2- PHP Data Types, Variables, Constant, Operators,

PHP Comments PHP Data Types

A Data type is the classification of data into a category according to its attributes;

- Alphanumeric characters are classified as strings
- Whole numbers are classified integers
- Numbers with decimal points are classified as floating points.
- True or false values are classified as Boolean.

PHP is a loosely typed language; it does not have explicit defined data types. PHP determines the data types by analyzing the attributes of data supplied. PHP implicitly supports the following data types

- Integer – whole numbers e.g. -3, 0, 69. The maximum value of an integer is platform-dependent. On a 32 bit machine, it's usually around 2 billion. 64 bit machines usually have larger values. The constant PHP_INT_MAX is used to determine the maximum value. <?php echo PHP_INT_MAX;

  ?>

  Output:

  9223372036854775807
- Floating point number – decimal numbers e.g. 3.14. they are also known as double or real numbers. The maximum value of a float is platform-dependent. Floating point numbers are larger than integers. □ Character string – e.g. Hello World □ Boolean – e.g. True or false.

Before we go into more details discussing PHP data types, let's first discuss variables.

PHP Variable

A variable is a name given to a memory location that stores data at runtime.

The scope of a variable determines its visibility.

A Php global variable is accessible to all the scripts in an application.

A local variable is only accessible to the script that it was defined in.

Think of a variable as a glass containing water. You can add water into the glass, drink all of it, refill it again etc.

The same applies for variables. Variables are used to store data and provide stored data when needed. Just like in other programming languages, PHP supports variables too.

Let's now look at the rules followed when creating variables in PHP.

- All variable names must start with the dollar sign e.g.
- $my_var
- Variable names are case sensitive; this means $my_var is different from $MY_VAR
- $my_var ≠ $MY_VAR
- All variables names must start with a letter follow other characters e.g. $my_var1. $1my_var is not a legal variable name.
- ✓ $my_var1; ✗ ~~$1my_var;~~
- Variable names must not contain any spaces, "$first name" is not a legal variable name. You can instead use an underscore in place of the space e.g. $first_name. You cant use characters such as the dollar or minus sign to separate variable names.
- ✓ $my_var; ✗ $my var

Let's now look at how PHP determines the data type depending on the attributes of the supplied data.

```
<?php $my_var
= 1; echo
$my_var;
```

```
?>
```

Output:

1

Floating point numbers

```
<?php
$my_var = 3.14; echo
$my_var;
?>
```

Output:

3.14

Character strings

```
<?php
$my_var ="Hypertext Pre Processor";
echo $my_var; ?> Output:
```

Hypertext Pre Processor

Use of Variables

Variables help separate data from the program algorithms.

The same algorithm can be used for different input data values.

For example, suppose that you are developing a calculator program that adds up two numbers, you can create two variables that accept the numbers then you use the variables names in the expression that does the addition.

Variable Type Casting

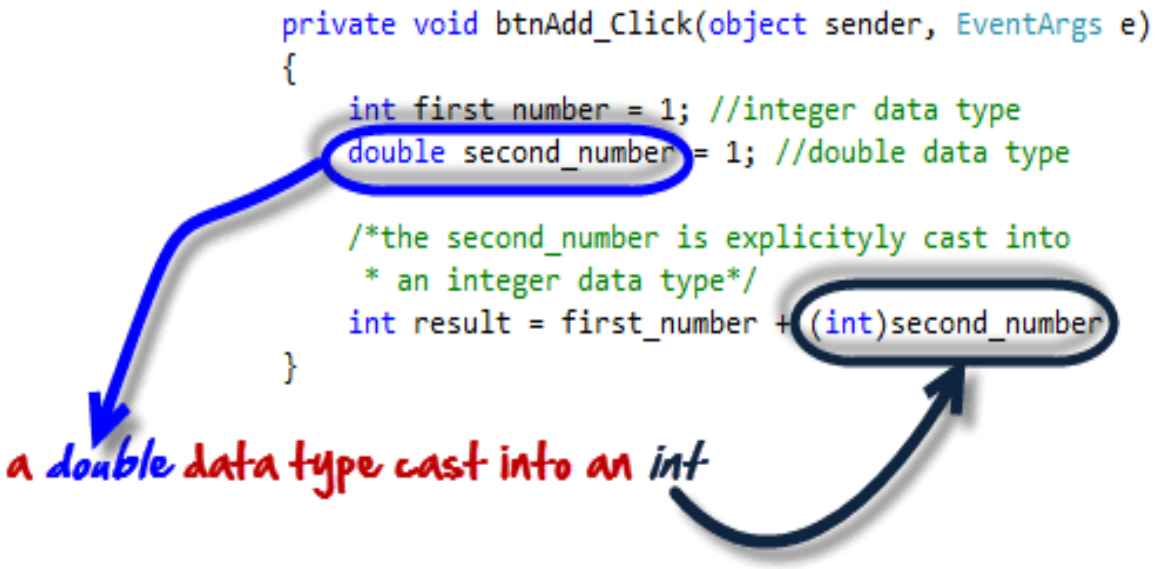
Performing arithmetic computations using variables in a language such as C# requires the variables to be of the same data type.

Type casting is converting a variable or value into a desired data type.

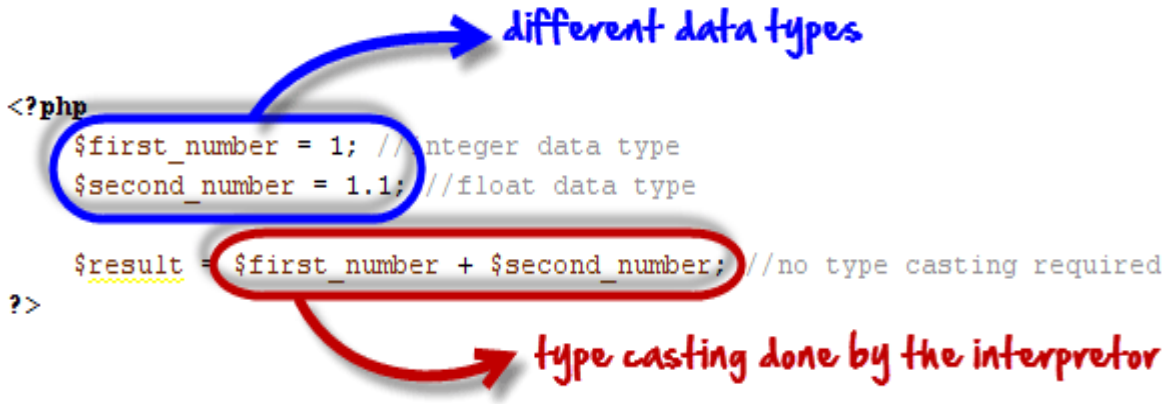
This is very useful when performing arithmetic computations that require variables to be of the same data type.

Type casting in PHP is done by the interpreter.

In other languages such as C#, you have to cast the variables. The code below shows type casting in C#.



The diagram below shows PHP implementing the above example.



PHP also allows you to cast the data type. This is known as explicit casting. The code below demonstrates explicit type casting.

```
<?php
$a = 1;
$b = 1.5;
$c = $a + $b; $c =
$a + (int) $b;
echo $c;
```

```
?>
```

Output:
2

The var_dump function is used to determine the data type. The code below demonstrates how to use the var_dump function.

```
<?php $a = 1;
var_dump($a); $b
= 1.5;
var_dump($b); $c
= "I Love PHP";
var_dump($c); $d
= true;
var_dump($d);
?>
```

Output:
int(1) float(1.5) string(10) "I Love PHP" bool(true)

PHP Constant

Define constant– A constant is a variable whose value cannot be changed at runtime. Suppose we are developing a program that uses the value of PI 3.14, we can use a constant to store its value.

Let's now look at an example that defines a constant. define('PI',3.14); //creates a constant with a value of 3.14 Once you define PI as 3.14 , writing a code like below will generate an error PI = 4; //PI has been defined as a constant therefore assigning a value is not permissible.

PHP Operators

Arithmetic operators

Arithmetic operators are used to perform arithmetic operations on numeric data. The concatenate operator works on strings values too. PHP supports the following operators.

Operator	Name	Description	Example	Output

+	Addition	Summation of x and y	1 + 1;	2
–	Subtraction	Difference between x and y	1 – 1;	0
*	Multiplication	Multiplies x and y	3 * 7;	21
/	Division	Quotient of x and y	45 / 5;	9
%	PHP Modulus	Gives remainder of dividing x and y	10 % 3;	1
-n	Negation	Turns n into a negative number	-(-5);	5
x . y	Concatenation	Puts together x and y	“PHP” . ” ROCKS”;10 . 3;	PHP ROCKS103

Assignment Operators

Assignment operators are used to assign values to variables. They can also be used together with arithmetic operators.

Operator	Name	Description	Example	Output
x = ?	assignment	Assigns the value of x to ?	$x = 5;	5
x += ?	addition	Increments the value of x by ?	$x = 2;$x += 1;	3
X -= ?	subtraction	Subtracts ? from the value of x	$x = 3;$x -= 2;	1
Operator	**Name**	**Description**	**Example**	**Output**
X *=?	multiplication	Multiplies the value of x ? times	$x = 0;$x *=9;	0
X /=?	division	Quotient of x and ?	$x = 6;$x /=3;	2

X %=?	modulus	The reminder of dividing x by?	$x = 3;$x %= 2;	1
X .=?	concatenate	Puts together items	” $x = ‘Pretty’;$x .= ‘ Cool!’;”	Pretty Cool!

Comparison operators

Comparison operators are used to compare values and data types.

Operator	Name	Description	Example	Output
X == y	Equal	Compares x and y then returns true if they are equal	1 == “1”;	True or 1
X === y	identical	Compares both values and data types.	1 === “1”;	False or 0. Since 1 is integer and “1” is string
X != y, x <> y	PHP Not equal	Compares values of x and y. returns true if the values are not equal	2 != 1;	True or 1
X > y	Greater than	Compares values of x and y. returns true if x is greater than y	3 > 1;	True or 1
X < y	Less than	Compares values of x and y. returns true if x is less than y	2 < 1;	False or 0
X >= y	Greater than or equal	Compares values of x and y. returns true if x is greater than or equal to y	1 >=1	True or 1
X <= y	Less than or equal	Compares values of x and y. returns true if x is greater than or equal to y	8 <= 6	False or 0

Logical operators

When working with logical operators, any number greater than or less than zero (0) evaluates to true. Zero (0) evaluates to false.

Operator	Name	Description	Example	Output

X and y, x && y	And	Returns true if both x and y are equal	1 and 4;True&& False;	True or 1False or 0
X or y, x || y	Or	Returns true if either x or y is true	6 or 9;0 || 0;	True or 1False or 0
X xor y	Exclusive or, xor	Returns true if only x is true or only y is true	1 xor 1;1 xor 0;	False or 0True or 1
!x	Not	Returns true if x is false and false if x is true	!0;	True or 1

PHP Comments

Why use Comments?

- If you don't work on the source code for some time, it's easy to forget what the code does. Commenting the source code helps remember what the code does.
- Commenting source code is also very important when multiple developers have to work on the same project. The changes made by one developer can be easily understood by other developers by simply reading the comments.
- As the best practice, you must have 3 lines of comments for every 10 lines of code

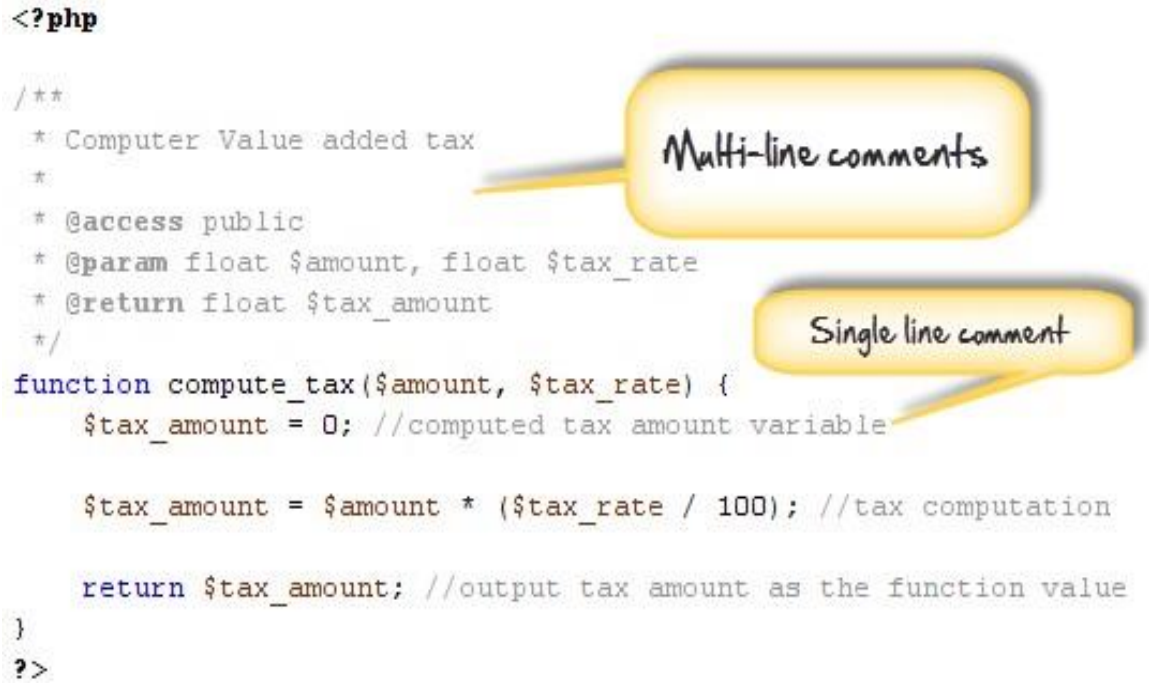
PHP Comments

- Comments help us to understand the code
- Comments are explanations that we include in our source code. These comments are for human understanding.
- Single line comments start with double forward slashes // and they end in the same line.
- `//`
- Multiple line comments start with a forward slash followed by the asterisk /* and end with the asterisk followed by the forward slash */.

```
/*this is a multiple-line
*comment
/*example
```

The diagram below shows a PHP file with both multiple line and single line comments

PHP Example




```php
<?php

/**
 * Computer Value added tax
 *
 * @access public
 * @param float $amount, float $tax_rate
 * @return float $tax_amount
 */
function compute_tax($amount, $tax_rate) {
    $tax_amount = 0; //computed tax amount variable

    $tax_amount = $amount * ($tax_rate / 100); //tax computation

    return $tax_amount; //output tax amount as the function value
}
?>
```

Summary

- PHP is a loosely typed language.
- Variables are memory locations used to store data
- The value of constants cannot be changed at runtime
- Type casting is used to convert a value or variable into a desired data type
- Arithmetic operators are used to manipulate numeric data
- Assignment operators are used to assign data to variables
- Comparison operators are used to compare variables or values
- Logical operators are used to compare conditions or values
- Comments are used to help understand source code. They are for human understanding
- Single line comment statements start with double forward slashes //.
- Multi-line comment statements are enclosed between /* statements */.