ម៉ូឌីលទី ១១ អនុគមន៍នៅក្នុង PHP (PHP Function)

១. What is function?

Function គឺជាប្លុកនៃកូដដែលសរសេរក្នុងកម្មវិធីដើម្បីអនុវត្តការងារជាក់លាក់មួយចំនួន។ យើងអាចទាក់ទង function នៅក្នុងកម្មវិធីទៅបុគ្គលិកនៅក្នុងការិយាល័យក្នុងជីវិតពិត ដើម្បីយល់កាន់តែច្បាស់អំពីរបៀបដែល function ដំណើរការ។ ឧបមាថាចៅហ្វាយចង់ឱ្យបុគ្គលិករបស់គាត់គណនាថវិកាប្រចាំឆ្នាំ។ ដូច្នេះតើដំណើរការនេះនឹងបញ្ចប់ដោយរបៀបណា? និយោជិតនឹងយកព័ត៌មានអំពីស្ថិតិពីចៅហ្វាយ ធ្វើការគណនា និងគណនាថវិកា ហើយបង្ហាញលទ្ធផលទៅចៅហ្វាយរបស់គាត់។ function ដំណើរការក្នុងលក្ខណៈស្រដៀងគ្នា។ ពួកគេយកព័ត៌មានជា parameter ប្រតិបត្តិប្លុកនៃសេចក្តីថ្លែងការណ៍ ឬអនុវត្តប្រតិបត្តិការលើ parameter នេះ ហើយត្រឡប់លទ្ធផល។ PHP ផ្តល់ឱ្យយើងនូវ function ធំៗពីរប្រភេទ៖

១.១. Built-in function

Built-in Function គឺជា function ដែលមានស្រាប់ភ្ជាប់មកជាមួយនឹងភាសាកម្មវិធីនិងនៅក្នុងភាសា PHP មាន Built-in Function ជាច្រើនដែលផ្តល់នូវភាពងាយស្រួលសម្រាប់ការសរសេរកូដរបស់អ្នក ។ PHP ផ្តល់ឱ្យយើងនូវការប្រមូលផ្តុំដ៏ធំនៃ built-in library functions ដែលភ្ជាប់មកជាមួយ។ មុខងារទាំងនេះត្រូវបានសរសេរកូដរួចហើយ និងរក្សាទុកជាទម្រង់ Functions ដើម្បីប្រើវា យើងគ្រាន់តែត្រូវការហៅពួកវាតាមតម្រូវការរបស់យើងដូចជា **var_dump, fopen(), print_r(), gettype()** ជាដើម។

១.២. User Defined Function

User Defined Function ជា function ដែលបង្កើតឡើងដោយអ្នកប្រើប្រាស់ផ្ទាល់ ។ ក្រៅពី built-in function PHP អនុញ្ញាតឱ្យយើងបង្កើត function ផ្ទាល់ខ្លួនរបស់យើងដែលហៅថា user defined functions ។ ដោយប្រើវា យើងអាចបង្កើតកញ្ចប់កូដផ្ទាល់ខ្លួនរបស់យើង ហើយប្រើវានៅពេលណាដែលចាំបាច់ដោយគ្រាន់តែហៅវា ។

Syntax

```
function functionName(){
//code to be executed
}
```

Example:

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output: *Hello PHP Function*

២. Why should we use functions?

- **Reusability:** ប្រសិនបើយើងមានកូដធម្មតាដែលយើងចង់ប្រើនៅផ្នែកផ្សេងៗនៃកម្មវិធី យើងអាចផ្ទុកវានៅក្នុង function មួយ ហើយហៅវាតាមតម្រូវការ។ នេះកាត់បន្ថយពេលវេលា និងការខិតខំប្រឹងប្រែងនៃពាក្យដដែលៗនៃកូដតែមួយ ។ នេះអាចត្រូវបានធ្វើទាំងនៅក្នុងកម្មវិធីមួយ និងដោយការនាំចូលឯកសារ PHP ដែលមាន function នៅក្នុងកម្មវិធីមួយចំនួនផ្សេងទៀត ។
- **Functions reduces the repetition of code within a program:** Function អនុញ្ញាតឱ្យអ្នកទាញយកប្លុកកូដដែលប្រើជាទូទៅទៅជាសមាសភាគតែមួយ។ ឥឡូវនេះ អ្នកអាចអនុវត្តភារកិច្ចដូចគ្នាដោយហៅ function នេះទៅគ្រប់ទីកន្លែងដែលអ្នកចង់បាននៅក្នុងស្ត្រីបរបស់អ្នកដោយមិនចាំបាច់ចម្លង និងបិទភ្ជាប់ប្លុកដូចគ្នានៃកូដម្តងហើយម្តងទៀត។
- **Functions makes the code much easier to maintain:** ដោយសារ function ដែលបានបង្កើតម្តងអាចប្រើបានច្រើនដង ដូច្នេះការផ្លាស់ប្តូរណាមួយដែលបានធ្វើឡើងនៅក្នុង function ត្រូវបានអនុវត្តដោយស្វ័យប្រវត្តិនៅគ្រប់ទីកន្លែងដោយមិនប៉ះឯកសារជាច្រើន ។
- **Functions makes it easier to eliminate the errors:** នៅពេលដែលកម្មវិធីត្រូវបានបែងចែកទៅជា function ប្រសិនបើមានកំហុសណាមួយកើតឡើង អ្នកដឹងច្បាស់ថា function អ្វីដែលបណ្តាលឱ្យមានកំហុស និងកន្លែងដែលត្រូវរកវា ។ ដូច្នេះការជួសជុលកំហុសកាន់តែងាយស្រួល ។
- **Functions can be reused in other application:** ដោយសារតែ function មួយត្រូវបានបំបែកចេញពីស្ត្រីបដែលនៅសល់ វាជាការងាយស្រួលក្នុងការប្រើ function ដូចគ្នានៅក្នុងកម្មវិធីផ្សេងទៀតដោយគ្រាន់តែបញ្ចូលឯកសារ PHP ដែលមាន function ទាំងនោះ។
- ផ្នែកខាងក្រោមនឹងបង្ហាញអ្នកពីរបៀបយ៉ាងងាយស្រួលដែលអ្នកអាចកំណត់ function ផ្ទាល់ខ្លួនរបស់អ្នកនៅក្នុង PHP ។

៣. Creating a Function

នៅពេលបង្កើត user defined function យើងត្រូវចងចាំរឿងមួយចំនួន៖

1. ឈ្មោះណាមួយដែលបញ្ចប់ដោយវង់ក្រចកបើក និងបិទ គឺជា function មួយ។
2. ឈ្មោះ function តែងតែចាប់ផ្តើមដោយ keyword function ។
3. ដើម្បីហៅ function មួយ យើងគ្រាន់តែត្រូវសរសេរឈ្មោះរបស់វាតាមវង់ក្រចក
4. ឈ្មោះ function មិនអាចចាប់ផ្តើមដោយលេខបានទេ ។ វាអាចចាប់ផ្តើមដោយអក្សរក្រម(A-Z, a-z) ឬសញ្ញាគូសពីក្រោម _ (underscore) ។
5. ឈ្មោះ function មិនប្រកាន់អក្សរតូចធំទេ ។

Syntax:

```
function functionName(){
   // Code to executed
}
```

Example:

```
<?php

function GroupA1()
{
   echo "This is Group A1";
}

// Calling the function
GroupA1();

?>
```

Output: *This is Group A1*

៤. Function with Parameters or Arguments

ព័ត៌មាន ឬអថេរនៅក្នុងវង់ក្រចករបស់ function ត្រូវបានគេហៅថា parameters។ ទាំងនេះត្រូវបានប្រើដើម្បីរក្សាតម្លៃដែលអាចប្រតិបត្តិបានក្នុងអំឡុងពេលដំណើរការ។ អ្នកប្រើប្រាស់មានសេរីភាពក្នុងការទទួលយក parameters ជាច្រើនតាមដែលគាត់ចង់បាន ដោយបំបែកដោយសញ្ញាក្បៀស(,) operator។ Parameters ទាំងនេះត្រូវបានប្រើដើម្បីទទួលយកធាតុបញ្ចូលក្នុងអំឡុងពេល ដំណើរការ។ ខណៈពេលដែលឆ្លងកាត់តម្លៃដូចជាអំឡុងពេលហៅ function ពួកវាត្រូវបានគេហៅថា Argument ។ Argument គឺជាតម្លៃដែលបានបញ្ជូនទៅ function មួយ ហើយ parameter ត្រូវ

បានប្រើដើម្បីរក្សា arguments ទាំងនោះ ។ នៅក្នុងពាក្យទូទៅ ទាំង parameter និង argument មានន័យដូចគ្នា ។ យើងត្រូវចងចាំថាសម្រាប់គ្រប់ parameter យើងត្រូវឆ្លងកាត់ argument ដែលត្រូវគ្នារបស់វា ។

Syntax:

```
function function_name($first_parameter, $second_parameter) {
    executable code;
}
```

Example1:

```php
<?php

// function along with three parameters
function proGeek($num1, $num2, $num3)
{
    $product = $num1 * $num2 * $num3;
    echo "The product is $product";
}

// Calling the function
// Passing three arguments
proGeek(2, 3, 5);

?>
```

Output: *The product is 30*

Example2:

```php
<?php
        // Definding function
        function getSum($num1, $num2){
            $sum = $num1 + $num2;
            echo "Sum of the two numbers $num1 and $num2 is : $sum";
        }
        // Calling function
        getSum(10,20);
    ?>
```

Output: *Sum of the two numbers 10 and 20 is : 30.*

៥. Setting Default Values for Function parameter

PHP អនុញ្ញាតឱ្យយើងកំណត់តម្លៃ default argument សម្រាប់ function parameters ។ ប្រសិនបើយើងមិនឆ្លងកាត់ argument ណាមួយសម្រាប់ parameter ដែលមាន default value នោះ PHP នឹងប្រើ default set value សម្រាប់ parameter នេះក្នុងការហៅ function ។

Example:

```php
<?php
// function with default parameter
function defVa($str, $num=12)
{
    echo "$str is $num years old \n";
}

// Calling the function
defVa("Ram", 15);

// In this call, the default value 12
// will be considered
defVa("Adam");

?>
```

Output: *Ram is 15 years old*
Adam is 12 years old

ក្នុងឧទាហរណ៍ខាងលើ parameter **$num** មាន default value 12 ប្រសិនបើយើងមិនហុចតម្លៃណាមួយសម្រាប់ parameter នេះក្នុងការហៅ function ទេ នោះ default value 12 នឹងត្រូវបានពិចារណា។ parameter **$str** ក៏មិនមាន default value ដែរ ដូច្នេះវាជាកាតព្វកិច្ច ។

៦. Return Values from Functions

Functions ក៏អាចត្រឡប់តម្លៃទៅផ្នែកនៃកម្មវិធីពីកន្លែងដែលវាត្រូវបានហៅ ។ **Return** Keyword ត្រូវបានប្រើដើម្បីត្រឡប់តម្លៃត្រឡប់ទៅផ្នែកនៃកម្មវិធីពីកន្លែងដែលវាត្រូវបានគេហៅ ។ Return value អាចជាប្រភេទណាមួយ រួមទាំង array និង objects ។ Return statement ក៏សម្គាល់ការបញ្ចប់នៃ function និងបញ្ឈប់ការប្រតិបត្តិបន្ទាប់ពីនោះ ហើយ returns value ។

Example1:

```php
<?php

// function along with three parameters
function proGeek($num1, $num2, $num3)
{
    $product = $num1 * $num2 * $num3;

    return $product; //returning the product
}

// storing the returned value
$retValue = proGeek(2, 3, 5);
echo "The product is $retValue";

?>
```

Output: *The product is 30*

Example2:

```php
<?php
        // Definding function
        function getSum($num1, $num2){
            $total = $num1 + $num2;
            return $total;
        }
        // Printing returned value
        echo getSum(5, 10); // Output : 15
    ?>
```

៧. Parameter or Arguments passing to Functions

PHP អនុញ្ញាតឱ្យយើងមានវិធីពីរយ៉ាងដែល arguments អាចត្រូវបានឆ្លងចូលទៅក្នុង function មួយ:

- **Pass by Value:** នៅលើ argument ឆ្លងកាត់ដោយប្រើតម្លៃ pass by value តម្លៃនៃ arguments ត្រូវបានផ្លាស់ប្តូរនៅក្នុង function មួយ ប៉ុន្តែតម្លៃដើមនៅខាងក្រៅ function នៅតែមិនផ្លាស់ប្តូរ ។ នោះមានន័យថាស្ទួន (duplicate) នៃតម្លៃដើមត្រូវបានឆ្លងកាត់ជា argument ។
- **Pass by Reference:** នៅលើ argument ដែលឆ្លងកាត់តាមឯកសារយោង តម្លៃដើមត្រូវបានឆ្លងកាត់។ ដូច្នេះតម្លៃដើមត្រូវបានផ្លាស់ប្តូរ ។ នៅក្នុង pass by reference យើងពិតជាឆ្លងកាត់អាសយដ្ឋាននៃតម្លៃ ដែលវាត្រូវបានរក្សាទុកដោយប្រើសញ្ញា ampersand (&) ។

Example1:

```php
<?php

// pass by value
function valGeek($num) {
 $num += 2;
   return $num;
}

   // pass by reference
   function refGeek(&$num) {
         $num += 10;
         return $num;
   }

   $n = 10;

   valGeek($n);
   echo "The original value is still $n \n";

   refGeek($n);
   echo "The original value changes to $n";

   ?>
```

Output:

The original value is still 10
The original value change to 20

Example2:

```php
<?php
        /* Definding a function that multiply a number by itself and return
        the new value. */
        function selfMulti(&$number){
            $number *= $number;
            return $number;
        }
        $mynum = 5;
        echo $mynum; // Output: 5


        echo "<br>";
        selfMulti($mynum);
        echo $mynum; // Ouput: 25
?>
```

៨. Understanding the variable scope

ទោះយ៉ាងណាក៏ដោយ អ្នកអាចប្រកាសអថេរនៅកន្លែងណាមួយនៅក្នុងស្ក្រីប PHP។ ប៉ុន្តែទីតាំងនៃការប្រកាសកំណត់វិសាលភាពនៃភាពមើលឃើញរបស់អថេរនៅក្នុងកម្មវិធី PHP ពោលគឺកន្លែងដែលអថេរអាចប្រើប្រាស់ ឬចូលប្រើបាន ។ ភាពងាយស្រួលនេះត្រូវបានគេស្គាល់ថាជាវិសាលភាពអថេរ ។ តាមលំនាំដើម អថេរដែលបានប្រកាសនៅក្នុង function គឺ **local** ហើយពួកវាមិនអាចមើលបាន ឬរៀបចំពីខាងក្រៅ function នោះ ដូចដែលបានបង្ហាញក្នុងឧទាហរណ៍ខាងក្រោម៖

Example:

```php
<?php
        // Defined function
        function test(){
            $great = "Hello World!";
            echo $great;
        }

        test(); // Output: Hello World!
        echo $great; // Generate undefined variable error
    ?>
```

ស្រដៀងគ្នានេះដែរ ប្រសិនបើអ្នកព្យាយាមចូលប្រើ ឬនាំចូលអថេរខាងក្រៅនៅខាងក្នុង function អ្នកនឹងទទួលបានកំហុសអថេរដែលមិនបានកំណត់ ដូចដែលបានបង្ហាញក្នុងឧទាហរណ៍ខាងក្រោម៖

Example:

```php
<?php
        $great = "Hello World!";

        // Defining function
        function test(){
            echo $great;
        }
        test(); // Generate undefined variable error
        echo $great; // Output: Hello World!
    ?>
```

ដូចដែលអ្នកអាចឃើញនៅក្នុងឧទាហរណ៍ខាងលើ អថេរដែលបានប្រកាសនៅខាងក្នុង function គឺមិនអាចចូលប្រើបានពីខាងក្រៅទេ ដូចគ្នាដែរអថេរដែលបានប្រកាសនៅខាងក្រៅ function គឺមិនអាចចូលប្រើនៅខាងក្នុង function បានទេ ។ ការបំបែកនេះកាត់បន្ថយឱកាសនៃអថេរនៅក្នុង function ដែលរងផលប៉ះពាល់ដោយអថេរនៅក្នុងកម្មវិធីចម្បង ។ វាអាចទៅរួចក្នុងការប្រើឈ្មោះដូចគ្នាឡើងវិញសម្រាប់អថេរក្នុង

function ផ្សេងៗគ្នា ចាប់តាំងពីមូលដ្ឋាន អថេរត្រូវបានទទួលស្គាល់ តែដោយ function ដែលពួកគេត្រូវបានប្រកាស ។

៩. The global keyword

វាអាចមានស្ថានភាពនៅពេលដែលអ្នកត្រូវការនាំចូលអថេរពីកម្មវិធីមេទៅក្នុង function ឬ ផ្ទុយទៅវិញ។ ក្នុងករណីបែបនេះ អ្នកអាចប្រើ **global keyword** មុនពេលអថេរនៅក្នុង function មួយ ។ Keyword នេះប្រែក្លាយអថេរទៅជាអថេរ **global** ដែលធ្វើឱ្យវាអាចមើលឃើញ ឬអាចចូលប្រើបានទាំងខាងក្នុង និងខាងក្រៅ function ដូចបង្ហាញក្នុងឧទាហរណ៍ខាងក្រោម៖

Example:

```php
<?php
        $great = "Hello World!";

        // Definding function
        function test(){
            global $great;
            echo $great;
        }
        test(); // Output: Hello World!
        echo "<br>";
        echo $great; // Output: Hello World!
        // Assign a new value to variable
        $great = "Goodbye";
        echo "<br>";

        test(); // Output: Goodbye
        echo "<br>";
        echo $great; // Output: Goodbye
    ?>
```

អ្នកនឹងស្វែងយល់បន្ថែមអំពីលទ្ធភាពមើលឃើញ និងការគ្រប់គ្រងការចូលប្រើនៅក្នុងមេរៀន class និង object របស់ PHP ។

១០. Creating recursive function

recursive function គឺជា function ដែលហៅខ្លួនឯងម្តងហើយម្តងទៀតរហូតដល់លក្ខខណ្ឌមួយត្រូវបានពេញចិត្ត។ Recursive function ជាញឹកញាប់ត្រូវបានគេប្រើដើម្បីដោះស្រាយការគណនាគណិតវិទ្យាដ៏ស្មុគស្មាញ ឬដើម្បីដំណើរការរចនាសម្ព័ន្ធដែលដាក់គ្នាយ៉ាងជ្រៅជាដើម ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីរបៀបដែល recursive function ដំណើរការ ។

Example1:

```php
<?php
        // Definding Recursive Function
        function printValues($arr){
            global $count;
            global $items;

            // Check input is an array
            if(!is_array($array)){
                die("ERROE: Input is not an array");
            }

            /*Loop through array, if value is itself an array recursively call the
            function else add the value found to the output items array, and
            increment counter by 1 for each value found. */

            foreach($arr as $a){
                if(is_array($a)){
                    printValues($a);
                }else{
                    $items[] = $a;
                    $count++;
                }
            }
            // Return total count and value found in array
            return array('total' => $count, 'values' => $items);
        }
        // Define nested array
        $species = array("birds" => array("Eagle","Parrot","swan"),
        "mammalss" => array(
            "Human","cat" => array(
                "Lion","Tiger","Jaquar"),
            ),
             "Elephan", "Monkey",

             "reptiles" => array(
                "snake" => array(


                    "Cobra" => array(
                        "King Cobra", "Egyptian cobra"
                    ),

                    "Viper", "Anaconda"
```

```
            ),
            "Crocodile", "Dinosaur" => array(
                "T-rex", "Almonsaurus"
            )
        )

    );
    // Count and print values in nested array
    $result = printValues($species);
    echo $result['total'] .'value($) found:';
    echo impode(',',$result['value']);
?>
```

Example2:

```
<?php
      function NaturalNumbers($number) {
            if($number<=10){
                  echo "$number <br/>";
            NaturalNumbers($number+1);
            }
      }

      NaturalNumbers(1);
?>
```

Output: *1, 2, 3, 4, 5, 6, 7, 8, 9, 10*

ចំណាំ៖ សូមប្រយ័ត្នពេលកំពុងបង្កើត function ដដែលៗ ព្រោះប្រសិនបើកូដត្រូវបានសរសេរមិនត្រឹមត្រូវ វាអាចបណ្តាលឱ្យមានការហៅចូល function មិនកំណត់ ។