CSU 07307: Internet and Web Technologies

Lecture 3

Agenda:

- Variables
- Constants
- Operators

Variables Intro

- As your program is processed, it needs to hold information in its memory
- When calculating $5 + 6$ in your program, for instance, a memory is required to store number 5 and the result (after adding 6)
- Although the result of the above calculation is in a memory somewhere, it cannot be accessed and manipulated
Variables Intro

- A variable allows us to store values, like text and numbers in a memory and retrieve them.
- Think of a variable as a reference to a memory where through it you can store, change and access values stored.
- When a variable is declared, it can be used over and over again in your program.

Real World Application of Variables

- In a computer game, things like lives and health are variables that constantly change.
  - You can make a mistake and lose health/live or collect items/weapons and gain health/live.
- A “Shopping Cart” could use a variable to store information about items you buy.
  - As you choose more items from an online shop, the variable’s total increases.

Declaring Variables

- All variables in PHP should start with a dollar sign followed by a variable name.
  - A variable name should start with a letter or the underscore character.
  - If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString).
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9 and _).
Declaring Variables

• To be usable, a variable MUST be assigned a value to store

```php
<?php
$firstname;  // Declare
$firstname = "NabilAbioji";  // Assign a value to the variable. In most cases only the 3rd line is used to both declare and assign a value/

<?

– The examples in the previous slide, variables are declared without assigning any value to them

Declaring Variables

• PHP is a loosely type language
  – Data type of a variable is determined by a value assigned to the variable

```php
<?php
$a = "Hello";  // The first value is overridden by the second value */
```
Declaring Variables

- PHP supports eight data types
  - String
  - Integer
  - Float
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

Using Variables

- A value stored in a variable can be retrieved by simply referring to that variable

```php
<?php
$firstname = "Mkulili";
echo $firstname; /* Use the name of the variable and not the value assigned to the variable */
?>
```

- A declared variable can be used as many times as required in a program

Using Variables

- Variables names are case-sensitive. So you should always keep an eye on their cases

```php
<?php
$firstname = "Mkulili";
echo $firstname; /* You are going to get an error because the processor will be looking for $firstname with capital 'F', which it does not exist */
?>
```

- A variable is only used after being declared. Thus mismatch of cases between variables will result into an error
Using Variables

- When variables are used in double quotes (""), PHP processor uses a dollar sign ($) to differentiate a variable from a string

```
<?php
firstname = "Mpilupilu";
echo "Hello $firstname"; /* The output will be Hello Mpilupilu because the processor knows that $firstname is a variable so it access the stored value */
?>
```

Constants Intro

- Variables give us flexibility of using the same value over and over again thus limiting typos
  - For instance whenever $firstname is referred, the value ‘Mpilupilu’ is accessed

```
<?php
 firstname = "Mpilupilu";
 echo "Hello $firstname"; /* The output will be Hello Mpilupilu because the processor knows that $firstname is a variable so it access the stored value */
?>
```

Constants Intro

- Values in variables, however, can be changed and thus with variables we can never be sure that the same value is used over and over
  - If the value of $firstname is later changed to ‘Mponjoro’, then reference to $firstname will give us ‘Mponjoro’ and not ‘Mpilupilu’

```
<?php
 firstname = "Mpilupilu";
 echo "Hello $firstname";
 firstname = "Mponjoro";
 echo "Hello $firstname";
?>
```
Constants Intro

• To ensure that the same value is used over and over again, constants can be used
• Constants are like variables except that values stored in them cannot be changed once they are defined
• Unlike variables, constants are automatically global across the entire program/script

Real World Application of Constants

• When dealing with applications that deals with calculations
  – You can use a constant to store a value of a π i.e 22/7 or 3.14
  – You can use a constant to store value of a VAT i.e 18% or 0.18
• The values need to be used over and over again without being changed

Declaring Constants

• A valid constant name starts with a letter and the underscore (no $ sign before the name)
  – All other rules for declaring variables apply
• Syntax
  
  define (name, value, case-sensitive)

  name: constant name 
  value: value to be assigned 
  case-sensitive: specifies whether the constant name should be case sensitive or not
Declaring and Using Constants

• Declaring a case-sensitive constant

```php
define("VAT", 0.18);
/* If case-sensitive is not specified, it is by default false (meaning
the letters should have the same caps as specified in when defining the constant)
*/
```

Declaring and Using Constants

• Declaring a case-insensitive constant

```php
define("VAT", 0.18, true);
/* The letters of the constant
name should not have the same
caps as specified when defining the constant
*/
```

PHP Operators

• Operators are special symbols used to perform operations on variables and values

• In PHP, we have 7 groups of operators but in this session we are going to discuss 3 groups;
  – String Operators
  – Arithmetic Operators
  – Assignment Operators
PHP Operators: String

- PHP has two operators designed for strings;

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Concatenation</td>
<td>Put two or more string values together</td>
</tr>
<tr>
<td>.</td>
<td>Concatenation assignment</td>
<td>Joins one string value to another</td>
</tr>
</tbody>
</table>

- Concatenation assignment only works with string values stored in variables.

PHP Operators: String

- Below are examples of using string operators;

```php
<?php

$firstname = "James";
$surname = "Brown";

// The statement below shows how five string values can be joined using the concatenation operator.
echo "This is ",$firstname," ",$surname," ";

// This will raise an error because . only works with string values from variables */
// This will not raise an error because . is used with variables */
```

PHP Operators: Arithmetic

- Arithmetic operators are used with numeric values to perform arithmetic operations.

- PHP has 6 arithmetic operators;

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>$a + $b</td>
<td>Add two or more numbers</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>$a - $b</td>
<td>Subtract a number from another number</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>$a * $b</td>
<td>Multiply two or more numbers</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>$a / $b</td>
<td>Divide a number from another number.</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
<td>$a % $b</td>
<td>Returns a remainder of a division</td>
</tr>
<tr>
<td>**</td>
<td>Exponentiation</td>
<td>$a ** $b</td>
<td>$a power of $b</td>
</tr>
</tbody>
</table>
PHP Operators: Arithmetic

- Division operator returns a float value unless the two numbers are integers
- Exponentiation operator is introduced in PHP 5.6
- In other cases – symbol is used as a negation operator, which is used to negate a value

PHP Operators: Arithmetic

- Below are examples of using arithmetic operators;

```php
<?php
    $a = 10;
    $b = 5;
    echo $a + $b."\n"; // The output is 15
    echo $a - $b."\n"; // The output is 5
    echo $a * $b."\n"; // The output is 50
    echo $a / $b."\n"; // The output is 2
    echo $a % $b; // The output is 0
?>
```

PHP Operators: Arithmetic

- Below is an example of using constant and an arithmetic operator;

```php
<?php
    $price = 10000;
    define("VAT", 0.18, true);
    echo $price*VAT; /* This example illustrates how a constant can be used in a real world scenario */
?>
```
PHP Operators: Assignment

- Assignment operators are used to assign values into variables
- PHP has 6 assignment operators;

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Same as</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>$a += $b</td>
<td>$a = $a + $b</td>
</tr>
<tr>
<td>-=</td>
<td>$a -= $b</td>
<td>$a = $a - $b</td>
</tr>
<tr>
<td>*=</td>
<td>$a *= $b</td>
<td>$a = $a * $b</td>
</tr>
<tr>
<td>/=</td>
<td>$a /= $b</td>
<td>$a = $a / $b</td>
</tr>
<tr>
<td>%=</td>
<td>$a %= $b</td>
<td>$a = $a % $b</td>
</tr>
</tbody>
</table>

PHP Operators: Assignment

- Below are examples of using assignment operators;

```php
<?php
  $a = 10; // Assign 10 to a variable
  $b = 5;
  echo $a += $b; // Assign by adding what is existing in the variable. This is a short form of $a = $a + $b;
  $a /= $b; // Assign by dividing
  $a -= $b; // Assign by subtracting
  $a *= $b; // Assign by multiplying
  echo $a %= $b; // Assign by dividing

  var_dump($a);
?>
```