UNIT-1

Topics:

Introduction to PHP: Declaring variables, data types, arrays, strings, operators, expressions, control structures, functions, Reading data from web form controls like text boxes, radio buttons, lists etc., Handling File Uploads. Connecting to database (MySQL as reference), executing simple queries, handling results, Handling sessions and cookies

File Handling in PHP: File operations like opening, closing, reading, writing, appending, deleting etc. on text and binary files, listing directories.

Introduction

What is PHP? What are the common uses of PHP? (2 marks May 2017)

- PHP stands for "PHP: Hypertext Preprocessor". Earlier it was called Personal Home Page.
- PHP is a server side scripting language that is embedded in HTML.
- The default file extension for PHP files is ".php".
- PHP supports all the databases that are present in the market.
- PHP applications are platform independent. PHP application developed in one OS can be easily executed in other OS also.

Common uses of PHP:

- PHP performs system functions, i.e. Using System function we can create, open, read, write & close Files.
- Using PHP we can create Dynamic Page Content i.e. anew Feature is added without disturbing the Old feature.
- Collect FORM data- User data is collected to Databasei.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete and modify elements within your database through PHP.
- Control Access- Using PHP, you can restrict users to access some pages of your website.
- Cookies- We can create,access cookies variables and set cookies.
- PHP can encrypt data.

Characteristics/Features of PHP:

- Cross Platform-PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.
- Cross Server- We can access data from different servers on our php application.
- Cross database- PHP supports all the database(Oracle, MySQL, SQLite etc.) that are present in the market.
- Interpreted- It is an interpreted language, i.e. there is no need for compilation.
- Open Source-Open source means you no need to pay for use php, you can free download and use.

PHP Basic Syntax-

Universal Style Tag A PHP script starts with **<?php** and ends with **?>**	<?php // PHP code goes here ?>
Comments in PHP	// This is single-line comment # This is also a single-line comment /* This is a multiple-lines comment block that spans over multiple lines */

Output Functions in PHP

In PHP there are two basic ways to get output: echo and print.

1. **The PHP echo Statement**

   The echo() function outputs one or more strings.

   The echo statement can be used with or without parentheses: echo or echo().

   Syntax: echo(*strings*)

   Example

   ```
   <?php
   echo "Hello world!";
   echo ("Hello world!");
   ?>
   ```

2. **PHP print() Function :**

   Syntax: print(*strings*)

   Example:

   ```
   <?php
   print "Hiee!";
   ?>
   ```

   echo and print are used to output data to the screen. The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

Printing Some Text

```
<html>
<head><title>Displaying TEXT from PHP</title></head>
<body>
<?php
echo 'Welcome to PHP.'; //echo statment using single quote
echo "Welcome to PHP."; //echo statment using double quote
echo ("Welcome to PHP."); //We can pass text inside parenthesis
?>
</body>
</html>
```

Mixing HTML and PHP

```
<html>          // Page starts with standard <html> <head> <title> section.
<head>
<title>
Using PHP and HTML together
</title>
</head>

<body>                  // <body> section
<h1> Using PHP and HTML together</h1>  //Contain <h1> header and some text
Here is PHP info: </br></br>
<?php      // <?php starts a PHP section
phpinfo(); //PHP function phpinfo(); displays information about the PHP installation, Every PHP
statement ends with semicolon
?>
</body>
</html>
```

When this page is run by PHP engine on the server HTML will be passed through browser and PHP part will be executed

---

Variables :

Variables are "containers" that are used for storing information.
In PHP, a variable starts with the $ sign, followed by the name of the variable.

What are the rules for naming a variable in PHP?

- Every variable starts with the $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character(can't start with a number)
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are **case-sensitive** ($age and $AGE are two different variables)

Storing Data in Variables

PHP variables can hold numbers or strings of characters. We can store information in variables with an assignment operator/ single equal sign (=).

```
<?php
$name="ACE";
$age = 21;
?>
```

After the execution of the statements above, the variable $name will hold the value ACE**,** the variable $age will hold the value **21.**

Destroying Variable

Want to uncreate a variable? We can indeed uncreate a variable in PHP using unset() function. The unset() function destroys a given variable.

```
 <?php
$car="TATA";
echo "Before unset(), My car is $car";  //Interpolating String: PHP will place the value held in
$car, inside a double-quoted text String
unset($car);
echo "After unset(), My car is ", $car;
?>
```

Creating Variable Variables ($ and $$)

Lets create a variable named $name and set the value='Siri'. $name="Siri"
$$name uses the value of the variable $name
$$var is known as reference variable where as $var is normal variable.

```
<?php
$name="Siri";
$$name="Siri CSEA";
echo $name."<br/>";
echo $$name."<br/>";
echo $Siri;
?>
```

Data Types

Explain about various data types in PHP (3 marks)

In other programming language C, C++, JAVA, we need to specify the data type for each variable, but In PHP we don't have to specify data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value.

PHP supports the following data types:

1) Boolean Holds true/false values
2) Integer Holds non decimal numbers like -1, 0 5 and so on (Positive or negative)
3) String Holds text like "Welcome to PHP" (text inside quotes)
4) Float Holds floating point numbers (also called double) like 3.14159 or 2.7128 with decimal point
5) Array Holds array of data items
6) Object Holds Programming Objects
7) Resource Holds a data resource
8) NULL Holds a value of NULL

Program Example:

```
<?php
$x = "Hello world!";  //String in double Quote
$y = 'Hello world!';  //String in single quote
echo $x;
echo "<br>";
echo $y;
$num = 5985;          //Integer
var_dump($num);       //var_dump() Prints the data type and value
$fnum = 10.365;              //float
var_dump($fnum);      //var_dump() Prints the data type and value
$boovar = true;              //Boolean represents two possible states: TRUE or FALSE
echo $boovar;
$cars = array("Volvo","BMW","Toyota");   //array stores multiple values in one single variable
var_dump($cars);                         //var_dump() Prints the data type and value
$myval = null;               //NULL has no value assigned to it
var_dump($myval);
?>
```

Arrays

- An array is collection of multiple values stored under a single variable name.
- In PHP, we have 3 types of arrays:
  1. Indexed arrays - Arrays with a numeric index
  2. Associative arrays - Arrays with named keys
  3. Multidimensional arrays - Arrays containing one or more arrays

Creating an Array in PHP

We use the array construct to create an array in PHP.

$actor=array("Pavan","kalyan","mahesh","babu");

Modifying the data in Arrays

We have seen how to create arrays, Next how can we modify the data in the arrays?

Example: We have created an array:

$actor=array("Pavan", "kalyan", "mahesh", "babu");

Now we want to change $actor[0] from Pavan to Pawan

We do : **$actor[0] ="Pawan";**

If we want to add new actor, "Prabhas"

We do : **$actor[]="Prabhas"**

Deleting Array Elements:

we can remove an element from an array using unset function.
unset($actor[0]);

Handling Arrays with Loops:

1) **The for Loop**
2) **The print_r Function**
3) **The foreach Loop**
4) **The while Loop**

The for Loop

- The for loop is used to iterate over the elements of an array **when we know in advance the number of elements of an Array.**
- The **count() function** is used to return the length (the number of elements) of an array & this number is useful when we want to set the number of times the for loop to repeat.

Syntax

```
for (init counter; test counter; increment/decrement counter) {
  code to be executed;
}
```

Program Example:

```
<html>
<head>
<title>
Using a for loop to loop over an Array
</title>
</head>
<body>
<h1>Using a for loop to loop over an Array</h1>
</br></br>

<?php
$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandigarh");
$arrayLength=count($cities);

for($i=0; $i<$arrayLength;$i++)
{
        echo $cities[$i],"<br>";
}

?>
</body>
</html>
```

The print_r Function

There is simple way to print the contents of an array using print_r function.

For example:

```
<?php
$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandi
garh");
print_r($cities);
?>
```

The foreach Loop

- The for each loop is designed to iterate over an elements of an Array/Associative array.
- We have two syntax to use foreach loop:
    1. **foreach (array as $value) statement**
    2. **foreach (array as $key => $value) statement**

The following example, puts the foreach loop to work, looping over the $cities array like:

```
<html>
<head>
<title>
Using a foreach loop to loop over an Array
</title>
</head>

<body>
<h1>Using a foreach loop to loop over an Array</h1>
</br></br>
<?php

$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandi
garh");

foreach ($cities as $value)
{
        echo "$value<br>";
}

?>
</body>
</html
```

when foreach starts executing, the array pointer is automatically set to the first element of the array. On each iteration, the value of the current element is assigned to $value and the array pointer is incremented by one.

The second form of foreach loop lets us to work with keys as well as values.

- Element is a combination of element key and element value.
- PHP also support slightly different type of array in which index numbers are replaced with user defined string keys. This type of array is known as **Associative Array.**
- The keys of the array must be unique, each key references a single value. The key value relationship is expressed through **=> symbol.**

For example:

```
<html>
<head>
<title>
Using a foreach loop with keys and values in an Array
</title>
</head>

<body>
<h1>Using a foreach loop with keys and values in n Array</h1>
</br></br>
<?php

$details=array("name"=>"Siri","Type"=>"College","Place"=>"Hyderabad");

foreach ($cities as $key=>$value)
{
        echo "$key : $value<br>";
}

?>
</body>
</html>
```

Note: If we don't provide an explicit key to an array element, The new key value depends upon the previous key.

```
<?php
$a=array(10,100=”ACE”,30);          //[0]=>10      [100]=>ACE    [101]=30
print_r($a);
?>
```

The while loop

- We can use while loop to iterate over an array.
- It tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE.
- The value of the expression is checked each time at the beginning of the loop.
- if the while expression evaluates to FALSE the execution of nested statement will stop.

To handle multiple-item return value from the each function, we can use list() function .
list() function assign the two return value from each separate variable.

Example of using list() function:

```
<?php
$Language=array("PHP","XML","Servlet","JSP","JavaScript");
list($x, $y, $z)=$Language;
echo "We have covered $x, $y and $z.";
?>
```

Example (Displaying array using while):

```
<html>
<head>
<title>
Using a while loop with keys and values in an Array
</title>
</head>

<body>
<h1>Using a while loop with keys and values in n Array</h1>
</br></br>
<?php

$details=array("name"=>"ACE","Type"=>"College","Place"=>"Hyderabad");

while (list($key, $value)=each($details))
{
        echo "$key : $value<br";
}

?>
</body>
</html>
```

PHP Array Function

array() - function creates and returns an array.
array_change_key_case() - function changes the case of all key of an array.
array_chunk() - function splits array into chunks. we can divide array into many parts.
count() - function counts all elements in an array.
sort() - function sorts all the elements in an array.
array_reverse() - function returns an array containing elements in reversed order.
array_search() - function searches the specified value in an array.
array_intersect() - function returns the intersection of two array. It returns the matching elements of two array.
array_push() — function pushes one or more elements onto the end of array.
array_pop() — function pops the element off the end of array.
array_merge() — function merge one or more arrays.
array_sum() — function calculate the sum of values in an array.

Converting Between String and Arrays Using implode and explode

- **The implode function** is used to "join elements of an array with a string".
- **implode()** accepts two argument, first argument the separator which specifies what character to use between array elements, and second one is array.

```
 <?php
$ice_cream[0]="Chocolate";
$ice_cream[1]="Mango";
$ice_cream[2]="Strawberry";

$text=implode(" , ", $ice_cream);     //Outputs: Chocolate, Mango, Strawberry

?>
```

- The **explode function** breaks a string into an array.
- **explode()** accepts three argument, first one is the delimiter, second one is the strings that needs splitting, and third one is not mandatory.

```
<?php

$ice_cream="Chocolate, Mango, Orange";
$ice_cream=explode(" , ", $text);
print_r($ice_cream);  //Outputs: Array( [0]=>Chocolate   [1]=>Mango  [2]=>Strawberry)

?>
```

Handling Multidimensional Arrays

Aditya	25	23
Srikhar	24	22
Surya	22	21

We are keeping track of student mid exam score

```
<?php
$midScore=array(
        array("Aditya",25,23),
        array("Srikhar",24,22),
        array("Surya",22,21)
        )

echo $midScore[0][0]." ".[0][1]." ".[0][2]."<br>";
echo $midScore[1][0]." ".[1][1]." ".[1][2]."<br>";
echo $midScore[2][0]." ".[2][1]." ".[2][2]."<br>";
```

Moving through Arrays

PHP supports number of functions to move through arrays.

1. **current()** – to get current element
2. **next()** – to get next element
3. **prev()** – to get the previous element
4. **end()** – to get the last element

```
<?php
$ice_cream=array("chocolate","strawberry","butterscotch");
echo "current element: ",current($ice_cream),"<br>";
echo "next element: ",next($ice_cream),"<br>";
echo "previous element: ",prev($ice_cream),"<br>";
echo "Last element : ",end($ice_cream),"<br>";
?>
```

Strings

The String functions

- **explode() - Breaks a string into an array**
- **implode() - Returns a string from the elements of an array**
- **str_replace() - Replaces characters in a string**
- **strrev() - Reverses a string**
  Example: echo strrev("Hello world!"); // outputs !dlrow olleH

- **strtolower() - Converts a string to lowercase letters**
  Example: strtolower("HELLO WORLD");// outputs hello world
- **strtoupper() - Converts a string to uppercase letters**
  Example: strtoupper("hellow World");// outputs HELLO WORLD
- **strlen() - Returns the length of a string**
  Example: echo strlen("Hello world!"); // outputs 12
- **str_word_count() - counts the number of words in a string**
  Example: echo str_word_count("Hello world!"); // outputs 2

Operators

Operators supported by PHP are:

1. Math operators
2. Assignment operators
3. Increment/Decrement operators
4. String operators
5. Comparison operators
6. Logical operators

PHP's Math Operators

Operator	Name	Example	Description
+	Addition	$x + $y	Sum of x and y
-	Subtraction	$x - $y	Difference of x and y
*	Multiplication	$x * $y	Product of x and y
/	Division	$x / $y	Quotient of x divided by y
%	Modulus	$x % $y	Remainder of x divided by y
**	Exponentiation	$x ** $y	Result of x raised to the power of y

PHP's Assignment Operators

The main **assignment operator is the = operator**, which just assigns a value.
Example: To store the value 99 in a variable $number: **$number = 99;**
By using = operator, we can **make multiple assignments** on the same line:
Example: **$numOne = $numTwo = $numThree = 99;**
Assignms the value 99 to three variable $numOne, $numTwo and $numThree

PHP gives us a set of Combination Assignment Operator

Assignment	Description
x += y	Adding x and y and store the result in x
x -= y	Subtracting y from x and store the result in x
x *= y	Multiplying x and y and store the result in x

Incrementing and Decrementing Operators

Increment operators increments the value of a variable and Decrement operator's decrements the value of a variable.

++$a	Pre-increment
$a++	Post-increment
--$a	Pre-decrement
$a--	Post-decrement

PHP's string Operators

PHP has two operators that work with Strings:

1. The concatenation Operator (.)
2. The combined concatenation assignment operator (.=)

Example:

```
<?php
echo "Hello"."Wolrd"; //Output: Hello World (Hello and World are concatenated)
$str1="Hello";
$str2="World";
echo $str1.=$str2; //str2 is appended to str1
?>
```

The PHP Comparison operators

We use PHP Comparison Operators to compare two values:

==	Is equal to
!=	Is not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

The PHP Logical operators

We use Logical operators when we want to test more than one condition at a time.

Logical operators supported by PHP are **and, or, not.**

&&	and operator	[need both values to be true]
||	or operator	[need one of our conditions to be true]
!	not operator	

Using logical operator to combine conditions.

Example: both username and password values needs to be true in our test.

```
    if( $username =='user' && $password =='password')
```

Expressions

The **most basic forms of expressions** is "$num = 5", we are assigning '5' into variable $num.

$num=5

Another example of expression is pre- and post-increment and decrement.

Example:

```
$c = $num++;   /* post-increment, assign original value of $num (5) to $c */
$d = ++$num;   /* pre-increment, assign the incremented value of $num to $d */
```

One more type of expressions are **comparison expressions.**

These expressions evaluate to either FALSE or TRUE.

PHP supports: > (greater than) >= (greater than or equal to) == (equal)
!= (not equal) < (less than) <= (less than or equal to)

Control Structures/ Flow Control

Using the if Statement

if statement is a conditional statement that allows us to make decisions on what code to execute.

The structure looks like**:**

```
if (expression)
  statement
```

Here, expression evaluates to a TRUE or FALSE

If expression is TRUE, The statement that follows is executed; if it is FALSE, statement is not executed
We use conditional and Logical Operators to create expression of if statement.

Example: We want to display some text if the outside temperature is above 28 degree.

```
<?php
$temperature=35;
if($temperature>28)
{
        echo "Its hotter outside";
}
?>
```

Example: To check how many minutes someone has been in the pool- if its more than 30minute, it's time to get out.

```
<html>
<head>
<title> Using the if statement</title>
</head>
<body>
<h1>Using the if Statement</h1>
<?php
$minutes=31;
if($minutes > 30)
{
        echo "Your time is UP!!<br>";
        echo "Please get out of the pool.";
}
?>
</body>
</html>
```

The else statement

The else statement

Often we want to execute a statement if a condition is TRUE, and we execute different statement if the condition is FALSE.

```
if (expression) {
   code to be executed if condition is true;
} else {
   code to be executed if condition is false;
}
```

Example: We want to print a message if the outside temperature is outside the range 24 degree and 32 degree, and another message if the temperature is inside the range.

```
<html>
<head>
<title> Using the else statement</title>
</head>
<body>
<h1>Using the else Statement</h1>
```

```
<?php
$temperature=44;
if($temperature<24 || $temperature>32)
{
        echo "Better Stay inside today!!";
}
else
{
        echo "Nice weather outside";
}
?>
</body>
</html>
```

The elseif statement

elseif, as its name suggests, is a combination of if and else.
If an if statements condition is false, we can make additional tests with elseif.

```
if (condition1) {
   code to be executed when condition1 is true;
} elseif (condition2) {
   code to be executed when condition2 is true;
} else {
   code to be executed when conditions are false;
}
```

Example:

```
<html>
<head>
<title> Using the elseif statement</title>
</head>
<body>
<h1>Using the elseif Statement</h1>
<?php

$a=40;
$b=30;
if ($a > $b)
{
  echo "$a is bigger than $b";
} elseif ($a == $b)
{
  echo "$a is equal to $b";
} else {
  echo "$a is smaller than $b";
}

?>
</body>
</html>
```

If the if statement's conditional expression is false → It will check the first elseif statements expression, → **if its true** elseif code is executed and **if its false** it moves to the next elseif statement and so on.
At he end else statament is executed if the no other code is executed in the if statement up to this point.

The switch statement:

- switch statement lets us replace long if-elseif-else ladder of condition checking with switch statement.
- switch statement compares a value against case statement and execute a code block whose value matches the case value.
- if no case value matches the value, default statament is executed.
- break statementends execution of the switch statement, if we don't write break statement, execution will continue with the code.

Syntax:

```
<?php
switch(value){
        case value1:
                // code block 1
                break;
        case value2:
                // code block 2
                break;

        default:
                // default code block
                break;
}
```

Example:

```
<html>
<head>
<title> Using the switch statement</title>
</head>
<body>
<h1>Using the switch Statement</h1>
<?php

$favcolor = "red";

switch ($favcolor)
{
   case "red":
      echo "Your favorite color is red!";
      break;
   case "blue":
      echo "Your favorite color is blue!";
      break;
   case "green":
      echo "Your favorite color is green!";
      break;
   default:
      echo "Your favorite color is neither red, blue, nor green!";
}

?>
</body>
</html>
```

Using for loop

- for loop is a repetition control structure that allows us to execute a block of code a specified number of times.
- When we want to execute same block of code over and over again specified number of times.
- Syntax:

```
        for( expression1 ; expression1 ; expression1 )
                        statement
```

  - First, for loop executes expression1; then it checks the value of expression2 - if true , loop executes statement once.

- Then it executes expression3 and after that checks the expression2 again-- if true loop execute statement once again.
- Then loop executes expression3 again, and keeps going untill expression2 becomes false.

Example:

```
<html>
<head>
<title>Using the for loop</title>
</head>
<body>
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
</body>
</html>
```

for loop can handle multiple loop counters by seperating them with comma operator.

```
<?php
for ($i=2, $k=2; $i<6 && $k<6 ; $i++, $k++) {
  echo "$i  *  $k = ",$i *$k,"<br>";
}
?>
```

The two variables $i and $k are initialized with 2.
At the end of each each loop $i and $k will be incremented by one.

We can put one loop inside a another loop, call as nested loop.

```
<?php
for ($row = 1; $row <= 5; $row++)
{
  for ($col = 1; $col <= 5; $col++)
  {
    echo '*';
  }
  echo "\n";
}
?>
```

Outputs:

```
*****
*****
*****
*****
*****
```

<u>Using While Loop</u>

- When we want to execute same block of code over and over again as long as the condition is true, we go for while loop.
- <u>Syntax:</u>

```
                    while (expression)
                    {
                            statement
                    }
```

  - While expression is true, the loop executes statement. When expression is false, the loop ends.

<u>Example</u>:

```
<html>
<head>
<title>
Using the while loop
</title>
</head>
<body>
<h1> Using the while loop </h1>
<?php
$x=1;

while($x<10)
{
        echo "Now \$x holds : ",$x,"<br>";
        $x++;
}

?>
</body>
</html>
```