

INTRODUCTION TO PHP

Dr. Prabu M, Ph.D.,(CSE)

Assistant Professor, Department of Computer Science,
CHRIST (Deemed to be University), Bangalore-560029.

PHP INTRODUCTION



PHP is a recursive acronym for “PHP: Hypertext Preprocessor”

It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

PHP INTRODUCTION

- > PHP is a server-side scripting language
- > PHP scripts are executed on the server
- > PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- > PHP is open source software
- > PHP is free to download and use

PHP INTRODUCTION

- > PHP runs on different platforms (Windows, Linux, Unix, etc.)
- > PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- > PHP is FREE to download from the official PHP resource:
www.php.net
- > PHP is easy to learn and runs efficiently on the server side

FEATURES OF PHP

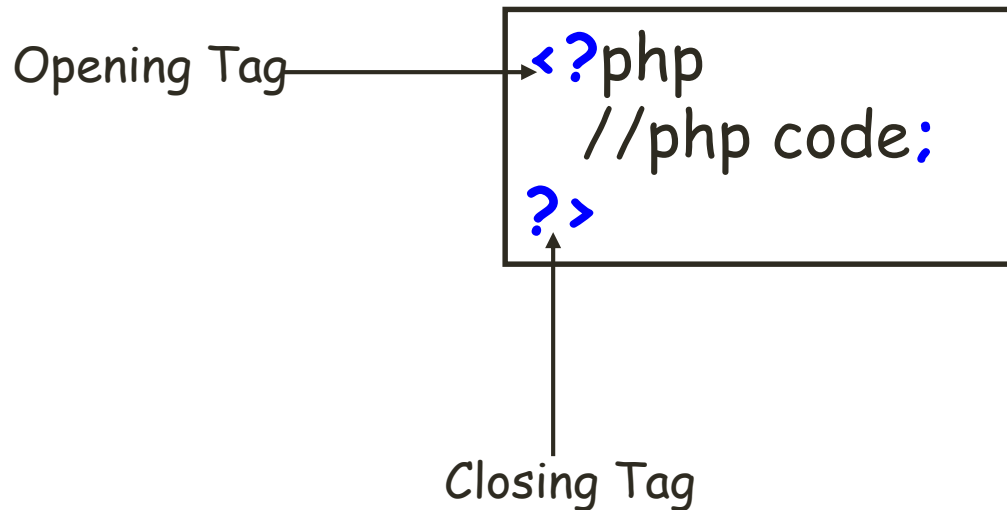
- Cross-platform compatibility.
- No need to specify the data type for variable declarations, and predefined variables can be used.
- Availability of predefined error reporting constants.
- Supports extended regular expressions.
- Has the ability to generate dynamic page content.
- Allows the user to create, open, read, write, delete and close files on the server
- Output can be in HTML, images, PDF, Flash, XHTML and XML file formats.
- Runs on various platforms such as Windows, Linux, UNIX, Mac OSx, etc.
- It is compatible with almost all servers being used currently.

PHP with HTML

- ❖ What makes PHP so different is that it not only allows HTML pages to be created.
- ❖ It is **invisible** to your Web site visitors. The only thing they see the resulting HTML output. This gives you more security for your PHP code and more flexibility in writing it.
- ❖ **HTML** can also be written **inside the PHP** section of your page.
- ❖ **PHP** can also be written as a **standalone** program, with no HTML

PHP Syntax

PHP is denoted in the page with opening and closing tags as follows:

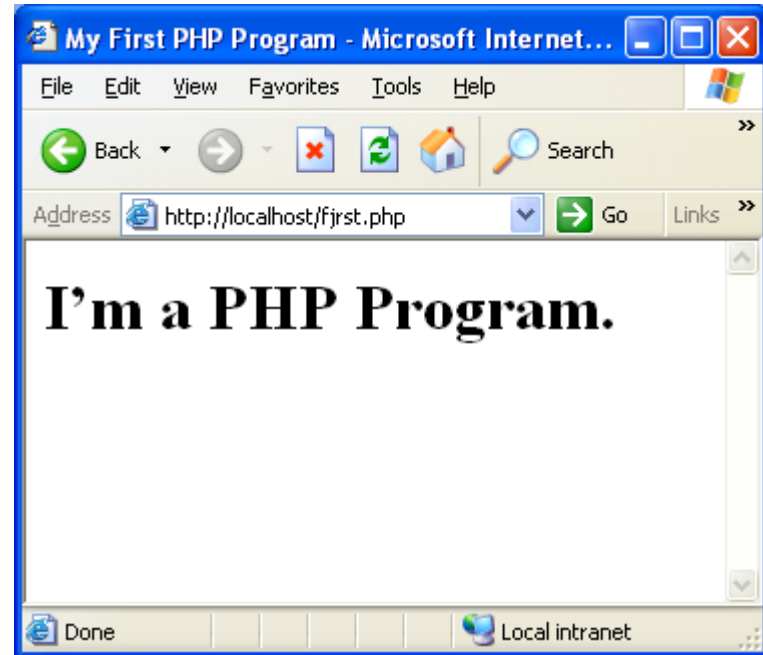


PHP Programs

Code

```
<HTML>
<HEAD>
<TITLE>
My First PHP Program
</TITLE> <h1>
</HEAD>
<BODY>
<?php
echo "I'm a PHP Program.";
?>
</BODY>
</HTML>
```

Result



PHP Programs

Code

```
<?php  
$name="Khan ";  
echo 'My name is '.$name;  
?>
```

Result

?

Comment Line in PHP

1. **Single Line Comment** - Single line comment in PHP is identified with `//`.

```
<?Php  
    // my first prg  
?>
```

2. **# -comment**

3. **Multi Lines Comment** - Multi lines comment in PHP is identified by `/* ... */`

```
<?Php  
    /* line1  
        line2  
        line 3 */  
?>
```

PHP Reserved words: Keywords

| | | | | |
|-------------------|------------|-----------|------------|--------------|
| __halt_compiler() | abstract | and | array() | as |
| break | callable | case | catch | class |
| clone | const | continue | declare | default |
| die() | do | echo | else | elseif |
| empty() | enddeclare | endfor | endforeach | endif |
| endswitch | endwhile | eval() | exit() | extends |
| final | finally | for | foreach | function |
| global | goto | if | implements | include |
| include_once | instanceof | insteadof | interface | isset() |
| list() | namespace | new | or | print |
| private | protected | public | require | require_once |
| return | static | switch | throw | trait |
| try | unset() | use | var | while |
| xor | yield | | | |

Output in PHP

1. **Echo & Print** is the common method in outputting data. it is a language construct
2. Output Text Usage:

```
<?php
    echo "Hello World";
    print "Hello World"; // prints out Hello World
?>
```

- Echo display the outputs one or more strings separated by commas.
- It accepts an list of argument (multiple arguments can be passed) and returns no value or returns void.
- Print always returns the value 1.
- Unlike echo, print accepts only one argument at a time.
- The print outputs only the strings.
- It is slow compared to that of echo.

Output in PHP

3. Output the value of a PHP variable:

```
<?php
    echo $hits; // prints out the number of hits
    print $hits; // returns a value if print process success.
?>
```

4. Can use "
", to print in next line:

```
<?php
    echo $a, "<br>",$b ; //prints a & b in separate
lines
?>
```

Data Type

- Four basic data types:
 1. Integer
 2. Float(Double)
 3. String
 4. Boolean
- More data types
 1. Array
 2. Object
- PHP is an untyped language
 - variables type can change on the fly.

PHP STRING

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

```
<?php  
$x = "Hello world!";  
$y = 'Hello world!';
```

```
echo $x;  
echo "<br>";  
echo $y;  
?>
```


Quotes (" Vs. ' ")

- ❖ In a double-quoted string any variable names are expanded to their values.
- ❖ In a single-quoted string, no variable expansion takes place.

```
$name = "abc";  
$age = 23;  
echo '$name is $age';
```

Output is
\$name is \$age

```
$name = "abc";  
$age = 23;  
echo "$name is $age";
```

Output is
abc is 23

PHP INTEGER

```
<?php
$x = 5985;
var_dump($x);
?>
```

- An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.
- Rules for integers:
- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats:
 - decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP FLOAT

```
<?php  
$x = 10.365;  
var_dump($x);  
?>
```

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example \$x is a float. The PHP var_dump() function returns the data type and value

PHP BOOLEAN

```
$x = true;  
$y = false;
```

- A Boolean represents two possible states: TRUE or FALSE.
- Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

Constants

- values that never changes.
- A constant is case-sensitive by default.
- A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- Constants are defined in PHP by using the `define()` function.
 - For e.g.
`define("CS", "Computer Science");`
`echo CS;//Extract constant value`
- `defined()` function says whether the constant exists or not.

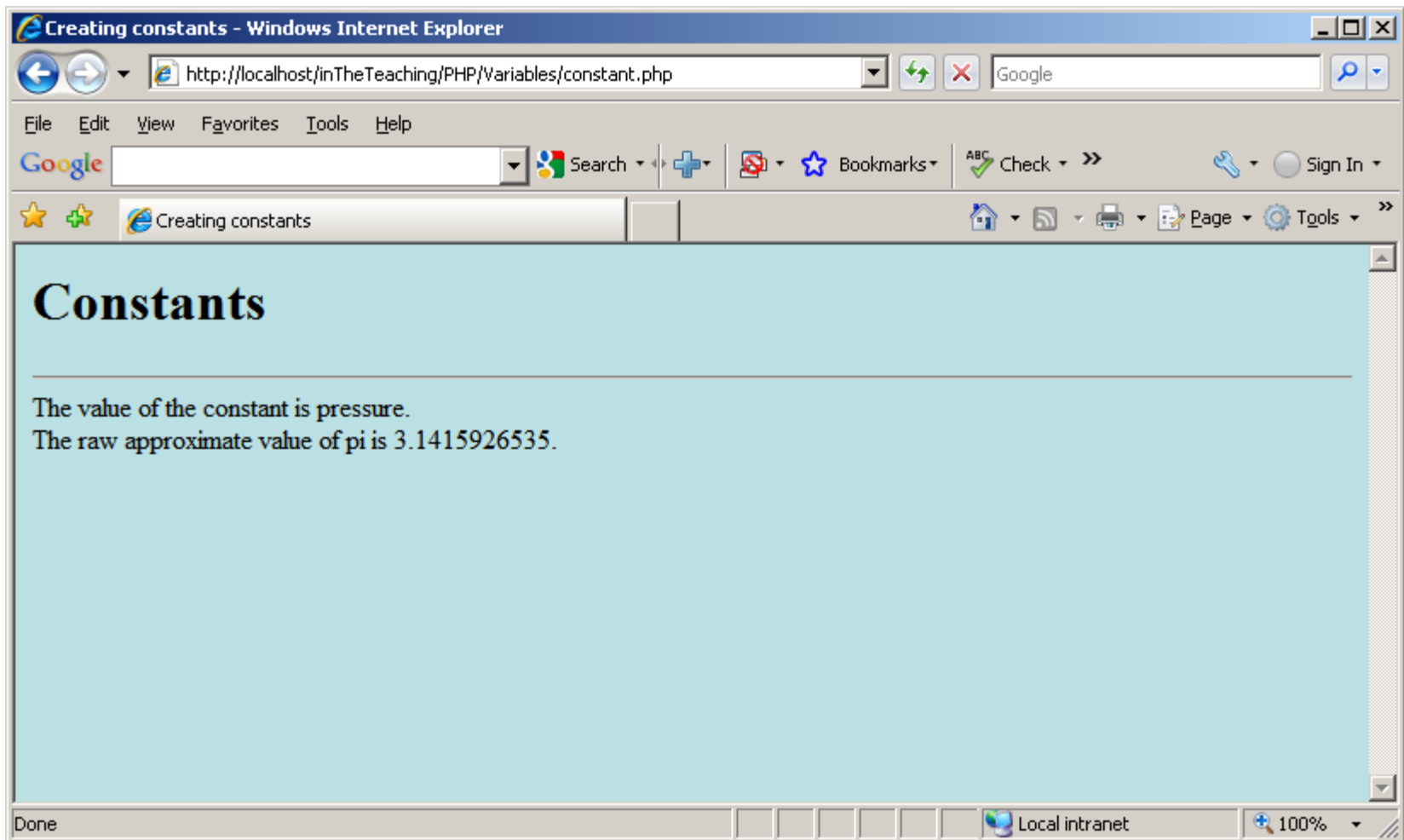
Crimson Editor - [C:\Program Files\Apache Group\Apache2\htdocs\inTheTeaching\PHP\Variables\constant.php]

File Edit Search View Document Project Tools Macros Window Help

constant.php

```
1 <html>
2   <head>
3     <title>Creating constants</title>
4   </head>
5   <body bgcolor="#bbe0e3">
6     <h1>Constants</h1><hr>
7     <?php
8       define("nameOfTheConstant", "pressure");
9       define("PI", 3.1415926535);
10      echo "The value of the constant is " . nameOfTheConstant . "<br>";
11      echo "The raw approximate value of pi is ", PI, "<br>";
12    ?>
13   </body>
14 </html>
```

Ready Ln 11, Ch 53 14 ASCII, DOS READ REC COL



Variables

- The variables in PHP are declared by appending the \$ sign to the variable name.
 - For e.g.
\$company = "NCST";
\$sum = 10.0;
- Variable's data type is changed by the value that is assigned to the variable.
- PHP is a Loosely Typed Language which automatically converts the variable to the correct data type, depending on its value.
- Type casting allows to change the data type explicitly.

Variables

Rules for Naming User Defined Variables:

- ❖ In PHP, unlike some other programming languages, there is no restriction on the size of a variable name.
- ❖ Variable names should be identified by dollar (\$) symbol.
 - ❖ Variable names can begin with an underscore.
 - ❖ Variable names cannot begin with a numeric character.
 - ❖ Variable names must be relevant and self-explanatory.

Variables

| Valid Examples | In Valid Examples |
|---|---|
| <code>\$prod_desc</code> <code>\$Intvar</code> <code>\$_Salesamt</code> | <code>\$9OctSales</code> <code>Sales123</code> <code>\$*asgs</code> |

Variables (cont)

```
$MyStrVal = "This is an example of a string value";  
$MyIntVal = 145665; // An integer value  
$MyBoolval = True; // A Boolean value can either be  
                    True or False  
$MyFloatVal=2346.45 // A float value  
$MyArrVal[0] = "My"; //An array containing three  
                    elements  
$MyArrVal[1] = "First";  
$MyArrVal[2] = "Array";
```


VARIABLES (CONT)

```
$Ball = "a";  
$Foo = "Ball";  
$World = "Foo";  
$Hello = "World";  
$a = "Hello";
```

```
$a;  
$$a;  
$$$a;  
$$$$a;  
$$$$$a;
```

1. \$\$\$\$\$\$a; -----
2. \$\$\$\$\$\$\$a; -----

\$a; //Returns Hello
\$\$a; //Returns World
\$\$\$a; //Returns Foo
\$\$\$\$a; //Returns Ball
\$\$\$\$\$a; //Returns a
1. \$\$\$\$\$\$a; -----
2. \$\$\$\$\$\$\$a; -----



PHP also offers another way to assign values to variables: assign by reference. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable).

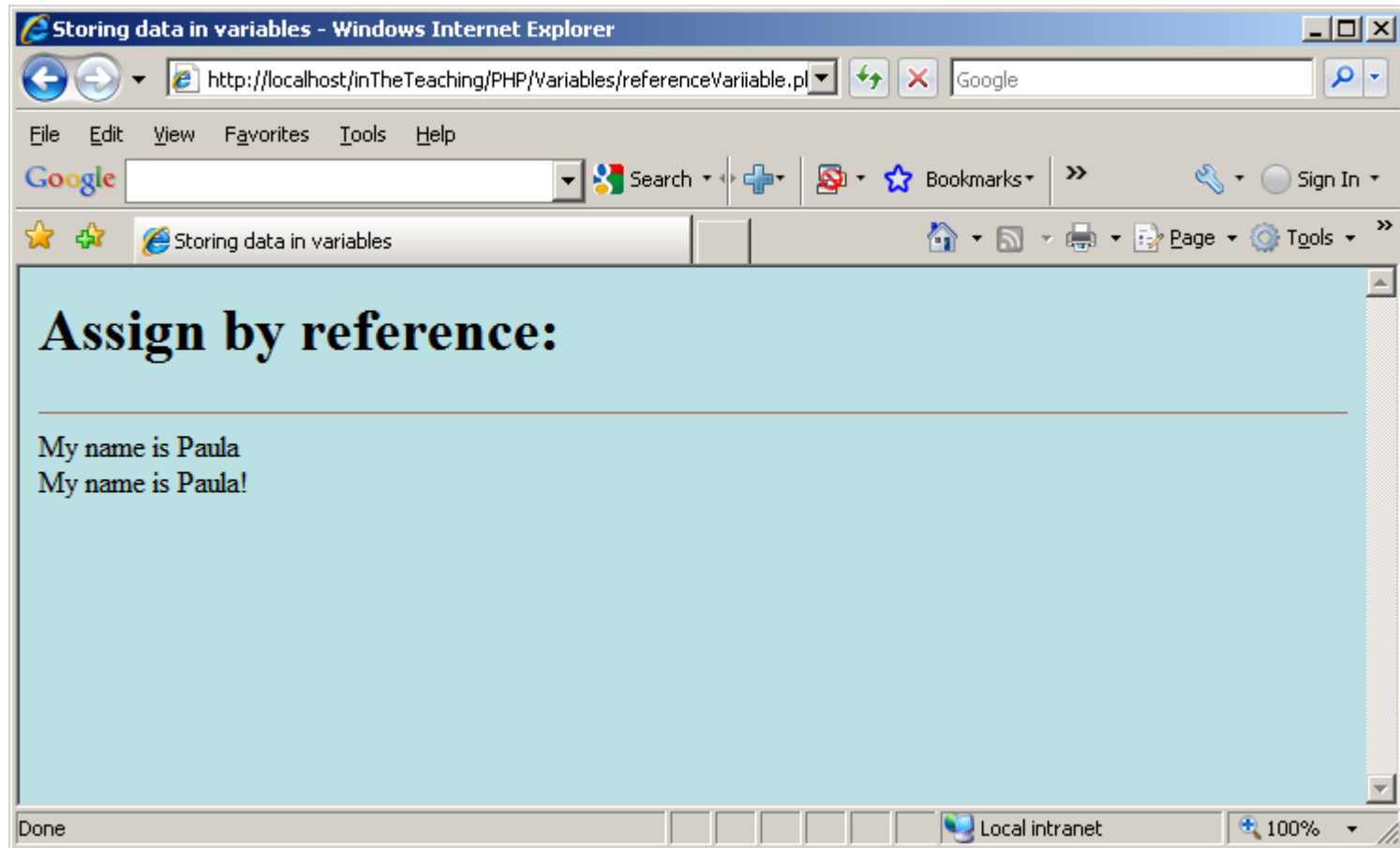
Crimson Editor - [C:\Program Files\Apache Group\Apache2\htdocs\inTheTeaching\PHP\Variables\referenceVariable.php]

File Edit Search View Document Project Tools Macros Window Help

referenceVariable.php

```
1 <html>
2   <head>
3     <title>Storing data in variables</title>
4   </head>
5   <body bgcolor="#bbe0e3">
6     <h1>Assign by reference:</h1><hr>
7     <?php
8       $yourName="Paula";
9       $myName=&$yourName;
10      $myName="My name is $myName";
11      echo $myName, "<br>";
12      echo $yourName, "!<br>";
13    ?>
14  </body>
15 </html>
```

Ready Ln 15, Ch 8 15 ASCII, DOS READ REC



PHP VARIABLES SCOPE

PHP has three different variable scopes:

local

Global-global and `$GLOBALS[index]`

static

```
<?php
$count = 0;
function calculate_count() {
    global $count;
    // will print 0 and increment global variable
    echo $count++ . "<br/>";
}
calculate_count();
echo $count;
?>
```

```
<?php
$count = 0;
function calculate_count() {
    // will print 0 and increment global variable
    echo $GLOBALS["count"]++ . "<br/>";
}
calculate_count();
echo $count;
?>
```

```
<?php
function counter()
{
    static $count = 0;
    echo $count;
    $count++;
}
?>
```

PHP SUPER GLOBALS

Super Globals are variables that are automatically available throughout all program code in all scopes.

These variables do not require declaration and then too they can be accessed.

Super global variables provide :-

- Useful information about the environment.
- Allow access to HTML form variables or parameters.
- Access to cookies stored on a client.
- Keeping track of sessions and file uploads.

| Name | Functionality |
|-------------------------|---|
| <code>\$GLOBALS</code> | Contains all global variables in your script, including other super globals. This is not generally recommended for use, unless you are, for some reason, not sure where a variable will be stored. |
| <code>\$_GET</code> | Contains all variables sent via a HTTP GET request. That is, sent by way of the URL. |
| <code>\$_POST</code> | Contains all variables sent via a HTTP POST request. |
| <code>\$_FILES</code> | Contains all variables sent via a HTTP POST file upload. |
| <code>\$_COOKIE</code> | Contains all variables sent via HTTP cookies. |
| <code>\$_SESSION</code> | Contains all variables stored in a user's session. |
| <code>\$_SERVER</code> | Contains all variables set by the web server you are using, or other sources that directly relate to the execution of your script. |

| Name | Functionality |
|-------------------------|---|
| <code>\$_REQUEST</code> | Contains all variables sent via HTTP GET, HTTP POST, and HTTP cookies. This is basically the equivalent of combining <code>\$_GET</code> , <code>\$_POST</code> , and <code>\$_COOKIE</code> , and is less dangerous than using <code>\$GLOBALS</code> . However, as it does contain all variables from untrusted sources (that is, your visitors), you should still try to steer clear unless you have very good reason to use it. |
| <code>\$_ENV</code> | Contains all environment variables set by your system or shell for the script. |

Settype

In PHP, Variable type can be changed by **Settype**

Settype

Syntax:

```
Settype(Variablename, "newDataType");
```

E.g.

```
$pi = 3.14 //float
```

```
Settype($pi,"string"); //now string - "3.14"
```

```
Settype($pi,"integer");// now integer - 3
```

Settype & Gettype

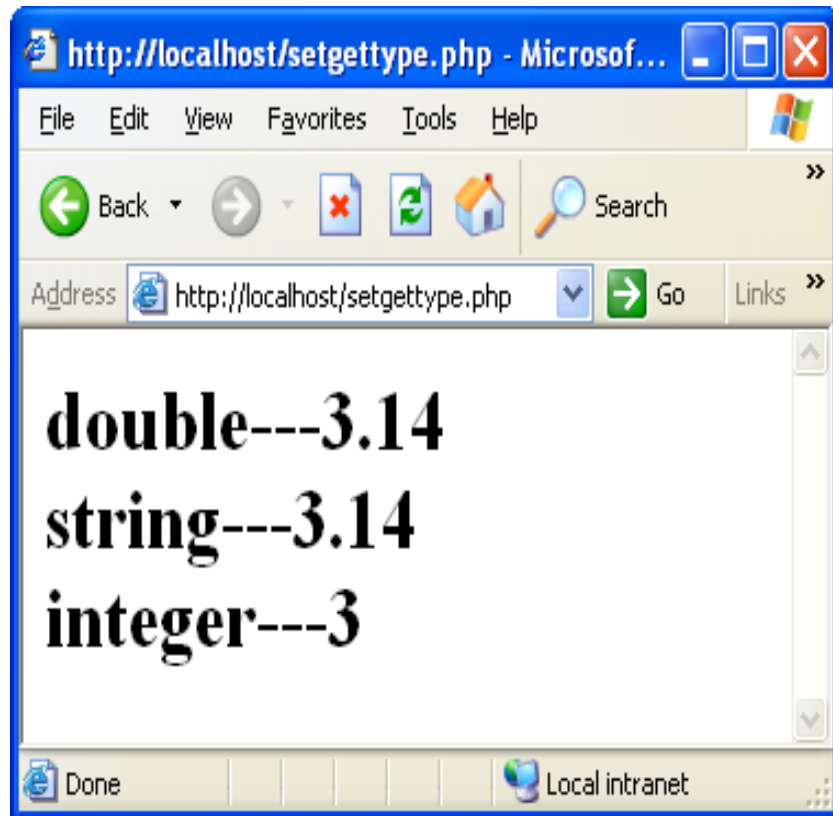
In PHP, Variable type and can know by **Gettype**.

Syntax:

Gettype(Variablename);

E.g.

```
$pi = 3.14;  
print gettype($pi);  
Print "---$pi <br>";  
Settype($pi,"string");  
print gettype($pi);  
Print "---$pi <br>";  
Settype($pi,"integer");  
print gettype($pi);  
Print "---$pi <br>";
```



Operators

- All the operators such as arithmetic, assignment, Comparison, and logical operators are similar to the operators in C and C++.
- In PHP the string concatenation operator is denoted by `'.'`
 - For e.g.

`$name = "My name is".$myname;`

PHP ARITHMETIC OPERATORS

| Operator | Name | Example | Result |
|----------|----------------|--------------|--|
| + | 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 $\$x$ and $\$y$ |
| % | Modulus | $\$x \% \y | Remainder of $\$x$ divided by $\$y$ |
| ** | Exponentiation | $\$x ** \y | Result of raising $\$x$ to the $\$y$ 'th power (Introduced in PHP 5.6) |

PHP ASSIGNMENT OPERATORS

| Assignment | Same as... | Description |
|-------------------------|------------------------------|---|
| | | |
| <code>\$x = \$y</code> | <code>\$x = \$ y</code> | The left operand gets set to the value of the expression on the right |
| <code>\$x += \$y</code> | <code>\$x = \$x + \$y</code> | Addition |
| <code>\$x -= \$y</code> | <code>\$x = \$x - \$y</code> | Subtraction |
| <code>\$x *= \$y</code> | <code>\$x = \$x * \$y</code> | Multiplication |
| <code>\$x /= \$y</code> | <code>\$x = \$x / \$y</code> | Division |
| <code>\$x %= \$y</code> | <code>\$x = \$x % \$y</code> | Modulus |

PHP COMPARISON OPERATORS

| Operator | Name | Example | Result |
|----------|--------------------------|-------------------------------|---|
| == | Equal | <code>\$x == \$y</code> | Returns true if \$x is equal to \$y |
| === | Identical | <code>\$x === \$y</code> | Returns true if \$x is equal to \$y, and they are of the same type |
| != | Not equal | <code>\$x != \$y</code> | Returns true if \$x is not equal to \$y |
| <> | Not equal | <code>\$x <> \$y</code> | Returns true if \$x is not equal to \$y |
| !== | Not identical | <code>\$x !== \$y</code> | Returns true if \$x is not equal to \$y, or they are not of the same type |
| > | Greater than | <code>\$x > \$y</code> | Returns true if \$x is greater than \$y |
| < | Less than | <code>\$x < \$y</code> | Returns true if \$x is less than \$y |
| >= | Greater than or equal to | <code>\$x >= \$y</code> | Returns true if \$x is greater than or equal to \$y |
| <= | Less than or equal to | <code>\$x <= \$y</code> | Returns true if \$x is less than or equal to \$y |

PHP INCREMENT / DECREMENT OPERATORS

| Operator | Name | Description |
|--------------------|----------------|--|
| <code>++\$x</code> | Pre-increment | Increments <code>\$x</code> by one, then returns <code>\$x</code> |
| <code>\$x++</code> | Post-increment | Returns <code>\$x</code> , then increments <code>\$x</code> by one |
| <code>--\$x</code> | Pre-decrement | Decrements <code>\$x</code> by one, then returns <code>\$x</code> |
| <code>\$x--</code> | Post-decrement | Returns <code>\$x</code> , then decrements <code>\$x</code> by one |

PHP LOGICAL OPERATORS

| Operator | Name | Example | Result |
|----------|------|-------------|---|
| and | And | \$x and \$y | True if both \$x and \$y are true |
| or | Or | \$x or \$y | True if either \$x or \$y is true |
| xor | Xor | \$x xor \$y | True if either \$x or \$y is true, but not both |
| && | And | \$x && \$y | True if both \$x and \$y are true |
| | Or | \$x \$y | True if either \$x or \$y is true |
| ! | Not | !\$x | True if \$x is not true |

PHP STRING OPERATORS

| Operator | Name | Example | Result |
|----------|--------------------------|------------------|------------------------------------|
| . | Concatenation | \$txt1 . \$txt2 | Concatenation of \$txt1 and \$txt2 |
| .= | Concatenation assignment | \$txt1 .= \$txt2 | Appends \$txt2 to \$txt1 |

PHP ARRAY OPERATORS

| Operator | Name | Example | Result |
|-----------------------|--------------|-------------------------------|---|
| + | Union | <code>\$x + \$y</code> | Union of <code>\$x</code> and <code>\$y</code> |
| <code>==</code> | Equality | <code>\$x == \$y</code> | Returns true if <code>\$x</code> and <code>\$y</code> have the same key/value pairs |
| <code>===</code> | Identity | <code>\$x === \$y</code> | Returns true if <code>\$x</code> and <code>\$y</code> have the same key/value pairs in the same order and of the same types |
| <code>!=</code> | Inequality | <code>\$x != \$y</code> | Returns true if <code>\$x</code> is not equal to <code>\$y</code> |
| <code><></code> | Inequality | <code>\$x <> \$y</code> | Returns true if <code>\$x</code> is not equal to <code>\$y</code> |
| <code>!==</code> | Non-identity | <code>\$x !== \$y</code> | Returns true if <code>\$x</code> is not identical to <code>\$y</code> |

Conditional Statements (Branching)

1. If...else Statement - single decision stmt

Syntax

```
if (<condition>)  
{  
    // True part  
    Php code  
}  
else  
{  
    // False part  
    php code  
}
```

E.g.

```
if ($num1 > $num2)  
{  
    $max = $num1;  
}  
else  
{  
    $max = $num2;  
}
```


Conditional Statements (Branching)

2. If...else if Statement - multiple decision stmt

Syntax

```
if (<condition1>)  
{  
    // True part Php code  
}  
else if (<condition2>)  
{  
    // 1st False 2nd true part  
    php code  
}  
else  
{  
    // false part PhP code  
}
```

E.g.

```
if ($num1 > $num2)  
{  
    $max = $num1;  
}  
else if ($num2 > $num3)  
{  
    $max = $num3;  
}  
else  
    $max = $num2;
```

Conditional Statements (Branching)

3. Switch Statement - replacement of "if else if stmt".

Syntax

```
switch (expression)
{
    case value1: {stmts1;
break;}
    case value2: {stmts2;
break;}
    .....
    .....
    [default: stmts;]
}
```

E.g.

```
switch ($day)
{
    case 1: {echo "its Sunday";
break;}
    case 2: {echo "its Monday";
break;}
    .....
    .....
    case 7: {echo "its Saturday";
break;}
    default: {echo "not specified
value";}
}
```

Conditional Statements (Looping)

4. For Statement - replacement of "if else if stmt".

Syntax

```
for( initial_expression;  
    termination_check;  
    index updation)  
{  
    statement (S);  
}
```

E.g.

```
for($i=1; $i < 10; $i++)  
    echo $i;
```

(or)

```
$i=1;  
for(; $i<10; )  
{  
    echo $i;  
    $i=$i+1;  
}
```

FOREACH LOOP

The *foreach* construct provides an easy way to iterate over arrays.

foreach works only on arrays and objects.

foreach will issue an error when you try to use it on a variable with a different data type or an uninitialized variable.

Syntax : 1 `foreach (array_expression as $value)`
 `statement // indexed array`

Syntax : 2 `foreach (array_expression as $key => $value)`
 `statement // associative array`

When *foreach* first starts executing, the internal array pointer is automatically reset to the first element of the array.

Indexed Array :

```
<?php
$array = array(1,2,3,4);
foreach($array as $val) {
    print $val; }
?>
```

Associative Array :

```
<?php
$arr = array(1=>'ABC',2=>'PQR');
foreach ($arr as $key => $val) {
    print "$key = $val\n";
}
?>
```

Conditional Statements (Looping)

5. Do ... While Statement

Syntax

```
do
{
    Statement (S);
}
while (<condition>);
```

E.g.

```
do
{
    $i=$i+1;
    echo $i;
} while ($i < 10);
```

Conditional Statements (Looping)

5. While Statement

Syntax

```
while(<condition>)  
{  
    Statement (S);  
}
```

E.g.

```
while ($i < 10)  
{  
    $i=$i+1;  
    echo $i;  
}
```

```
<html>
<body>
<form method=POST action="<?php $_SERVER["PHP_SELF"]; ?>">
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Enter your name &nbsp;<input type=text name="tname" value=" ">
<br>
<input type=submit name="submit" value="Click">
</form>
<?php
//if ($_SERVER["REQUEST_METHOD"] == "POST")
if($_POST['submit']=="Click")
{
$var=$_POST['tname'];
echo "Your name is ".$var;
}
?>
</body>
</html>
```