PHP Study material

Expressions

Four simple Boolean expressions

```
<?php
echo "a: [" . (20 > 9) . "]<br>";
echo "b: [" . (5 == 6) . "]<br>";
echo "c: [" . (1 == 0) . "]<br>";
echo "d: [" . (1 == 1) . "]<br>";
?>
```

The output from this code is as follows:

a: [1]
b: []
c: []

d: [1]

Outputting the values of TRUE and FALSE

```
<?php // test2.php
echo "a: [" . TRUE . "]<br>";
echo "b: [" . FALSE . "]<br>";
?>
```

which outputs the following:

a: [1]

b: []

Literals and Variables

```
<?php
$myname = "Brian";
$myage = 37;
echo "a: " . 73 . "<br>"; // Numeric literal
echo "b: " . "Hello" . "<br>"; // String literal
echo "c: " . FALSE . "<br>"; // Constant literal
echo "d: " . $myname . "<br>"; // String variable
echo "e: " . $myage . "<br>"; // Numeric variable
?>
```

And, as you'd expect, you see a return value from all of these with the exception of c:, which evaluates to FALSE, returning nothing in the following output:

a: 73
b: Hello
c:
d: Brian

e: 37

Assigning a value and testing for equality

```
<?php
$month = "March";
if ($month == "March") echo "It's springtime";

?>
```

The equality and identity operators

```
<?php
$a = "1000";
$b = "+1000";
if ($a == $b) echo "1";
if ($a === $b) echo "2";

?>
```

The four comparison operators

```
<?php
$a = 2; $b = 3;
if ($a > $b) echo "$a is greater than $b<br>";
if ($a < $b) echo "$a is less than $b<br>";
if ($a >= $b) echo "$a is greater than or equal to $b<br>";
if ($a <= $b) echo "$a is less than or equal to $b<br>";

?>
```

The logical operators in use

```
<?php
$a = 1; $b = 0;
echo ($a AND $b) . "<br>";
echo ($a or $b) . "<br>";
echo ($a XOR $b) . "<br>";
echo !$a . "<br>";

?>
```

Conditional Statements:- perform different action based on different Condition

if

if__else

if__elseif__else

switch

switch(n)

{

Case ‘A’:
Code to be executed if n = A;
break;
Case ‘B’:
Code to be executed if n = B;
break;
default:
Code to be executed if n is different from all labels;

}

While loop:- executed block of code if n is different from all labels.

```
while ( Condition is true)
{
        Code to be executed;
}
```

do__while

```
do
{
        Code to be executed;
} while (condition id true)
```

For loops:-How many times the script should run

```
For (init counter; test condition; increment)
{
        Code to be executed;
}
```

Foreach:- works only on array

```
<?php
        $colors = array(“red, “green”, “blue”, “yellow”);
```

```
foreach($colors @value)
{
      echo “$value <br>”;
}
?>
```

PHP **Break, Continue and goto**

```
<?php
          $x= 1;
          while($x)
          {
              echo($x, “  “);
              if ($x == 5)
              {
                  break; //use of break
              } x++;
          }
?>
```

```
<? php
          $x = 2;
while ($x)
{
          for ($j = 0; $j++)
          {
               echo  $j*$x;
              if ($j * $x >= 10)
              {
                    break2;//use of break2;
               }
          }
          $x++;
}
?>
```

Continue statement:-

```
<?php
          for ($i = 0; $i< 5; $i++)
          {
```

```
            If ($x == 2)
             {
                continue ;
             }
           echo $i, “”;
        }
?>
```

Goto statement:-

```
<?php
        for ($i = 0; $i< 5; $i++)
        {
            If ($x == 2)
             {
                goto end ;
             }
           echo $i, “”;
        }
        end:
?>
```

Static Keyword:-once initialized cannot change value of static variable through re-initialization

```
<?PHP

Function myStatic()
{
        static $no = 0;
        echo $no;
        $no++;
}
myStatic();
myStatic();
myStatic();

?>
```

Accessing character with a string

```
<?PHP

$str = “This is my PHP file”  ;    //$char = $str[position];
echo  $str[0];    //it will print  T
```

```
echo $str[7];    //check  what it will print?

For ($i = 1;  $i<strlen($str); $i++)
{
        echo $str[$i];
}
?>
```

ucfirst -> converts the first character of each word in a string to uppercase

```
<?php

$str = “this is my first php”;
For ($i = 1;  $i<strlen($str); $i++)
{
        $str = ucfirst($str[$i]);
}
?>
```

Q1. Implement a *groupByOwners* function that:

- Accepts an associative array containing the file owner name for each file name.
- Returns an associative array containing an array of file names for each owner name, in any order.

For example, for associative array *["Input.txt" => "Randy", "Code.py" => "Stan", "Output.txt" => "Randy"]* the *groupByOwners* function should return *["Randy" => ["Input.txt", "Output.txt"], "Stan" => ["Code.py"]]*.

Q2. Implement the *unique_names* function. When passed two arrays of names, it will return an array containing the names that appear in **either or both** arrays. The returned array should have no duplicates.

For example, calling *unique_names(['Ava', 'Emma', 'Olivia'], ['Olivia', 'Sophia', 'Emma'])* should return *['Emma', 'Olivia', 'Ava', 'Sophia']* in any order.

Q3. The user interface contains two types of user input controls: *TextInput*, which accepts all texts and *NumericInput*, which accepts only digits.

Implement the class *TextInput* that contains:

- Public function *add($text)* - adds the given text to the current value.
- Public function *getValue()* - returns the current value (string).

Implement the class *NumericInput* that:

- Inherits from *TextInput.*
- Overrides the *add* method so that each non-numeric text is ignored.

For example, the following code should output '10':

```
$input = new NumericInput();
$input->add('1');
$input->add('a');
$input->add('0');
echo $input->getValue();
```

From:-
Ritu Meena
Assistant Professor
Shivaji College
Delhi University