ตัวอย่าง

```php
<?php
    $favcolor = "red";
    switch ($favcolor) {
        case "red":
            echo "สีที่คุณชอบ คือ สีแดง";
            break;
        case "blue":
            echo "สีที่คุณชอบ คือ สีฟ้า";
            break;
        case "green":
            echo "สีที่คุณชอบ คือ สีเขียว";
            break;
        default:
            echo "สีที่คุณชอบนอกเหลือจาก สีแดง สีฟ้า และ สีเขียว";
    }
?>
```

ผลลัพธ์

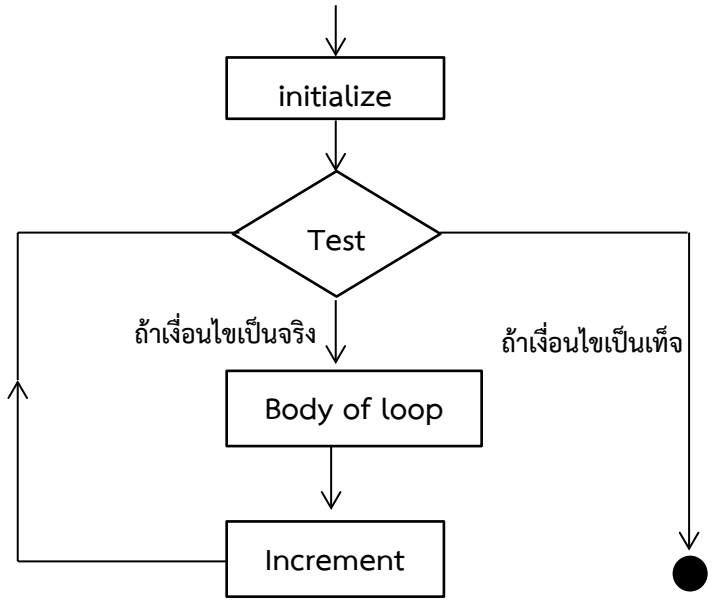
สีที่คุณชอบ คือ สีแดง

คำสั่งทำซ้ำ (Loop)

คำสั่งทำซ้ำ (Loop) คือ คำสั่งที่ทำงานซ้ำๆ วนไปมา เมื่อเงื่อนไขเป็นจริง แต่เมื่อไหร่ที่เงื่อนไขเป็นเท็จ คำสั่งทำซ้ำ จะจบการทำงานทันที ซึ่งสามารถแบ่งออกเป็น 2 คำสั่ง คือ For และ While

1. For Loop

ใช้ For Loop เมื่อทราบว่าต้องการให้โปรแกรมทำงานซ้ำๆ จำนวนกี่รอบ ดังรูปดังต่อไปนี้



ภาพที่ 4 โครงสร้างการทำงานของลูป For

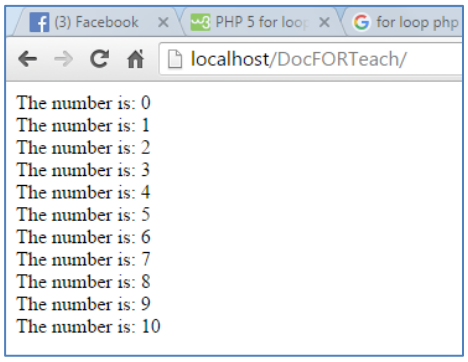
รูปแบบคำสั่ง

for (*เริ่มต้นการนับ; ทดสอบโค๊ด; เพิ่มค่าให้กับการนับ*) {
    โค๊ดจะทำงานเมื่อคำสั่ง for เป็นจริง
}

ตัวอย่างที่ 1 จงแสดงเลข 0-10

```php
<?php
    for ($x = 0; $x <= 10; $x++) {
        echo "The number is: $x <br>";
    }
?>
```

ผลลัพธ์ 1



ภาพที่ 5 ผลลัพธ์การทำงานของลูป For

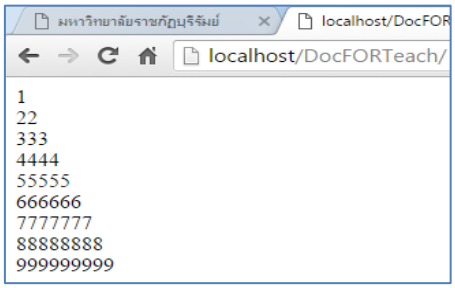
ตัวอย่างที่ 2 จงแสดงจำนวนเต็มที่เป็นเลขคี่ตั้งแต่เลข 1-10

```php
<?PHP
    for ($x = 1;$x <= 10;$x++){
        if($x % 2 == 1){
            echo $x;
        }
    }
?>
```

ผลลัพธ์ที่ 2

13579

ตัวอย่างที่ 3 จงเขียนโค๊ดให้แสดงผลลัพธ์ดังต่อไปนี้



ภาพที่ 6 ผลลัพธ์ของตัวอย่างที่ 3

คำตอบตัวอย่างที่ 3

```php
<?PHP
    for ($x = 1;$x <= 9;$x++){
        for ($y = 1 ;$y <= $x; $y++){
            echo $x;
        }
        echo "<br>";
    }
?>
```

2. Foreach Loop

Foreach Loop เป็นคำสั่งที่ใช้ทำงานกับตัวแปรที่เป็น Array ใช้การวนซ้ำไปเรื่อยๆ ตามจำนวนสมาชิกใน Array นั้นๆ

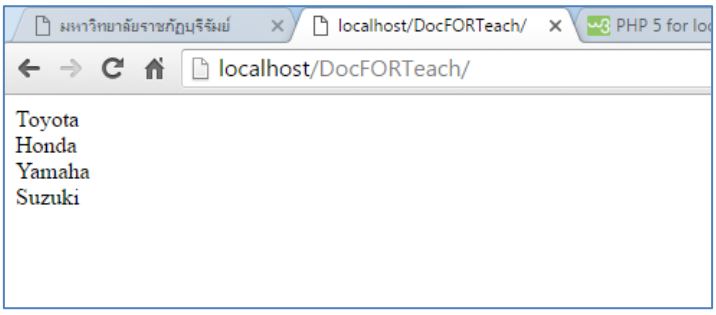
รูปแบบคำสั่ง

```
foreach ($array as $value) {
    คำสั่งที่จะทำงานเมื่อเงื่อนไข foreach เป็นจริง
}
```

ตัวอย่าง

```php
<?php
    $cars = array("Toyota", "Honda", "Yamaha", "Suzuki");
    foreach ($cars as $value) {
        echo "$value <br>";
    }
?>
```

ผลลัพธ์



ภาพที่ 7 ผลลัพธ์ของ foreach loop

3. While Loop

While Loop จะดำเนินการทำงานกับโค๊ดภายใต้คำสั่ง while loop เมื่อเงื่อนไขเป็นจริงเท่านั้น

รูปแบบคำสั่ง

```
while (เงื่อนไขเป็นจริง) {
    คำสั่งที่จะทำงานเมื่อเงื่อนไข while เป็นจริง
}
```

ตัวอย่างที่ 1

```php
<?php
    $x = 1;
    while($x <= 5) {
        echo "$x";
        $x++;
    }
?>
```

ผลลัพธ์ที่ 1

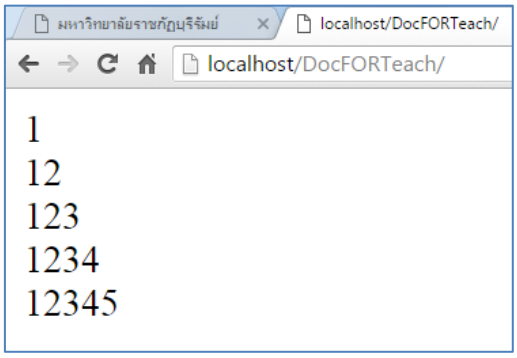
12345

คำสั่ง while loop สามารถเขียนคำสั่งซ้อนกันได้ตามความต้องการของผู้เขียนโค๊ดดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2

```php
<?php
    $x = 1;
    while($x <= 5) {
        $y = 1;
        while ($y<=$x){
            echo "$y";
            $y++;
        }
        echo "<br>";
        $x++;
    }
?>
```

ผลลัพธ์ที่ 2



ภาพที่ 8 ผลลัพธ์ของ while loop

ฟังก์ชัน (Function)

ฟังก์ชันการทำงานในภาษา php มีหลายฟังก์ชัน สามารถเรียกใช้ฟังก์ชันไหนก็ได้ตามที่ต้องการให้เหมาะสมกับงานที่ทำ แต่บางครั้งฟังก์ชันในภาษา php ไม่สามารถทำงานได้ตามที่ต้องการทั้งหมด จึงจำเป็นต้องเขียนฟังก์ชันขึ้นมาด้วยตนเองซึ่งมีข้อกำหนดดังนี้

1. ฟังก์ชันเป็นชุดคำสั่งที่สามารถใช้ซ้ำๆ ในโปรแกรมได้
2. ฟังก์ชันจะไม่ทำงานในทันทีเมื่อโหลดหน้าเว็บ
3. ฟังก์ชันจะทำงานก็ต่อเมื่อมีการเรียกใช้ฟังก์ชัน
4. การตั้งชื่อฟังก์ชันจะต้องไม่ตรงกับชื่อฟังก์ชันที่มีอยู่ภาษา php
5. ขึ้นต้นด้วยตัวอักษรภาษาอังกฤษ
6. ห้ามเว้นวรรคระหว่างตั้งชื่อฟังก์ชัน

รูปแบบของฟังก์ชัน

```
function ชื่อฟังก์ชัน() {
    โค๊ดที่ต้องการให้ฟังก์ชันทำงาน
}
```

ตัวอย่างที่ 1

```php
<?php
    function Message() {
        echo "Hello world";
    }
    Message(); // การเรียกใช้ฟังก์ชัน
?>
```

ผลลัพธ์ที่ 1

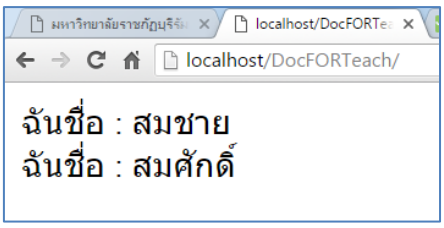
Hello world

จากตัวอย่างที่ 1 เป็นฟังก์ชันที่แสดงผลตามที่ต้องการให้แสดงเท่านั้น นอกจากนี้ยังสามารถส่งข้อมูลเข้าไปในฟังก์ชันได้ โดยผ่านอาร์กิวเมนต์ของฟังก์ชัน ซึ่งจะกำหนดอาร์กิวเมนต์ไว้ในวงเล็บของฟังก์ชัน และสามารถกำหนดได้มากกว่า 1 อาร์กิวเมนต์ โดยขั้นด้วยเครื่องหมายจุลภาค (,) แสดงดังตัวอย่างดังต่อไปนี้

ตัวอย่างที่ 2

```php
<?php
    echo '<meta charset="utf-8">';
    function name($fname) {
        echo "ฉันชื่อ :  $fname <br>";
    }
    name("สมชาย");//การเขียนใช้ฟังก์ชัน
    name("สมศักดิ์");//การเขียนใช้ฟังก์ชัน
?>
```

ผลลัพธ์ที่ 2

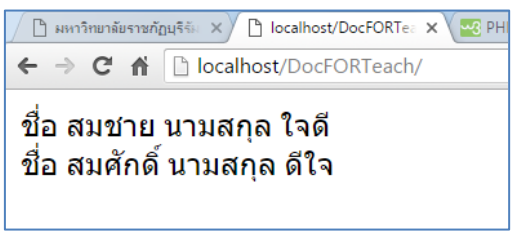


ภาพที่ 9 ผลลัพธ์ของฟังก์ชันอาร์กิวเมนต์

ตัวอย่างที่ 3

```php
<?php
    echo '<meta charset="utf-8">';
    function name($fname,$lname) {
        echo "ชื่อ  $fname นามสกุล  $lname <br>";
    }
    name("สมชาย","ใจดี");  //การเขียนใช้ฟังก์ชัน
    name("สมศักดิ์","ดีใจ");//การเขียนใช้ฟังก์ชัน
?>
```

ผลลัพธ์ที่ 3



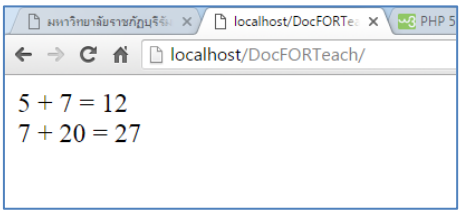
ภาพที่ 10 ผลลัพธ์ของฟังก์ชันอาร์กิวเมนต์มากกว่า 1 ตัว

นอกจากนี้ฟังก์ชันยังสามารถส่งค่ากลับ (return) เพื่อนำค่าที่ได้ไปคำนวณทำอย่างอื่นได้อีก เช่น นำค่าไปคำนวณทางคณิตศาสตร์ เป็นต้น เรียกว่า ฟังก์ชันรีเทิร์น แสดงดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 4

```php
<?php
    function sum($x, $y) {
        $z = $x + $y;
        return $z;
    }
    echo "5 + 7 = " . sum(5, 7) . "<br>"; //การเรียกใช้ฟังก์ชัน
    echo "7 + 20 = " . sum(7, 20);  //การเรียกใช้ฟังก์ชัน
?>
```

ผลลัพธ์ที่ 4



ภาพที่ 11 ผลลัพธ์ของฟังก์ชันรีเทิร์น

ตัวแปรหลายค่า (Array)

ตัวแปรอาร์เรย์ เป็นตัวแปรที่สามารถเก็บค่าไว้ได้หลายค่าในตัวแปรเดียว ซึ่งมีรูปแบบการเขียนดังนี้

$val = array(“แดง”,20) ;

$val หมายถึง ชื่อตัวแปร

“แดง” หมายถึง ค่าที่เป็นตัวอักษร

20 หมายถึง ค่าที่เป็นตัวเลข

การใช้งานตัวแปร array ให้เรียกตามตำแหน่ง index ของค่านั้นๆ ซึ่งจะเริ่มจาก 0 ดังตัวอย่างดังต่อไปนี้

ตัวอย่างที่ 1

```php
<?php
    echo '<meta charset="utf-8">';
    $col = array("แดง", "ม่วง", "เขียว");
    echo "สีที่ฉันชอบคือ " . $col[0] . " " . $col[1] . " และ " . $col[2] ;
?>
```

ผลลัพธ์ที่ 1

สีที่ฉันชอบคือ แดง ม่วง และ เขียว

ถ้าหากต้องการเปลี่ยนค่าของตัวแปรอาร์เรย์ให้อ้างไปที่ index ของค่านั้นๆ เช่น

ตัวอย่างที่ 2

```php
<?php
    echo '<meta charset="utf-8">';
    $col = array("แดง", "ม่วง", "เขียว");
    $col[0] = "ดำ";
    $col[1] = "ขาว";
    $col[2] = "น้ำเงิน";
    echo "สีที่ฉันชอบคือ " . $col[0] . " " .  $col[1] . " และ " . $col[2];
?>
```

ผลลัพธ์ที่ 2

สีที่ฉันชอบคือ ดำ ขาว และ น้ำเงิน

การนับจำนวนสมาชิก array โดยใช้ฟังก์ชัน count( )

```php
<?php
    $x = array("A", "B", "C");
    echo count($x);
?>
```

ผลลัพธ์

3