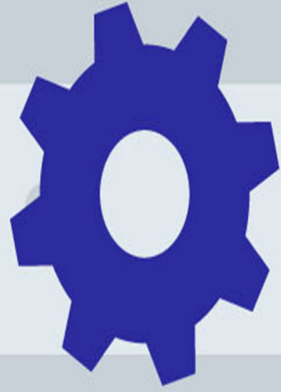




+

.PHP



<?>

+

ကျွန်းကျွန်း

ကျွန်းကျွန်း

មេរៀនទី ១

Introduction PHP

History

PHP ត្រូវបានផ្តល់ឈ្មោះជាផ្លូវការ HyperText Preprocessor វាជាភាសាដែលតំណើរការនៅលើ Server ដែលជា ទូទៅ ត្រូវបាន សរសេរក្នុងបរិបទ HTML ប៉ុន្តែមិនដូចជា HTML page ធម្មតានោះទេ PHP script មិនត្រូវបានបញ្ជូន ទៅអោយ client ដោយ server ផ្ទាល់នោះទេ វាធ្វើការបញ្ជូនដោយ PHP engine ។ PHP code ដែលស្ថិតនៅក្នុង script អាចប្រើដើម្បីធ្វើ ការជាមួយ Databases, បង្កើតជាប្រភេទ , អាន និងបញ្ចូលទិន្នន័យ កែប្រែអត្ថបទដែលមាននៅខាងក្នុង files ឬ ធ្វើការ ទំនាក់ទំនងទៅកាន់ Remote Server ព្រមជាមួយនឹងលទ្ធភាពជាច្រើនផ្សេងទៀត ។

Rasmus Lerdorf គឺជាមនុស្សដំបូងដែលអភិវឌ្ឍន៍នូវ PHP/FI ហើយត្រូវបានមនុស្សរាប់លានអ្នកប្រើប្រាស់វា ។ ជំនាន់ដំបូងនៃ PHP/FI ឈ្មោះថា **Personal Homepage Tools/Form Interpreter** ដែលលក្ខណៈជាមូលដ្ឋានគឺស្រដៀងទៅនឹង ភាសា Perl ព្រោះបានប្រមូលផ្តុំនូវ Perl scripts មកអភិវឌ្ឍន៍ នៅក្នុងអំឡុងឆ្នាំ ១៩៩៥ ប៉ុន្តែវាមានកង្វះខាតជាច្រើននៅក្នុង ភាសានេះ ដូចជា for loops ជាដើម ។

PHP/FI 2

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

PHP 3

Zeev and Andi សំរេចចិត្តសរសេរ scripting language ឡើងវិញប៉ុន្តែពុំវត្តមាន Rasmus ចូលរួមដើម្បីអភិវឌ្ឍន៍នូវ PHP3 ឡើយហើយបានផ្តល់នូវឈ្មោះថា **Hypertext Preprocessor** ដើម្បីបញ្ជាក់ថា PHP គឺជាផលិតផលផ្សេងមួយទៀតហើយនិង មិនត្រឹមតែយកមកប្រើសំរាប់តែការងារផ្ទាល់ ខ្លួននោះទេ ។ Zeev and Andi ក៏បានបង្កើតនូវ Extension API ដែល API បង្កើតថ្មីនេះវាមានលទ្ធភាពបំពេញនូវការងារជាច្រើនដូចជា Accessing databases , spell checkers ហើយនិង បច្ចេក វិទ្យាដទៃទៀត ដែលធ្វើអោយមានការចាប់អារម្មណ៍ពីសំណាក់អ្នកអភិវឌ្ឍន៍ ជាច្រើនមកចូលរួមក្នុងគំរោង PHP ។ នៅខណៈពេលនោះដែរ PHP ក៏ត្រូវបានបញ្ចេញនូវជំនាន់ថ្មីរបស់ខ្លួនគឺ PHP 3 នៅ ថ្ងៃទី ៣ ខែ មិថុនា ឆ្នាំ ១៩៩៨ ដែលតាមការប៉ាន់ស្មាន PHP នឹងត្រូវបានតំឡើង ប្រមាណ ជាង ៥០ ០០០ domains ប៉ុន្តែជាមួយគ្នាពិតប្រាកដ លើកដំបូងរបស់ PHP ត្រូវបានគេតំឡើងច្រើនជាងមួយលាន Domain ទៅទៀត ។

PHP 4

នៅក្រោយឆ្នាំ ១៩៩៨ Zeev និង Andi ងាកទៅពិចារិកលើការងារ PHP 3 ហើយពួកគេមានគំនិតថាពួកគេអាចសរសេរ script language ឡើងវិញដោយមានលក្ខណៈល្អប្រសើរជាងមុនទៅទៀត ក្នុងខណៈពេលដែល PHP 3 កំពុងបន្តការធ្វើសម្អាត និងប្រតិបត្តិការនោះ PHP 4 ក៏ចាប់ផ្តើមបង្កើតនូវគំរូថ្មី គឺ “compile first, execute later.” តំណាក់កាលនៃការ compile មិនត្រូវបាន compile PHP Script អោយទៅជា machine code នោះទេ វាជំនួសដោយការ compile ទៅជា byte code ដែលធ្វើ ការប្រតិបត្តិការ ដោយ **Zend Engine** (Zend មកពីពាក្យថា Zeev និង Andi) ។ វិធីសាស្ត្រថ្មីសំរាប់ការប្រតិបត្តិ script នេះអាចធ្វើអោយ PHP 4 តំណើរការបានល្អប្រសើរច្រើនជា PHP 3 ហើយត្រូវបានដាក់បង្ហាញនូវ PHP 4 នេះក្នុង ខែ ឧសភា ឆ្នាំ ២០០២ ប៉ុន្តែដោយមានការផ្លាស់ប្តូរនៅក្នុងភាសានេះជាបន្តបន្ទាប់ទើប PHP 4 បានបង្កើតនូវជំនាន់របស់ខ្លួន ជា PHP 4.1.0 នឹង បានបង្ហាញនូវ **Superglobals** ដូចជា \$_GET និង \$_POST ។ ដែល Superglobals នេះអាចយកមកប្រើប្រាស់ពីខាងក្នុង Functions ដោយមិនចាំបាច់ប្រើ global keyword ។ រហូតដល់ជំនាន់ចុងក្រោយរបស់ PHP4 ត្រូវបាន បង្ហាញ ជាចុងក្រោយបង្អស់ នៅថ្ងៃទី 27 ខែ ធ្នូ ឆ្នាំ ២០០២ ។

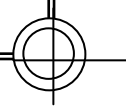
PHP 5

ដោយមានតំរូវការជាច្រើននូវលក្ខណៈរបស់ object-oriented Andi ក៏មានគំនិតសរសេរនូវ Object-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 ផ្ទុកនូវមុខងារថ្មីជាច្រើនដែលបញ្ចូលជាមួយមុខងារសំរាប់ 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)

Variable គឺជាគ្រឹះដ៏សំខាន់សំរាប់ភាសាកុំព្យូទ័រ វាត្រូវបានគេប្រើដើម្បីផ្ទុកនូវតំលៃជាបណ្តោះអាសន្ននៅពេលដែលប្រតិបត្តិការ script ម្តងៗ ។ variable ត្រូវបានផ្តល់តំលៃទៅអោយ នៅពេលដែល Script ចាប់ផ្តើមតំណើរការ ឬ នៅពេលអ្នកប្រើប្រាស់បញ្ចូល ឬបានមកពីការចាប់យកទិន្នន័យពី Database អ្នកអាចប្រើប្រាស់ variable គ្រប់ពេលទាំងអស់ នៅពេលដែល script ចាប់ផ្តើមតំណើរការជាមួយទិន្នន័យ ហើយអ្នកអាចកែប្រែ តំលៃដែល Variable នោះផ្ទុកពីការប្រតិបត្តិការ script មួយទៅកាន់ការប្រតិបត្តិការ របស់ script មួយផ្សេងទៀត រហូតដល់ពេលដែល script របស់អ្នកត្រូវបានបញ្ចប់ ។

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

ឧទាហរណ៍ ខាងក្រោមគឺជាការផ្តល់ឈ្មោះទៅអោយ variable ដែលត្រឹមត្រូវ :

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

ឧទាហរណ៍ ការផ្តល់ឈ្មោះទៅអោយ variable ដែលមិនត្រឹមត្រូវ :

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

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

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

ការប្រកាស variable ពីរ ខាងលើអ្នកបានប្រើប្រាស់នូវ assignment operator (=) ដែលនឹងរៀបរាប់លំអិតនៅក្នុងមេរៀន “ Operator and Expression “ ។ បន្ទាប់ពីអ្នកបានផ្តល់នូវតំលៃទៅ variable រួចរាល់ហើយ អ្នកអាចយកវាមក ប្រើប្រាស់បាន ដូចខាងក្រោម :

print \$num2; វាមានតំលៃស្មើនឹងការប្រើ print 8; ដូច្នេះមានន័យថា \$num2 ផ្ទុកតំលៃ ៨ ចូរចងចាំថារាល់ចុងបញ្ចប់នៃរបាយការណ៍របស់ PHP នីមួយៗត្រូវតែបញ្ចប់ដោយសញ្ញា (;) semicolon ។

2 Data Types

ប្រភេទខុសៗគ្នានៃទិន្នន័យត្រូវបានប្រើប្រាស់នូវចំនួនសរុបរបស់អង្គចងចាំផ្សេងៗគ្នា ហើយវាអាចប្រព្រឹត្តទៅនៅពេលអ្នក រៀបចំវានៅក្នុង script របស់អ្នក ។ មានភាសាកម្មវិធីមួយចំនួនទាមទារអោយអ្នកសរសេរកម្មវិធី ប្រកាសនូវប្រភេទ នៃទិន្នន័យ

ពីមុខ variable ដោយឡែកសំរាប់ភាសា PHP ការប្រើប្រាស់ variable គឺមានភាពងាយស្រួល ដោយវាងងឹតធ្វើការគណនា នូវប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តិនៅពេលដែលអ្នកបានផ្តល់តំលៃទៅអោយវា ។

Standard Data Types		
ប្រភេទ	ឧទាហរណ៍	ពិពណ៌នា
Integer	5	ផ្ទុកនូវតំលៃលេខជាចំនួនគត់
Double	3.234	ផ្ទុកនូវតំលៃលេខជាចំនួនទសភាគ
String	"hello"	ផ្ទុកនូវតំលៃជាបន្តនៃតួអក្សរ
Boolean	True	ផ្ទុកនូវតំលៃពិសេសគឺ true ឬ false
Array		មេរៀនទី
Object		មេរៀនទី

ឧទាហរណ៍ 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 ដូច្នេះនៅពេលដែលប្រើប្រាស់ gettype() function ដើម្បីត្រួតពិនិត្យ variable នៅបន្ទាត់ទី១០ នោះអ្នកនឹង ទទួលបាននូវ លទ្ធផល ជាអក្សរ Null ។ បន្ទាប់ពីនេះ variable \$testing ត្រូវបានផ្តល់នូវតំលៃជាតួលេខនៅបន្ទាត់ទី ១១ គឺលេខ ៥ ដែលជាចំនួនគត់ ឬ អ្នកអាចនិយាយបានថាតំលៃលេខដែលគ្មានក្បៀសដូច្នេះលទ្ធផលដែលនឹងទទួលបានបន្ទាប់ពីការប្រើប្រាស់នូវ gettype() function បន្ទាត់ទី១២ គឺ // integerចំណែកការផ្តល់នូវតំលៃ "five" ទៅអោយ variable \$testing នៅបន្ទាត់ទី ១៥ គឺជាបន្តិចនៃតួអក្សរ ។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 ។

សំគាល់ :

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

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

```
$name = "Rithya"; .
```

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

ប្រសិនបើអ្នកប្រើប្រាស់នូវ single quotation mark variable មិនត្រូវបានជំនួសដោយតំលៃនោះទេ ។

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

ឧទាហរណ៍ ២

Quotation.php

```
01:     <html>
02:     <head><title>Single Quotation and Double Quotation mark</title>
03:     </head>
04:     <body>
05:     <h2> Using double quotation and Signle quotation 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 ដែលប្រើដើម្បីទទួលបានប្រភេទ variable ដោយឡែក var_dump() ប្រើដើម្បីប្រាប់ប្រភេទ variable និង ចំនុះរបស់វា ។ ច្រើនជាងនេះទៅទៀត សំរាប់ប្រភេទតំលៃដែលស្មុគស្មាញដូចជា arrays និង object var_dump() ផ្តល់អោយនូវព័ត៌មានគ្រប់ប្រភេទដែលមាននៅក្នុង variable នោះ ។

ឧទាហរណ៍ 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

PHPផ្តល់អោយនូវវិធីដើម្បីធ្វើការផ្លាស់ប្តូរប្រភេទទិន្នន័យដោយប្រើប្រាស់cast operators ដូចមានរៀបរាប់ក្នុងតារាងខាងក្រោម:

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

ឧទាហរណ៍ 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 គឺជានិមិត្តសញ្ញាទំនាក់ទំនងដែលអ្នកអាចប្រើតំលៃមួយ ឬ ច្រើន បង្កើតចេញជាតំលៃថ្មីមួយទៀត ហើយតំលៃដែលប្រតិបត្តិដោយ operator នោះត្រូវបានគេហៅថា operand ។

Operand គឺជាតំលៃដែលចូលរួមជាមួយ operator ដែលជាទូទៅ មាន operand ពីរជាមួយ Operator មួយ ។

ឧទាហរណ៍ការប្រើ 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 ខាងឆ្វេងដូចឧទាហរណ៍

ខាងក្រោម :

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


ជាទូទៅខាងឆ្វេងនៃ Assignment operator ត្រូវតែជា variable ។

5.2 Arithmetic Operators

Arithmetic Operators			
Operator	Name	Example	Result
+	បូក (Addition)	10+5	15
/	ចែក (Division)	10/3	3.33333333333333
*	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 ត្រូវបានបំប្លែងទៅជា string " 2007 " មុន

ពេលដែលវាត្រូវបានភ្ជាប់ជាមួយ 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

<i>Some Combined Assignment Operators</i>		
Operator	Example	Equivalent to
<code>%=</code>	<code>\$x %= 5</code>	<code>\$x = \$x % 5</code>
<code>.=</code>	<code>\$x .= " test"</code>	<code>\$x = \$x." test"</code>

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	(<code>\$x</code> គឺ 4)
<code>==</code>	សមមូល	តំលៃខាងឆ្វេង ស្មើ តំលៃខាងស្តាំ	<code>\$x == 5</code>
<code>!=</code>	Non-equivalence	តំលៃខាងឆ្វេង ខុសពី តំលៃខាងស្តាំ	<code>\$x != 5</code>
<code>===</code>	Identical	តំលៃខាងឆ្វេង ស្មើ តំលៃខាងស្តាំ ហើយ តំលៃទាំងពីរត្រូវតែមានប្រភេទដូចគ្នា	<code>\$x===5</code>
<code>></code>	ធំជាង	តំលៃខាងឆ្វេងធំជាង តំលៃខាងស្តាំ	<code>\$x > 4</code>
<code>>=</code>	ធំជាង ឬ ស្មើ	តំលៃខាងឆ្វេងធំជាង ឬ ស្មើ តំលៃខាងស្តាំ	<code>\$x >= 4</code>
<code><</code>	តូចជាង	តំលៃខាងឆ្វេងតូចជាង តំលៃខាងស្តាំ	<code>\$x < 4</code>
<code><=</code>	តូចជាង ឬ ស្មើ	តំលៃខាងឆ្វេងតូចជាង ឬ ស្មើតំលៃខាងស្តាំ	<code>\$x <= 4</code>

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 បំបែក operand អោយទៅជាតំលៃ Boolean រួចធ្វើការប្រៀបធៀបតំលៃទាំងនោះ ។

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

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

ឧទាហរណ៍ ២. true && false លទ្ធផលគឺ false ។

ឧទាហរណ៍ 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).

ឧទាហរណ៍

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

ឧទាហរណ៍ :

```
$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 ប្រតិបត្តិការក្នុងរបស់អ្នក អ្នកអាចប្រើប្រាស់នូវ constant ។ PHP បានផ្តល់នូវ define() function ដើម្បី បង្កើតនូវ constant ។

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

តំលៃដែលអ្នកអាចផ្តល់ទៅអោយ constant គឺត្រូវតែជា តំលៃលេខ ឬ តួអក្សរ ហើយសំរាប់ការផ្តល់ឈ្មោះរបស់ constant អ្នកគួរតែប្រើអក្សរធំ ។ រាល់ការប្រើប្រាស់ constant variable អ្នកត្រូវយក ឈ្មោះរបស់ constant នោះមកប្រើ ប៉ុន្តែមិនមានសញ្ញា (\$) dollar symbol នៅពីមុខនោះទេ ។

ឧទាហរណ៍ 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 ១០ 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 ដូចមានក្នុងឧទាហរណ៍ខាងក្រោម ។

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

ដូច្នេះអ្នកអាចប្រើប្រាស់ constant ដោយមិនមានការខ្វាយខ្វល់អំពីអក្សរតូចឬធំឡើយ ។

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

ឧទាហរណ៍ 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 គឺជាវិធីសាស្ត្រដែលប្រើដើម្បីត្រួតពិនិត្យទៅលើការប្រតិបត្តិការរបស់ statement ដែលនៅបន្ទាប់វា (អាចជា single statement ឬ ជា block of code ដែលបិទនៅក្នុងសញ្ញា {-----})

If statement ធ្វើការវាយតម្លៃ expression ដែលនៅក្នុងសញ្ញា (---) ប្រសិនបើ expression របស់ if ផ្តល់តម្លៃ true នោះ statement ដែលនៅខាងក្រោមនឹងត្រូវអនុវត្តន៍ ។

ក្នុងខាងក្រោមបង្ហាញពី ទំរង់នៃ if statement ដែលត្រួតពិនិត្យ expression ជា string ។

```
if(expression)
{
    // code ដែលនឹងត្រូវអនុវត្តនៅពេលដែល expression ផ្តល់តម្លៃ true
}
```

ឧទាហរណ៍ ៨

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

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

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

ការប្រើប្រាស់ comparasion operator (==) ដើម្បីប្រៀបធៀប variable \$user និង តំលៃជាអក្សរ " Thanith " variable \$password ជាមួយនឹងតំលៃ "123" ប្រសិនបើតំលៃដែលត្រូវប្រៀបធៀបនិងតំលៃរបស់ variable ដូចគ្នានោះ expression នឹងផ្តល់តំលៃ true ហើយ code block នឹងត្រូវអនុវត្តប៉ុន្តែប្រសិនបើតំលៃរបស់ \$user ឬរកទៅជា "Romchong" ឬ តំលៃរបស់ \$password ឬរកទៅជា "124" ហើយតំលៃការ script ឡើងវិញ នោះ expression ដែលនៅក្នុង if statement នឹងផ្តល់តំលៃ false ហើយ Code block នឹងមិនត្រូវអនុវត្តន៍ ដែល script នឹងបែរទៅអនុវត្តន៍នូវ else statement ជំនួសវិញ ។

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

1.1 Using the else if Clause with the if Statement

អ្នកអាចប្រើប្រាស់នូវទំរង់ 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 ផ្តល់តម្លៃ false

    // ហើយ expression របស់វា true
}
else
{
    // code ដែលត្រូវអនុវត្តន៍ ប្រសិនបើពុំមាន expression ណាមួយ true
}
```

ប្រសិនបើ expression ផ្តល់តម្លៃ true នោះ block of code ដំបូងក៏មិនត្រូវបាន អនុវត្តន៍ដែរ else if ចាប់ផ្តើមធ្វើការជាមួយ expression របស់ខ្លួន ប្រសិនបើ expression នេះផ្តល់តម្លៃ True នោះ block ក្នុងនោះ នឹងត្រូវអនុវត្តន៍ ផ្ទុយមកវិញក្នុងករណីដែលស្ថិតនៅក្នុង else clauseនឹងត្រូវអនុវត្តន៍ ជំនួសវិញ ។ អ្នកអាចប្រើប្រាស់ else if បានជាច្រើនទៅ តាមការចង់ បានរបស់អ្នក ហើយប្រសិនបើ អ្នកមិន ចង់អោយមាន Default action ទេ អ្នកមិនចាំបាច់ប្រើប្រាស់ else clause នោះទេ ។

2. The switch Statement

switch statement គឺជាវិធីសាស្ត្រដែលប្រើដើម្បីផ្លាស់ប្តូរលំដាប់នៃការអនុវត្តកូដរបស់កម្មវិធីដែលអាស្រ័យទៅលើ ការវាយតម្លៃរបស់ expression ។ ការប្រើប្រាស់ if statement ជាមួយ else if អ្នកអាចប្រើប្រាស់ expressionបានច្រើន ដោយ ឡែក switch ប្រើប្រាស់តែ expression មួយប៉ុណ្ណោះ ។ការអនុវត្តន៍ code ខុសៗគ្នាគឺអាស្រ័យទៅលើលទ្ធផលនៃexpression ដែលផ្តល់តម្លៃជា 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:

```
// ក្នុងនឹងត្រូវអនុវត្តប្រសិនបើមិនមាន expression ណាមួយនឹង expression របស់ case
}
```

Expression របស់ switch statement ជាទូទៅត្រូវបានប្រើជា variable ហើយ code របស់ switch statement ត្រូវសរសេរនៅក្នុង case statement ។ រាល់តំលៃ expression របស់ case នីមួយៗ ត្រូវបានយកមកធ្វើការផ្ទៀងផ្ទាត់ជាមួយ expression របស់ switch statement ប្រសិនបើមានតំលៃរបស់ case ណាមួយដូចនឹង expression របស់ switch statement នោះ code block នឹងត្រូវអនុវត្ត បន្ទាប់ មក break statement នឹងបញ្ចប់ការអនុវត្ត switch statement ប៉ុន្តែប្រសិនបើពុំមាន case expression ណាមួយ ដូចនឹង switch expression នោះ default statement គឺជាអ្នកអនុវត្ត ។

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 program របស់អ្នក រហូតដល់វាសំរេច លក្ខខណ្ឌ ឬ អ្នកបញ្ជាអោយចាកចេញពី loop ។

3.1 The while Statement

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

```
while ( expression )
```

```
{
```

```
    // do something
```

```
}
```

ឧទាហរណ៍ ១០

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 ប៉ុន្តែលក្ខណៈពិរិយលក្ខណៈគឺ while statement គឺ block code របស់ do while statement អនុវត្តន៍មុនពេលដែល Expression របស់វាត្រូវបាន test និង ផ្តល់តំលៃ true ឬ false ។

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

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

ឧទាហរណ៍ 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 នីមួយៗដែលមាននៅក្នុងសញ្ញាវង់ក្រចករបស់ for statement គឺត្រូវបែងចែកគ្នាដោយ semicolon (;)

។ expression ទីមួយ ចាប់ផ្តើមរាប់ variable ហើយ expression ទី២ធ្វើការត្រួតពិនិត្យលក្ខណៈរបស់ for loop និង expression ទី៣ បង្កើន ឬ បន្ថយនូវចំនួនការរាប់ ។

ឧទាហរណ៍ 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 នឹងត្រូវអនុវត្តន៍ បន្ទាប់មក \$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 ឬ ដោយប្រើប្រាស់ break statement ។

ឧទាហរណ៍ 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 ដើម្បីត្រួតពិនិត្យនូវតំលៃរបស់ variable\$counter ប្រសិនបើតំលៃរបស់វាស្មើនឹងសូន្យ 0 break statement នឹងត្រូវអនុវត្តន៍ ដែលត្រូវចាកចេញពីBlock code របស់ for loop statement ហើយអនុវត្តន៍នូវ statement ដែលនៅបន្ទាប់ពី for statement ។

3.5 Skipping an Iteration with the continue Statement

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

ឧទាហរណ៍ 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 នឹងត្រូវរំលងការអនុវត្តន៍ ហើយបន្តអនុវត្តន៍ iteration ជាបន្តទៀត ។

3.6 Nesting Loops

ឧទាហរណ៍ ១៥:

nestingLoop.php

```
01:    <html>
02:    <head><title>using nested loop</title>
03:    </head>
04:    <body>
05:    <h2>Using nested loop</h2>
06:
07:    <?php
08:
09:    $j=0;$i=0;
10:    for($i=0;$i<=5;$i++)
11:    {
12:        print "The Value i=".$i."<br/>";
13:        for($j=0;$j<=3;$j++)
14:        {
15:            print "value J=".$j."<br/>";
16:        }
17:    }
18:
19:    ?>
20:
21:    </body>
22:    </html>
```

ការប្រើប្រាស់ for statement ដើម្បី បង្ហាញ table ទៅកាន់ browser ។

Nesting Two for Loops

```
01:    <!
02:    Nesting loops
03:    >
04:    <html>
05:    <head>
```

```

06:     <title>Nesting Two for Loops</title>
07:     </head>
08:     <body>
09:     <div>
10:
10:     <?php
11:
12:     print "<table border=\"1\">\n";
13:     for ( $y=1; $y<=12; $y++ )
14:     {
15:         print "<tr>\n";
16:         for ( $x=1; $x<=12; $x++ )
17:         {
18:             print "\t<td>";
19:             print ($x*$y);
20:             print "</td>\n";
21:         }
22:         print "</tr>\n";
23:     }
24:     print "</table>";
25:
26:     ?>
27:
28:     </div>
29:
30: </body>
31: </html>

```

មេរៀនទី ៤

Function

1. What is Functions ?

Function គឺជាបន្ទុកនៃ block code ដែលអ្នកបង្កើត ហើយអនុញ្ញាតិអោយអ្នកហៅ យកមកប្រើ នៅក្នុង script របស់អ្នក ។ នៅពេលដែលអ្នកហៅ function ក្នុង ដែលនៅខាងក្នុងនឹងត្រូវអនុវត្តន៍ ហើយអ្នកក៏អាច បញ្ជូនតម្លៃទៅអោយ function ឬ ទទួលតម្លៃពី function មកវិញ ។

1.1 Calling Functions

Function មានពីរប្រភេទគឺ Function ដែលមានស្រាប់ ភ្ជាប់មកជាមួយនឹងភាសាម៉ូឌី និង Function ដែលអ្នកបង្កើតឡើងដោយខ្លួនឯង ។ នៅក្នុងភាសា PHP មាន built-in function ជាច្រើនដែលផ្តល់នូវភាពងាយស្រួលសំរាប់ ការសរសេរកូដរបស់អ្នក ។

1.2 Defining a Function

អ្នកអាចបង្កើត function ដោយប្រើ function statement function function_name(\$argument1, \$argument2)
{
//
}

ប្រសិនបើ function របស់អ្នកតម្រូវអោយមាន argument ចាប់ពីពីរឡើងទៅ អ្នកត្រូវតែចែកវាដោយប្រើប្រាស់សញ្ញា comma(,) ដែលជាទូទៅ argument ទាំងនោះគឺជា variable ហើយតម្លៃរបស់វានឹងត្រូវផ្តល់អោយនៅពេលដែល function ត្រូវបានហៅ ។

ឧទាហរណ៍ ១៦

Declaring a Function

```
<html>
<head>
<title>Declaring a Function</title>
</head>
<body>
<?php
function bigsum()
{
    $sum=0;
    for ($i=1; $i<=10; $i++ )
    {
        $sum+=$i;
        $str= $str."$i+";
    }
    echo substr($str,0,-1).="$sum";
}
bigsum();
?>
</body>
```


</html>

លទ្ធផលដែលបង្ហាញនៅលើ browser គឺជា string នៃតួអក្សរ 1+2+3+.....+10 = 55 ឧទាហរណ៍ខាងលើនេះគឺ ជាការបង្កើតនូវ function មួយឈ្មោះថា bigsum() ដែលជា function មិនមាននូវ Arguments នោះទេ ។ សំរាប់ឧទាហរណ៍ទី ២០ យើងនឹងបង្កើត function ដែលប្រើប្រាស់ argument ។

ឧទាហរណ៍ ២០

```
01: <html>
02: <head>
03: <title>Declaring a Function</title>
04: </head>
05: <body>
06:
07: <?php
08:
09: function bigsum($count)
10: {
11:     $sum=0;
12:     for($i=1; $i<=$count; $i++ )
13:     {
14:         $sum+=$i;
15:         $str= $str."$i+";
16:     }
17:     echo substr($str,0,-1)."$sum";
18: }
19:
20: bigsum(10);
21:
22: ?>
23:
24: </body>
25: </html>
```

1.2.1 Returning Values from User-Defined Functions

នៅឧទាហរណ៍ខាងលើយើងទទួលបាននូវលទ្ធផលជា string នៅលើ browser ដោយការប្រើប្រាស់ bigsum() function ។ function របស់អ្នកអាចធ្វើការផ្លាស់ប្តូរតំលៃដែលអ្នកបានផ្តល់អោយឬ មានពេលខ្លះអ្នកនឹងចង់អោយ function ផ្តល់នូវតំលៃថ្មីអោយអ្នក បន្ទាប់ពីអ្នកបានផ្តល់តំលៃទៅឱ្យវា។

Function មួយអាចទទួលបានតំលៃដោយការប្រើប្រាស់ នូវ return statement ជាមួយនឹងតំលៃដែលត្រូវបញ្ជូនត្រឡប់ ទៅកាន់ function ។ return បញ្ជប់ប្រតិបត្តិការរបស់ function ហើយ បញ្ជូនតំលៃត្រឡប់ ទៅវិញនៅពេលដែល function ត្រូវបានហៅមកប្រើ ។ ខាងក្រោមគឺជាឧទាហរណ៍នៃការបង្កើត function ដែល return តំលៃពី ផលបូកចំនួនពីរតំលៃ ។

ឧទាហរណ៍ ២១

```
01: <html>
02: <head>
03: <title>A Function That Returns a Value</title>
04: </head>
05: <body>
06:
07: <?php
08:
09: function addNums($num1,$num2)
10: {
```

```

11:     $result = $num1 + $num2;
12:     return $result;
13: }
14: print addNums(3,5);
15:
16: ?>
17: </body>
18: </html>

```

addNums() Function ត្រូវបានហៅយកមកប្រើប្រាស់ជាមួយនឹង argument ចំនួនពីរដែលជាតំលៃលេខ 3 និង លេខ 5 ដែលតំលៃទាំងនេះត្រូវបានរក្សាទុកនៅក្នុង variable \$num1 និង \$num2 addNums function ធ្វើការគណនានូវផលបូកនៃតំលៃដែលរក្សាទុកនៅក្នុង variable ទាំងពីរហើយផ្តល់ជាលទ្ធផលទៅអោយ variable \$result ។ អ្នកក៏អាច return តំលៃដោយការហៅ function ដទៃទៀតបានផងដែរ return (another_function(\$an_argument));

2. Variable Scope

Variable ដែលប្រកាសនៅខាងក្នុង function គឺអាចប្រើប្រាស់បានតែនៅខាងក្នុង function នោះប៉ុណ្ណោះ មានន័យថាមិនអាចប្រើប្រាស់បាននៅខាងក្រៅ function ឬ នៅក្នុង function ដទៃទៀត ។

សំរាប់ project ធំៗអាចជួយអ្នកពីគ្រោះថ្នាក់នៃការកែប្រែតំលៃដែលមាននៅក្នុង variable ពេលដែលអ្នកប្រកាស variable ពីរ ហើយមានឈ្មោះដូចគ្នា និង ប្លង់នៅក្នុង function ផ្សេងៗគ្នា ។

ខាងក្រោមគឺជាឧទាហរណ៍ ស្តីពីការបង្កើត variable មួយនៅក្នុង function ហើយព្យាយាមយក Variable នោះមកបង្ហាញលើ browser ពី ក្រៅ function ។

ចំណាំ : Variable ដែលបង្កើតនៅខាងក្នុង function មិនអាចប្រើប្រាស់ពីខាងក្រៅ function បាននោះទេ ។

ឧទាហរណ៍ ២២:

```

01: <html>
02: <head>
03: <title>Local Variable Unavailable Outside a Function</title>
04: </head>
05: <body>
06: <div>
07:
08: <?php
09:
10: function test()
11: {
12:     $testvariable = "this is a test variable";
13: }
14: print "test variable: $testvariable<br/>";
15:
16:
17: ?>
18: </div>
19: </body>
20: </html>

```

អ្នកនឹងបានឃើញនូវលទ្ធផលនៃឧទាហរណ៍ខាងលើ ដែលតំលៃរបស់ variable \$testvariable មិនត្រូវបានបង្ហាញនោះទេ ពីព្រោះពុំមាន variable ណាមួយត្រូវបានបង្កើតនៅខាងក្រៅ function test() នោះទេ ។ គួរចំណាំថា ការយក variable ដែលមិនមាន មកប្រើប្រាស់គឺមិនមានភាព error ឡើយ ។ ម្យ៉ាងវិញទៀត variable ដែលប្រកាសនៅខាងក្រៅ function គឺមិនអាចយកមកប្រើក្នុង function ដោយស្វ័យប្រវត្តិនោះទេ ។

2.1 Accessing Variables with the global Statement

មានពេលខ្លះអ្នកប្រហែលជាត្រូវការប្រើប្រាស់ variable នៅក្នុង function របស់អ្នកដោយពុំត្រូវ អោយមានការបញ្ជូនតំលៃតាម argument នោះទេ ។ ខាងក្រោមគឺជាឧទាហរណ៍ដែលប្រើប្រាស់នូវ global statement ដើម្បីប្រើប្រាស់នូវ variable ដែលប្រកាសខាងក្រៅ function ។

ឧទាហរណ៍ ២៣:

```
01: <html>
02: <head>
03: <title>The global Statement</title>
04: </head>
05: <body>
06:
07: <?php
08:
09: $lottery=42;
10:
11: function lotteryToday()
12: {
13:     global $lottery;
14:     print "Lottery today is $lottery<br />";
15: }
16:
17: lotteryToday();
18: ?>
19: </body>
20: </html>
```

នៅបន្ទាត់ទី ១៣ នៃឧទាហរណ៍ទី២៣ យើងបានប្រើប្រាស់នូវ **global** ពីមុខ variable \$lotteryដែលបានប្រកាសនៅក្នុង function lotteryToday() ដើម្បីសំដៅទៅកាន់ global variable \$lottery ដែលបានប្រកាសនៅខាងក្រៅ function នៅបន្ទាត់ទី ០៩ ។

អ្នកត្រូវតែប្រើប្រាស់នូវ **global** statement នៅរាល់ function ដែលអ្នកចង់ប្រើប្រាស់នូវ global variable ។ អ្នកគួរប្រុងប្រយ័ត្នផងដែរនៅពេលដែលអ្នកផ្តល់តំលៃទៅអោយ variable \$lottoeryនៅក្នុង function ពីព្រោះ \$lottery នឹងធ្វើការផ្លាស់ប្តូរតំលៃរបស់ខ្លួនទាំងអស់នៅក្នុង script របស់អ្នក ។អ្នកក៏អាចប្រកាស variable ជាមួយ **global** statement បានច្រើនក្នុងពេលតែមួយបានផងដែរ ដោយការចែកនូវ variable នីមួយៗដោយសញ្ញា (,) comma ។

ឧទាហរណ៍ : **global** \$var1, \$var2, \$var3;

មេរៀនទី ៥

Array

1. What Is an Array?

អ្នកក៏បានស្គាល់រួចមកហើយអំពី variable ដែលត្រូវបានប្រើដើម្បីផ្ទុកនូវតំលៃផ្សេងៗ ។ ដោយការប្រើប្រាស់ variable អ្នកក៏អាចបង្កើតជា script អោយប្រតិបត្តិការ ឬ បង្ហាញជាព័ត៌មានផ្សេងៗនៅរាល់ពេលដែលអ្នកតំណើរការវា ប៉ុន្តែគួរអោយសោកស្តាយ ដោយអ្នកអាច រក្សាទុកនូវតំលៃតែមួយប៉ុណ្ណោះក្នុងពេលតែមួយ និង ក្នុង variable មួយ ។

Array គឺជា variable ពិសេសដែលអនុញ្ញាតអោយអ្នក ផ្ទុកនូវតំលៃបានច្រើននៅក្នុង variable មួយ ។ រាល់តំលៃនីមួយៗត្រូវបានរក្សាទុកនៅក្នុង index របស់ array ដែលអាច ជាលេខ ឬ ជា តួអក្សរ ។ ជា default ធាតុរបស់ array ដែលជា index គឺចាប់ផ្តើមពីលេខ 0 ។ ហេតុអ្វីបានជាត្រូវប្រើប្រាស់ array ?

ប្រសិនបើអ្នកមានតំលៃចំនួន ៥ ដែលត្រូវរក្សាទុក នោះអ្នកប្រាកដជាត្រូវបង្កើត variable ចំនួនប្រាំផងដែរ **Array** គឺ flexible ព្រោះវាអាចផ្ទុកតំលៃបានពីរ ឬ ពីររយ តំលៃ ដោយពុំមានការផ្អាកនូវ variable ថ្មីទៀតហើយ array ក៏អាចអោយអ្នកធ្វើការជាមួយតំលៃរបស់វា បានយ៉ាងងាយ ដូចជា ការ loop ធាតុរបស់ array នីមួយៗ ឬ តំរាប់ធាតុរបស់វាទៅលំដាប់នៃលេខរៀង ឬ ជាតួអក្សរ ទៅតាមការកំណត់នៅក្នុង system របស់អ្នក ។ ខាងក្រោមគឺជាការបង្ហាញនូវធាតុរបស់ users array ដែលមានធាតុទី៤ ជា index ទី៣ នៃ users ។

The Elements in the users Array		
Index Number	Value	Which Element?
0	Sok	First
1	Sao	Second
2	Setha	Third
3	Mesa	Fourth

PHP ក៏បានផ្តល់នូវលទ្ធភាពដើម្បីរៀបចំនូវ indexed របស់ array ដោយការប្រើប្រាស់ទាំង លេខ និង អក្សរផងដែរ ។

2. Creating Arrays

អ្នកអាចបង្កើតនូវ array variable ដោយប្រើវិធីពីរយ៉ាងគឺ ការប្រើប្រាស់ array() construct ឬ ការប្រើប្រាស់នូវសញ្ញា square brackets ([]) ។

2.1 Defining Arrays with the array () Construct.

array()construct វាមានសារៈប្រយោជន៍នៅពេលដែលអ្នកចង់ផ្តល់តំលៃច្រើនទៅអោយ array ក្នុងពេលតែមួយ ។ ខាងក្រោមគឺឧទាហរណ៍នៃការបង្កើត array មួយដែលមានឈ្មោះថា \$users ហើយយើងបានផ្តល់តំលៃជា string ចំនួនបួន តំលៃទៅអោយវា ។

```
$users = array ("Sok", "Sao", "Mata", "Mesa");
```

```
ឥឡូវនេះអ្នកអាចយកធាតុរបស់ $users មកប្រើប្រាស់ដោយការប្រើប្រាស់ index របស់ array  
print $users[2];
```

លទ្ធផលនៃឧទាហរណ៍ខាងលើនឹង បង្ហាញជាអក្សរ Mata ដែល index របស់វាហើយត្រូវបានដាក់នៅចន្លោះ square brackets ([2]) បន្ទាប់ពីឈ្មោះរបស់ array (\$users) ។

2.2 Defining or Adding to Arrays with the Array Identifier.

អ្នកអាចបង្កើត array ថ្មី ឬ បន្ថែមតំលៃទៅអោយ array បានដោយការប្រើប្រាស់នូវ **array Identifier** ។
array identifier គឺជាបន្តនៃ square brackets និង index របស់វា ជាលេខ ឬ ជាឈ្មោះ។ខាងក្រោមជា ឧទាហរណ៍នៃការបង្កើត \$users array ដោយប្រើវិធីសាស្ត្រមួយផ្សេងទៀត ។

```
$users[] = " Sok";  
$users[] = " Sao";  
$users[] = " Dara";  
$users[] = " Mesa";
```

ឧទាហរណ៍ខាងលើមិនទាមទារអោយអ្នកដាក់នូវតំលៃលេខដែលជា index នៅចន្លោះ square Brackets នោះឡើយ ព្រោះ PHP នឹងផ្តល់តំលៃ index ដោយស្វ័យប្រវត្តិទៅអោយ array ។ ម្យ៉ាងវិញទៀតអ្នកក៏អាចដាក់នូវតំលៃជា index របស់ array បានផងដែរ ប៉ុន្តែមិនមានការគាំទ្រអោយធ្វើបែបនេះនោះទេសូមពិនិត្យមើល ការសរសេរកូដខាងក្រោម៖

```
$users[0] = "Phanit";  
$users[200] = "Ratana";
```

Array ខាងលើមានធាតុត្រឹមតែពីរប៉ុណ្ណោះ ប៉ុន្តែ index ចុងក្រោយរបស់វាគឺជា index ទី២០០ PHP នឹងមិនកំណត់តំលៃធាតុទៅអោយ index ដែលនៅចន្លោះ នោះឡើយ ដែលធ្វើអោយមានការភាន់ច្រឡំនៅពេលដែលចង់ព្យាយាមយកធាតុដែលមានក្នុង array នោះមកប្រើ ខណៈពេលដែលអ្នកចង់ប្រើនូវ ធាតុរបស់វា ណាមួយទៅតាមចិត្តរបស់អ្នក ។ សំរាប់ការបង្កើត array អ្នកអាចប្រើ នូវ array() construct រួចប្រើនូវ array identifier ដើម្បីបន្ថែមនូវធាតុថ្មីទៀតក៏បាន ។
សូមពិនិត្យឧទាហរណ៍ខាងក្រោម ៖

```
$users = array ("Sok", "Sao", "Mata", "Mesa");  
$users[] = "Menear";
```

3. Populating an Array with array fill()

ប្រសិនបើអ្នកចង់ដាក់នូវតំលៃជា default នៅចន្លោះនៃ index array នោះអ្នកប្រហែលជាប្រើនូវ array() function ដូចខាងក្រោម ។

\$member = array ("cambodia", "cambodia", "cambodia","cambodia "); ឬ អ្នកអាចប្រើប្រាស់នូវ array ម្យ៉ាងទៀតដូចជា

```
$member[] = "Cambodia";  
$member[] = "Cambodia";  
$member[] = "Cambodia";  
$member[] = "Cambodia";
```

PHP បានផ្តល់នូវ function ដើម្បីដោះស្រាយនូវបញ្ហានេះដោយប្រើនូវ array_fill() function ដែលត្រូវអោយអ្នកផ្តល់នូវ arguments ចំនួន៣សំរាប់ function នេះ ទី១គឺជាតំលៃ index ដែលចាប់ផ្តើម

ទីពីរគឺជា ចំនួនធាតុដែលអ្នកចង់បាន ហើយ argument ទី៣ គឺជាតំលៃដែលអ្នកត្រូវផ្តល់អោយជាធាតុរបស់Array ។
ដោយការប្រើប្រាស់នូវ array_fill() function អ្នកអាចសរសេរនូវបំណែកកូដខាងលើឡើងវិញ ដូចខាងក្រោម :

```
$member = array_fill( 0, 4, "Cambodia" );
```

ឧទាហរណ៍: ២៤

```
<?
$member = array_fill(0,4,"Cambodia");
$member[] = "Malaysia";
print $users[2];
?>
```

4. Associative Arrays

Associate array គឺជា array ដែលប្រើប្រាស់នូវ index ជាអក្សរនៅ ចន្លោះ square brackets អ្នកអាចបង្កើតនូវ
associate array ដោយ array construct(array()) ឬ array identifier (array[])

ឧទាហរណ៍

```
$character = array("name" => "Sok",
                  "occupation" => "Student",
                  "age" => 12,
                  "Adr" => "Phnom Pehn" );
```

ឥឡូវនេះយើងអាចប្រើប្រាស់នូវធាតុ array របស់ \$character ដូចខាងក្រោម:

```
print $character['occupation'];
```

keys ដែលនៅក្នុង associate array គឺជា string ដូច្នេះ engine នឹងបង្ហាញរបាយការណ៍ error ប្រសិនបើ keys នៃ array មិនមាន
quoted ។ ដូច្នេះអ្នកគួរតែប្រើនូវ quotation marks នៅពេលដែលអ្នកប្រើkeys នៃ array ជា string ។

```
print $character[occupation]; // មិនត្រឹមត្រូវ
print $character["occupation"]; // ត្រឹមត្រូវ
```

ប្រសិនបើ key ត្រូវផ្ទុកនៅក្នុង variable នោះ អ្នកមិនចាំបាច់ត្រូវប្រើ quotation mark នោះឡើយ ។

```
$occ = "occupation";
print $character[$occ]; // ត្រឹមត្រូវ
```

4.1 Directly Defining or Adding to an Associative Array

អ្នកអាចបង្កើត ឬ បន្ថែម ឈ្មោះ/តំលៃ ទៅអោយ associate array ដោយការផ្តល់នូវតំលៃទៅអោយឈ្មោះនៃធាតុរបស់
array ដូចខាងក្រោម ។

```
$character["name"] = "sok";
$character["occupation"] = "Programmer";
$character["age"] = 30;
$character["address"] = "Phnom Penh";
```

5. Getting the Size of an Array

count() function ផ្តល់នូវចំនួនធាតុរបស់ array ដែលមាន បំណែកក្នុងខាងក្រោមយើងបានប្រើប្រាស់ នូវ count()function ដើម្បីយកធាតុចុងក្រោយនៃ array \$users មកប្រើ ។

```
$users = array ("Sambath", "Ratana", "Davy", "Ryda" );
print $users[count($users)-1];
```

ឧទាហរណ៍ ២៦

```
<html>
<body>
<head>
<title>Getting size of array</title>

<body>
<h2>Getting size of array</h2>

<?php

$users[]="Sok";
$users[]="Som";
$users[]="Mom";
$users[]="Mab";

print "<br/>".count($users);

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

ចំណាំ : count() ផ្តល់តំលៃនៃចំនួនសរុបរបស់ធាតុ array មិនមែនជាតំលៃនៃ index ចុងក្រោយនោះទេ ។

ឧទាហរណ៍ខាងក្រោមនេះគឺការបង្កើត array ដោយការផ្តល់នូវ index ទៅអោយ array ដោយខ្លួនអ្នក

```
$users[35] = "Sok";
$users[52] = "Som";
$users[890] = "Mom";
$users[52] = "Mab";
```

count() នៅតែផ្តល់នូវចំនួនសរុបនៃធាតុ array ដដែល គឺមានចំនួន ៤ ធាតុ ។ ហើយអ្នកក៏អាចទទួលនូវធាតុរបស់ array ចុងក្រោយបានដោយការប្រើនូវ end() function ដែលទាមទារត្រឹមតែ argument មួយ ប៉ុណ្ណោះ ។ ឧទាហរណ៍ print end(\$users);

6. Looping Through an Array

foreach statement គឺជាវិធីសាស្ត្រមួយយ៉ាងប្រសើរដើម្បី loop រាល់ធាតុនិមួយៗដែលមាននៅ ក្នុង array ។

អ្នកអាចប្រើប្រាស់នូវ foreach statement ដូចខាងក្រោម :

```
foreach( $arr as $temp )
{
//.....
}
```

\$arr គឺជា array ដែលអ្នកត្រូវ loop ហើយ \$temp គឺជា variable ដែលអ្នកនឹងត្រូវរក្សាទុកជាបណ្តោះអាសន្ន នូវធាតុនិមួយៗ ក្នុងគ្រា loop ។ ក្នុងខាងក្រោមគឺជាការបង្កើត array មួយហើយយើង ប្រើ foreach statement ដើម្បីយកធាតុនិមួយៗមក បង្ហាញលើ browser ។

ឧទាហរណ៍ ២៦

```
<html>
<head>
<title>Looping through array</title>
</head>

<body>
<h2>Looping through array</h2>

<?php

$users = array ("Sok", "Som", "Mab", "Mab" );

foreach ( $users as $val )
{
print "$val<br />";
}

?>

</body>
</html>
```

ឧទាហរណ៍ ២៧

```
<html>
<head>
<title>Looping through array</title>
</head>

<body>
<h2>Looping through array</h2>

<?php

$users[]="Sok";
$users[]="Som";
$users[]="Mom";
$users[]="Mab";

foreach( $user as $val )
{
print "$val<br/>";
}

?>

</body>
</html>
```


6.1 Looping Through an Associative Array

ដើម្បីប្រើប្រាស់នូវ foreach statement ជាមួយ associat array ទាំង keys និង values អ្នកអាចប្រើប្រាស់ជាមួយនឹង foreach statement ដូចខាងក្រោម :

```
foreach( $arr as $key=>$value )
{
    //.....
}
```

\$arr គឺជា array ដែលយើងនឹងត្រូវ loop ហើយ \$key គឺជា variable ដែលផ្ទុកនូវ key និមួយៗជាបណ្តោះអាសន្ន ហើយ \$val គឺជា variable ដែលផ្ទុកនូវ value របស់ array និមួយៗជាបណ្តោះអាសន្នក្នុងគ្រា loop ម្តងៗ ។

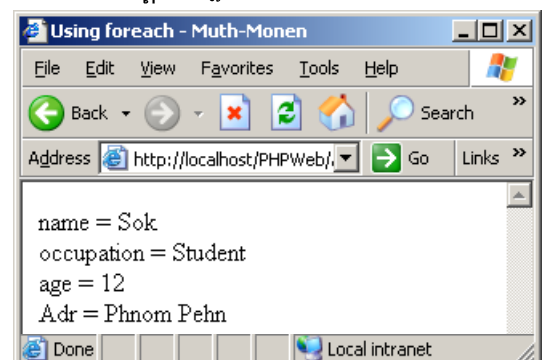
ឧទាហរណ៍ ២៨

```
01: <html>
02: <head>
03: <title>Looping thought associate array</title>
04: </head>
05: <body>
06: <h2> Looping thought associate array</h2>
07:
08: <?php
09:
10: $character = array ("name" => "Sao",
11:                    "occupation" => "Student",
12:                    "age" => 12,
13:                    "Adr" => "Phnom Pehn");
14:
15: foreach ( $character as $key=>$val )
16: {
17: print "$key = $val<br />";
18: }
19:
20: ?>
21:
22: </body>
23: </html>
```

Arrayដែលបានបង្កើតនៅបន្ទាត់ទី១១ ហើយប្រើប្រាស់នូវ foreach statement នៅបន្ទាត់ទី ១៥

ដើម្បី loop នូវធាតុដែលជាតំលៃ និង key របស់វា ។ រាល់ key និមួយៗត្រូវបានផ្ទុកនៅក្នុង variable ដែលផ្តល់ឈ្មោះអោយថា \$key និង រាល់ value និមួយៗត្រូវបានផ្ទុកនៅក្នុង variable ដែលផ្តល់ឈ្មោះថា \$val

ហើយត្រូវបានបង្ហាញជាលទ្ធផលទៅកាន់ browser នៅបន្ទាត់ទី ១៧ ។



7. Joining Two Arrays with array_merge()

`array_merge()` មានតួនាទីផ្គុំនូវធាតុរបស់ `array` ពីរ ឬ `array` ច្រើនបញ្ចូលគ្នា រួចផ្តល់ជា `Array` ថ្មីមួយទៀតដែលជាបន្តនៃធាតុ `array` ទាំងអស់នោះ។ ឧទាហរណ៍ខាងក្រោមគឺជាឧទាហរណ៍នៃការបង្កើតនូវ `array` ពីរ គឺ `array $arr1` និង `array $arr2` ហើយធ្វើការបញ្ចូលគ្នារវាងធាតុនៃ `array` ទាំងពីររួចផ្តល់តំលៃទាំងនោះទៅអោយ `$arr3` បន្ទាប់មក `loop` រាល់ធាតុដែលមាននៅក្នុង `$arr3` ។

ឧទាហរណ៍ ២៩

`array_merge.php`

```
<html>
<head>
<title>Joining Two array with array_merge()</title>
</head>

<body>
<h2>Using array_merge()</h2>

<?php

$arr1 = array("a", "b", "c");
$arr2 = array(1,2,3);
$arr3 = array_merge( $arr1,$arr2 );

foreach ( $arr3 as $val )
{
    print "$val<br />";
}

?>

</body>
</html>
```

`$arr3` `array` ផ្គុំនូវរាល់ធាតុដែលមាននៅក្នុង `$arr1` និង `$arr2` `array` ។ `foreach` statement នឹងបង្ហាញនូវបន្តនៃធាតុ `array` ('a', 'b', 'c', 1, 2, 3) ជាមួយនឹង `
` នៅចន្លោះធាតុនីមួយៗ ។

8. Adding Multiple Variables to an Array

`array_push()` ទទួលយកនូវ `array` និង តំលៃផ្សេងពី `parameters` ដែលតំលៃទាំងនោះគឺជាធាតុដែលនឹងត្រូវបន្ថែមទៅអោយ `array` ។ `array_push()` function មិនមានលក្ខណៈដូច `array_merge()` នោះទេ ពីព្រោះ `array` ដែលបញ្ជូនទៅជា `argument` ដំបូងនឹងត្រូវប្រែប្រួល ហើយ `array_push()` នឹង ផ្តល់មកវិញនូវតំលៃសរុបទាំងអស់ ដែលមាននៅក្នុង `array` នោះវិញ ។

ឧទាហរណ៍ ៣០

```

01: <html>
02: <head><title>Adding Mulples variables to an Array</title></head>
04: <body>
05:
06: <?php
07:
08: $arr1 = array ("a", "b", "c");
09: $total = array_push( $arr1,1,2,3);
10: print "There are $total elements in \$arr1 ";
11:
12: foreach ( $arr1 as $val )
13: {
14: print "$val<br/>";
15: }
16:
17: ?>
18:
19: </body>
20: </html>

```

ឥឡូវនេះ \$arr1 បានបន្ថែមនូវធាតុជា integer ចំនួនបីធាតុ ដែលបានមកពីការប្រើប្រាស់នូវ array_push() function ដូច្នេះ \$arr1 ផ្ទុកនូវតំលៃសរុបចំនួន ៦ ធាតុហើយត្រូវបានបង្ហាញនូវធាតុនីមួយៗទៅកាន់ browser ដោយកូដដែលមាននៅក្នុង foreach statement ។

ចំណាំ: យើងបានប្រើនូវ backslash ពេលដែល print ជា string "-----\\$arr1" នៅបន្ទាត់ទី ១០

ប្រសិនបើអ្នកប្រើនូវ និមិត្តសញ្ញានៅពីមុខពាក្យ ឬ ឃ្លាណាមួយនោះ PHP នឹងបញ្ចូលជាតំលៃទៅអោយទៅតាមឈ្មោះ variable ។ ឧទាហរណ៍ ខាងលើយើងចង់បង្ហាញនូវ string '\$arr1' ដែលមិនមែនជាតំលៃរបស់ variable នោះទេ ដូច្នេះដើម្បី print នូវ special character (\$) យើងត្រូវតែប្រើនូវ backslash (\)

នៅពីមុខ ។ ប្រសិនបើអ្នកចង់បន្ថែមធាតុទៅអោយ array ហើយជាធាតុដំបូងទៀតនោះ អ្នកអាចប្រើប្រាស់ជាមួយនឹង array_unsift() ។

ឧទាហរណ៍ ៣១

```

<html>
<head>
<title>Using array_unsift() </title>
</head>

<body>
<h2>Using array_unsift()</h2>

<?php

```

```
$arr1 = array ("a", "b", "c");
$total = array_unshift( $arr1, 1, 2, 3 );
print "There are $total elements in \$arr1 ";
```

```
foreach ( $arr1 as $val )
{
print "$val<br/>";
}
```

:

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

ឥឡូវនេះ \$arr1 array ផ្ទុកនូវតំលៃ 1, 2, 3, "a", "b", "c" ។

9. Removing the First Element of an Array with array_shift()

array_shift() មានតួនាទីយកចេញនូវធាតុទីមួយនៃ array ។សូមពិនិត្យ ឧទាហរណ៍ខាងក្រោម:

ឧទាហរណ៍ ៣២

```
<html>
<head>
<title>Remove first array element with array_shift()</title>
</head>
<body>
<h2>Using array_shift()</h2>

<?php

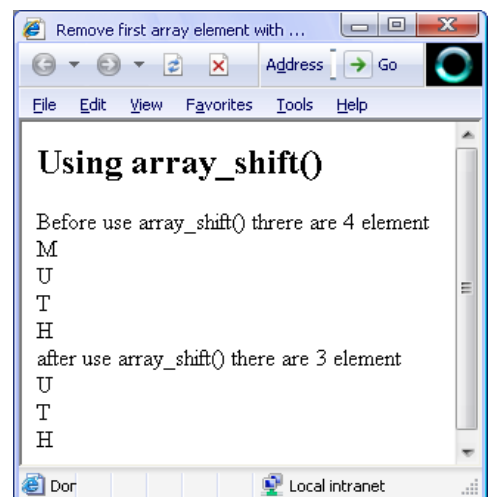
$user=array("M","U","T","H");
$total=count($user);
echo "Before use array_shift() there are $total element<br/>";

foreach($user as $val)
{
echo "$val <br/>";
}

array_shift($user);
$total=count($user);
echo "after use array_shift() there are $total element<br/>";

foreach($user as $val)
{
print "$val <br/>";
}

?>
```



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

10. Slicing Arrays with array_slice()

array_slice() អនុញ្ញាតអោយអ្នកទាញយកចំនួននៃធាតុ array ដែលទាមទារអោយអ្នកផ្តល់នូវ argument ចំនួនបី ទីមួយគឺជា array variable ទី២ជាទីតាំងដែលត្រូវចាប់ផ្តើមយក និងទីបី គឺជាចំនួនធាតុ របស់ array ដែលត្រូវការហើយ argument ទីបីនេះគឺ (optional) ប្រសិនបើអ្នកមិនប្រើនូវ argument នេះទេ array_slice() នឹងចាប់យកនូវធាតុទាំងអស់នៃ array ចាប់ពីទីតាំងដែលអ្នកបានផ្តល់ជាតំលៃនៅ ក្នុង argument ទី១មក ។

ឧទាហរណ៍ ៣៣

```
<html>
<head><title>Using array_slice()</title>
</head>
<body>
<h2>Using array_slice()</h2>

<?php

$first = array ("a", "b", "c", "d", "e", "f");
$second = array_slice($first, 2, 3);

foreach ( $second as $val )
{
    print "$val<br />";
}

?>

</body>
</html>
```

ក្នុងដែលបង្កើតនៅឧទាហរណ៍ ៣៣ នឹងបង្ហាញនូវលទ្ធផល 'c', 'd', និង 'e' ដែលជាធាតុរបស់

\$second បន្ទាប់ពីប្រើនូវ array_slice() ដោយយកធាតុទាំងអស់របស់ \$first ។

11. Sorting Arrays

11.1 Sorting Numerically Indexed Arrays with sort()

sort() ទទួលយកនូវ argument មួយដែលជា array ហើយធ្វើការតំរៀបវារៈទៅតាមលំដាប់នៃតួ អក្សរឬពីតូចទៅធំ ឧទាហរណ៍ ៣៤ គឺជាការបង្កើត array ដោយផ្តល់នូវតំលៃជា string ទៅអោយបន្ទាប់មកយើងប្រើ sort() ដើម្បីតំរៀបធាតុទាំងនោះពី A-Z ឬ ពីតូចទៅធំ រួចបង្ហាញជាលទ្ធផលទៅកាន់ Browser ។

សូមពិនិត្យក្នុងឧទាហរណ៍ ៣៤

```
<html>
<head><title> sorting array with sort() function</title>
</head>
<body>
<h2>Using sort() function </h2>

<?php

$_array = array ("Bayon", "Taprom", "Angkor", "Presh vihear");
sort( $_array );

foreach ( $_array as $var )
{
print "$var<br />";
}

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

អ្នកក៏អាចផ្លាស់ប្តូរនូវលំដាប់នៃការតំរៀបធាតុរបស់ array ពីលំដាប់ តូចទៅធំ ឬ ពីលំដាប់ធំ ទៅតូចវិញដោយការប្រើនូវ rsort() ។ សូមកុំព្យាយាមប្រើ sort()ឬ rsort() ជាមួយ associate array ពីព្រោះ វានឹងធ្វើអោយអ្នកបាត់បង់នូវ key របស់ array ។

11.2 Sorting an Associative Array by Value with asort()

asort() ទទួលយក argument ដែលជាប្រភេទ associate array ហើយធ្វើការតំរៀបតំលៃ របស់ array នោះពី A-Z ឬ ជាតំលៃលេខ ពីតូចទៅធំ អាស្រ័យទៅលើប្រភេទនៃតំលៃ ។

ឧទាហរណ៍ ៣៥

```
<html>
<head><title> sorting an associate array by value with asort() </title>
</head>
<body>
<h2>sorting an associate array by value with asort()</h2>

<?php

$first = array("Cocacola"=>3,"Pepsi"=>2,"Fanta"=>1);
echo "<b>an associate Array before use asort </b> <br/>";

foreach ( $first as $key => $val )
{
```

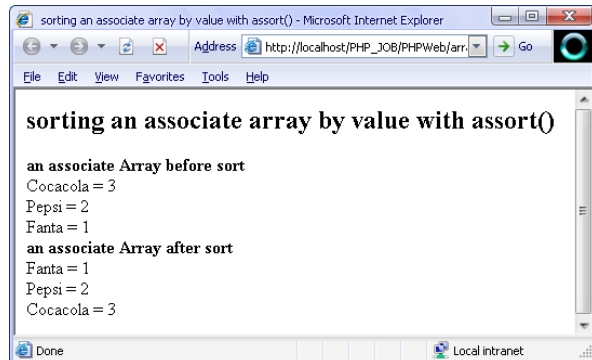
```
print "$key = $val<br />";
}
echo "<b>an associate Array after asort </b><br/>";
asort( $first );
```

```
foreach ( $first as $key => $val )
{
print "$key = $val<br />";
}
```

```
?>
```

```
</body>
```

```
<html>
```



សំរាប់ការប្តូរលំដាប់នៃការតំរៀបពីតូចទៅធំ ឬ ពីធំទៅតូច អ្នកអាចប្រើរន្ទូវ **arsort()** ។

11.3 Sorting an Associative Array by Key with ksort()

ksort() ទទួលនូវ argument ដែលជា associate array ហើយធ្វើការតំរៀបតំលៃ របស់វា និមួយៗទៅតាមលំដាប់ពី A-Z ឬ ជាលេខ ពីធំទៅតូច ដោយអាស្រ័យទៅលើ key របស់ array នោះ ។

ឧទាហរណ៍ ៣៦

```
<html>
```

```
<head><title> sorting an associate array by key with asort() </title>
```

```
<head>
```

```
<body>
```

```
<h2>sorting an associate array by key with ksort()</h2>
```

```
<?php
```

```
$first = array("Cocacola"=>3,"Pepsi"=>2,"Fanta"=>1);
echo "<b>an associate Array before use ksort </b> <br/>";
```

```
foreach ( $first as $key => $val )
```

```
{
```

```
print "$key = $val<br />";
```

```
}
```

```
echo "<b>an associate Array after use ksort </b><br/>";
```

```
ksort( $first );
```

```
foreach ( $first as $key => $val )
```

```
{
```

```
print "$key = $val<br />";
```

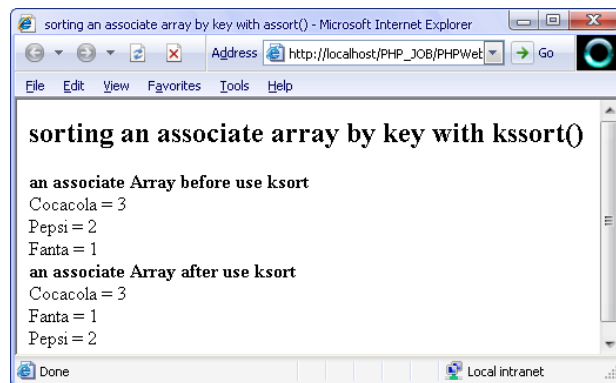
```
}
```

```
?>
```

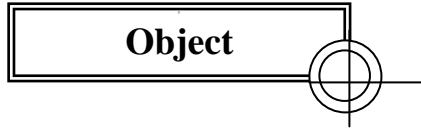
```
</body>
```

```
<html>
```

អ្នកក៏អាចផ្លាស់ប្តូរលំដាប់នៃការតំរៀប ពីធំទៅតូច ឬ ពី Z-A វិញដោយប្រើ **ksort()** ។



មេរៀនទី ៦



1. What Is an Object ?

Object គឺជា បណ្តុំនៃ variables និង functions ដែលបិទនៅក្នុង template ពិសេសមួយដែល គេហៅថា class ។ object ណាមួយដែលមាននៅខាងក្នុង ពីការប្រើប្រាស់វា ដោយផ្តល់ជា Interface សំរាប់អោយអ្នកបញ្ជូននូវ បញ្ហាដើម្បីទទួលបានព័ត៌មានត្រលប់មកវិញ ។ interface ទាំងនោះគឺជា Function ពិសេសដែលគេអោយឈ្មោះ methods ហើយរាល់ methods ទាំងអស់នៃ object គឺត្រូវតែ Access ទៅកាន់ special variable ហៅថា properties ។

គុណសម្បត្តិរបស់ object-oriented code គឺផ្តល់នូវការកាត់បន្ថយការសរសេរកូដ ដដែលៗពីព្រោះ class ដែលបាន បង្កើត ជា object មានភាពងាយស្រួលយកទៅប្រើប្រាស់ពី project មួយទៅកាន់Project មួយផ្សេងទៀត ។ បន្ថែមពីនេះវាមានលទ្ធភាពបង្កើត នូវ child classes ដែល inherit ហើយនឹងOverride នូវលក្ខណៈផ្សេងៗពី parents class ។

2. Creating an Object

ដើម្បីបង្កើត object មួយអ្នកត្រូវតែរៀបចំនូវ template មួយដើម្បីអោយវាអាច instantiated ដែល template នេះគឺជា class នៅក្នុងភាសា PHP ដើម្បីប្រកាស class មួយអ្នកត្រូវតែប្រើ class Keyword ។

```
class Person
{
    // a very minimal class
}
```

Person class គឺជាមូលដ្ឋានដ៏សំខាន់មួយដែលអ្នកអាច instantiate រាល់តំលៃនិមួយៗពីPersonObject ។ ដើម្បីបង្កើត instance មួយនៃ object អ្នកត្រូវតែប្រើ new statement ។

```
$obj1 = new Person();
$obj2 = new Person();
print "\$obj1 is an ".gettype($obj1)."<br />";
print "\$obj2 is an ".gettype($obj2)."<br />";
```

អ្នកអាច test នូវ \$obj1 និង \$obj2 ជាមួយនឹង gettype() function ។ gettype function ទទួលតំលៃVariable និមួយៗហើយផ្តល់ជា string មកវិញដែលប្រាប់អ្នកអំពីអ្វីដែលអ្នកកំពុងតែធ្វើការជាមួយ ។

3. Object Properties

Object ដែល access ទៅកាន់ special variable ត្រូវបានគេហៅថា properties ។ អ្នកអាចប្រកាសនូវ special variable ទាំងនោះបាននៅគ្រប់ទីកន្លែងទាំងអស់នៃ class ប៉ុន្តែដើម្បីអោយមានភាពងាយស្រួលអ្នកគួរតែប្រកាសវានៅខាងលើ ។ សំរាប់ property មួយវាអាចជា value ឬ ជា array ឬ ជាObject ផ្សេងៗ ។

```
class Person
{
    var $name = "Vichet";
}
```

យើងបាន declared នូវ variable ជាមួយនឹង var keyword ដែលនេះគឺជាវិធីសាស្ត្រតែមួយគត់ដើម្បី declare នូវ property មួយនៅក្នុងភាសា PHP 4 យើងនឹងឃើញពីលក្ខណៈបន្ថែមរបស់ PHP 5 នៅឧទាហរណ៍ក្រោយៗទៀត ។ ដូច្នេះប្រសិនបើអ្នកចង់សរសេរកូដអោយ compatible ជាមួយ PHP 4 នោះ អ្នកត្រូវតែប្រើ var keyword ។

ឥឡូវនេះ Person objet ដែលបានបង្កើតមានផ្ទុកនូវ property មួយ ដែលអោយឈ្មោះថា \$name ជាមួយនឹងតំលៃរបស់វាគឺ "Vichet" ។ អ្នកអាច access នូវ property នេះ ពីខាងក្រៅ object និងធ្វើការផ្លាស់ប្តូរតំលៃរបស់វាដោយប្រើឧទាហរណ៍ ៣៧ ។

```
<?
class Person
{
    var $name = "Vichet";
}
$obj1 = new Person();
$obj2 = new Person();
$obj1->name = "Soporn";
print "$obj1->name<br />";
print "$obj2->name<br />";

?>
```

(->) operator នេះអនុញ្ញាតអោយអ្នកធ្វើការ access ឬ ផ្លាស់ប្តូរនូវតំលៃរបស់ properties នៃ Object ។ យើងបានផ្តល់នូវតំលៃ "Soporn" ទៅអោយ \$name property តាមរយៈ \$obj1->name ។

4. Object Methods

Method គឺជា function ដែល defined នៅក្នុង class ។ គ្រប់ object ដែលបាន instantiated ពី class តែងតែមាន method's ជានិច្ច យើងនឹងបន្ថែមនូវ method មួយទៅអោយ Person class នៅ ឧទាហរណ៍ ៣៨ ។

```
01: <?php
02:
03: class Person
04: {
05:     var $name = "Vichet";
06:
07:     function getName()
08:     {
09:         return "Ratana";
10:     }
11:
12: }
13:
14: $person = new Person();
15: print $person->getName();
16: // outputs "Ratana"
17: ?>
```

ដូចដែលអ្នកបានជួបប្រទះមកហើយអំពី method នៃឧទាហរណ៍ ៣៧ ដែលមើលទៅវាមានលក្ខណៈដូចគ្នាទៅនឹង function ធម្មតាដែរ ។ អ្នកអាច call object method ដោយការប្រើប្រាស់និមិត្តសញ្ញា (->) ។ method ដែលបាន access ទៅកាន់ member variables នៃ class ខាងលើបាន return នូវ string "Ratana" ដែលនេះមិនមែនជាការអនុវត្តន៍ដែលត្រឹមត្រូវនោះទេ method គួរតែ return តំលៃដែល copy ពី \$name property និង មិនមែនជា string literal ។ អ្នកក៏បានស្គាល់រួចមកហើយអំពីការ access a property ពីខាងក្រៅ object ប៉ុន្តែតើត្រូវធ្វើយ៉ាងណាដើម្បី refer វានៅខាងក្នុង class ខ្លួនឯង? សូមពិនិត្យមើលឧទាហរណ៍ ៣៩ ។

5. Accessing a Property from Within a Method

ឧទាហរណ៍ ៣៩

```
01: <html>
02: <head><title>Accessing a property from within a method</title>
03: </head>
04: <body>
05: <h2>Accessing a property from within a method</h2>
06:
07: <?php
08:
09: class Person
10: {
11:     var $name = "Thary";
12:
13:     function getName()
14:     {
15:         return $this->name;
16:     }
17: }
18:
19: $person = new Person();
20: $person->name = "Bopha";
21: print $person->getName();
22: //outputs "Bopha"
23: ?>
24: </body>
25: </html>
```

Class ដែលបានបង្កើតក្នុងឧទាហរណ៍ ៣៩ យើងបានប្រើប្រាស់នូវ special variable \$this ដើម្បី refer ទៅកាន់ current instantiated object នៅបន្ទាត់ទី ១៥ គឺ \$name ។ object ដែល refer ទៅកាន់ខ្លួនឯងត្រូវតែប្រើ \$this variable ភ្ជាប់ជាមួយសញ្ញា (->) ដោយការប្រើវិធីនេះអ្នកអាច access រាល់ property ឬ method ដែលស្ថិតនៅក្នុង class ខ្លួនឯង ។ អ្នកអាចគិតថាចង់អោយ object មាននូវតំលៃនៃ \$name property ផ្សេងៗគ្នា ដោយអ្នកអាចធ្វើការរៀបចំនូវតំលៃរបស់ \$name property ដូចដែលបានអនុវត្តនៅក្នុងឧទាហរណ៍ ៣៩ ឬ អ្នកអាចបង្កើតជា method សំរាប់ធ្វើការជាមួយវាដូចមានបង្ហាញក្នុងឧទាហរណ៍ ៤០ ។

6. Changing the Value of a Property from Within a Method

ឧទាហរណ៍ ៤០

```

01: <html>
02: <body>
03: <head><title>Changing the value of a property from within a method</title>
04: </head>
05: <body>
06: <h2>Changing the value of a property from within a method</h2>
07:
08: <?php
09:
10: class Person
11: {
12:     var $name = "tepy";
13:
14:     function setName( $n )
15:     {
16:         $this->name = $n;
17:     }
18:
19:     function getName()
20:     {
21:         return $this->name;
22:     }
23: }
24:
25: $person = new Person();
26: $person->setName("darya");
27: print $person->getName();
28: // outputs "darya"
29:
30: ?>
31:
32: </body>
33: </html>

```

\$name property នៃ object ចាប់ផ្តើមដោយតំលៃ string "tepy" នៅបន្ទាត់ទី១២ ប៉ុន្តែបន្ទាប់ពី setName() method ត្រូវបានហៅនៅបន្ទាត់ទី 26 តំលៃរបស់វាត្រូវបានប្តូរទៅជា "darya" វិញ ។ Object គឺមានលទ្ធភាពផ្លាស់ប្តូរនូវ property របស់ខ្លួនឯងបាន ហើយសំរាប់ការបញ្ជូននូវ arguments ទៅកាន់ method វិញគឺអ្នកអាចប្រើនូវវិធី ដូចដែលអ្នកអនុវត្តន៍វាជាមួយ function ធម្មតាដែរ ។

7. Object Constructors

នៅឧទាហរណ៍មុនយើងបានប្រើប្រាស់ method មួយឈ្មោះថា setName() ដើម្បីធ្វើការផ្លាស់ប្តូរតំលៃរបស់ \$name property ម្យ៉ាងវិញទៀត ការផ្តល់នូវតំលៃតំបូងសំរាប់ \$name property នៅក្នុង Class គឺ hard-code ។ var \$name = "tepy";

ប្រសិនបើយើងគិតថា \$name property ផ្ទុកនូវតំលៃផ្សេងៗគ្នា រាល់ពេលដែល instance នៃ Person class យើងអាចធ្វើអោយកាន់តែប្រសើរជាងមុនដោយការ set \$name property នៅពេលដែល Object ត្រូវបាន initialize ។ យើងអាចប្រើប្រាស់ special function ដែលគេអោយឈ្មោះថា constructor ដើម្បី set properties

និងបំពេញការងារផ្សេងៗទៅតាមតំរូវការនៃការងារ។ constructor គឺត្រូវហៅដោយ ស្វ័យប្រវត្តិនៅពេលដែល object ត្រូវបាន instantiated ដោយការប្រើប្រាស់ new keyword ។

អ្នកអាចបង្កើត constructor តាមវិធីសាស្ត្រពីរយ៉ាង ទី១ គឺ constructor ដែលជា function មានឈ្មោះដូច class ។ ឧទាហរណ៍ ៤១ នឹងបង្ហាញអ្នក constructor សាមញ្ញមួយទៅអោយ Person Class ដែលក្នុងខាងក្រោមនេះប្រើប្រាស់បានសំរាប់តែ PHP 5 ប៉ុណ្ណោះ ។

ឧទាហរណ៍ ៤១ A Class with a Constructor

```
01: <html>
02: <head>
03: <title>A Class with a Construct</title>
04: </head>
05: <body>
06: <h2>A Class with a Construct</h2>
07:
08: <?php
09:
10: class Person
11: {
12:     var $name;
13:
14:     function Person($name="tepy")
15:     {
16:         $this->name = $name;
17:     }
18:     function setName( $n)
19:     {
20:         $this->name = $n;
21:     }
22:
23:     function getName()
24:     {
25:         return $this->name;
26:     }
27: }
28:
29: $person = new Person("darya");
30: print $person->getName ();
31: // outputs "Darya"
32:
33: ?>
34:
35: </body>
36: </html>
```

Person() constructor method បន្ទាត់ទី១៤គឺត្រូវបានហៅដោយស្វ័យប្រវត្តិនៅពេលដែលយើងInstantiate នូវ Person object បន្ទាត់ទី២៩ យើងក៏បានរៀបចំនូវតំលៃ default មួយជា string "tepy" ផ្តល់ទៅអោយ parameter ផងដែរ វាក៏ជាការជំនួសអោយ ខណៈដែលយើងមិនបានផ្តល់ជា argument នៅពេលដែលយើងបង្កើត Object ។

PHP 5 បានបង្ហាញនូវ syntax ថ្មីមួយដើម្បីបង្កើត constructor methods ដោយជំនួសអោយការប្រើប្រាស់នូវ function ដែលមានឈ្មោះដូច class មកជាការប្រើ special syntax ថ្មីគឺ__construct()ដូច្នេះយើងអាចធ្វើការផ្លាស់ប្តូរបន្ទាត់ទី៥នៃ ឧទាហរណ៍ខាងលើ មកប្រើនូវ syntax ថ្មីដោយធ្វើការជំនួសនូវ function Person() មកប្រើ __construct() វិញ ។

```
function __construct( $name="tepy")
{
.....
}
```

8. Limiting Access to Object Properties

PHP 4 មិនបានផ្តល់នូវការ ការពារសំរាប់ object properties នោះទេ Client code អាច get ឬ set object propertiesបានទៅតាមការគិតរបស់ពួកគេ ។មានសំណួរសួរថាតើមានបញ្ហាអ្វីទេក្នុងការអនុវត្តន៍បែបនេះ? វាមិនមានជាបញ្ហាក្នុងការប្រើប្រាស់នូវ Public accessible properties នោះទេ ដែល ជាទូទៅ វាជាការអនុវត្តន៍សំរាប់ការ access ទៅកាន់ object ដែលមានលក្ខណៈតូច ។ នៅឧទាហរណ៍ខាង ក្រោមយើងនឹងបានឃើញនូវលក្ខណៈមួយដែលកំណត់នូវព្រំដែននៃការ access ទៅកាន់ \$name propertyរបស់ Person class ។

ឧទាហរណ៍ ៤២ Class with Public Properties

```
01: <?php
02: class Person
03: {
04:     var $name;
05:     var $pid;
06:     var $personStr;
07:
08:     function Person( $name="somphy", $pid=0 )
09:     {
10:         $this->name = $name;
11:         $this->pid = $pid;
12:     }
13:
14:     function setName( $n )
15:     {
16:         $this->name = $n;
17:         $this->$personStr=$this->name." ".$this->pid;
18:     }
19:
20:     function getName ()
21:     {
22:         return $this->name;
23:     }
24: }
25: $person = new Person("sovan",5233);
26: print $person->PersonStr();
27: // outputs "sovan 5233"
```

```
28:    print "<br />";
29:    $person->name = "makara";
30:    ?>
```

PHP 5 ផ្តល់នូវវិធីសាស្ត្រផ្សេងដើម្បី declare នូវ properties របស់យើងដោយការជំនួសនូវ Var keyword មកប្រើនូវ keywords ថ្មីមួយក្នុងចំណោម keyword ទាំងបី ដែលមានលក្ខណៈ ស្រដៀង ទៅនឹង programming Java យើងនឹងបង្ហាញពីការ declare នូវ property ថ្មីនេះនៅក្នុងតារាងខាងក្រោម

PHP 5 Property Declaration Keywords	
Privacy Level	Description
public	Accessible to all. Equivalent to var.
private	Available only to the containing class.
protected	Available only to the containing class and subclasses.

ដូច្នេះយើងអាចផ្លាស់ប្តូរ properties របស់យើងដែលមាននៅក្នុងឧទាហរណ៍ខាងលើដោយប្រើprivate ជំនួស ដោយ var keyword វិញ ។

ex.

```
private $name;
```

```
private $pid;
```

ឥឡូវនេះការព្យាយាមផ្លាស់ប្តូរនូវតំលៃរបស់ \$name property នៃ Person object នៅបន្ទាត់ទី ៣០ នឹងបង្ហាញនូវ error message ដូចខាងក្រោម ។

Fatal error: Cannot access private property Person::\$name in **c:\Inetpub\wwwroot\classPrivateProperty.php** on line 30

ដូច្នេះ Client coders ត្រូវតែប្រើប្រាស់នូវ setName() method ដើម្បីធ្វើការកែប្រែទៅលើតំលៃរបស់ \$name property ។

មានពេលណាមួយអ្នកប្រហែលជាចង់អោយ child classes អាច access ទៅកាន់ property

ដែលនៅក្នុងពេលនោះអ្នកគួរតែប្រើនូវ protected keyword ព្រោះវាអនុញ្ញាតអោយអ្នក accesse វាពី Class ដែលអ្នកបាន derived ហើយយើងនឹងបានឃើញវានៅក្នុងផ្នែកមួយដែលនិយាយពី Inheritance ។

9. Limiting Access to Object Methods

លក្ខណៈសំខាន់នៃ object-oriented code គឺជា class ។ Object ត្រូវតែកំណត់នូវមុខងារនិង Public interface អោយបានច្បាស់លាស់នៅពេលអ្នកបង្កើតនូវ methods ផ្សេងៗ ។បំណែកដែលមាននៅក្នុង class គឺផ្ទុកនូវមុខងារផ្សេងៗសំរាប់តំ

ណើរការដូច្នេះអ្នកគួរតែពិចារណាទាំងនោះពីពិភពខាងក្រៅ ។ សំរាប់ឧទាហរណ៍ខាងលើយើងគួរតែបង្កើតនូវ method សំរាប់ \$personStr property ពីព្រោះវាជាការទាំងអស់របស់ \$personStr ត្រូវបានផ្ទុកនៅក្នុង setName() method ។

```
function setName( $n )
{
```

```
$this->name = $n;
$this->$personStr=$this->name." ".$this->pid;
}
```

អ្នកប្រហែលជាត្រូវការនូវ method ដើម្បី reset នូវ string របស់ \$personStr ដូច្នេះយើងនឹងបង្កើតនូវ

Method ថ្មីមួយសំរាប់ ផ្តល់តំលៃទៅអោយ \$personStr property ។

```
function setName( $n )
{
    $this->name = $n;
    $this->makePersonStr( $n, $this->code );
}
function makePersonStr( $string , $code)
{
    return $this->personStr = "$string $code";
}
```

ឥឡូវនេះយើងនៅតែមានបញ្ហានៅឡើយជាមួយនឹង method របស់យើង ពីព្រោះ client code នៅតែអាច Access នូវ makePersonStr() method ហើយវាអាចធ្វើឱ្យ ទិន្នន័យរបស់យើងមានការពិបាក គ្រប់គ្រង ។ យើងចង់អោយត្រឹមតែ object ដែលជាអ្នកបង្កើតនូវ property តែមួយគត់ដែលមានសិទ្ធិ Access មក កាន់ property នេះ នៅក្នុងជំនាន់របស់ PHP 5 យើងអាចផ្តល់នូវលក្ខណៈ privacy ទៅអោយ methods ដូចដែលយើងបានអនុវត្តន៍ជាមួយនឹង private property ពីឧទាហរណ៍មុន ។

```
private function makePersonStr($string , $code)
{
    // ...
}
```

ឥឡូវនេះ makePersonStr() function អាច access បានតែនៅក្នុង method ទាំងឡាយណាដែលស្ថិតក្នុង Person class ប៉ុណ្ណោះ ។

Public, protected ហើយនិង private វាតំណើរបានតែជាមួយនឹង PHP 5 តែប៉ុណ្ណោះ ដូច្នេះប្រសិនបើអ្នកព្យាយាមប្រើនូវ keyword ទាំងបីនេះជាមួយ PHP 4 នោះ script របស់អ្នកនឹងត្រូវបរាជ័យទាំងស្រុង ។

10. Inheritance

ដើម្បីបង្កើត class មួយដែលអាច inherits function ពី parent class យើងប្រហែលជាត្រូវកែប្រែនូវការ declare class របស់យើងបន្តិចបន្តួច ។ ឧទាហរណ៍ថ្មីខាងក្រោមនេះ គឺជាការបង្កើតនូវ Item Class ហើយនិង បង្កើតនូវ inheriting class ដែលផ្តល់ឈ្មោះថា PriceItem ។

ឧទាហរណ៍ ៤៣

Creating a Class That Inherits from Another

```
01: <html>
02: <body>
03: <h1>Creating Class That Inherits from Another</h1>
04: <?php
05:
06: class Item
07: {
```



```

08:         var $name;
09:
10:         function Item( $name="item", $code=0)
11:         {
12:             $this->name = $name;
13:             $this->code = $code;
14:         }
15:
16:         function getName()
17:         {
18:             return $this->name;
19:         }
20:     }

21:     class PriceItem extends Item
22:     {
23:         .....
24:         .....
25:     }
26:     $item = new PriceItem( "Angkor", 4545 );
27:     print $item->getName ();
28:     // outputs "Angkor"
29:     ?>
30: </body>
31: </html>
    
```

យើងបានបង្កើតនូវ class មួយទៀតដែលមានឈ្មោះថា PriceItem នៅបន្ទាត់ទី ២១ ។ គួរចំណាំ

ថា extends clause ដែលប្រើប្រាស់នៅខាងក្នុងការ declare class នៅបន្ទាត់ទី២១ នេះមានន័យថា PriceItem object inherits រាល់ function ទាំងអស់ដែលមាននៅក្នុង Item class ដូច្នេះ PriceItem Object និមួយៗគឺមានលទ្ធភាព access ទៅកាន់ getName() method ឬ \$name property ប៉ុន្តែ ក៏អាស្រ័យទៅលើការប្រើប្រាស់នូវ privacy settings ផងដែរ ។ ដោយសារតែយើងពុំបានបង្កើតនូវ Constructor method សំរាប់ PriceItem class ដូច្នេះតើ \$name property វាអាចធ្វើការផ្លាស់ផ្លូវតំលៃ ពី default "item" ទៅជា "Angkor" ដោយបញ្ជូនទៅតាម PriceItem បានយ៉ាងដូចម្តេច ? ពីព្រោះយើង

ពុំបានផ្តល់នូវ constructor នៅក្នុង PriceItem នោះទេ ដូច្នេះប្រសិនបើ class ដែល extend ពី class ដទៃ ទៀតមិនមាននូវ constructor method នោះ constructor ដែលជាប់រស់ parent class នឹងត្រូវបាន ហៅដោយស្វ័យប្រវត្តិនៅពេលដែល child object ចាប់ផ្តើមបង្កើតឡើង ។

11. Overriding the Method of a Parent Class

នៅក្នុងលក្ខណៈនៃ object-oriented ក្នុងរូបសំណាក child classes អាច override methods ពី Parents class នឹងអនុញ្ញាតិអោយ objects អាច instantiated ពី parent class ។

The Method of a Child Class Overriding That of Its Parent

ឧទាហរណ៍ ៤៤

```
01: <?php
```

```

02: class Item
03: {
04:     var $name;
05:
06:     function Item( $name="item", $code=0)
07:     {
08:         $this->name = $name;
09:         $this->code = $code;
10:     }
11:
12:     function getName()
13:     {
14:         return $this->name;
15:     }
16: }
17:
18: class PriceItem extends Item
19: {
20:     function getName()
21:     {
22:         return "(price)."$this->name;
23:     }
24: }
25:
26: $item = new PriceItem( "widget", 5442 );
27: print $item->getName();
28: // outputs "(price) Angkor"
29: ?>

```

getName() method ដែលបង្កើតនៅក្នុង PriceItem class នៅបន្ទាត់ទី ២០ ត្រូវបានហៅ ដោយប្រើប្រាស់នូវ \$name property របស់ parent class ដែលនៅចំនុចនេះយើងគួរតែធ្វើការសម្រេចចិត្ត បង្កើតនូវ \$name property ដែលមាននៅក្នុង Item class ជា private ។

```

class Item
{
    private $name;
    // ...
}

```

ការផ្លាស់ប្តូរនូវឧទាហរណ៍ខាងលើធ្វើអោយលទ្ធផលមានភាពប្រែប្រួលដូចខាងក្រោម-

លទ្ធផលដែលទទួលបានមុនពេលធ្វើការផ្លាស់ប្តូរ គឺ (price) Angkor លទ្ធផលថ្មីដែលនឹងទទួលបានគឺ (price) ឥឡូវនេះ PriceItem class មិនអាច access ទៅកាន់ \$name property ទៀតបានទេ ប៉ុន្តែប្រសិនបើ Child class ត្រូវការ access ទៅកាន់ methods ឬ property នៃ ancestor classes យើងគួរតែប្រើនូវ Protected keyword ជំនួសអោយ private វិញ ។

12. Calling an Overridden Method

មានពេលខ្លះ function ដែលមាននៅក្នុង parent class អាចមានផលប្រយោជន៍សំរាប់អ្នក ដូច្នេះ សំរាប់ Object-oriented-programming អនុញ្ញាតិអោយអ្នក refer ទៅកាន់ parent class ដោយប្រើនូវ **parent** keyword ។ ឧទាហរណ៍ខាងក្រោមនេះ getName() method ដែលស្ថិតនៅខាងក្នុង PriceItem Class នឹង call នូវ method នៅក្នុង Item class ដែលបានត្រូវ override ។

Calling an Overridden Method (PHP 5 Syntax)

ឧទាហរណ៍ ៤៥

```

01:  <?php
02:      class Item {
03:          private $name;
04:
05:          function __construct( $name="item", $code=0 ) {
06:              $this->name = $name;
07:              $this->code = $code;
08:          }
09:
10:          function getName() {
11:              return $this->name;
12:          }
13:      }
14:
15:      class PriceItem extends Item {
16:          function getName() {
17:              return "(price) ".parent::getName ();
18:          }
19:      }
20:
21:      $item = new PriceItem ("widget", 5442);
22:      print $item->getName();
23:      // outputs "(price) widget"
24:
25:  ?>

```

យើងបានធ្វើការជាមួយនឹង getName() method នៅក្នុង PriceItem class បន្ទាត់ទី ១៧ ។

PriceItem class មិនធ្វើការផ្តល់ជាមួយនឹង \$name property របស់ Item class នោះទេ ដូច្នេះនៅចំណុច នេះយើងអាច declare \$name property ជា private ដែលមិនធ្វើអោយមានផលប៉ះពាល់ជាមួយនឹង លទ្ធផលនោះទេ ហើយប្រសិនបើយើងប្រើវាជាមួយនឹង PHP 5 ទៀតនោះ វាក៏ជាការអនុវត្តន៍ដ៏ល្អ ដើម្បី Lock នូវ methods និង property របស់យើង ។

13. Working with Constructors

យើងធ្លាប់បានឃើញមកហើយថា constructor របស់ parent class នឹងត្រូវ called ដោយស្វ័យប្រវត្តិប្រសិនបើ child class មិនបានបង្កើតនូវ constructor សំរាប់ខ្លួនវានោះទេ ។ យើងនឹងបន្ថែមនូវ Constructor method ទៅអោយ PriceItem class របស់យើងជាមួយនឹងឧទាហរណ៍ខាងក្រោម ។

ឧទាហរណ៍ ៤៦ Adding a Constructor to PriceItem

```

01:  <?php
02:      class Item
03:      {
04:          private $name;
05:          function __construct( $name="item", $code=0 ) {
06:              $this->name = $name;
07:              $this->code = $code;
08:          }
09:          function getName () {
10:              return $this->name;
11:          }

```

```

12:     }
13:     class PriceItem extends Item
14:     {
15:         private $price;
16:         function __construct( $name, $code, $price ) {
17:             parent::__construct( $name, $code );
18:             $this->price = $price;
19:         }
20:
21:         function getName() {
22:             return "(price) ".parent::getName ();
23:         }
24:     }
25:
26:     $item = new PriceItem ("widget", 5442, 5.20);
27:     print $item->getName ();
28:     // outputs "(price) widget"
29:
30:     ?>

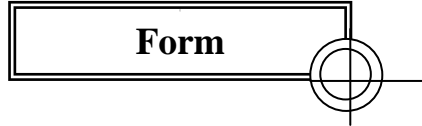
```

យើងបានបង្កើតនូវ constructor method នៅបន្ទាត់ទី១៦ ដើម្បីទទួល argument សំរាប់ \$name និង \$code ព្រមជាមួយនឹង argument ថ្មីសំរាប់ price ។ យើងបានប្រើប្រាស់ **parent** keyword ដើម្បី call constructor របស់ Item class នៅបន្ទាត់ទី២២ មុនពេលដែលយើងធ្វើការរៀបចំតំលៃអោយ

\$price property ដែលនេះបង្ហាញអោយយើងឃើញពីហេតុផលមួយសំរាប់ប្រើនូវ syntax ថ្មីរបស់ PHP5 ដើម្បីបង្កើត constructor ។

```
parent::__construct( $name, $code );
```

មេរៀនទី ៧



នៅលើ internet (world wide web) Form ផ្តល់នូវលទ្ធភាពនៃការបញ្ជូននូវព័ត៌មានពីអ្នក ប្រើប្រាស់ពី Client ទៅកាន់ Server ។ PHP ត្រូវបានរចនាឡើងដើម្បីធ្វើការជាមួយនឹងព័ត៌មានទាំងនោះនៅពេលដែល HTML forms ត្រូវបាន submit ។

1. User Input

អ្នកប្រាកដជាចង់អោយកម្មវិធីរបស់អ្នកមានសកម្មភាព ឬ ទំនាក់ទំនងជាមួយនឹងអ្នកប្រើប្រាស់ក្នុងខណៈពេលណាមួយ ។ ឧទាហរណ៍ខាងក្រោមគឺជាការបង្កើតនូវ Form មួយដើម្បីទទួលនូវព័ត៌មានពីអ្នកប្រើប្រាស់ដូចជា first-name , last-name , date of birth , email address និង password ។

```
<html>
<head>
<title>Registration form </title>
</head>
<body>
<CENTER>
<h2>Registration Form</h2>
<hr width="50%">
<form method="GET" action="register.php" >
<table>

<tr><td>First-Name</td>
<td><input type="Text" name="txtFname"></td>
</tr>

<tr><td>Last-Name</td>
<td><input type="Text" name="txtLname"></td>
</tr>

<tr><td>Date of Birth</td>
<td>
<Select name="cboday">
<option value="01">01</Option>
<option value="02">02</Option>
<option value="03">03</Option>
</select>
<Select name="cbomonth">
<option value="01">Jan</Option>
<option value="02">Feb</Option>
<option value="03">Mar</Option>
</select>
<Select name="cboyear">
<option value="1980">1980</Option>
<option value="1981">1981</Option>
<option value="1982">1982</Option>
<option value="1983">1983</Option>
</select>
</td>
</tr>
```

```

<tr><td>E-mail</td>
    <td> <input type="text" name="txtemail"></td>
</tr>

<tr><td align=right colspan=2> <input type="submit" value="Submit">
    </td>
</tr>

<tr><td>Password</td>
    <td> <input type="password" name="txtpwd" size=22></td>
</tr>

</table>
</body> </html>

```

នៅបន្ទាត់ដែលយើងបានប្រើនូវ form tag <form method="get" action="register.php"> យើងបានប្រើនូវ get method ដែលជា attribute ដំបូងនៅក្នុង form tag ។ HTTP get method ធ្វើការបញ្ជូនទិន្នន័យពី form ទៅតាម URL ដែលធ្វើអោយទិន្នន័យទាំងនោះអាចបង្ហាញនៅលើ address bar របស់ browser ជាហេតុធ្វើអោយលទ្ធផលដែលបានពី form អាចត្រូវរក្សាទុកជា bookmark ។ អ្នកប្រហែលជាត្រូវប្រើ post method វិញជាការប្រសើរជាង ប្រសិនបើអ្នកចង់ប្រើប្រាស់នូវ password ពីព្រោះ POST ធ្វើការបញ្ជូនទិន្នន័យដែលមាននៅក្នុង form ទៅតាម body នៃ HTTP request ដូច្នេះទិន្នន័យទាំងនោះមិនត្រូវបានបង្ហាញអោយឃើញជាមួយ URL ឬ ជា bookmarked នោះទេ ។

ដើម្បី processes data ពី form ដែលប្រើ get method នៅក្នុង Script អ្នកអាចប្រើជាមួយ \$_GET ឬប្រសិនបើអ្នកប្រើ នូវ post method អ្នកអាចប្រើជាមួយ \$_POST ឬ ប្រើ \$_REQUEST អ្នកអាចប្រើបានទាំងពីរ method ។

ឧទាហរណ៍ ការប្រើប្រាស់ get method នៅក្នុង script

register.php

```

<html>
<body>
<head>
<title>Personal Information</title>
</head>
<center>

<h2>Personal Information</h2>
<hr width=50%>

<?php

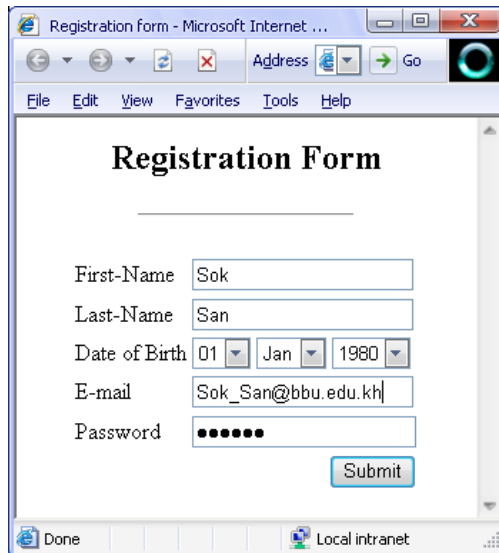
$name = $_GET['txtFname'];
$lname = $_GET['txtLname'];
$day = $_GET['cboDay'];
$month = $_GET['cboMonth'];
$year = $_GET['cboYear'];
$dob = $day."-$month."-$year";

print "<table>";

print "<tr>";

    print "<td><b>First-Name :</b></td>";
    print "<td> $fname </td>";

```



```
print "</tr>";

print "<tr>";

    print "<td><b>Last-Name :</b></td>";
    print "<td> $lname </td>";

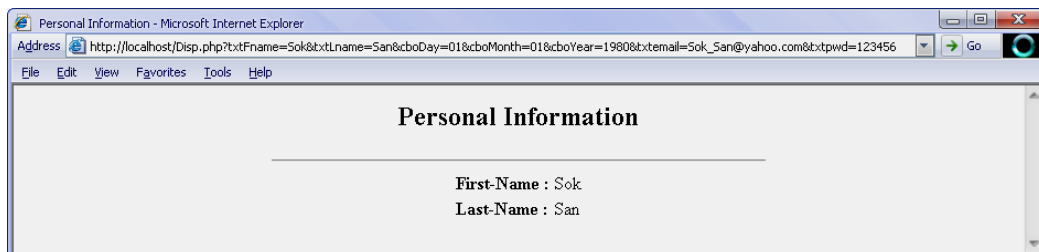
print "</tr>";

print "</table>";

?>
```

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

ខាងក្រោមគឺជាលទ្ធផលដែលទទួលបានបន្ទាប់ពី form ត្រូវបាន submit ដោយប្រើ get method ។ សូមពិនិត្យមើលនូវ address bar របស់ browser ដែលព័ត៌មានបញ្ជូនមកនោះ ត្រូវបានភ្ជាប់មកជាមួយនឹង url



ឧទាហរណ៍ ៤៩ ការប្រើប្រាស់ post method នៅក្នុង script

register.php

```
<html>
<body>

<?php

$fname = $_POST['txtFname'];
$lname = $_POST['txtLname'];

print "<table>";

print "<tr>";

    print "<td><b>First-Name :</b></td>";
    print "<td> $fname </td>";

print "</tr>";

print "<tr>";

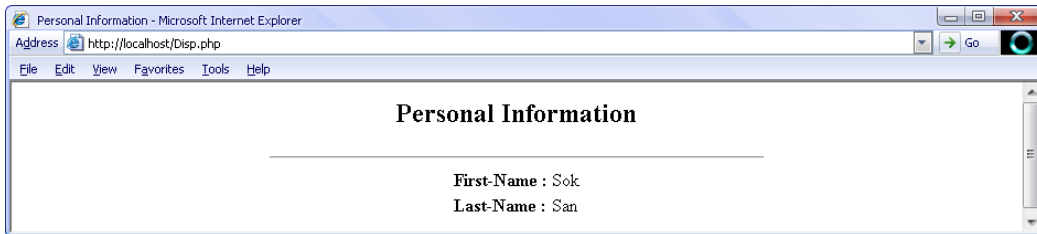
    print "<td><b>Last-Name :</b></td>";
    print "<td> $lname </td>";

print "</tr>";

print "</table>";
```

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

អ្នកនឹងបានឃើញពីភាពខុសគ្នារវាងការប្រើប្រាស់នូវ post method និង get method នៅលើ Address bar របស់ browser នីមួយៗ ។ ខាងក្រោមគឺជាលទ្ធផលដែលទទួលបានបន្ទាប់ ពី form ត្រូវបាន Submit ដោយប្រើប្រាស់នូវ post method ។



2. Accessing Form Input with User-Defined Arrays

ឧទាហរណ៍ ៤៩ ផ្តល់លទ្ធភាពអោយអ្នកទទួលបាននូវព័ត៌មានពី HTML element ដែលបាន Submit នូវតំលៃមួយសំរាប់តែ element មួយប៉ុណ្ណោះដែលនេះពុំទាន់មានលក្ខណៈគ្រប់គ្រាន់នៅឡើយនោះទេប្រសិន បើអ្នកចង់ធ្វើការជាមួយនឹង multiple select ។

ឧទាហរណ៍ ៥០

```
01: <html>
02: <head>
03: <title>HTML Form with a 'select' Element</title>
04: </head>
05:
06: <body>
07: <center>
08: <h2>Multiple Select</h2>
09:
10: <hr width="50%">
11: <form action="multiple.php" method="POST">
12: <select name="products[]" multiple="multiple">
13: <option>Cocacola </option>
14: <option>Fanta </option>
15: <option>Sprite </option>
16: <option>Merinda </option>
17: <option>Pepsi </option>
18: </select><br><br>
19: <input type="submit" value="submit">
20:
21: </body>
22: </html>
```

យើងនឹងបង្កើត script ដើម្បីស្វែងរក input ពី "products[]" form element ដែលបង្កើតនៅបន្ទាត់ទី 12 ដោយផ្តល់ឈ្មោះជា array indexed ហៅថា products ។ products[] គឺជា select element ដែលផ្តល់អោយនូវជំរើសសំរាប់អ្នកប្រើប្រាស់ដោយការប្រើ option element នៅបន្ទាត់ទី ១៣ ដល់ បន្ទាត់ទី ១៧ យើងនឹងបង្ហាញពីលទ្ធផលដែលអ្នកប្រើប្រាស់បានជ្រើសរើសពី form ខាងលើ ដែលបង្កើតជា array នៅក្នុង ឧទាហរណ៍ ៥១

```
01: <html>
02: <body>
03: <head>
```



```

04: <title>Reading Input from the form multiple select</title>
05: </head>
06:
07: <?php
08:
09: if ( is_array( $_POST['products'] ) )
10: {
08:     echo "<b>Your products choice are :</b><br/>";
10:     foreach($_POST['products'] as $val)
11:     {
12:         print "$val<br>";
13:     }
14: }
15:
16: ?>
17:
18: </body>
19: </html>

```

នៅបន្ទាត់ទី ០៩ នៃ ឧទាហរណ៍ ៥១ យើងធ្វើការត្រួតពិនិត្យទៅលើ \$_POST['products'] Element ប្រសិនបើ element នេះជា array យើងនឹង loop រាល់ធាតុនីមួយៗរបស់វានៅបន្ទាត់ទី ១០ ដើម្បី បង្ហាញជាលទ្ធផលទៅកាន់ Browser ។ អ្នកក៏អាចអនុញ្ញាតិអោយអ្នកប្រើប្រាស់ ជ្រើសរើសនូវជំរើសច្រើនដោយប្រើ check boxes ដែលត្រូវផ្តល់នូវឈ្មោះដូចគ្នាសំរាប់ element នីមួយៗ ហើយភ្ជាប់ជាមួយ empty brakets ។ PHP នឹង compiles នូវអ្វីដែលអ្នកប្រើប្រាស់បាន select ទៅជា array ។ យើងនឹងធ្វើការផ្លាស់ប្តូរពីការប្រើ select element នៅឧទាហរណ៍ខាងលើ មកប្រើ check boxes វិញដូចមានក្នុងឧទាហរណ៍ ៥២ ។

ឧទាហរណ៍ ៥២

```

<html>
  <head>
  <title>HTML Form with a 'select' Element</title>
  </head>

  <body>
  <center>
  <h2>Multiple Select</h2>

  <hr width="50%">
  <form action="multiple.php" method="POST">

  <input type="checkbox" name="products[]" value="Cocacola" />Cocacola
  <input type="checkbox" name="products[]" value="Fanta" />Fanta
  <input type="checkbox" name="products[]" value="Sprite" />Sprite
  <input type="checkbox" name="products[]" value="Merinda" />Merinda
  <input type="checkbox" name="products[]" value="Pepsi" />Pepsi

  <br><br>
  <input type="submit" value="submit">

  </body>
</html>

```



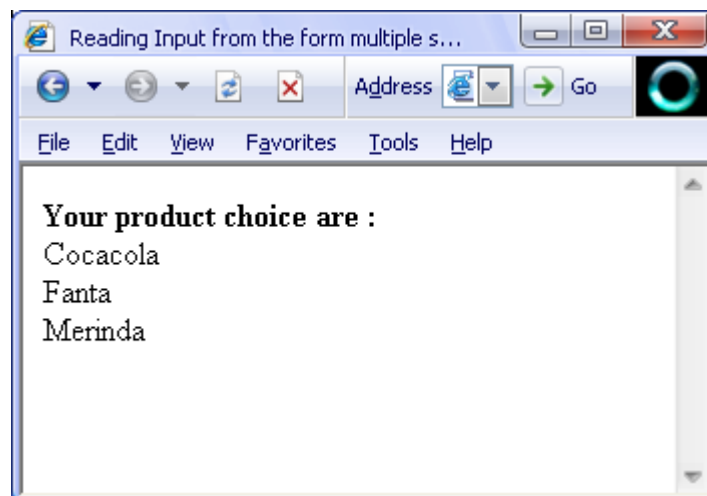
multiple.php

ឧទាហរណ៍ ៥២

```

01: <html>
02: <body>
03: <head>
04: <title>Reading Input from the form multiple select</title>
05: </head>
06:
07: <?php
08:
09: if ( is_array( $_POST['products'] ) )
10: {
08:     echo "<b>Your products choice are :</b><br/>";
10:     foreach($_POST['products'] as $val)
11:     {
12:         print "$val<br>";
13:     }
14: }
15:
16: ?>
17:
18: </body>
19: </html>

```



មេរៀនទី ៨



1. What is MySQL ?

MySQL គឺជាប្រភេទ Open Source Sql databases management system ដែលមានប្រជាប្រិយបំផុតសំរាប់ ការប្រើប្រាស់ ដោយត្រូវបានអភិវឌ្ឍន៍និងគាំទ្រដោយ MySQL AB ។ MySQL AB គឺជាក្រុមហ៊ុនដែលបង្កើតឡើង ដោយក្រុមអ្នកអភិវឌ្ឍន៍ MySQL ។

1.1 MySQL is a relational database management system.

A relation databases គឺជាក្រុមទុកនូវទិន្នន័យក្នុង table ផ្សេងពីគ្នាដែលមានលក្ខណៈប្រសើរជាងការរក្សាទុកនូវទិន្នន័យទាំង អស់នៅក្នុង storeroom ដ៏ធំមួយ ហើយវិធីនេះអាចជួយបង្កើនល្បឿន និង មានភាពងាយស្រួលប្រើកាន់តែប្រសើរឡើងជាមួយ នឹងទិន្នន័យ ។ SQL គឺជាបំណែកមួយនៃ MySQL ដែល ប្រើ ជាពាក្យពេញថា “Structured Query Language” SQL គឺជាភាសាដែលប្រើប្រាស់ជា standard ដើម្បី access ទៅកាន់ databases ហើយត្រូវបានកំណត់ដោយ ANSI/ISO SQL Standard ។ SQL standard ត្រូវបានវិវត្តន៍ឡើងនៅក្នុងអំឡុងឆ្នាំ ១៩៨៦ ដែលបង្កើតបានជាជំនាន់របស់ SQL-92 ដោយសំដៅទៅលើ SQL standard ដែលធ្វើការដាក់ចេញនូវជំនាន់របស់ខ្លួននៅក្នុងអំឡុងឆ្នាំ ១៩៩២ និង បន្ទាប់មកគឺ SQL-1999 និងក្រោយមកទៀតគឺ SQL-2003 ។

1.2 MySQL software is Open Source.

Open Source គឺមានន័យថាអ្នកប្រើប្រាស់មានលទ្ធភាពកែប្រែនូវ software នោះបាន ហើយ អាចទាញយកនូវ Software ទាំងនោះនៅលើ Internet និងអាចប្រើប្រាស់វា ដោយមិនមានការបង់ប្រាក់ ទៅលើសេវាកម្មអ្វីទាំងអស់ ។ MySQL software ប្រើប្រាស់ GPL(General Public License) ដើម្បីផ្តល់អោយអ្នកនូវអ្វីដែលត្រូវការ និង អ្វីដែលអ្នកមិនត្រូវការចំពោះ software នេះ ។ គឺមានន័យថា ប្រសិនបើអ្នកមិនមានអារម្មណ៍ទុកចិត្ត ជាមួយនឹង GPL ឬ អ្នកចង់បង្កប់នូវកូដទៅក្នុង MySQL ដើម្បីធ្វើជា Commercial application អ្នកអាចទិញនូវច្បាប់អនុញ្ញាតិអោយធ្វើពាណិជ្ជកម្មពីក្រុមហ៊ុន MySQL ដែល ព័ត៌មានលំអិតអ្នកអាចប្រើប្រាស់នូវគេហទំព័រ (<http://www.mysql.com/company/legal/licensing/>) ។

2. SQL commands in mysql

ផ្នែកខាងក្រោមនឹងបង្ហាញពីការប្រើប្រាស់ SQL commands នៅក្នុង MySQL context ។ គ្រប់ commands ទាំងអស់សុទ្ធតែបានទទួលស្គាល់ដោយ MySQL system ក្នុងអំឡុងពេលដែល design SQL databases environment ។ command របស់ MySQL នីមួយៗត្រូវតែបញ្ចប់ជាមួយនឹងសញ្ញា Semicolon (;) ។

show databases គឺជា command ដំបូងដែលប្រើដើម្បីបង្ហាញនូវឈ្មោះ database ទាំងអស់ដែលមាននៅក្នុង MySQL ក្នុងនោះមាន database ចំនួនពីរភ្ជាប់មកស្រាប់ជាមួយ MySQL package ។

```
mysql> show databases;
+-----+
| Database |
+-----+
| mysql    |
| test     |
```

```
+-----+
2 rows in set (0.00 sec)
```

Database ដែលមានឈ្មោះថា mysql ផ្ទុកនូវ MySQL settings ហើយនិង users ផ្សេងៗ ។សំរាប់ជាឧទាហរណ៍ យើងនឹងបង្កើត database មួយឈ្មោះថា BbuDb ។

create database databaseName ;

syntax : mysql>create database BbuDb ;

យើងអាចពិនិត្យមើល database ដែលបានបង្កើតខាងលើដោយប្រើប្រាស់ **show database** command ។

```
mysql> show databases;
```

```
+-----+
| Database |
+-----+
| mysql    |
| BbuDb   |
| test     |
+-----+
3 rows in set (0.00 sec)
```

ដើម្បីលុបបំបាត់ database ណាមួយនោះអ្នកអាចប្រើ SQL command : **drop database databaseName** អ្នកគួរប្រុងប្រយ័ត្នផងដែរ

ចំពោះការប្រើប្រាស់ command នេះ ពីព្រោះវានឹងធ្វើការលុប database ដោយមិនមានការសាកសួរ ឬ ធ្វើការបញ្ជាក់ពី

អ្នកម្តងទៀតឡើយ ។SQL syntax ដែលប្រើដើម្បីបង្កើត table យើងនឹងបង្ហាញជាមួយឧទាហរណ៍ខាងក្រោម ប៉ុន្តែមុនពេលដែលប្រើ command នេះអ្នកត្រូវតែកំណត់នូវ database មួយអោយជាក់លាក់ជាមុនសិន ដោយប្រើ command use:

mysql>use **BbuDb** ; ។ យើងនឹងប្រើ command ដើម្បីបង្កើត table ដែលមានattributes ដូចជា id , username , password , name និង email ជាមួយនឹងឧទាហរណ៍ខាងក្រោម ។The general form of the syntax is:

create table tableName (col1Name datatype otherSettings, etc)

```
mysql> create table tblpeople
      ( id int(5) not null auto_increment ,
        username varchar(20) binary not null,
        password varchar(20) binary not null,
        name      varchar(20)          not null,
        email     varchar(30)          not null,
        primary key(id)
      );
```

រាល់ attribute នីមួយៗសុទ្ធតែត្រូវបានកំណត់នូវចំនួនតួអក្សរដែលត្រូវរក្សាទុកនៅក្នុង table ហើយ fieldនីមួយៗនៃ table សុទ្ធតែទាមទារអោយមាននូវតំលៃជានិច្ច ។ ចំណែក id attribute តំលៃរបស់វានឹងត្រូវកើនឡើងដោយស្វ័យប្រវត្តិនៅពេល

ដែលទិន្នន័យត្រូវបានបញ្ចូលទៅកាន់ tblpeople មួយលើកៗ ។varchar គឺជាតំលៃដែលប្រើជាតួអក្សរ ឬជាតួលេខ ចំណែក binary គេប្រើដើម្បីធ្វើអោយតំលៃ ទៅជា case-sensitive រីឯ name ហើយ និង email attribute គឺមិនត្រូវបានប្រើ binary នោះទេ ដូច្នេះវាមិនមានលក្ខណៈ case-sensitive ឡើយ ។ id គឺត្រូវបានបង្កើតជា primary key សំរាប់ table ដើម្បី

identify row នីមួយៗនៃ table ។

```
mysql> desc tblpeople ;
```

Field	Type	Null	Key	Default	Extra
id	int(5)		PRI	NULL	auto_increment
username	varchar(20)				
password	varchar(20)				
name	varchar(20)				
email	varchar(30)				

5 rows in set (0.01 sec)

បន្ទាប់ពីមកទៀតអ្នកអាចប្រើប្រាស់ **insert** command ដើម្បីបញ្ចូលទិន្នន័យទៅក្នុង table ។

ឧទាហរណ៍:

```
mysql>insert into tblpeople values (null, 'muth', 'muth07 ', 'monen', 'muth_monen@hotmail.com');
```

```
mysql>insert into tblpeople values(null, 'lou', ' army ', ' lou cy ', ' lou_cy@gmail.com ');
```

នៅពេលដែល id attribute ជា auto_increment យើងប្រើ null សំរាប់ជា data entry ដែល null Value

មិនមានលក្ខណៈដូចនឹង blank (" ") នោះទេ ។ id attribute ចាប់ផ្តើមដោយតំលៃជា integer 1 ហើយវានឹងធ្វើការបង្កើនតំលៃ ១

នៃនៅពេលដែលមានការបញ្ចូលទិន្នន័យទៅកាន់ row ថ្មីម្តងៗ។ អ្នកអាចពិនិត្យមើលព័ត៌មាន ទាំងអស់ដែលមាននៅក្នុង

tableបន្ទាប់ពីបានបញ្ចូលដោយប្រើប្រាស់ SQL command mysql>**select * from tblpeople ;**

id	username	password	name	email
1	muth	muth07	monen	muth_monen@hotmail.com
2	lou	army	lou cy	lou_cy@gmail.com

2 rows in set (0.00 sec)

ដើម្បីលុបនូវ table ណាមួយចេញពី database អ្នកអាចប្រើ command :mysql> **drop table tableName**សំរាប់ command

ដែលប្រើដើម្បីលុប row ពី table ទាមទារអោយអ្នកផ្តល់នូវឈ្មោះនៃ table ហើយនិងwhere condition ដែលបញ្ជាក់ពី row(s)

ដែលនឹងត្រូវលុប ។

delete from tableName where Where-condition

សូមប្រុងប្រយ័ត្នប្រសិនបើអ្នកមិនប្រើ where-condition នោះទេ គ្រប់ rows ទាំងអស់ដែលមាននៅក្នុងtable

នឹងត្រូវលុបចោលទាំងអស់។ នៅក្នុងឧទាហរណ៍ខាងក្រោមយើងនឹងប្រើតំលៃរបស់ attribute name និង username ដើម្បីលុប row

ចេញពី table tblpeople ។

delete from tblpeople where name=' monen ' and username = ' muth ' ;

UPDATE command ត្រូវបានប្រើដើម្បីកែប្រែព័ត៌មានដែលមាននៅក្នុង table អ្នកអាច select យក

ព័ត៌មានណាមួយមកកែប្រែព័ត៌មានដោយមិនមានការបាត់បង់នូវ record ឬ row ដើមទាំងស្រុងឡើយ ។

The syntax is:

UPDATE table-name SET field1='val1', field2='val2', field3='val3'.....

WHERE condition;

ex.

update tblpeople set password='maco' where username='muth' ;

3. PHP/MySQL Functions

3.1 Connecting to MySQL

មុនពេលដែលចាប់ផ្តើមធ្វើការជាមួយនឹង database របស់អ្នក អ្នកត្រូវតែ connect ទៅកាន់ Server ជាមុនសិន ។
PHP ផ្តល់នូវ mysql_connect () function ដើម្បីអនុវត្តន៍នូវភារកិច្ចនេះ ។

បំណែកកូដខាងក្រោមនឹងបង្ហាញពីការប្រើប្រាស់ mysql_connect() ភ្ជាប់ទៅកាន់ MySQL database Server ។

```
$link = mysql_connect( "localhost", "root", "123" );
if ( ! $link )
{
    die( "Couldn't connect to MySQL" );
}
```

3.2 Selecting a Database

បន្ទាប់ពីបានបង្កើត connection ដើម្បីភ្ជាប់ទៅកាន់ MySQL រួចរាល់ហើយ អ្នកត្រូវតែជ្រើសរើស Database ណាមួយដើម្បីធ្វើការជាមួយវា ដោយអ្នកអាចប្រើប្រាស់នូវ mysql_select_db() function ។

បំណែកកូដខាងក្រោមយើងនឹង select យក database មួយដែលមានឈ្មោះថា **BbuDb** ។

```
$database = "BbuDb";

mysql_select_db( $database ) or die ( "Couldn't open $database );
```

3.3 Adding Data to a Table

ឥឡូវនេះយើងមានលទ្ធភាពគ្រប់គ្រាន់ដើម្បី access ទៅកាន់ database ហើយយើងអាចបញ្ចូលនូវព័ត៌មានទៅកាន់ table នៃ database ។ សំរាប់ជាឧទាហរណ៍យើងនឹងប្រើប្រាស់ table ដែលបានបង្កើតនៅក្នុង database BbuDb ដែលមានឈ្មោះថា tblpeople ។

Adding a Row to a Table

```
01: <html>
02: <head>
03: <title>Listing 13.2 Adding a Row to a Database</title>
04: </head>
05: <body>
06: <div>
07: <?php
08: $user = "root";
09: $pass = " ";
10: $db = "BbuDb";
11: $link = @mysql_connect( "localhost", $user, $pass );
```

```

12:  if ( ! $link )
13:  {
14:  die( "Couldn't connect to MySQL: ".mysql_error() );
15:  }
16:  print "<h2>Successfully connected to server</h2>\n\n";
17:  @mysql_select_db( $db )
18:  or die ( "Couldn't open $db: ".mysql_error() );
19:  print "Successfully selected database \"$db\"<br />\n";
20:
21:  $query = "insert into tblpeople(username , password , name , email )
22:  values( ' ratana07 ', ' 1234 ', ' som ratana ', ' ratana@example.com ' )";
23:  mysql_query( $query, $link )
24:  or die ( "INSERT error: ".mysql_error() );
25:
26:  mysql_close( $link );
27:  ?>
28:  </div>
29:  </body>
30:  </html>

```

សំរាប់ឧទាហរណ៍នេះយើងមិនបានបញ្ចូលតំលៃទៅអោយ id column នោះទេ ព្រោះ field នេះជាប្រភេទ auto_increments ។ វាជាការពិត រាល់ពេលដែលយើង reload script នៃឧទាហរណ៍ខាងលើទិន្នន័យដែលនឹងត្រូវបញ្ចូលទៅកាន់ row ថ្មីមួយទៀតនៃ table ។

3.4 Adding User Input to a Database

Registration.html

```

<html>
<head>
<title>Adding user input to a database</title>
</head>
<body>
<CENTER>
<h2> Adding user input to a databases </h2>
<hr width=550>
    <form action="insert.php" method="Post">
        <table>
        <tr>
            <td>Name :</td>
            <td><input type="TextBox" size="18" name="txtname"/></td>
        </tr>
        <tr>
            <td>UserName :</td>
            <td><input type="TextBox" size="18" name="txtusr"/></td>
        </tr>
        <tr>
            <td>Password :</td>
            <td><input type="Password" name="txtpwd"/></td>
        </tr>
    </form>

```

```

        <tr>
            <td>Email :</td>
            <td><input type="TextBox" size="18" name="txtemail"/></td>
        </tr>

        <tr>
            <td align="right" colspan="2">
                <input type="Submit" value="Submit">
            </td>
        </tr>

    </table>
</body>
</html>

```

Inser.php

```

01  <html>
02  <head>
03  <title>Listing 13.3 Adding user input to a database</title>
04  </head>
05  <body>
06
07  <?php
08
09  $username = $_REQUEST['txtusr'] ;
10  $name = $_REQUEST['txtname'] ;
11  $pwd = $_REQUEST['txtpwd'] ;
12  $email = $_REQUEST['txtemail'] ;
13  if ( ! empty($username) && ! empty( $name ) && ! empty($pwd )&&
14      !empty($email) ) // check user input here!
15  {
16      $db = "BbuDb";
17      $link = @mysql_connect( "localhost", $user, $pass );
18      if ( ! $link )
19      {
20          die( "Couldn't connect to MySQL: ".mysql_error() );
21      }
22      @mysql_select_db( $db ) or die ( "Couldn't open $db: ".mysql_error() );
23
24      $query = " insert into tblpeople(username , password , name , email )
25      values( '" . $username. "', '" . $name. "', '" . $pwd . "', '" . $email. "' )";
26      mysql_query( $query, $link ) or die ( "INSERT error: ".mysql_error() );
27
28      mysql_close( $link );
29  ?>
30  </body>
31  </html>

```


យើងបាន select នូវ database ដែលផ្ទុក table ឈ្មោះថា tblperson នៅបន្ទាត់ទី 22 ហើយនឹងបង្កើត SQL query ដើម្បីបញ្ជូនតំលៃដែលបានមកពី user-submitted ដោយបញ្ជូនព័ត៌មានទៅអោយmysql_query() នៅបន្ទាត់ទី ២៦ ។

3.5 Accessing a Resultset

អ្នកអាចមានភាពងាយស្រួលដើម្បីទទួលបាន array ពីជួរ និងមួយៗនៃ fields ដោយការប្រើប្រាស់ ជាមួយនឹង mysql_fetch_row() function ។ function នេះត្រូវអោយអ្នកផ្តល់ result resource មួយ ហើយ វានឹង return វិញនូវ row និងមួយៗដែលមាននៅក្នុង field ។

Listing All Rows and Fields in a Table

```
01: <html>
02: <head>
03: <title> Selecting Data</title>
04: </head>
05: <body>
06: <?php
07: $user = "root";
08: $pass = "1235";
09: $db = "BbuDb";
10: $link = mysql_connect( "localhost", $user, $pass );
11: if ( ! $link )
12: {
13: die( "Couldn't connect to MySQL: ".mysql_error() );
14: }
15: mysql_select_db( $db, $link )
16: or die ( "Couldn't open $db: ".mysql_error() );
17:
18: $result = mysql_query( " SELECT * FROM tblperson " );
19: $num_rows = mysql_num_rows( $result );
20:
21: print "<table border='1'\n";
22: while ( $a_row = mysql_fetch_row( $result ) )
23: {
24: print "<tr>\n";
25: foreach ( $a_row as $field )
26: {
27: print "\t<td>".$field."</td>\n";
28: }
29: print "</tr>\n";
30: }
31: print "</table>\n";
32: mysql_close( $link );
33: ?>
34: </body>
35: </html>
```

បន្ទាប់ពីយើងបាន connected ទៅកាន់ database server ហើយនឹង selected database រួចរាល់យើង បានប្រើ mysql_query() នៅបន្ទាត់ទី ១៨ ដើម្បីបញ្ជូន select statement ទៅកាន់ database serverបន្ទាប់មកយើងរក្សានូវ return result resource នៅក្នុង variable មួយឈ្មោះថា \$result ហើយប្រើវាដើម្បី ទទួលយកចំនួន row ដែលមាននៅក្នុង table ។

សំរាប់ test expression នៃ while statement នៅបន្ទាត់ទី ២២ យើងបានផ្តល់នូវលទ្ធផលដែលបានពី mysql_fetch_row() ទៅអោយ variable \$a_row ។ បិតក្នុងតំណើរការរបស់ while statement

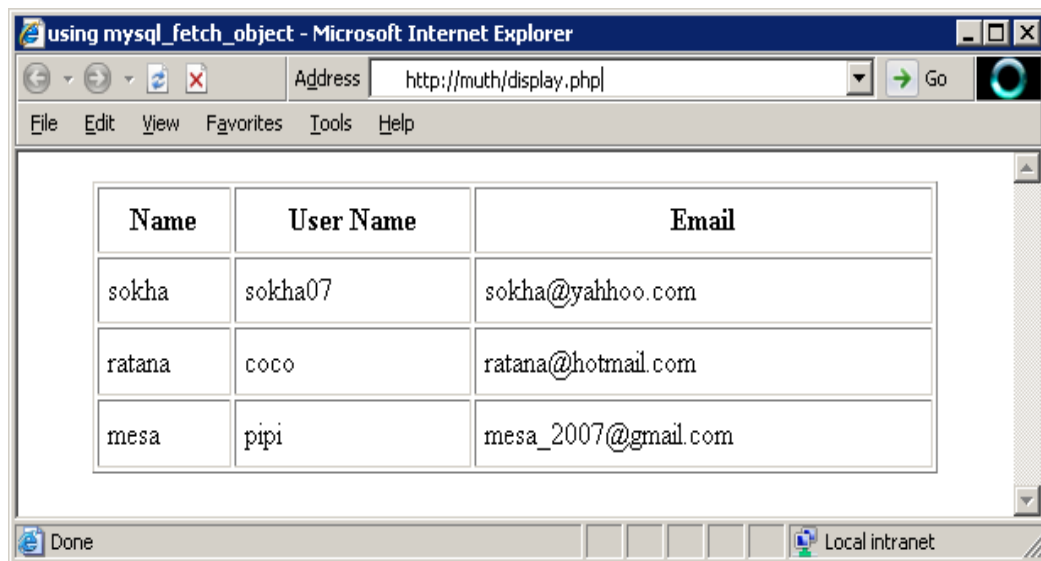
យើងបាន loop នូវ row array ដែលផ្ទុកក្នុង \$a_row នៅបន្ទាត់ទី ២៥ ហើយធ្វើការបច្ចេកទេសវិលវិលផលពីធាតុនីមួយៗ បញ្ចូលទៅក្នុង table cell រួចត្រូវបានបង្ហាញដោយ web browser ។ អ្នកក៏អាច accessFields ដោយប្រើប្រាស់ឈ្មោះរបស់វាទៅតាមវិធីសាស្ត្រពីរយ៉ាងគឺ : mysql_fetch_row() វា return ជា Numerics array និង mysql_fetch_row() វា return ជា associative array ជាមួយនឹងឈ្មោះរបស់ Fields ដោយប្រើជា keys របស់ array ។ បំណែកក្នុងខាងក្រោមយើងនឹងជំនួសបន្ទាត់ទី ២៣-៣៣ ដោយការប្រើប្រាស់ mysql_fetch_array() ជំនួសវិញ ។

```
print "<table border='1'\>\n";
while ( $a_row = mysql_fetch_array( $result ) )
{
    print "<tr>\n";
    print "<td>".$a_row['name']."</td>";
    print "<td>".$a_row['username']."</td>";
    print "<td>".$a_row['email']."</td>";
    print "</tr>\n";
}
print "</table>\n";
```

ម្យ៉ាងវិញទៀតអ្នកអាចចាប់យក fields ពីជួរណាមួយមកធ្វើជា properties នៃ object ដោយ ប្រើ mysql_fetch_object() នោះឈ្មោះ field ទាំងអស់នឹងក្លាយជាឈ្មោះរបស់ properties ។

```
print "<table border='1'\>\n";
while ( $a_row = mysql_fetch_object( $result ) )
{
    print "<tr>\n";
    print "<td>".$a_row->name."</td>";
    print "<td>".$a_row->username."</td>";
    print "<td>".$a_row->email."</td>";
    print "</tr>\n";
}
print "</table>\n";
```

ចំពោះការប្រើប្រាស់ mysql_fetch_array() និង mysql_fetch_object() ទាំងពីរនេះសុទ្ធតែផ្តល់នូវភាពងាយស្រួលសំរាប់ចាប់យក ព័ត៌មានពី row ណាមួយ ។ ទោះបីជាយ៉ាងណាក៏ដោយជាធម្មតា ភាគ ច្រើនគេនិយមប្រើប្រាស់ mysql_fetch_array() ។



using mysql_fetch_object - Microsoft Internet Explorer

Address <http://muth/display.php> Go

File Edit View Favorites Tools Help

Name	User Name	Email
sokha	sokha07	sokha@yahoo.com
ratana	coco	ratana@hotmail.com
mesa	pipi	mesa_2007@gmail.com

Done Local intranet

មេរៀនទី ៩

Saving State with Cookies and Session

1. Cookie

cookie គឺជាទិន្នន័យដែលរក្សាទុកដោយ user's browser ដែលប្រព្រឹត្តទៅបានដោយការ request ពី server ឬ script ។ វាអាចរក្សាទុកបានត្រឹមតែ 20 cookies និង ទំហំផ្ទុកទិន្នន័យបានត្រឹម 4 kilobyte ប៉ុណ្ណោះ ហើយរាល់ cookie នីមួយៗត្រូវតែមាននូវ ឈ្មោះ តំលៃ និង កាលបរិច្ឆេទឈប់ប្រើប្រាស់ ។

បន្ទាប់ពី cookie ត្រូវបាន set មានតែ ម៉ាស៊ីនកុំព្យូទ័រដែល set នូវ cookie នោះប៉ុណ្ណោះទើបអាចមានសិទ្ធិ read នូវទិន្នន័យដែលមាននៅខាងក្នុងបាន ប៉ុន្តែក៏អាស្រ័យទៅលើ user's privacy ដែល user អាចធ្វើការ configure នៅលើ browser របស់ពួកគេ ដើម្បីអោយ cookie ទាំងអស់ មានលទ្ធភាព set ឬ ក៏បដិសេធមិនអោយមានការ request cookie ។ cookie អាចជាវិធីសាស្ត្រដ៏ប្រសើរសំរាប់រក្សាទុកនូវព័ត៌មានខ្លីពីអ្នកប្រើប្រាស់ ពី page មួយទៅ page មួយទៀត ។

Cookies របស់ PHP បញ្ជូនទៅកាន់ web server ដោយប្រើប្រាស់ setcookie() function ។ ប្រសិនបើ cookie បាន set នូវ time-out browser នឹងធ្វើការចងចាំនូវ cookie ទោះបីជាអ្នក restart ម៉ាស៊ីនកុំដោយ ប៉ុន្តែប្រសិនបើអ្នកមិនបាន set នូវ time-out សំរាប់ cookie នោះទេ browser នឹងបំបាត់ចោលនូវ cookie នោះភ្លាមនៅពេលដែលអ្នក close browser ។ សំរាប់ជាឧទាហរណ៍ យើងនឹង set cookie នៅពេលដែល user បញ្ចូលនូវ username និង password បានត្រឹមត្រូវជាមួយនឹង login form ។

Auth.php

```
<?php
class Auth
{
    function Auth()
    {
        mysql_connect('localhost', 'root');
        mysql_select_db('my_own_bookshop');
    }
    public function authUser($user , $password)
    {
        $q = ' SELECT username , password FROM tblpeople WHERE          username="'. $user. "' AND
        password = '". $password. "' ";
        $r = mysql_query($q);
        if (mysql_num_rows($r) == 1)
        { return TRUE; }
        else { return FALSE; }
    }
}
}??>
```

login.php

```
<?
ob_start();
?>

<html>
<head><title>Login</title></head>
<body>

<?php
require("auth.php");
$auth = new Auth();
```

```

if (isset($_POST['login']) && ($_POST['login'] == 'Log in') &&
    $auth->authUser($_POST['txtuid'] , $_POST['txtpwd'] ) )
{
    $uid = $_POST['txtuid'];
    /* User successfully logged in, setting cookie */
    setcookie('uid', $uid, time() + 14400, '/');
    header("Location: http://localhost/index.php");
    exit();
}
else {
?>
    <h1>Log-in</h1>
    <form method="post" action="login.php">
    <table>
    <tr><td>User name :</td>
    <td><input type='text' size='18' name='txtuid' /></td></tr>
    <tr><td>Password :</td>
    <td><input type='password' name='txtpwd' /></td></tr>
    <tr><td colspan='2' align='right'>
    <input type='submit' name='login' value='Log in' /></td>
    </tr>
    </table>
    </form>
<?php
}
?>
</body>
</html>

```

ចំពោះ superglobal ដែលប្រើដើម្បី read cookies គឺ \$_COOKIE ដែលយើងនឹងអនុវត្តន៍វាជា មួយនឹងឧទាហរណ៍នៅក្នុង

file មួយឈ្មោះថា index.php ។

index.php

```

<?php
if (isset($_COOKIE['uid']) && $_COOKIE['uid'])
{
?>
<html>
<head><title>Index page</title></head>
<body>
    Logged in with UID: <?php echo $_COOKIE['uid']; ?><br />
    <a href='logout.php'>Log out</a>.
</body>
</html>

<?php
}
else {
/* If no UID is in the cookie, we redirect to the login .page */
header('Location: http://localhost/login.php');
}
?>

```

សំរាប់ការប្រើប្រាស់ user id របស់យើងនេះ គឺវាមានភាពចាំបាច់ណាស់ ដូចជាការចងចាំសំរាប់Authentication data (ដូចដែលយើងបានអនុវត្តនៅក្នុង script របស់យើង) ។ប៉ុន្តែវាមិនទាន់ជាគំនិតដែលត្រឹមត្រូវនោះទេ ពីព្រោះអ្នកប្រើប្រាស់អាចកែប្រែ cookies ទាំងនោះ ឬ install តំលៃផ្សេងៗបានទៅតាមអំពើចិត្ត ។

មានតំណោះយ៉ាងប្រសើរមួយសំរាប់បញ្ហានេះគឺការប្រើប្រាស់ PHP Sessions ដែលយើងនឹងអនុវត្តនៅក្នុងឧទាហរណ៍ក្រោយមួយទៀត ។ ដើម្បី លប់ cookie អ្នកគ្រាន់តែប្រើប្រាស់ parameters ដូចដែលអ្នក set cookie ប៉ុន្តែអ្នកមិនចាំបាច់ផ្តល់នូវ value សំរាប់ cookie នោះទេ ហើយ កាលបរិច្ឆេទលប់ប្រើប្រាស់អ្នកត្រូវផ្តល់អោយជាកាលបរិច្ឆេទនៃអតីតកាល ។ នៅក្នុង logout page យើងនឹងលប់ cookie ដោយប្រើវិធី ដូចខាងក្រោម ។

logout.php

```
<?php
setcookie('uid', '', time() - 86400, '/');
header('Location: http://localhost/login.php');
?>
```

time()-86400 គឺជាកាលបរិច្ឆេទដែលកន្លងហួសទៅមួយថ្ងៃ នឹងជាកាលបរិច្ឆេទនៃអតីតកាល

ដើម្បីប្រាប់អោយ browser បំបាត់ចោលនូវ cookie data នោះ ។

2. SESSIONS

PHP session អនុញ្ញាតិអោយ application របស់អ្នករក្សាទុកព័ត៌មាននៅក្នុងcurrent session ខណៈពេលដែលអ្នកប្រើប្រាស់បាន logged in ទៅកាន់ application របស់អ្នក ។ session ត្រូវបានកំណត់ ដោយ session ID តែមួយគត់ PHP បង្កើត session ID ដោយ MD5 hash នៃ remote IP addressនិង បន្ថែមនូវសញ្ញាផ្សេងៗដោយ random ទៅក្នុង hexadecimal string ។ យើងនឹងសរសេរឡើងវិញនូវឧទាហរណ៍លើកមុន ដែលយើងបានប្រើប្រាស់ cookie មកជំនួសដោយការប្រើប្រាស់ session វិញ ។

login.php

```
<html>
<body>
<?php

require("auth.php");
$auth = new Auth();
session_start();
if (isset($_POST['login']) && ($_POST['login'] == 'Log in') &&
$auth->authUser($_POST['txtuid'] , $_POST['txtpwd'] ) )
{
    $suid = $_POST['txtuid'];
    $_SESSION['uid'] = $suid;
    header("Location: http://localhost/index.php");
    exit();
}
else
{
    ?>
    /* HTML form comes here */
```

```
<?php
}
?>
```

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

index.php

```
<?php
if (isset ($_SESSION['uid']) && $_SESSION['uid'])
{
?>

    <html>
    <head><title>Index page</title></head>
    <body>
    Logged in with UID: <?php echo $_SESSION['uid']; ?><br />
    <a href='logout.php'>Log out</a>.
    </body>
    </html>
```

```
<?php
}
```

```
else
{
/* If no UID, we redirect to the login page */
header('Location: http://localhost/login.php');
}
```

```
?>
```

logout.php

```
<?php
session_start();
$_SESSION = array();
session_destroy();
header('Location: http://localhost/login.php');
?>
```

យើងនៅតែ initialize នូវ session ជាមួយនឹង session_start() បន្ទាប់ពីយើងអាច clear នូវ session បានដោយធ្វើអោយ \$_SESSION subperglobal ទៅជា empty array បន្ទាប់មកយើងធ្វើការDestroy session ហើយនឹង associate data ដោយការ call session_destroy() function ។

ចំនាំ : ដើម្បីលប់ចេញនូវធាតុទាំងអស់របស់ session អ្នកត្រូវតែផ្តល់ empty array ទៅអោយ variable នោះ ។

Good Luck For You