

# PHP BASIC

# BEFORE WE GET STARTED

1. Installed Git.
2. Installed Docker (recommend) or LEMP/LAMP stack or XAMPP/WAMP.
3. Installed Text Editor VS Code (recommend), Atom or IDE PHP Storm (recommend).
4. Chrome Web Browser (optional).
5. Checkout :  
<https://hub.docker.com/r/namnh06/lemp/>
6. Checkout :  
<https://github.com/namnh06/aptech-php-course>

# SET UP ENVIROMENT

1. Turn on Git Bash (on Windows – recommend)/ Terminal (on Ubuntu).
2. Type in : `docker pull namnh06/lamp` and waiting to pull image docker.
3. Do the following things in hub docker : <https://hub.docker.com/r/namnh06/lamp>.
4. Access into docker : `docker exec -it -u0 lamp bash`.
5. Move to : `cd /var/www/html && git clone https://github.com/namnh06/aptech-php-course.git`

WHAT IS THE DOCKER ?

MORE INFORMATION ABOUT MY DOCKER :

<https://hub.docker.com/r/namnh06/lamp/>

# SYNTAX, ECHO, COMMENT

- PHP files have “.php” extension.
- PHP script always starts with <?php and end with ?>, in some case no need to close ?> in file .php. The end of statement of code always need “;”.
- Comment starts with “//”, “#” or “/\* \*/”.
- Output into HTML Dom by using “echo”, “print”, “print\_r” or “var\_dump”.
- Can write HTML, CSS, Javascript inside “.php” file.

```
35 <?php
36 // This is a single-line comment
37
38 # This is also a single-line comment
39
40 /*
41 This is a multiple-lines comment block
42 that spans over multiple
43 lines
44 */
45
46 echo "Hello World";
```

# VARIABLE, CONSTANT

- Variable starts with “\$” (dollar sign), no need to declare type of variable.
- Variable name must start with a letter or underscore character “\_” and can not start with number.
- Variable name can only contain alpha-numeric characters and underscores (A-z, 0-9 and \_).
- Using “global”, “static” keyword to make variable “special”.
- Using function “define” to create a constant.

```
1 <?php
2 $currentYear = 2018;
3
4 function forecast()
5 {
6     global $currentYear;
7     $nextYear = ++$currentYear;
8     echo "Next Year is $nextYear";
9 }
10
11 forecast();
```

```
1 <?php
2 define("CURRENT_YEAR", 2018);
3
4 function forecast()
5 {
6     $nextYear = CURRENT_YEAR + 1;
7     echo "Next Year is $nextYear";
8 }
9
10 forecast();
```

# DATA TYPES

- String : a sequence of characters.
- Integer : non-decimal number between -2,147,483,648 and 2,147,483,647.
- Float : decimal number.
- Boolean : TRUE/FALSE.
- Array : stores multiple values.
- Object : stores data and information or properties and function.
- NULL : is NULL.

```
1  <?php
2
3  $string = "Hello World";
4
5  $integer = 2018;
6
7  $float = 99.99;
8
9  $boolean = true;
10
11 $array = [1, 2, 3, 4, 5];
```

# OPERATORS

## 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)

```
1 <?php
2 $three = 3;
3 $nine = 9;
4
5 // Additional
6 $twelve = $three + $nine;
7
8 // Subtraction
9 $six = $nine - $three;
10
11 // Multiplication
12 $twentySeventh = $nine * $three;
13
14 // Division
15 $anotherThree = $nine / $three;
16
17 // Modulus
18 $zero = $nine % $three;
19
20 // Exponentiation
21 $sevenHundredAndTwentyNinth = $nine ** $three;
```

# OPERATORS (cont. 1)

- 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



# OPERATORS (cont. 2)

## Comparison Operators

## Increment/Decrement Operators

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

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

# OPERATORS (cont. 3)

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

- String Operators

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

- Logical Operators

# OPERATORS (cont. 4)

- 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>&lt;&gt;</code>	Inequality	<code>\$x &lt;&gt; \$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>

# CONDITION STATEMENTS

- If
- If ... else
- If ... else if ... else
- Switch case

```
1  <?php
2
3  $number = 10;
4
5  if ($number === 10) {
6      echo "$number === 10";
7  } else {
8      echo "$number is someone else";
9  }
10
11 if ($number > 10) {
12     echo "$number is greater than 10.";
13 } else if ($number < 10) {
14     echo "$number is less than 10.";
15 } else {
16     echo "$number is equal 10.";
17 }
```

```
1  <?php
2  define("SPRING", 5);
3  define("SUMMER", 10);
4  define("AUTUMN", 15);
5  define("WINTER", 20);
6  $yourCoupon = "spring";
7  $discountPercent = 0;
8
9  switch (strtoupper($yourCoupon)) {
10     case "SPRING":
11         $discountPercent = SPRING;
12         break;
13     case "SUMMER":
14         $discountPercent = SUMMER;
15         break;
16     case "AUTUMN":
17         $discountPercent = AUTUMN;
18         break;
19     case "WINTER":
20         $discountPercent = WINTER;
21         break;
22     default:
23         $discountPercent = 0;
24 }
25
26 echo $discountPercent;
```

# LOOPS

- While
- Do ... While
- For

```
1  <?php
2
3  $number = 0;
4
5  do {
6      echo "$number";
7      $number++;
8  } while ($number < 5);
```

```
1  <?php
2
3  $number = 0;
4
5  for ($number; $number < 5; $number++) {
6      echo "$number";
7  }
```

- Foreach

```
1  <?php
2
3  $array = ["ten" => 10, "nine" => 9];
4
5  foreach ($array as $key => $value) {
6      echo "$key is $value <br>";
7  }
```

# FUNCTIONS

- A function name can start with a letter or underscore (not a number).
- Give the function a name that reflects what the function does!
- PHP has more than 1000 built-in functions.

```
1  <?php
2
3  define("variable", 1);
4  $array = array(1, 2, 3, 4, 5);
5  count($array);
6  date("Y/m/d H:i:s");
7  rsort($array);
8  strotoupper("hello");
```

```
1  <?php
2
3  function displayName($yourName = "Nam NH")
4  {
5      echo "Hello $yourName<br>";
6  }
7
8  displayName();
9  displayName("Smith");
```

# SUPERGLOBALS

- Built-in variables that are always available in all scopes.
- Always accessible, regardless of scope, can access from any function or file.

❖ \$GLOBALS

❖ \$\_SERVER

❖ \$\_REQUEST

❖ \$\_POST

❖ \$\_GET

❖ \$\_FILES

❖ \$\_ENV

❖ \$\_COOKIE

❖ \$\_SESSION

**TIME TO PRACTICE.**