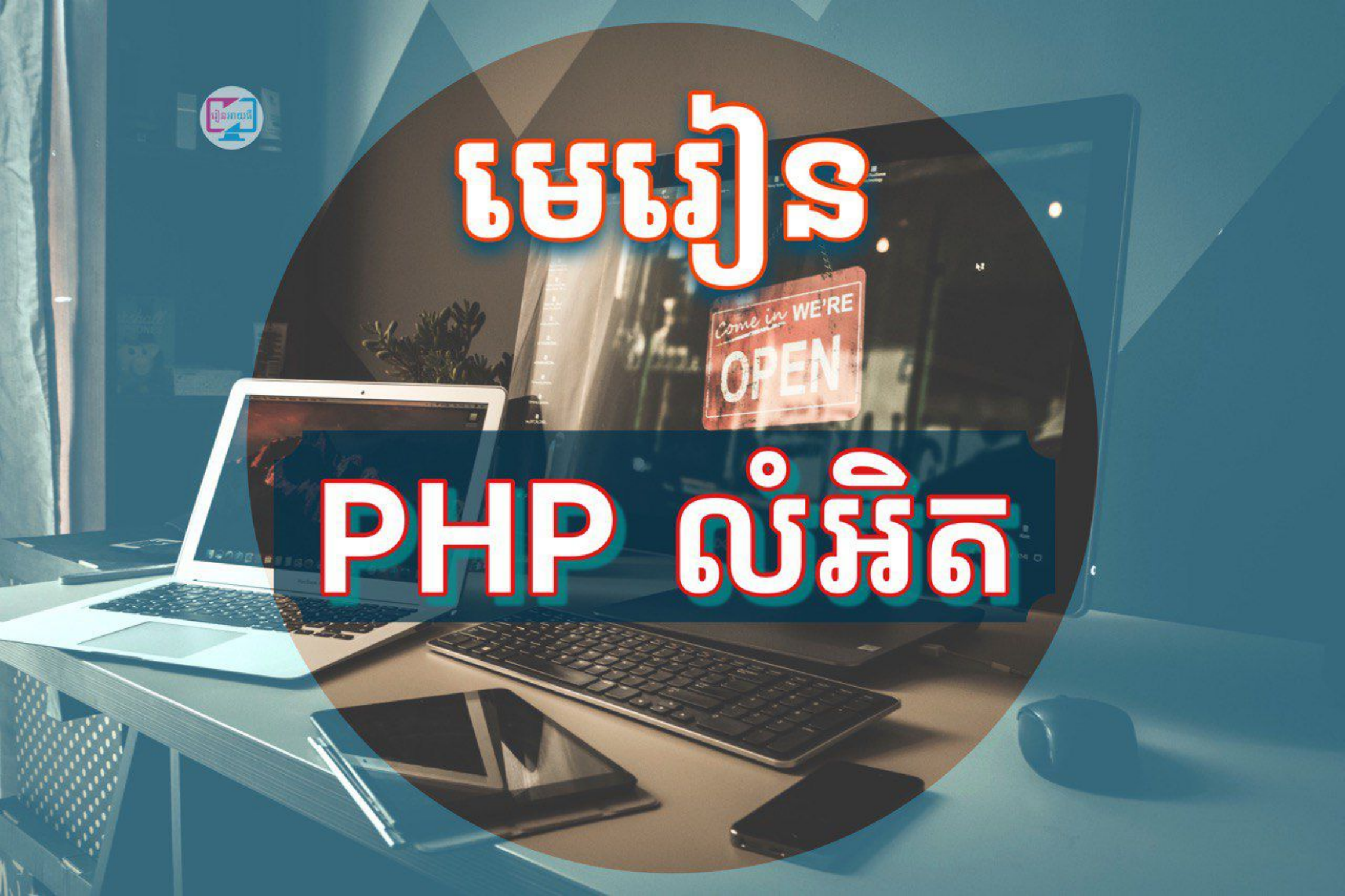
មេរៀន
Come in WE'RE
OPEN
PHP លំអិត

មេរៀនទី ១

Introduction PHP

History

PHP ត្រូវបានផ្តល់ឈ្មោះជាផ្លូវការ HyperText Preprocessor វាជាភាសាដែលតំណើរការនៅលើ Serverដែលជា ទូទៅត្រូវបាន សរសេរក្នុងបរិបទ HTML ប៉ុន្តែមិនដូចជា HTML page ធម្មតានោះទេPHP script មិនត្រូវបានបញ្ជូន ទៅអោយ client ដោយ server ផ្ទាល់នោះទេ វាធ្វើការបញ្ជូនដោយ PHP engine ។ PHP code ដែលឋិតនៅក្នុង script អាចប្រើដើម្បីធ្វើការជាមួយ Databases, បង្កើតជារូបភាព ,អាន និងបញ្ចូលទិន្នន័យឬ កែរប្រែអត្ថបទដែលមាននៅខាងក្នុង files ឬ ធ្វើការទំនាក់ទំនងទៅកាន់ Remote Server ព្រមជាមួយនឹងលទ្ធភាពជាច្រើនផ្សេងទៀត ។

Rasmus Lerdorf គឺជាមនុស្សដំបូងដែលអភិវឌ្ឍន៍នូវ PHP/FI ហើយត្រូវបានមនុស្សរាប់លានអ្នកប្រើប្រាស់វា ។ ជំនាន់តំបូងនៃ PHP/FI ឈ្មោះថា **Personal Homepage Tools/Form Interpreter**ដែលលក្ខណៈជាមួលដ្ឋានគឺស្រដៀងទៅនឹងភាសា Perl ព្រោះបានប្រមូលផ្តុំនូវ Perl scripts មកអភិវឌ្ឍន៍ នៅក្នុងអំឡុងឆ្នាំ ១៩៩៥ ប៉ុន្តែវានៅមានកង្វះខាតជាច្រើននៅក្នុងភាសានេះ ដូចជា for loops ជាដើម ។

PHP/FI 2

នៅក្នុងឆ្នាំ ១៩៩៧ Rasmus បន្តការអភិវឌ្ឍន៍នូវ PHP/FI 2 រហូតមកដល់ខែ វិច្ឆិកា ឆ្នាំដដែលបន្ទាប់ពី Andi Gutmans ហើយនិង Zeev Suraski បានប្រទះឃើញនូវ PHP/FI ចំពេលដែលពួកគេកំពុងស្វែងរកនូវភាសាដើម្បីអភិវឌ្ឍន៍នូវគំរោងបង្កើត E-commerce solution សំរាប់សកលវិទ្យាល័យ របស់ពួកគេ ។ ពួកគេបានអោយដឹងថា PHP/FI ពុំទាន់មានលទ្ធភាពគ្រប់គ្រាន់នៅឡើយហើយខ្វះខាតនូវ លក្ខណៈពិសេសជាច្រើន ។ ចំនុចមួយដែលមានការចាប់អារម្មណ៍ជាងគេ គឺ while loops ដែលពួកគេនឹងត្រូវប្រតិបត្តិ ។

PHP 3

Zeev and Andi សំរេចចិត្តសរសេរ scripting language ឡើងវិញប៉ុន្តែពុំវត្តមាន Rasmusចូលរួមដើម្បីអភិវឌ្ឍន៍នូវ PHP3ឡើយហើយបានផ្តល់នូវឈ្មោះថ្មីថា**Hypertext Preprocessor**ដើម្បីបញ្ជាក់ថា PHPគឺជាផលិតផលផ្សេងមួយទៀតហើយនឹងមិនត្រឹមតែយកមកប្រើសំរាប់តែការងារផ្ទាល់ ខ្លួននោះទេ ។ Zeev and Andi ក៏បានបង្កើតនូវ Extension API ដែល API បង្កើតថ្មីនេះវាមានលទ្ធភាពបំពេញនូវការងារជាច្រើនដូចជា Accessing databases , spell checkers ហើយនឹង បច្ចេកវិទ្យាដទៃៗទៀត ដែលធ្វើអោយមានការចាប់អារម្មណ៍ពីសំណាក់អ្នកអភិវឌ្ឍន៍ ជាច្រើនមកចូលរួមក្នុងគំរោង PHP ។ នៅខណៈពេលនោះដែរ PHP ក៏ត្រូវបានបញ្ចេញញូវជំនាន់ថ្មីរបស់ខ្លួនគឺ PHP 3 នៅ ថ្ងៃទី ៣ ខែ មិថុនា ឆ្នាំ ១៩៩៨ ដែលតាមការប៉ាន់ស្មាន PHP នឹងត្រូវបានតំឡើង ប្រមាណ ជាង ៥0 000 domains ប៉ុន្តែជាមួយតួលេខពិតប្រាកដ លើកដំបូងរបស់ PHP ត្រូវបានគេតំឡើងច្រើនជាងមួយលាន Domain ទៅទៀត ។

PHP 4

នៅក្រោយឆ្នាំ ១៩៩៨ Zeev និង Andi ងាកទៅពិតនិត្យលើការងារ PHP 3 ហើយពួកគេមានគំនិតថាពួកគេអាចសរសេរនូវ script language ឡើងវិញអោយមានលក្ខណៈល្អប្រសើរជាងមុនទៅទៀត ក្នុងខណៈពេលដែល PHP 3 កំពុងបន្តការធ្វើសម័ន្ធ និងប្រតិបត្តិការនោះ PHP 4 ក៏ ចាប់ផ្តើមបង្កើតនូវគំរូថ្មី គឺ “compile first, execute later.” តំណាក់កាលនៃការ compile មិនត្រូវបាន complie PHP Script អោយទៅជា machine code នោះទេ វាជំនួសដោយការ compile ទៅជា byte code ដែលធ្វើ ការប្រតិបត្តិការ ដោយ **Zend Engine** (Zend មកពីពាក្យថា **Ze**ev និង **And**i) ។ វិធីសាស្ត្រថ្មីសំរាប់ការប្រតិបត្តិ script នេះអាចធ្វើអោយ PHP 4 តំណើរការបានល្អប្រសើរច្រើនជា PHP 3 ហើយត្រូវបានដាក់បង្ហាញនូវ PHP 4 នេះក្នុង ខែ ឧសភា ឆ្នាំ ២00២ ប៉ុន្តែដោយមានការផ្លាស់ប្តូរនៅក្នុងភាសានេះជាបន្តបន្ទាប់ទើប PHP 4 បានបង្កើតនូវជំនាន់របស់ខ្លួន ជា PHP 4.1.0 នឹង បានបង្ហាញនូវ **Superglobals** ដូចជា $_GET និង $_POST ។ ដែល Superglobals នេះអាចយកមកប្រើប្រាស់ពី ខាងក្នុង Functions ដោយមិនចាំបាច់ប្រើ global keyword ។ រហូតដល់ជំនាន់ចុងក្រោយរបស់ PHP4 ត្រូវបាន បង្ហាញ ជាចុងក្រោយបង្អស់ នៅថ្ងៃទី 27 ខែ ធ្នូ ឆ្នាំ ២00២ ។

PHP 5

ដោយមានតំរូវការជាច្រើននូវលក្ខណៈរបស់ object-oriented Andi ក៏មានគំនិតសរសេរនូវObject-Oriented សំរាប់ជាផ្នែកនៃ Zend Engine. Zeev and Andi បានសរសេរនូវឯកសារអំពី “ Zend Engine II : Feature Overview and Design ” ហើយចាប់ផ្តើមពិភាក្សាគ្នាអំពី PHP’s ទៅថ្ងៃអនាគតដែលក្នុងជំនាន់ PHP 5 មានចំនុចជាច្រើនដែលនឹងត្រូវកែប្រែ បន្ថែម ឬ រំលោះចោល ។
PHP’s មិនត្រឹមតែប្រែប្រួលដោយអាចអោយប្រើប្រាស់នូវលក្ខណៈ Object-Oriented ប៉ុណ្ណោះនោះទេ វាថែមទាំង ផ្តុំកនូវមុខងារថ្មីៗជាច្រើនដែលបញ្ចូលជាមួយមុខងារសំរាប់ XML ហើយជាពិសេសនោះ គឺ SimpleXML extension ដែលធ្វើអោយមានភាពងាយស្រួល ក្នុងការសម្របសម្រួលជាមួយឯកសារ XML និង SOAP ឬ MySQLi ថ្មី ហើយនឹង extensions ផ្សេងៗទៀតដែលជាចំនុចសំខាន់នៅក្នុង PHP’s ។ គេរំពឹងថា PHP 5 នឹងអាចក្លាយជាអ្នកនាំមុខគេ នៅលើទីផ្សារនៃការអភិវឌ្ឍន៍ web ។

មេរៀនទី ២

Building Block

1.អថេរ (Variables)

Variableគឺជាគ្រឹះដ៏សំខាន់សំរាប់ភាសាកុំព្យូទ័រវាត្រូវបានគេប្រើដើម្បីផ្ទុកនូវតំលៃជាបណ្តោះអាសន្ននៅពេលដែលប្រតិបត្តិការ script ម្តងៗ ។ variable ត្រូវបានផ្តល់តំលៃទៅអោយ នៅពេលដែល Script ចាប់ផ្តើមតំណើរការ ឬ នៅពេលអ្នកប្រើប្រាស់បញ្ចូលឬបានមកពីការចាប់យកទិន្នន័យពី Databaseអ្នកអាចប្រើប្រាស់ variable គ្រប់ពេលទាំងអស់ នៅពេលដែល script ចាប់ផ្តើមតំណើរការជាមួយទិន្នន័យ ហើយអ្នកអាចកែប្រែ តំលៃដែល Variable នោះផ្ទុកពីការប្រតិបត្តិការ script មួយទៅកាន់ការប្រតិបត្តិការរបស់ script មួយផ្សេងទៀត រហូតដល់ពេលដែល script របស់អ្នកត្រូវបានបញ្ចប់ ។

សំរាប់ភាសា PHP ឈ្មោះរបស់ variable ត្រូវតែចាប់ផ្តើមដោយនិមិត្តសញ្ញា $ (dollar sign)អ្នកអាចផ្តល់នូវឈ្មោះរបស់ variable ជាតួអក្សរ តំលៃលេខ ឬ underscore (_) ក៏ប៉ុន្តែអ្នកមិនអាចប្រើអក្សរដកឃ្លាជាមួយឈ្មោះរបស់ variable នោះទេ ។

ឧទាហរណ៏ ខាងក្រោមគឺជាការផ្តល់ឈ្មោះទៅអោយ variable ដែលត្រឹមត្រូវ :

```
$sok ;
$sok_som_neang ;
$_Dara ;
$Chenda22 ;
```

ឧទាហរណ៏ ការផ្តល់ឈ្មោះទៅអោយ variable ដែល**មិនត្រឹមត្រូវ** :

```
$123 ;
$*ABC ;
$A+B ;
```

Variable របស់ PHP អាចផ្ទុកនូវប្រភេទទិន្នន័យដែលជាតំលៃលេខ តួអក្សរ object , arrayBooleans ហើយរាល់ចំនុះរបស់ variable អាចធ្វើការផ្លាស់ប្តូរគ្រប់ពេលវេលា ។ ជាទូទៅការប្រកាសVariable ឬ ការផ្តល់តំលៃទៅអោយ variable អ្នកអាចអនុវត្តន៏នូវ statement ដូចខាងក្រោម ។

```
$num1 = 5;
$num2 = 8;
```

ការប្រកាស variable ពីរ ខាងលើអ្នកបានប្រើប្រាស់នូវ assignment operator (=) ដែលនឹងរៀបរាប់លំអិតនៅក្នុងមេរៀន “ Operator and Expression “ ។បន្ទាប់ពីអ្នកបានផ្តល់នូវតំលៃទៅ variableរួចរាល់ហើយ អ្នកអាចយកវាមក ប្រើប្រាស់បានដូចខាងក្រោម :

print $num2; វាមានតំលៃស្មើនឹងការប្រើ print 8; ដូចឆ្នេះមានន័យថា $num2 ផ្ទុកតំម្លៃ ៨ ចូរចងចាំថារាល់ចុងបញ្ចប់នៃរបាយការណ៏របស់ PHP និមួយៗត្រូវតែបញ្ចប់ដោយសញ្ញា (;) semicolon ។

2 Data Types

ប្រភេទខុសៗគ្នានៃទិន្នន័យត្រូវបានប្រើប្រាស់នូវចំនួនសរុបរបស់អង្គចងចាំផ្សេងៗគ្នា ហើយវាអាចប្រព្រឹត្តទៅនៅពេលអ្នករៀបចំវានៅក្នុង script របស់អ្នក ។ មានភាសាកម្មវិធីមួយចំនួនទាមទារអោយអ្នកសរសេរកម្មវិធី ប្រកាសនូវប្រភេទ នៃទិន្នន័យ

ពីមុខ variable ដោយឡែកសំរាប់ភាសា PHP ការប្រើប្រាស់ variable គឺមានភាពងាយស្រួល ដោយវានឹងធ្វើការគណនានូវប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តិនៅពេលដែលអ្នកបានផ្តល់តំលៃទៅអោយវា ។

Standard Data Types		
ប្រភេទ	**ឧទាហរណ៏**	**ពិពណ៌នា**
Integer	`5`	ផ្ទុកនូវតំលៃលេខជាចំនួនគត់
Double	`3.234`	ផ្ទុកនូវតំលៃលេខជាចំនួនទសភាគ
String	`"hello"`	ផ្ទុកនូវតំលៃជាបន្តុំនៃតួអក្សរ
Boolean	`True`	ផ្ទុកនូវតំលៃពិសេសគឺ true ឬ false
Array		មេរៀនទី
Object		មេរៀនទី

ឧទាហរណ៏ ១

Gettype.php

```
01:    <html>
02:    <head><titile>Display Data Type</title>
03:    </head>
04:    <body>
05:
06:    <h2>Using gettype</h2>
07:
08:    <?php
09:    $testing;
10:    Print gettype($testing);// NULL
11:    $testing = 5;
12:    print gettype($testing); // integer
14:    print "<br />";
15:    $testing = "five";
16:    print gettype($testing); // string
17:    print "<br />";
18:    $testing = 5.0;
19:    print gettype($testing); // double
20:    print "<br />";
21:    $testing = true;
22:    print gettype($testing); // boolean
23:    print "<br />";
24:    ?>
```

```
25:
26:     </body>
27:     </html>
```

នៅពេលដែល variable $testing ត្រូវបានប្រកាសនៅបន្ទាត់ទី ០៩ ដោយមិនបានផ្តល់តំលៃទៅអោយវា ដូចច្នេះនៅពេលដែលប្រើប្រាស់ gettype() function ដើម្បីត្រួតពិនិត្យ variable នៅបន្ទាត់ទី១0 នោះអ្នកនឹង ទទួលបាននូវ លទ្ធផលជាអក្សរ Null ។ បន្ទាប់ពីនេះ variable $testing ត្រូវបានផ្តល់នូវតំលៃជាតួលេខនៅបន្ទាត់ទី ១១ គឺលេខ ៥ ដែលជាចំនួនគត់ ឬអ្នកអាចនិយាយបានថាតំលៃលេខដែលគ្មានក្បៀសដូច្នេះលទ្ធផលដែលនឹងទទួលបានបន្ទាប់ពីការប្រើប្រាស់នូវ gettype() function បន្ទាត់ទី១២ គឺ // integerចំណែកការផ្តល់នូវតំលៃ "five" ទៅអោយ variable $testing នៅបន្ទាត់ទី ១៥ គឺជាបន្តុំនៃតួអក្សរ ។នៅពេលដែលអ្នកចង់ធ្វើការជាមួយតំលៃជា string អ្នកត្រូវតែដាក់តំលៃនោះស្ថិតនៅក្នុងចន្លោះ សញ្ញា (“) Double quotation mark ឬ (‘) single quotation mark ។

តំលៃជាប្រភេទ double ត្រូវបានផ្តល់ទៅអោយ variable $testing ដែលស្ថិតនៅបន្ទាត់ទី ១៨ជាតំលៃលេខ 5.0 ដែលតំលៃនេះជាតំលៃលេខទសភាគ ឬ ជាតំលៃលេខដែលមានក្បៀស។ តំលៃជាប្រភេទ Boolean ត្រូវបានផ្តល់ទៅអោយ variable $testing នៅបន្ទាត់ទី ២១ ហើយតំលៃនេះអាចមានតែពីរប៉ុណ្ណោះគឺ true ឬ false ។

<u>សំគាល់ :</u>

ភាពខុសគ្នារវាការប្រើប្រាស់នូវ (“) double quotation mark និង (‘) single quotation mark

Double quotation mark អនុញ្ញាតិអោយយើងប្រើប្រាស់នូវ variable បញ្ចូលជាមួយ ពីព្រោះ PHP engine នឹងជំនួសនូវតំលៃដែលជា variable ។ សូមពិនិត្យមើលឧទាហរណ៏ខាងក្រោម-

```
$name = "Rithya"; .
print "hello, $name"; // hello,Rithya .
```

ប្រសិនបើអ្នកប្រើប្រាស់នូវ single quotation mark variable មិនត្រូវបានជំនួសដោយតំលៃនោះទេ ។

```
print 'hello, $name'; // hello, $name .
```

ឧទាហរណ៏ ២

<u>Quotation.php</u>

```
01:     <html>
02:     <head><title>Single Quotation and Double Quotation mark</title>
03:     </head>
04:     <body>
05:     <h2> Using double quatation and Signle quatation mark</h2>
06:
07:     <?
08:     $name = "Rithya";
09:     print "Hello,$name<br/>"; // Hello,Rithya
10:     print 'Hello,$name';  // Hello,$name
11:     ?>
12:
13:     </body>
14:     </html>
```

3.Displaying Type Information with `var_dump()`

`gettype()` គឺជា `function` ដែលប្រើដើម្បីទទួលនូវប្រភេទ `variable` ដោយឡែក `var_dump()` ប្រើដើម្បីប្រាប់នូវប្រភេទ `variable` និង ចំនុះរបស់វា ។ ច្រើនជាងនេះទៅទៀត សំរាប់ប្រភេទតំលៃដែលស្មុគស្មាញដូចជា `arrays` និង `object` `var_dump()` ផ្តល់អោយនូវពត៌មានគ្រប់ប្រភេទដែលមាននៅក្នុង`Variable` នោះ ។

ឧទាហរណ៏ ៣

Var_dump.php

```
01:     <html>
02:     <head>
03:     <title>Displaying Type Information with var_dump</title>
04:     </head>
05:     <body>
06:     <h2>using Var_dump</h2>
07:
08:     <?php
09:     $testing=5;
10:     print var_dump($testing);
11:     ?>
12:
13:     </body>
14:     </html>
```

4.The Cast Operators

PHPផ្តល់អោយនូវវិធីដើម្បធ្វើការផ្លាស់ប្តូរនូវប្រភេទទិន្នន័យដោយប្រើប្រាស់**castoperators** ដូចមានរៀបរាប់ក្នុងតារាងខាងក្រោម:

Operator	**Changes Type To**
(int), (integer)	Integer
(float), (real), (double)	Floating point
(string)	String
(bool), (Boolean)	Boolean
(array)	Array
(object)	Object

ឧទាហរណ៏ ៥

Casting.php

```
<html>
<head>
        <title>Casting variable</title>
</head>
<body>
<h2>Using Casting to Changing Type</h2>
<?php
        $unexpect=3.14;
```

```
$holder = (double)$unexpect;
print gettype($holder);
print "--- $holder<br/>";

$holder = (string)$unexpect;
print Gettype($holder);
print "-- $holder<br/>";

$holder = (integer)$unexpect;
print gettype($holder);
print "-- $holder<br/>";

$holder = (Double)$unexpect;
print  gettype($holder);
print   "---- $holder<br/>";

$holder = (boolean)$unexpect;
print gettype($holder);
print "-- $holder<br/>";

?>
</body>
</html>
```

5.Operators and Expressions

Operators គឺជានិមិ្តតសញ្ញាទាំឡាយណាដែលអ្នកអាចប្រើតំលៃមួយ ឬ ច្រើន បង្កើតចេញជាតំលៃថ្មីមួយទៀត ហើយតំលៃដែលប្រតិបត្តិដោយ operator នោះត្រូវបានគេហៅថា operand ។

Operand គឺជាតំលៃដែលចូលរួមជាមួយ operator ដែលជាទូទៅ មាន operand ពីរជាមួយOperator មួយ ។

ឧទាហរណ៏ការប្រើ operand ពីរ ជាមួយ operator ដើម្បីបង្កើតចេញជាតំលៃថ្មីមួយផ្សេងទៀត ។

4 + 5 .

៤ ហើយនិង ៥ គឺជា operand ដែលត្រូវបាន operated ដោយ addition operator (+) ដើម្បីបង្កើតនូវតំលៃថ្មី គឺ ៩ ។ ការផ្គុំនូវ operand និង operator ដើម្បីបង្កើតជាលទ្ធផល ត្រូវបានគេហៅថា expression

5.1 The Assignment Operator

Assignment operators ប្រើដើម្បីផ្តល់នូវតំលៃទៅអោយ variable ។ ដូចដែលអ្នកបានជួបខាងលើ assignment operator ត្រូវបាន initialize variable គ្រប់ពេល វាគឺជាអក្សរ (=) ។Assignment operator ចាប់យកតំលៃពី operand ខាងស្តាំទៅអោយ operand ខាងឆ្វេងដូចឧទាហរណ៏

ខាងក្រោម :

```
$name = " DYCHANDOEUN " ;
Print $name;  .
```

ជាទូទៅខាងឆ្វេងនៃ Assignment operator ច្រើនតែជា variable ។

5.2 Arithmetic Operators

Arithmetic Operators			
Operator	**Name**	**Example**	**Result**
+	បូក (Addition)	10+5	15
/	ចែក (Division)	10/3	3.333333333333
*	គុណ (Multiplication)	10*5	50
%	យកសំណល់ពីផលចែក (Modulus)	10%3	1
-	ដក (Subtraction)	10-2	2

5.3 The Concatenation Operator (.)

Concatenation operator ប្រើដើម្បីភ្ជាប់ string សំរាប់ operator មួយនេះវាធ្វើការជាមួយតែ string ប៉ុណ្ណោះ ។ ដូច្នេះរាល់ operand ដែលមិនមែនជា string វានឹង convert អោយទៅជា string សូមពិនិត្យមើលឧទាហរណ៍ :

"hello"." world" ការសរសេរបែបនេះវាស្មើនឹងការសរសេរ "hello world" ។

$year = 2007; .

Print "Happy khmer new year_".$year ;

Variable $year ដែលជាប្រភេទ integer ត្រូវបានបំលែងទៅជា string " 2007 " មុនពេលដែលវាត្រូវបានភ្ជាប់ជាមួយ string “ Happy khmer new year ” ។

5.4 Combined Assignment Operators

```
$x = 4;
$x = $x + 4; // លទ្ធផល $x គឺ 8
```

ឬ អ្នកអាចជំនួសដោយការសរសេរដូចខាងក្រោម

```
$x = 4;
$x += 4; // $ លទ្ធផល $x គឺ 8
```

Some Combined Assignment Operators		
Operator	**Example**	**Equivalent to**
+=	$x += 5	$x = $x + 5
-=	$x -= 5	$x = $x - 5
/=	$x /= 5	$x = $x / 5
*=	$x *= 5	$x = $x * 5

Some Combined Assignment Operators		
Operator	**Example**	**Equivalent to**
%=	$x %= 5	$x = $x % 5
.=	$x .= " test"	$x = $x." test"

5.5 Comparison Operators

Comparison operators ប្រើដើម្បីប្រៀបធៀបនូវ operands ទាំងឡាយ ដោយផ្តល់នូវតំលៃត្រលប់ជា Boolean (true or false) ។

ឧទាហរណ៍ដើម្បីត្រួតពិនិត្យតំលៃដែលមាននៅក្នុង $x និងជាតំលៃដែលតូចជាងប្រាំ អ្នកអាចប្រើជាមួយនឹង less than operator ។

```
$x < 5
```

ប្រសិនបើ $x ផ្ទុកតំលៃ លេខ 4 expression ខាងលើនឹងផ្តល់ជាតំលៃ true ប៉ុន្តែប្រសិនបើ $x ផ្ទុកតំលៃជាលេខ 7 នោះ expression នឹងផ្តល់នូវតំលៃ false ។

Comparison Operators			
Operator	**ឈ្មោះ**	**ផ្តល់តំលៃ True ប្រសិនបើ**	**($x គឺ 4)**
==	សមមូល	តំលៃខាងឆ្វេង ស្មើ តំលៃខាង ស្តាំ	$x == 5
!=	Non-equivalence	តំលៃខាងឆ្វេង ខុសពី តំលៃខាងស្តាំ	$x != 5
= = =	Identical	តំលៃខាងឆ្វេង ស្មើ តំលៃខាងស្តាំ ហើយ តំលៃទាំងពីរត្រូវតែមានប្រភេទដូចគ្នា	$x===5
>	ធំជាង	តំលៃខាងឆ្វេងធំជាង តំលៃខាងស្តាំ	$x > 4
>=	ធំជាង ឬ ស្មើ	តំលៃខាងឆ្វេងធំជាង ឬ ស្មើ តំលៃខាងស្តាំ	$x >= 4
<	តូចជាង	តំលៃខាងឆ្វេងតូចជាង តំលៃខាងស្តាំ	$x < 4
<=	តូចជាង ឬ ស្មើ	តំលៃខាងឆ្វេងតូចជាង ឬ ស្មើតំលៃខាងស្តាំ	$x <= 4

Operator ខាងលើនេះភាគច្រើនប្រើជាមួយ integers or double ហើយសំរាប់ operator(==) គឺប្រើដើម្បីប្រៀបធៀបតំលៃដែលជា strings ។

5.6 Logical Operators

Logical Operators				
Operator	**Name**	**Returns True if...**	**Example**	**Result**
`||`	Or	Left or right is true	`true || false`	`True`
`Or`	Or	Left or right is true	`true || false`	`true`
`Xor`	Xor	Left or right is true but not both	`true xor true`	`false`
`&&`	And	Left and right are true	`true && false`	`false`
`And`	And	Left and right are true	`true && false`	`false`
`!`	Not	The single operand is not true	`! true`	`false`

Logical operators បំលែង operand អោយទៅជាតំលៃ Boolean រួចធ្វើការប្រៀបធៀបតំលៃទាំងនោះ ។

Or operator ឬ (||) ផ្តល់តំលៃ true ប្រសិនបើ operand ខាងឆ្វេង ឬ ខាងស្តាំណាមួយមានតំលៃ true ។

ឧទាហរណ៏ ១. true || false លទ្ធផលគឺ true ។**And** operator ឬ (&&) ផ្តល់តំលៃ true នៅពេលដែល operand ទាំងពីរមានតំលៃ true ។

ឧទាហរណ៏ ២. true && false លទ្ធផលគឺ false ។

ឧទាហរណ៏ ៣. ($x > 2) && ($x < 15)

5.7 Increment/Decrement Operators

Increment/decrement operators ប្រើដើម្បីបង្កើន ឬ បន្ថយតំលៃ របស់ variable ដែលជា Integer ហើយជាទូទៅប្រើដើម្បីរាប់ Iteration របស់ loop ។

```
$x = $x + 1; // $x is incremented.
$x += 1; // $x is incremented.
$x++; // $x is incremented
$x = $x - 1; // $x is decremented.
$x--; // $x is decremented.
$x- = 1; // $x is decremented.
```

Operator	Name	Effect on $var	Value of the Expression
$var++	Post-increment	$var is incremented by 1	The previous value of $var
++$var	Pre-increment	$var is incremented by 1	The new value of $var (incremented by 1).
$var--	Post-decrement	$var is decremented by 1	The previous value of $var
--$var	Pre-decrement	$var is decremented by 1	The new value of $var (decremented by 1).

ឧទាហរណ៏

```
$num1 = 5;
$num2 = $num1++;  // post-increment, $num2 ត្រូវបានផ្តល់នូវតំលៃដើមរបស់ $num1
print $num1;  // លទ្ធផលដែលនឹងត្រូវបង្ហាញគឺ តំលៃរបស់ $num1 គឺ ៦
print $num2;  // លទ្ធផលដែលនឹងត្រូវបង្ហាញគឺ ជាតំលៃដើមរបស់ $num1 គឺ ៥
```

ឧទាហរណ៏ :

```
$num1 = 5;
$num2 = ++$num1;  // pre-increment, $num2 ត្រូវបានផ្តល់នូវតំលៃថ្មីរបស់ $num1
ទៅអោយ $num2 ។
print $num1;  // លទ្ធផលដែលនឹងត្រូវបង្ហាញគឺ តំលៃរបស់ $num1 គឺ ៦
print $num2;  // លទ្ធផលដែលនឹងត្រូវបង្ហាញគឺ ជាតំលៃរបស់ $num1 គឺ ៦
```

6.Constants

ជាទូទៅ Variables ត្រូវបានគេប្រើដើម្បីផ្ទុកនូវតំលៃ ពីព្រោះតំលៃ និង ប្រភេទរបស់វាអាច
នឹងត្រូវផ្លាស់ប្តូរបានគ្រប់ពេលវេលា ។ ប្រសិនបើអ្នកចង់ធ្វើការជាមួយតំលៃដែលមិនប្រែប្រួលនៅក្នុងការ ប្រតិបត្តិការកូដរបស់អ្នក
អ្នកអាចប្រើប្រាស់នូវ constant ។PHP បានផ្តល់នូវ define() function ដើម្បី
បង្កើតនូវ constant ។

```
define("CONSTANT_NAME", 42);
```

តំលៃដែលអ្នកអាចផ្តល់ទៅអោយ constant គឺត្រូវតែជា តំលៃលេខ ឬ តួអក្សរ ហើយសំរាប់ការផ្តល់ឈ្មោះរបស់ constant អ្នកគួរតែប្រើអក្សរធំ ។ រាលការប្រើប្រាស់ constant variable អ្នកគ្រាន់យក ឈ្មោះរបស់ constant នោះមកប្រើ ប៉ុន្តែមិនមានសញ្ញា ($) dollar symbol នៅពីមុខនោះទេ ។

ឧទាហរណ៏ ៦

constant.php

```
01:     <html>
02:     <head>
03:     <title>Defining a constant</title>
04:     </head>
05:     <body>
06:     <div>
07:
08:     <?php
09:     define("USER", "Ankor");
10:     print "Welcome ".USER;
11:     ?>
12:
13:     </div>
14:     </body>
15:     </html>
```

នៅបន្ទាត់ទី ១0 យើងបានប្រើ concatenation operator ដើម្បីភ្ជាប់តំលៃរបស់ constant និងអក្សរ "Welcome" ពីព្រោះ PHP engine មិនមានវិធីសាស្ត្រដើម្បីបែងចែក រវាង constant និង string ដែលនៅក្នុង quotation mark នោះទេ ។

ជា Default constant គឺ case sensitive ប៉ុន្តែអ្នកអាចផ្លាស់ប្តូរដោយទទួលយកនូវ argument ទី៣ ជា boolean មកប្រើក្នុង define() function ដើម្បីកំណត់អោយការប្រើប្រាស់ឈ្មោះ constant ជា Case insensitive ដូចមានក្នុងឧទហរណ៏ខាងក្រោម ។

define("USER", "Ankor", **true**); .

ដូច្នេះអ្នកអាចប្រើប្រាស់ constant ដោយមិនមានការខ្វាយខ្វល់អំពីអក្សរតូចឬធំឡើយ ។

```
print User;
print usEr;
print USER;
```

ឧទាហរណ៏ ៧

constant2.php

```
<html>
<head>
<title>Defining a constant</title>
</head>
```

```
<body>
<div>

<?php
        define ("USER", "Angkor",true);
        print "Welcome".uSER."<br/>";
        print "Welcome".uSeR."<br/>";
        print "Welcome".usEr."<br/>";
        print "Welcome".uSER."<br/>";
?>

</div>
</body>
</html>
```

មេរៀនទី ៣

Going With The Flow

1. The if Statement

If statement គឺជាវិធីសាស្ត្រដែលប្រើដើម្បីត្រួពិនិត្យទៅលើការប្រតិបត្តិការរបស់ statement ដែលនៅបន្ទាប់វា (អាចជា single statement ឬ ជា block of code ដែលបិតនូវក្នុងសញ្ញា {-------})

If statement ធ្វើការវាយតម្លៃ expression ដែលនៅក្នុងសញ្ញា (---) ប្រសិនបើ expression របស់ ifផ្តល់តំលៃ true នោះ statement ដែលនៅខាក្រោមនឹងត្រូវអនុវត្តន៏ ។

កូដខាងក្រោមបង្ហាញពី ទំរង់នៃ if statement ដែលត្រួតពិនិត្យ expression ជា string ។

```
if(expression)
{
  // code ដែលនឹងត្រូវអនុវត្តន៏នៅពេលដែល expression ផ្តល់តំលៃ true
}
```

ឧទាហរណ៏ ៨

```
<Html>
<head>
<title> Using if Statement </title>
</head>
<body>
<h2>Using if statement</h2>
<?php

$user="Thanith";
$pwd="123";
If(($user=  ="Thanith" && pwd=  ="123"))
print  "Login successful";
else
print "Login fail !";

?>
</body>
</Html>
```

ការប្រើប្រាស់ comparasion operator (==) ដើម្បីប្រៀបធៀប variable $user និង តំលៃជាអក្សរ " Thanith " variable $pwd ជាមួយនឹងតំលៃ "123" ប្រសិនបើតំលៃដែលត្រូវប្រៀបធៀបនិងតំលៃរបស់ variable ដូចគ្នានោះ expression នឹងផ្តល់តំលៃ true ហើយ code block នឹងត្រូវអនុវត្តន៏ប៉ុន្តែប្រសិនបើតំលៃរបស់ $user ប្តូរទៅជា "Romchong" ឬ តំលៃរបស់ $pwd ប្តូរទៅជា "124" ហើយតំណើរការ scriptឡើងវិញ នោះ expression ដែលនៅក្នុង if statement នឹងផ្តល់តំលៃ falseហើយCode block នឹងមិនត្រូវអនុវត្តន៏ ដែល script នឹងបែរទៅអនុវត្តន៏នូវ else statement ជំនួសវិញ ។

```
else
print "Login fail !";
```

1.1 Using the `else if` Clause with the `if` Statement

អ្នកអាចប្រើប្រាស់នូវចំរង់ if/else ឬ else/if ដើម្បីធ្វើការពិនិត្យលើ expression មុនពេលដែល Script របស់អ្នកត្រូវអនុវត្តន៍ នូវ default block of code ។

```
if ( expression )
{
// code ដែលត្រូវអនុវត្តន៍ ប្រសិនបើ expression ផ្តល់តម្លៃ true
}
else if ( another expression )
{
// code ដែលត្រូវអនុវត្តន៍ ប្រសិនបើ expression ដំបូងផ្តល់តម្លៃ false
// ហើយ expression របស់វា true
}
else
{
// code ដែលត្រូវអនុវត្តន៍ ប្រសិនបើពុំមាន expression ណាមួយ true
}
```

ប្រសិនបើ expression ដំបូងមិនផ្តល់នូវតំលៃ true នោះ block of code ដំបូងក៏មិនត្រូវបាន អនុវត្តន៍ដែរ else if ចាប់ផ្តើមធ្វើការជាមួយ expression របស់ខ្លួន ប្រសិនបើ expression នេះផ្តល់តំម្លៃ True នោះ block កូដទី២ នឹងត្រូវអនុវត្តន៍ ផ្ទុយមកវិញកូដដែលស្ថិតនៅក្នុង else clauseនឹងត្រូវអនុវត្តន៍ ជំនួសវិញ ។ អ្នកអាចប្រើប្រាស់ else if បានជាច្រើនទៅ តាមការចង់បានរបស់អ្នក ហើយប្រសិនបើ អ្នកមិន ចង់អោយមាន Default action ទេ អ្នកមិនចាំបាច់ប្រើប្រាស់ else clause នោះទេ ។

2. The switch Statement

switch statement គឺជាវិធីសាស្ត្រដែលប្រើដើម្បីផ្លាស់ប្តូរលំដាប់នៃការអនុវត្តន៍កូដរបស់កម្មវិធីដែលអាស្រ័យទៅលើការវាយតំលៃរបស់ expression ។ ការប្រើប្រាស់ if statement ជាមួយ else if អ្នកអាចប្រើប្រាស់ expressionបានច្រើន ដោយឡែក switch ប្រើប្រាស់តែ expression មួយប៉ុណ្ណោះ ។ការអនុវត្តន៍ code ខុសៗគ្នាគឺអាស្រ័យទៅលើលទ្ធផលនៃexpression ដែលផ្តល់តំលៃជា simple typeដូចជា (number , string , Boolean…… ។ល។) ។

```
switch (expression)
{
  case exp:
    // execute this if expression results in result1
    break;
  case exp:
    // execute this if expression results in result2
    break;
```

```
   default:
        // កូដនឹងត្រូវអនុវត្តន៍ប្រសិនបើមិនមាន expression ណាដូចនឹង expression របស់ case
}
```

Expression របស់ switch statement ជាទូទៅត្រូវបានប្រើជា variable ហើយ codeរបស់ switch statement ត្រូវសរសេរនៅក្នុង case statement ។ រាល់តំលៃ expression របស់ caseនិមួយៗ ត្រូវបានយកមកផ្ទៀងផ្ទាត់ជាមួយ expression របស់ switch statement ប្រសិនណាតំលៃរបស់case ណាមួយដូចនឹង expression របស់ switch statement នោះ code block នឹងត្រូវអនុវត្តន៍ បន្ទាប់ មក break statement នឹងបញ្ចប់ការអនុវត្តន៍ switch statement ប៉ុន្តែប្រសិនបើពុំមាន case expression ណាមួយ ដូចនឹង switch expression នោះ default statement គឺជាអ្នកអនុវត្តន៍ ។

ឧទាហរណ៍ ៩

```
<Html>
<head>
<title>
Using switch Statement
</title>
</head>
<body>
<h2>Using switch statement</h2>

<?php

$name="Daro";
switch($name)
{
        case "Dara":
                print " Hello Dara";
        break;

        case "Many":
                print "Hello Many";
        break;
        case "Daro":
                print "Hello Daro";
        break;

        Default:
                print "No one know";
}

?>
```

```
</body>
</Html>
```

3. Loops

Loop statement អាចអោយអ្នកអនុវត្តន៍នូវការងារម្តងហើយម្តងទៀតនៅក្នុង programរបស់អ្នក រហូតដល់វាសំរេចលក្ខ័ណ្ឌ ឬ អ្នកបញ្ជាអោយចាកចេញពី loop ។

3.1 The `while` Statement

While loops គឺជាប្រភេទមួយនៃ loops ។ expression របស់វាផ្តល់ជាតំលៃ true ឬ false ដូច្នេះប្រសិនបើ expression ផ្តល់តំលៃជាលទ្ធផល true នោះ code block នឹងត្រូវអនុវត្តន៏ ដែល blockCode ស្ថិតនៅក្នុង loop នោះត្រូវបានគេអោយឈ្មោះថា iteration ។

```
while ( expression )
{
  // do something
}
```

ឧទាហរណ៏ ១0

while.php

```
<html>
<head><title>The While Statement</title>
</head>
<body>
<h2>Using the While Statement</h2>

<?php

$sum=0;$i=1;
$str="";
While ($i<=10)
{
        $sum+=$i;
        $str= $str."$i+";
      $i++;
}
        echo substr($str,0,-1)."=$sum";

?>

</body>
</html>
```

3.2 The do...while Statement

`do..while` statement វាមានលក្ខណៈប្រហាក់ប្រហែលនឹង while statement ប៉ុន្តែលក្ខណៈពីរដែលខុសគ្នាពី while statement គឺ block code របស់ do while statement អនុវត្តន៏មុនពេលដែល Expression របស់វាត្រូវបាន test និង ផ្តល់តំលៃ true ឬ false ។

```
do
{
……
 // code to be executed
……
}
while (expression);
```

Test expression នៃ `do...while` statement ត្រូវតែបញ្ចប់ដោយ (;) semicolon.

ឧទាហរណ៏ ១១

dowhile.php

```
<html>
<head><title>The Do While Loop Statement</title>
</head>
<body>
<h2>Using the do while Statement</h2>
</body>
</html>
<?php
$sum=0;$i=1;$str="";
Do
{
        $sum=$sum+$i;
        $str= $str."$i+";
        $i=$i+1;
}
While ($i<=10);
{
  $i=$i-1;
  echo substr($str,0,-1)."=$sum";

}
?>
```

3.3 The `for` Statement

```
for ( initialization expression; test expression; modification expression )
{
  // code to be executed
}
```

រាល់ expression និមួយៗដែលមាននៅក្នុងសញ្ញាវង់ក្រចករបស់ for statement គឺត្រូវបែងចែកគ្នាដោយ semicolon (;) ។ expression ទីមួយ ចាប់ផ្តើមរាប់ variable ហើយ expression ទី២ធ្វើការត្រួតពិនិត្យលក្ខ័ណរបស់ for loop និង expression ទី៣ បង្កើន ឬ បន្ថយនូវចំនួនការរាប់ ។

ឧទាហរណ៏ ១២

forloop.php

```
<html>
<head>
<title>The for Statement</title>
</head>
<body>
```

```
<h2>Using for Statement</h2>

<?php
$sum=0;$str="";
for ($i=1; $i<=10; $i++ )
{
$sum+=$i;
$str= $str."$i+";
}
echo substr($str,0,-1)."=$sum";
 ?>

</body>
</html>
```

នៅពេលដែល program តំណើរការដល់ for loop variable $i ត្រូវបាន initialize ហើយTest expression ចាប់ផ្តើមត្រួតពិនិត្យទៅលើ expression របស់ខ្លួន ប្រសិនបើ expression ផ្តល់តំលៃTrue នោះ code block នឹងត្រូវអនុវត្តន៍បន្ទាប់មក $i variable ធ្វើការបង្កើនតំលៃមួយហើយ testExpression ចាប់ផ្តើមធ្វើការត្រួតពិនិត្យទៅលើ expression របស់ខ្លួនសារជាថ្មីម្តងទៀត។ ប្រតិបត្តិការនេះបន្តការអនុវត្តន៍រហូតដល់ test expression ផ្តល់តំលៃ false ។

3.4 Breaking Out of Loops with the `break` Statement

រាល់ loop statement គឺសុទ្ធតែមានភ្ជាប់មកជាមួយនូវ test expression ដែលអាចអោយអ្នក បញ្ឈប់វាបាន ឬដោយប្រើប្រាស់ break statement ។

ឧទាហរណ៏ ១៣

break.php

```
01: <html>
02: <head>
03: <title>the break Statement</title>
04: </head>
05: <body>
06: <div>
07:
08: <?php
09:
10: $counter = -5;
11: for ( ; $counter <= 10; $counter++ ) {
12:   if ( $counter == 0 ) {
13:   break;
14:   }
15:   $temp = 2000/$counter;
16:   print "2000 divided by $counter is.. $temp<br />";
17:   }
18: ?>
19:
```

```
20: </div>
21: </body>
22: </html>
```

យើងបានប្រើប្រាស់នូវ if statement នៅបន្ទាត់ទី ១៣ ដើម្បីត្រួពិនិត្យនូវតំលៃរបស់ variable$counter ប្រសិនបើតំលៃរបស់វាស្មើនឹងសូន្យ 0 break statement នឹងត្រូវអនុវត្តន៏ ដែលត្រូវចាកចេញពីBlock code របស់ for loop statement ហើយអនុវត្តន៏នូវ statement ដែលនៅបន្ទាប់ពី for statement ។

3.5 Skipping an Iteration with the `continue` Statement

`Continue` statement បញ្ឈប់តំណើរការរបស់ iteration ដែលកំពុងអនុវត្តន៏ ប៉ុន្តែមិនបញ្ឈប់ តំណើរការរបស់ loop ទាំងស្រុងនោះទេ វានឹងបន្តធ្វើការជាមួយ iteration ក្រោយៗបន្តទៀត រហូតដល់ Expression ផ្តល់តំលៃ false ឬ ជួប នឹង break statement ។

ឧទាហរណ៏ ១៤ continue.php

```
01: <!
02: exam continue statement
03: >
04:         <html>
05:         <head>
06:         <title>Using the continue Statement</title>
07:         </head>
08:         <body>
09:         <div>
10:
11:         <?php
12:
13:         $counter = -5;
14:         for( ; $counter <= 10; $counter++ )
15:         {
16:                     if ( $counter == 0 )
17:                     {
18:                     continue;
19:                     }
20:         $temp = 2000/$counter;
21:         print "2000 divided by $counter is .. $temp<br />";
22:         }
23:
24:         ?>
25:
26:         </div>
27:         </body>
28:         </html>
```

នៅបន្ទាត់ទី១៤យើងបានជំនួស break statement ដោយការប្រើប្រាស់ continue statementប្រសិនបើ `variable $counter` ស្មើ 0 `iteration` នឹងត្រូវរំលងការអនុវត្តន៏ ហើយបន្តអនុវត្តន៏ `iteration` ជាបន្តទៀត ។