```php
// In the following example the index is automatically assigned;
$courses = array("HDIT", "HDCS", "BIT", "BCS");

// Index of the elements starts at 0 so HDIT is in index 0.
```

```php
// In the following example the index is manually assigned;
$courses[0] = "HDIT";
$courses[1] = "HDCS";
$courses[2] = "BIT";
$courses[3] = "BCS";
```
PHP Numeric Array

• To access items stored in an array, you simply have to refer to the name of the array and an index of the item, see the example below:

```php
echo "The first course in the list is ". $courses[0];
// Output: The first course in the list is HDIT
```

Determine Size of an Array

• Size of an array is determined by the number of items stored in the array.
  – If 4 items are stored in an array, then the size of that array is 4.

• To determine the size of an array, we use `count()` function or `sizeof()` function.

```php
$courses = array("HDIT", "HDCS", "BIT", "BCS");
echo "The size of this array is ". count($courses);
echo "The size of this array is ". sizeof($courses);
// Output: The size of this array is 4
```
PHP Loops

• To access and display all the items in an array, we need to repeat `echo` statement four times, see the example below;

```php
echo "$courses[0] <br/>";
echo "$courses[1] <br/>";
echo "$courses[2] <br/>";
echo "$courses[3] <br/>";
```

PHP Loops

• Using loops, however, the `echo` statement can only be used once to access and display all the items in an array.

• PHP loops are used when same block of code is to ran over and over again.

PHP Loops

• In PHP, we have the following looping statements:
  – while
  – do...while
  – for
  – foreach
while Loop

- The **while** loop tests a condition before a block of code
  - The block of code is executed only if the condition is true.
- Below is the structure of the **while** loop;

```php
while (condition is true){
  code to be executed
}
```

while Loop

- Below is an example of the **while** loop;

```php
$i = 0;
while ($i <= 3){
  echo $i.' <br /';
  $i++;  
}
```

while Loop

- In the example, the loop starts with `$i = 0`.
- The loop will continue to run as long as `$i` is less than, or equal to 3.
- With `$i++`, `$i` will be increased by 1 each time the loop runs:
do...while Loop

• The do...while loop tests a condition after a block of code
  – The block of code will always be execute at least once.

• Below is the structure of the do...while loop;
  ```
  do{
  code to be executed
  } while (condition is true); //This semicolon is required
  ```

do...while Loop

• Below is an example of a do...while loop;
  ```
  $i = 0;
  do{
    echo $i.'<br>' ;
  } while ($i > 0);
  ```

• The semicolon after the condition is required, especially when there are more lines of code after the condition.

do...while Loop

• In the example, the loop starts with $i = 0.

• The loop will execute the echo statement and only then will test the condition i.e. testing if the value assigned to $i is greater than 0.

• In the example the loop will stop immediately after testing the condition because 0 is not greater than 0.
for Loop

• The for loop is used for repeating a known number of times.

• Below is the structure of the for loop;

```plaintext
for (init; condition; increment/decrement)
{
  code to be executed
}
```

for Loop

• init is used to set a counter.

• condition is evaluated for each loop iteration
  – If the condition evaluates to TRUE, the loop continues.
  – If the condition evaluates to FALSE, the loop ends.

• increment is used to increment or decrement a counter.

for Loop

• Below is an example of the for loop;

```plaintext
$courses = array("HDIT", "HDCS", "BIT", "BCS");
for ($i = 0; $i < sizeof($courses); $i++){
  echo $courses[$i] . '<br />
}
```

• In the example, the loop starts with $i=0. The loop will continue to run as long as $i is less than the size of the array. $i will be increased by 1 each time the loop runs.
foreach Loop

• The **foreach** loop provides an easy way to go through arrays.

• Below is the structure of the **foreach** loop;

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

foreach Loop

• Below is an example of the **foreach** loop;

```php
$courses = array(“HDIT”, “HDCS”, “BIT”, “BCS”);
foreach ($courses as $value) {
    echo $value. ‘<br/>’;
}
```

• For every iteration, the value of the current element is assigned to **$value** (and the array pointer is moved by one) - so the next iteration, the next element will be accessed.