บทที่ 3
ตัวแปร ตัวดำเนินการและนิพจน์

ในการศึกษาการเขียนโปรแกรมทุกภาษา สิ่งที่ควรเรียนรู้เป็นลำดับแรกๆ คือ การกำหนดตัวแปรสำหรับเก็บข้อมูลบางอย่างเอาไว้ก่อนจะนำไปใช้งานอื่นๆ ต่อไป และขั้นตอนถัดไปก็คือ ทำความรู้จักการใช้ตัวดำเนินการแบบต่างๆ เพื่อใช้ในการประมวลผลข้อมูลให้ได้ผลลัพธ์ตามต้องการ

3.1 ตัวแปร

ตัวแปร (Variable) ใช้ในการเก็บพักข้อมูลบางอย่างก่อนที่จะนำข้อมูลนั้นไปใช้งานอื่นๆ ต่อไป สำหรับตัวแปรในภาษา PHP ไม่จำเป็นต้องได้รับการประกาศอย่างชัดเจนเหมือนภาษาอื่นๆ แต่ก็ไม่เป็นสิ่งที่ดีนักหากขาดทักษะการเขียนโปรแกรมที่ดี ก็อาจจะทำให้เกิดความสับสนในการใช้งานได้ ในภาษา PHP มีข้อกำหนดสำหรับประกาศใช้ตัวแปรที่สำคัญ ดังต่อไปนี้

3.1.1 ตัวแปรในภาษา PHP ไม่จำเป็นต้องระบุชนิดของข้อมูล เนื่องจากตัวแปรแต่ละตัวสามารถเก็บข้อมูลชนิดใดก็ได้

3.1.2 ตัวแปรในภาษา PHP จะต้องขึ้นด้วยสัญลักษณ์เครื่องหมาย $ (Dollar Sign) แล้วตามด้วยชื่อของตัวแปรที่ต้องการใช้งาน เช่น $name, $value, $a, $x เป็นต้น

3.1.3 การตั้งชื่อตัวแปรต้องขึ้นต้น ด้วยอักษร a-z หรือ A-Z หรือเครื่องหมาย _ (Underscore) เท่านั้น ห้ามขึ้นต้นด้วยตัวเลข 0-9 หรืออักขระอื่นๆ นอกเหนือจากนี้ ตัวอย่างการกำหนดตัวแปรที่ถูกต้อง เช่น $name, $_price, $value1, $num2string เป็นต้น

3.1.4 การเขียนตัวแปรด้วยลักษณะ ตัวพิมพ์ที่แตกต่างกัน ถือว่าเป็นตัวแปรคนละตัว เช่น $abc, $ABC เป็นต้น จะถือว่าไม่ใช่ตัวแปรเดียวกัน

3.1.5 สามารถนำอักขระภาษาอื่นๆ มาตั้งเป็นชื่อตัวแปรได้ เช่น $ชื่อ, $จำนวน1, $ผลลัพธ์ เป็นต้น แต่โดยทั่วไปแล้ว นิยมตั้งชื่อตัวแปรเป็นภาษาอังกฤษมากกว่า

3.2 การกำหนดค่าตัวแปร

เมื่อได้ประกาศตัวแปรก็สามารถเริ่มต้นใช้งานได้ทันที แต่ก่อนที่จะนำตัวแปรไปใช้งานได้นั้น ตัวแปรจะต้องเก็บข้อมูลบางอย่างเอาไว้ หรือเรียกว่าการกำหนดค่าตัวแปร การกำหนดค่าตัวแปรแบบง่ายๆ คือ การคัดลอกค่าหรือโอนถ่ายค่าของนิพจน์หรือสมการต่างๆ โดยมีสัญลักษณ์ที่ใช้กำหนดค่า คือ เครื่องหมาย = (เท่ากับ) การกำหนดค่าตัวแปรหลักๆ จะมีอยู่ด้วยกัน 4 วิธี คือ 1) กำหนดค่าตัวแปรปกติ (Value assignment) 2) กำหนดค่าตัวแปรโดยการอ้างอิง (Reference assignment) 3) กำหนดค่าตัว

แปรซ้อนตัวแปร (Variable variables assignment) และ 4) กำหนดตัวแปรค่าคงที่ (Constants variables) ดังนี้

3.2.1 การกำหนดค่าตัวแปรปกติ

การกำหนดค่าตัวแปรรูปแบบนี้ เป็นการกำหนดค่าตัวแปรแบบง่ายที่สุดและนิยมใช้งานโดยทั่วไป สามารถกำหนดได้หลายลักษณะขึ้นอยู่กับชนิดข้อมูล ดังนี้

1) การกำหนดค่าตัวแปรชนิดตัวเลข ตัวอย่างดังนี้

ตัวอย่างที่ 3.1 การกำหนดค่าตัวแปรชนิดตัวเลข

```
1    <?php
2        $x = 123;
3        $y = 4.56;
4        $z = -789;
5    ?>
```

จากตัวอย่างที่ 3.1 เป็นการกำหนดค่าตัวแปรชนิดตัวเลข อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $x มีค่าเท่ากับ 123

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $y มีค่าเท่ากับ 4.56

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $z มีค่าเท่ากับ -789

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

2) การกำหนดค่าตัวแปรชนิดข้อความ สามารถกำหนดได้ตามรูปแบบการเขียนข้อความนั่นคือกำหนดข้อความไว้ภายในเครื่องหมาย Double Quotes ("...") หรือ Single Quotes ('...') เท่านั้น ตัวอย่างดังนี้

ตัวอย่างที่ 3.2 การกำหนดค่าตัวแปรชนิดข้อความ

```
1    <?php
2        $name = "Parinya";
3        $country = 'Thailand';
4        $phone = '0123456789';
5    ?>
```

จากตัวอย่างที่ 3.2 เป็นการกำหนดค่าตัวแปรชนิดข้อความ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $name มีค่าเท่ากับ "Parinya"

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $country มีค่าเท่ากับ 'Thailand'

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $phone มีค่าเท่ากับ '0123456789' เลขที่เขียนอยู่ภายในเครื่องหมาย Double Quotes หรือ Single Quotes จะถือว่าเป็น ชนิดข้อความ แต่สามารถนำไปใช้คำนวณได้

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

3) การกำหนดค่าตัวแปรชนิดบูลีน สามารถกำหนดค่าเป็น true หรือ false อย่างใดอย่างหนึ่ง ดังนี้

ตัวอย่างที่ 3.3 การกำหนดค่าตัวแปรชนิดบูลีน

```
1    <?php
2        $first_time = true ;
3        $is_valid = false ;
4    ?>
```

จากตัวอย่างที่ 3.3 เป็นการกำหนดค่าตัวแปรชนิดบูลีน อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าตัวแปร $first_time มีค่าเท่ากับ true

บรรทัดที่ 3 กำหนดค่าตัวแปร $is_valid มีค่าเท่ากับ false

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

การใช้งานตัวแปรใน PHP มีความยืดหยุ่น ไม่จำกัดชนิดของตัวแปรเป็นชนิดใดชนิดหนึ่ง และยังสามารถสลับปรับเปลี่ยนได้ในทันที มีตัวอย่าง ดังนี้

ตัวอย่างที่ 3.4 การกำหนดค่าตัวแปรแบบไม่จำกัดชนิดของตัวแปร

```
1    <?php
2        $color = "red";
3        $number = 12;
4        $age = '12 ';
5        $sum = 12 + "15";              // ผลลัพธ์ $sum = 27
6    ?>
```

จากตัวอย่างที่ 3.4 เป็นการกำหนดค่าตัวแปรแบบไม่จำกัดชนิดของตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าตัวแปร $color มีค่าเท่ากับ "red" (เป็นชนิดข้อความ)

บรรทัดที่ 3 กำหนดค่าตัวแปร $number มีค่าเท่ากับ 12 (เป็นชนิดเลขจำนวนเต็ม)

บรรทัดที่ 4 กำหนดค่าตัวแปร $age มีค่าเท่ากับ '12' (เป็นชนิดข้อความ)

บรรทัดที่ 5 กำหนดค่าตัวแปร $sum มีค่าเท่ากับ 12 + '15' (เมื่อ 12 คือ ข้อมูลชนิดเลขจำนวนเต็ม นำมาบวกกับ '15' ข้อความ) ลักษณะเช่นนี้ในภาษาอื่นๆ จะไม่สามารถคำนวณได้ แต่สำหรับในภาษา PHP จะปรับเปลี่ยนชนิดของตัวแปรโดยอัตโนมัติ ทำให้สามารถนำมารวมค่ากันได้ ดังนั้นในบรรทัดที่ 5 จึงหมายถึง กำหนดค่าตัวแปร $sum มีค่าเท่ากับ 27 นั่นเอง

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

3.2.2 การกำหนดค่าตัวแปรโดยการอ้างอิง

การกำหนดค่าตัวแปรโดยการอ้างอิง ช่วยให้สามารถสร้างตัวแปรใหม่ให้มีค่าเช่นเดียวกันกับตัวแปรใดๆ ก็ได้ สามารถสร้างความสัมพันธ์ระหว่างตัวแปรหลักและตัวแปรอ้างอิง เมื่อมีการเปลี่ยนแปลงค่าในตัวแปรใดๆ ค่าของตัวแปรทั้งหมดที่มีการอ้างอิงกันจะมีค่าหรือข้อมูลเดียวกัน การกำหนดตัวแปรโดยอ้างอิงสามารถทำได้โดยการเพิ่มเครื่องหมายแอนด์ (&) ด้านขวาของเครื่องหมายเท่ากับหรือด้านหน้าของตัวแปร มีตัวอย่างดังนี้

ตัวอย่างที่ 3.5 การกำหนดค่าตัวแปรโดยการอ้างอิง

```
1   <?php
2       $value1 = "Hello";
3       $value2 =& $value1;   // $value1 และ $value2 มีค่าเท่ากับ "Hello"
4       $value2 = "Goodbye"; // $value1 และ $value2 มีค่าเท่ากับ "Goodbye"
5       printf("Value 1 is %s and Value 2 is %s.", $value1, $value2);
6   ?>
```

ผลลัพธ์

```
Value 1 is Goodbye and Value 2 is Goodbye.
```

จากตัวอย่างที่ 3.5 เป็นการกำหนดค่าตัวแปรโดยการอ้างอิง อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าตัวแปร $value1 มีค่าเท่ากับ "Hello"

บรรทัดที่ 3 กำหนดค่าตัวแปร $value2 อ้างอิงค่ากับตัวแปร $value1 (เสมือนเป็นตัวแปรตัวเดียวกัน)

บรรทัดที่ 4 กำหนดค่าตัวแปร $value2 มีค่าเท่ากับ "Goodbye" ยังผลให้ตัวแปร $value1 เปลี่ยนแปลงค่าตามไปด้วย คือ ทั้งตัวแปร $value1 และ $value2 มีค่าเท่ากับ "Goodbye"

บรรทัดที่ 5 แสดงผลค่าที่เก็บในตัวแปร $value1 และ $value2

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

3.2.3 การกำหนดค่าตัวแปรซ้อนตัวแปร

การกำหนดค่าตัวแปรซ้อนตัวแปร เป็นการกำหนดค่าให้กับตัวแปร โดยกำหนดผ่านอีกตัวแปรหนึ่ง หรือการกำหนดชื่อตัวแปรตามค่าของตัวแปร จะสามารถกำหนดค่าลักษณะแบบนี้ได้นั้น ค่าของตัวแปรหลักจะต้องเป็นชนิดข้อความเท่านั้น ตัวอย่างดังนี้

ตัวอย่างที่ 3.6 การกำหนดค่าตัวแปรซ้อนตัวแปร

```
1   <?php
2       $test = "x";
3       $$test = 5;
4       printf("Test value = %s <br>", $test);
5       printf("X value = %d ", $x);
6   ?>
```

ผลลัพธ์

```
Test value = x
X value = 5
```

จากตัวอย่างที่ 3.6 เป็นการกำหนดค่าตัวแปรซ้อนตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าตัวแปร $test มีค่าเท่ากับ "x" (ข้อความเท่านั้น)

บรรทัดที่ 3 กำหนดค่าตัวแปร $$test มีค่าเท่ากับ 5 (ใช้สัญลักษณ์ $ ซ้อนอีก 1 เครื่องหมาย ความหมายคือ กำหนดค่าให้กับค่าของตัวแปร $test เมื่อค่าของตัวแปร $test มีค่าเท่ากับ "x" คือ มีตัวแปรเพิ่มขึ้นมาอีก 1 ตัวแปร โดยปริยาย คือ $x)

บรรทัดที่ 4 แสดงผลค่าที่เก็บในตัวแปร $test (เป็นชนิดข้อความ)

บรรทัดที่ 5 แสดงผลค่าที่เก็บในตัวแปร $x (เป็นชนิดเลขจำนวนเต็ม)

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

3.2.4 การกำหนดตัวแปรค่าคงที่

การกำหนดตัวแปรค่าคงที่ เป็นการกำหนดค่าให้กับตัวแปรมีค่าใดค่าหนึ่งตลอดการใช้งานในโปรแกรม โดยไม่มีการเปลี่ยนแปลงค่า เมื่อตัวแปรค่าคงที่ถูกกำหนดแล้ว ตัวแปรจะไม่สามารถเปลี่ยนหรือคืนค่าได้อีกต่อไป จะแตกต่างจากการกำหนดค่าตัวแปรปกติ นั่นก็คือ ตัวแปรค่าคงที่ไม่ต้องใช้เครื่องหมาย $ นำหน้าตัวแปร การสร้างและกำหนดค่าตัวแปรจำเป็นต้องใช้ฟังก์ชัน define ( ) สำหรับการสร้างและกำหนดค่า มีรูปแบบดังนี้

รูปแบบ

```
bool define ( string $name , mixed $value [, bool $case_insensitive = false ] )
```

เมื่อ $name หมายถึง ชื่อตัวแปร

$value	หมายถึง	ค่าที่ต้องการกำหนด
$case_insensitive	หมายถึง	false คือ ชื่อตัวแปรที่จะเรียกใช้ต้องเหมือนกับอักษรที่กำหนด เช่น กำหนดเป็นอักษรพิมพ์ใหญ่ เวลาเรียกใช้ก็ต้องใช้อักษรพิมพ์ใหญ่ด้วย เป็นต้น (ค่าโดยปริยาย) True คือ ไม่สนใจชื่อตัวแปรที่จะเรียกใช้ว่าเป็นอักษรพิมพ์เล็กหรือพิมพ์ใหญ่

ตัวอย่างที่ 3.7 การกำหนดตัวแปรค่าคงที่

```
1   <?php
2       define("CONSTANT", "Hello Suratthani.");
3       echo CONSTANT . "<br>";
4       echo Constant . "<br>";
5       define("GREETING", "Hello Suratthani.", true);
6       echo GREETING . "<br>";
7       echo Greeting . "<br>";
8   ?>
```

ผลลัพธ์

```
Hello Suratthani.
Constant
Hello Suratthani.
Hello Suratthani.
```

จากตัวอย่างที่ 3.7 เป็นการกำหนดตัวแปรค่าคงที่ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 ใช้ฟังก์ชัน define("CONSTANT", "Hello Suratthani.") หมายความว่า กำหนดตัวแปรค่าคงที่ CONSTANT มีค่าเท่ากับ "Hello Suratthani." (กำหนดให้เรียกใช้ตัวแปรด้วยอักษรพิมพ์ใหญ่เท่านั้น)

บรรทัดที่ 3 แสดงผลค่าที่เก็บในตัวแปรค่าคงที่ CONSTANT ผลลัพธ์ที่แสดงผล คือ คำว่า Hello Suratthani.

บรรทัดที่ 4 แสดงผลข้อความ Constant (ข้อความ ไม่ใช่ตัวแปร)

บรรทัดที่ 5 ใช้ฟังก์ชัน define("GREETING", "Hello Suratthani.", true) หมายความว่า กำหนดตัวแปรค่าคงที่ GREETING มีค่าเท่ากับ "Hello Suratthani." (โดยกำหนดพารามิเตอร์เพิ่มเป็น true คือ ไม่สนใจชื่อตัวแปรที่จะเรียกใช้ว่าเป็นอักษรพิมพ์เล็กหรือพิมพ์ใหญ่)

บรรทัดที่ 6 แสดงผลค่าที่เก็บในตัวแปรค่าคงที่ GREETING ผลลัพธ์ที่แสดงผล คือ คำว่า Hello Suratthani.

บรรทัดที่ 7 แสดงผลค่าที่เก็บในตัวแปรค่าคงที่ Greeting ผลลัพธ์ที่แสดงผล คือ คำว่า Hello Suratthani.

บรรทัดที่ 8 สิ้นสุดสคริปต์ PHP

3.3 ขอบเขตของตัวแปร

การประกาศและกำหนดค่าตัวแปรทั้งวิธีกำหนดค่าปกติ หรือกำหนดค่าแบบอ้างอิง หรือกำหนดค่าตัวแปรซ้อนตัวแปร หรือกำหนดค่าตัวแปรคงที่ ในภาษา PHP สามารถประกาศได้ทุกที่ในสคริปต์ของ PHP ตำแหน่งของการประกาศตัวแปรมีผลต่อการใช้งานหรือเรียกว่าขอบเขตของตัวแปร (Variable Scope) จะมีผลต่อการเข้าถึงและใช้งาน ขอบเขตของตัวแปรมี 6 ชนิด ประกอบด้วย 1) ตัวแปรเฉพาะที่ (Local variables) 2) ตัวแปรฟังก์ชันพารามิเตอร์ (Function parameters) 3) ตัวแปรสาธารณะ (Global variables) 4) ตัวแปรคงที่ (Static variables) 5) ตัวแปรพิเศษ (Super Global variables) และ 6) ตัวแปรที่มีค่า NULL มีรายละเอียด ดังนี้

3.3.1 ตัวแปรเฉพาะที่

ตัวแปรเฉพาะที่ คือ ตัวแปรที่ถูกประกาศไว้ภายในฟังก์ชัน จะมีขอบเขตเฉพาะภายในฟังก์ชันที่ประกาศใช้ตัวแปรเท่านั้น และฟังก์ชันนั้นๆ ก็ไม่สามารถเรียกใช้ตัวแปรที่ประกาศไว้ภายนอกฟังก์ชันได้ ถึงแม้ว่าชื่อตัวแปรจะเหมือนกันก็ตาม ถือว่าเป็นตัวแปรคนละตัว ลักษณะเช่นนี้จึงเรียกว่าขอบเขตตัวแปรเฉพาะที่ ตัวอย่างดังนี้

ตัวอย่างที่ 3.8 แสดงตัวอย่างการอ้างอิงตัวแปรเฉพาะที่

```
1   <?php
2       function update_counter ( ) {
3           $counter=$counter*2;
4       }
5       $counter = 10;
6       update_counter ( );
7       echo $counter;
8   ?>
```

ผลลัพธ์

```
10
```

จากตัวอย่างที่ 3.8 แสดงตัวอย่างการอ้างอิงตัวแปรเฉพาะที่ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 เริ่มต้นฟังก์ชัน update_count ( ) (ฟังก์ชันจะไม่ทำงานหากไม่มีการเรียกใช้งานฟังก์ชัน แต่ถ้าฟังก์ชันถูกเรียกใช้งานจะทำคำสั่งที่อยู่ภายในเครื่องหมาย { } ของฟังก์ชัน เมื่อฟังก์ชันทำงานตามลำดับคำสั่งแล้วเสร็จ จะไปทำงานต่อบรรทัดที่ 7)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $counter มีค่าเท่ากับค่าของตัวแปรก่อนหน้าคูณด้วย 2 (หากไม่มีการกำหนดค่าเริ่มต้นให้กับตัวแปร โดยปริยายตัวแปรจะมีค่าเป็น NULL คือ ไม่มีค่า เมื่อนำมาคูณด้วย 2 จะมีค่าเท่ากับ 0)

บรรทัดที่ 4 จบการทำงานของฟังก์ชัน

บรรทัดที่ 5 กำหนดให้ตัวแปร $counter มีค่าเท่ากับ 10

บรรทัดที่ 6 เรียกใช้งานฟังก์ชัน update_count ( )

บรรทัดที่ 7 แสดงผลค่าที่เก็บในตัวแปร $counter ผลลัพธ์ที่แสดง คือ 10 เป็นผลมาจากการกำหนดค่าตัวแปร ในบรรทัดที่ 5 โดยตัวแปรที่อยู่ในบรรทัดที่ 3 มีขอบเขตของตัวแปรเฉพาะที่ นั่นเองตามที่ได้กล่าวถึงไว้ข้างต้น

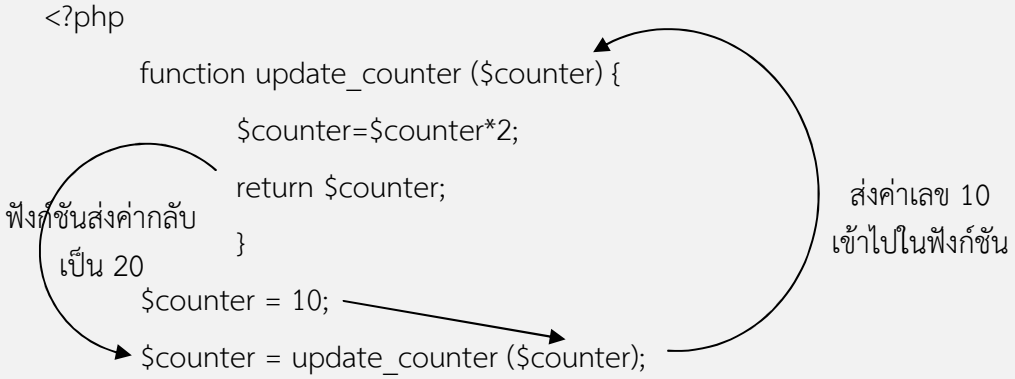
บรรทัดที่ 8 สิ้นสุดสคริปต์ PHP

3.3.2 ตัวแปรฟังก์ชันพารามิเตอร์

ตัวแปรฟังก์ชันพารามิเตอร์ คือ ตัวแปรที่ถูกส่งผ่านไปยังฟังก์ชันหนึ่งๆ การส่งค่าพารามิเตอร์นั้นสามารถส่งได้ครั้งละหลายๆ ตัวแปร โดยทำการคั่นแต่ละพารามิเตอร์ด้วยเครื่องหมาย comma (,) สำหรับรูปแบบการส่งค่าจะกล่าวถึงในบทถัดไป ตัวอย่างการส่งค่าตัวแปรฟังก์ชันพารามิเตอร์ตัวอย่างดังนี้

ตัวอย่างที่ 3.9 แสดงตัวอย่างการอ้างอิงตัวแปรฟังก์ชันพารามิเตอร์

```
1   <?php
2       function update_counter ($counter) {
3           $counter=$counter*2;
4           return $counter;
5           }
6       $counter = 10;
7       $counter = update_counter ($counter);
8       echo $counter;
9   ?>
```



ผลลัพธ์

```
20
```

จากตัวอย่างที่ 3.9 แสดงตัวอย่างการอ้างอิงตัวแปรฟังก์ชันพารามิเตอร์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 เริ่มต้นฟังก์ชัน update_count ($counter) (ฟังก์ชันจะไม่ทำงานหากไม่มีการเรียกใช้งานฟังก์ชัน แต่ถ้าฟังก์ชันถูกเรียกใช้งานจะทำคำสั่งที่อยู่ภายในเครื่องหมาย { } ของฟังก์ชัน เมื่อทำงานตามลำดับแล้วเสร็จจะ return ผลลัพธ์ไปให้กับตัวแปร $counter ในบรรทัดที่ 7 ลักษณะการทำงานจะคล้ายกับตัวอย่างที่ 3.8 แตกต่างที่ฟังก์ชันมีการกำหนดตัวแปรฟังก์ชันพารามิเตอร์สำหรับการรับและส่งค่า)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $counter มีค่าเท่ากับค่าของตัวแปร $counter ที่มาจากตัวแปรฟังก์ชันพารามิเตอร์ ที่ส่งเข้าไปในฟังก์ชัน จากตัวอย่าง คือ เลข 10 คูณด้วย 2 (มีค่าเท่ากับ 20)

บรรทัดที่ 4 ฟังก์ชันส่งค่ากลับด้วย return $counter (มีค่าเท่ากับ 20) กลับมาให้ตัวแปร $counter ในบรรทัดที่ 7

บรรทัดที่ 5 จบการทำงานของฟังก์ชัน

บรรทัดที่ 6 กำหนดให้ตัวแปร $counter มีค่าเท่ากับ 10

บรรทัดที่ 7 กำหนดค่าให้กับตัวแปร $counter มีค่าเท่ากับ ผลของการเรียกใช้งานฟังก์ชัน update_count ( ) คือ ค่าของผลลัพธ์ที่ได้จากการ return ของฟังก์ชัน

บรรทัดที่ 8 แสดงผลค่าที่เก็บในตัวแปร $counter ผลลัพธ์ที่แสดง คือ 20

บรรทัดที่ 9 สิ้นสุดสคริปต์ PHP

3.3.3 ตัวแปรสาธารณะ

ตัวแปรสาธารณะ คือ ตัวแปรที่ประกาศไว้ภายนอกฟังก์ชัน สามารถเข้าถึงจากส่วนใดๆ ของโปรแกรม แต่จะไม่สามารถใช้งานได้ภายในฟังก์ชัน แต่ถ้าต้องการเรียกใช้งานตัวแปรสาธารณะที่อยู่ภายนอกฟังก์ชัน ภายในฟังก์ชันจะต้องใช้คำสั่ง GLOBAL แล้วตามด้วยชื่อของตัวแปรที่ต้องการใช้งานหรืออ้างอิงถึงตัวแปรที่อยู่ภายนอกฟังก์ชัน ตัวอย่างดังนี้

ตัวอย่างที่ 3.10 แสดงตัวอย่างการอ้างอิงตัวแปรสาธารณะโดยใช้คำสั่ง GLOBAL

```
1   <?php
2       function update_counter ( ) {
3           GLOBAL $counter;
4           $counter=$counter*2;
5       }
6       $counter = 10;
7       update_counter ( );
8       echo $counter;              // ผลลัพธ์ คือ 20
9   ?>
```

จากตัวอย่างที่ 3.10 แสดงตัวอย่างการอ้างอิงตัวแปรสาธารณะ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 เริ่มต้นฟังก์ชัน update_count ( ) (ฟังก์ชันจะไม่ทำงานหากไม่มีการเรียกใช้งานฟังก์ชัน แต่ถ้าฟังก์ชันถูกเรียกใช้งานจะทำคำสั่งที่อยู่ภายในเครื่องหมาย { } ของฟังก์ชัน เมื่อทำงานแล้วเสร็จตามลำดับคำสั่ง จะไปบรรทัดที่ 8)

บรรทัดที่ 3 กำหนดให้ฟังก์ชันสามารถเรียกใช้ตัวแปร $counter (ตัวแปรสาธารณะ) ที่กำหนดไว้ภายนอกฟังก์ชัน ด้วยการเพิ่ม GLOBAL นำหน้าชื่อตัวแปร $counter

บรรทัดที่ 4 กำหนดให้ตัวแปร $counter มีค่าเท่ากับ ค่าของตัวแปรก่อนหน้าคูณด้วย 2 (มีค่าเท่ากับ 20)

บรรทัดที่ 5 จบการทำงานของฟังก์ชัน

บรรทัดที่ 6 กำหนดให้ตัวแปร $counter มีค่าเท่ากับ 10

บรรทัดที่ 7 เรียกใช้งานฟังก์ชัน update_count ( )

บรรทัดที่ 8 แสดงผลค่าที่เก็บในตัวแปร $counter ผลลัพธ์ที่แสดง คือ 20

บรรทัดที่ 9 สิ้นสุดสคริปต์ PHP

นอกจากการอ้างอิงโดยใช้คำสั่ง GLOBAL แล้ว ยังมีอีกวิธีหนึ่งในการอ้างอิงตัวแปรสาธารณะได้ คือ การใช้ตัวแปรพิเศษ มีลักษณะรูปแบบการใช้งานแบบอาร์เรย์ ตัวแปรพิเศษที่ใช้สำหรับอ้างอิงตัวแปรสาธารณะ มีชื่อเรียกเฉพาะ คือ $GLOBALS ตัวอย่างดังนี้

ตัวอย่างที่ 3.11 การอ้างอิงตัวแปรสาธารณะผ่านตัวแปรพิเศษ $GLOBALS

```
1   <?php
2       function update_counter ( ) {
3           $GLOBALS["counter"] = $GLOBALS["counter"]*2;
4       }
5       $counter = 10;
6       update_counter ( );
7       echo $counter;              // ผลลัพธ์ คือ 20
8   ?>
```

จากตัวอย่างที่ 3.11 แสดงตัวอย่างการอ้างอิงตัวแปรสาธารณะ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 เริ่มต้นฟังก์ชัน update_count ( ) (ฟังก์ชันจะไม่ทำงานหากไม่มีการเรียกใช้งานฟังก์ชัน แต่ถ้าฟังก์ชันถูกเรียกใช้งานจะทำคำสั่งที่อยู่ภายในเครื่องหมาย { } ของฟังก์ชัน เมื่อทำงานแล้วเสร็จตามลำดับคำสั่ง จะไปบรรทัดที่ 7)

บรรทัดที่ 3 เรียกใช้และกำหนดค่าตัวแปรสาธารณะผ่านตัวแปรพิเศษ $GLOBALS["counter"] มีค่าเท่ากับ $GLOBALS["counter"] คูณด้วย 2 (มีค่าเท่ากับ 20)

บรรทัดที่ 4 จบการทำงานของฟังก์ชัน

บรรทัดที่ 5 กำหนดให้ตัวแปร $counter มีค่าเท่ากับ 10

บรรทัดที่ 6 เรียกใช้งานฟังก์ชัน update_count ( )

บรรทัดที่ 7 แสดงผลค่าที่เก็บในตัวแปร $counter ผลลัพธ์ที่แสดง คือ 20

บรรทัดที่ 8 สิ้นสุดสคริปต์ PHP

3.3.4 ตัวแปรคงที่

ตัวแปรคงที่ จะทำการเก็บค่าของตัวแปรนั้นๆ ไว้ทุกครั้งที่เรียกใช้ฟังก์ชัน เมื่อเรียกใช้ฟังก์ชันกี่ครั้งค่าของตัวแปรที่ถูกประกาศเป็นแบบคงที่จะถูกปรับปรุงค่าไว้เสมอ สามารถใช้ได้กับตัวแปรเฉพาะที่เท่านั้น มีตัวอย่าง ตัวอย่างดังนี้

ตัวอย่างที่ 3.12 การใช้งานตัวแปรคงที่

```
1   <?php
2       function show_number ( ) {
3           STATIC $count = 0;
4           $count=$count+10;
5           echo $count;
6           echo "<br>";
7       }
8       show_number ( );
9       show_number ( );
10      show_number ( );
11  ?>
```

ผลลัพธ์ของการทำงานถ้าหากไม่มีการกำหนด STATIC ไว้ด้านหน้าของตัวแปร $count ดังนี้

10

10

10

แต่ผลลัพธ์ของการทำงานเมื่อมีการกำหนด STATIC ไว้ด้านหน้าตัวแปร $count จะได้ผลลัพธ์ ดังนี้

10

20

30

3.3.5 ตัวแปรพิเศษ

ตัวแปรพิเศษ คือ ตัวแปรที่กำหนดไว้ล่วงหน้าภายในระบบของภาษา PHP สามารถเข้าถึงและเรียกใช้ตัวแปรพิเศษได้จากทุกที่ภายในสคริปต์ที่ทำงานอยู่ เช่น เรียกดูรายละเอียดการใช้งานของผู้ใช้ ณ ปัจจุบัน รายละเอียดของเว็บเซิร์ฟเวอร์ ระบบปฏิบัติการ และอื่นๆ ตัวแปรพิเศษที่สำคัญ ประกอบด้วย 1) $_SERVER 2) $_GET 3) $_POST 4) $_ENV 5) $_COOKIE และ 6) $_SESSION รายละเอียด ดังนี้

1) $_SERVER เป็นตัวแปรพิเศษที่ใช้สำหรับเก็บค่าสารสนเทศต่างๆ ของเว็บเซิร์ฟเวอร์ที่กำลังทำงานอยู่ ตัวแปรพิเศษ $_SERVER จัดเก็บอยู่ในรูปแบบของอาร์เรย์ ประกอบไปด้วยอิลิเมนท์ (Element) ที่สามารถเรียกดูข้อมูลได้จำนวนมาก โดยอิลิเมนท์แต่ละตัวของตัวแปรพิเศษ $_SERVER จะทำหน้าที่บอกสารสนเทศแตกต่างกัน โดยมีรูปแบบการเรียกดูข้อมูลตัวแปรพิเศษ $_SERVER ดังนี้

รูปแบบ

```
$_SERVER["ชื่ออิลิเมนท์"]
```

ตารางที่ 3.1 แสดงชื่ออิลิเมนท์ของตัวแปรพิเศษ $_SERVER ที่สำคัญ และใช้งานบ่อย

ชื่ออิลิเมนท์	คำอธิบาย
PHP_SELF	เก็บชื่อไฟล์ที่กำลังทำงานอยู่ อิลิเมนท์นี้มักใช้เมื่อต้องการอ้างถึงหรือเรียกใช้การประมวลผลที่ไฟล์สคริปต์เดิมในขณะนั้น
HTTP_REFERER	เก็บค่าที่อยู่ URL ของหน้าเว็บเพจที่อ้างถึงในหน้าเว็บเพจปัจจุบัน
REMOTE_ADDR	เก็บค่า IP Address ของเครื่องไคลเอ็นท์ที่กำลังเข้าชมเว็บเพจนั้นอยู่
QUERY_STRING	เก็บค่า Query String ทั้งหมดที่ต่อท้ายจาก URL ของหน้าเว็บเพจปัจจุบัน

การเรียกใช้ตัวแปรพิเศษ $_SERVER เพื่อแสดง IP Address ที่กำลังใช้งานอยู่ ตัวอย่างดังนี้

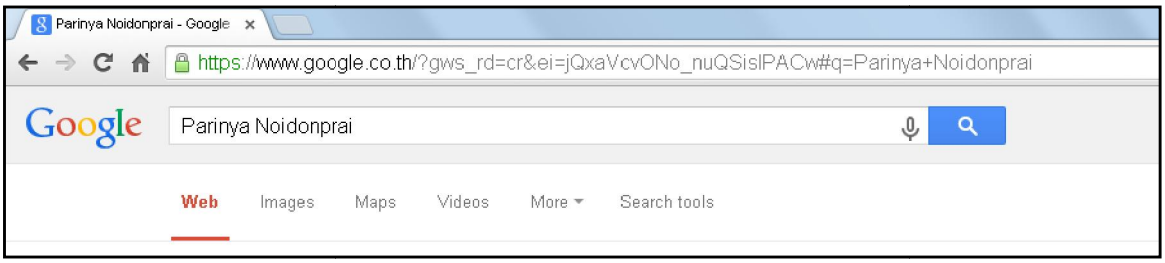
ตัวอย่างที่ 3.13 การใช้งานตัวแปรพิเศษ $_SERVER เพื่อเรียกดู IP Address

```
1   <?php
2       printf ("Your IP address is: %s", $_SERVER["REMOTE_ADDR"]);
3   ?>
```

ผลลัพธ์

```
Your IP address is: 127.0.0.1
```

2) $_GET เป็นตัวแปรพิเศษใช้เก็บค่าของตัวแปรที่ถูกส่งมากับสคริปต์ในรูปแบบของ Query String ต่อท้าย URL ดังภาพที่ 3.1



ภาพที่ 3.1 แสดงตัวอย่างการส่งค่าแบบ Query String

จากภาพที่ 3.1 เป็นการเชื่อมโยงไปยังหน้าค้นหาโดยมีการส่งค่าตัวแปรไป โดยตัวแปรที่ส่งไปจะอยู่หลังเครื่องหมาย "?" และสามารถส่งได้ครั้งละหลายๆ ตัวแปร โดยใช้สัญลักษณ์ "&" เชื่อมแต่ละตัวแปร

3) $_POST เป็นตัวแปรพิเศษใช้เก็บค่าของตัวแปรที่ถูกส่งมาจากเว็บเพจที่มีการส่งค่าของมูลแบบ POST ส่วนใหญ่จะอยู่ในรูปแบบของฟอร์มที่กำหนด "method" เป็น POST เช่นเดียวกับ $_GET แต่ไม่แสดงผลตัวแปรและค่าตัวแปรบน URL

การกำหนด "method" โดยส่วนใหญ่จะกำหนดเป็น POST เพื่อให้ส่งข้อมูลจาก field ต่างๆ ไป ในส่วนหัวของโปรโตคอล HTTP request สามารถส่งข้อมูลจำนวนมากได้ ส่วนการส่งด้วย method GET จะใช้ในกรณีที่มี field และความยาวข้อมูลน้อย และไม่เป็นความลับ เนื่องจากการส่งแบบ GET เป็นการสร้าง Query String จาก field ต่างๆ ต่อท้าย URL โดยอัตโนมัติ และการส่งแบบ GET ไม่สามารถใช้ได้กับฟอร์มที่มีการส่งไฟล์

4) $_ENV เป็นตัวแปรพิเศษใช้สำหรับเก็บตัวแปรที่จัดเก็บสภาพแวดล้อมทั่วไป และค่าต่างๆ ของระบบที่ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ ผลลัพธ์ที่ได้จะแตกต่างกันตามรูปแบบของตัวแปรที่ใช้ เช่น $COMPUTERNAME เป็นตัวแปร Environment ชนิดหนึ่งที่ใช้แสดงชื่อของเครื่องคอมพิวเตอร์

5) $_COOKIE เป็นตัวแปรพิเศษที่ใช้สำหรับเก็บค่าของข้อมูลไว้ระยะเวลาหนึ่ง เพื่อใช้งานกับเอกสารเว็บเพจทั้งเว็บไซต์ (การเก็บข้อมูลไว้ในตัวแปรพิเศษ $_COOKIE ระบบจะทำการสร้างไฟล์เก็บตัวแปรพิเศษ $_COOKIE ไว้ที่เครื่องไคลเอ็นต์) การที่จะใช้ตัวแปรพิเศษ $_COOKIE ได้นั้น จะต้องมีการสร้างตัวแปรพิเศษ $_COOKIE ก่อน โดยใช้ฟังก์ชัน setcookie ( ) รายละเอียดการใช้งาน จะกล่าวถึงในบทต่อไป

6) $_SESSION เป็นตัวแปรพิเศษที่ใช้สำหรับเก็บค่าของข้อมูลไว้ระยะเวลาหนึ่ง เพื่อใช้งานกับเอกสารเว็บเพจทั้งเว็บไซต์เหมือนกับตัวแปรพิเศษ $_COOKIE (การเก็บข้อมูลไว้ในตัวแปรพิเศษ $_SESSION นั้น ระบบจะทำการสร้างไฟล์เก็บตัวแปรพิเศษ $_SESSION ไว้ในฝั่งของเว็บเซิร์ฟเวอร์) การที่จะใช้ตัวแปรพิเศษ $_SESSION ได้นั้น จะต้องมีการสั่งเริ่มใช้เซสชัน โดยใช้คำสั่ง session_start ( ) รายละเอียดการใช้งาน จะกล่าวถึงในบทต่อไป

ตัวแปรพิเศษ $_COOKIE และ ตัวแปรพิเศษ $_SESSION เหมือนกัน คือ ใช้สำหรับเก็บค่าตัวแปรไว้เพื่อเรียกใช้ในหน้าเว็บเพจต่างๆ ภายในเว็บไซต์ และทั้งคู่มีความแตกต่างกัน คือ อายุของตัวแปร โดยอายุของตัวแปรพิเศษ $_COOKIE ถูกกำหนดด้วยเวลา ส่วนอายุของตัวแปรพิเศษ

$_SESSION ถูกกำหนดด้วยการทำงานของบราวเซอร์ เมื่อไคลเอนต์ทำการปิดโปรแกรมเว็บบราวเซอร์ ตัวแปรพิเศษ $_SESSION ก็จะถูกทำลาย

3.3.6 ตัวแปรที่มีค่า NULL

ค่า NULL คือ ตัวแปรว่างเปล่าหรือไม่มีค่าใดๆ ทั้งนี้หากสร้างตัวแปรขึ้นมาโดยไม่มีการกำหนดค่าใดๆให้กับตัวแปร จะทำให้ตัวแปรนั้นมีค่าเป็น NULL หรืออาจเกิดจากการสร้างตัวแปรแล้วกำหนดค่าให้เท่ากับ NULL เช่น ในกรณีต่อไปนี้ ตัวแปร $a และ $b จะมีค่าเป็น NULL

ตัวอย่างที่ 3.14 ตัวแปรที่มีค่า NULL

```
1    <?php
2        $a;
3        $b = NULL;
4        printf ("Value A is: %s <br> Value B is: %s", $a, $b);
5    ?>
```

สำหรับในภาษา PHP ตัวแปรที่มีค่า NULL เมื่อนำไปใช้งาน จะมีผลดังนี้

1) หากนำไปคำนวณจะมีค่า เทียบเท่ากับ 0
2) หากนำไปใช้ในแบบข้อความ จะมีค่า เทียบเท่ากับ "" หรือค่าว่าง
3) หากนำไปเปรียบเทียบทางตรรกะ จะเทียบเท่า กับค่า false

3.4 การตรวจสอบและยกเลิกตัวแปร

การเขียนโปรแกรมในระดับสูงขึ้นไปจะเป็นการทำงานแบบไดนามิก ในบางกรณีไม่อาจทราบล่วงหน้าได้ว่า ข้อมูล (ตัวแปร) ที่ต้องการใช้งานนั้นมีอยู่จริง หรือเก็บค่าใดๆ เอาไว้หรือไม่ จึงต้องมีคำสั่งในการตรวจสอบข้อมูลเหล่านี้ก่อนนำไปใช้เพื่อป้องกันข้อผิดพลาด นอกจากนี้ตัวแปรที่สร้างขึ้นอาจจะใช้งานเพียงชั่วระยะเวลาหนึ่ง และหลังจากนั้นหากมีตัวแปรนี้อยู่ต่อไปอาจจะส่งต่อการทำงานของระบบ ดังนั้นจึงต้องมีคำสั่งสำหรับการตรวจสอบและยกเลิกตัวแปร ฟังก์ชันที่สำคัญประกอบด้วย 1) isset ( ) 2) empty ( ) และ 3) unset ( ) มีรายละเอียดดังนี้

3.4.1 isset ( )

เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบว่ามีตัวแปรตามที่ระบุอยู่จริงหรือไม่ หากไม่มีตัวแปร (เนื่องจากยังไม่ได้ประกาศใช้ตัวแปร) หรือตัวแปรมีค่าเป็น NULL ฟังก์ชัน isset ( ) จะส่งค่า false แต่หากมีการประกาศใช้ตัวแปรอยู่จริง และไม่มีค่าเป็น NULL ฟังก์ชัน isset ( ) จะส่งค่า true มีรูปแบบดังนี้

```
bool isset ( mixed $var [, mixed $... ] )
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการตรวจสอบ

ตัวอย่างที่ 3.15 การใช้ฟังก์ชัน isset ( ) สำหรับตรวจสอบการกำหนดตัวแปร

```
1    <?php
2        $value1 = NULL;
3        $value2 = 2015;
4        var_dump(isset($value1));
5        echo "<br>";
6        var_dump(isset($value2));
7    ?>
```

ผลลัพธ์

```
bool (false)
bool (true)
```

จากตัวอย่างที่ 3.15 ตัวอย่างการใช้ฟังก์ชัน isset ( ) สำหรับตรวจสอบตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $value1 มีค่าเท่ากับ NULL

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $value2 มีค่าเท่ากับ 2015

บรรทัดที่ 4 แสดงผลการตรวจสอบตัวแปร $value1 ว่าได้ประกาศใช้ และ ไม่เป็นค่า NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool(true) หากเป็นเท็จจะแสดงคำว่า bool(false)

บรรทัดที่ 5 ขึ้นบรรทัดใหม่

บรรทัดที่ 6 แสดงผลการตรวจสอบตัวแปร $value2 ว่าได้ประกาศใช้ และ ไม่เป็นค่า NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool(true) หากเป็นเท็จจะแสดงคำว่า bool(false)

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

3.4.2 empty ( )

เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบตัวแปร มีค่าว่างจริงหรือไม่ ในภาษา PHP จะถือว่าตัวแปรมีค่าว่างในลักษณะต่อไปนี้

1) มีค่าเป็น NULL
2) มีค่าเป็นข้อความว่าง หรือ ""
3) มีค่า 0 หรือ "0"

ถ้าตัวแปรมีค่าว่างจะได้ผลลัพธ์เป็น true หากตัวแปรมีการกำหนดค่า จะได้ผลลัพธ์เป็น false มีรูปแบบดังนี้

รูปแบบ

```
bool empty ( mixed $var )
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการตรวจสอบ

ตัวอย่างที่ 3.16 การใช้ฟังก์ชัน empty ( ) สำหรับตรวจสอบค่าว่างของตัวแปร

```
1    <?php
2        $value1 = NULL;
3        $value2 = 2015;
4        var_dump (empty ($value1) );
5        echo "<br>";
6        var_dump (empty ($value2) );
7    ?>
```

ผลลัพธ์

```
bool (true)
bool (false)
```

จากตัวอย่างที่ 3.16 ตัวอย่างการใช้ฟังก์ชัน empty ( ) สำหรับตรวจสอบค่าว่างของตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $value1 มีค่าเท่ากับ NULL

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $value2 มีค่าเท่ากับ 2015

บรรทัดที่ 4 แสดงผลการตรวจสอบตัวแปร $value1 เป็นค่าว่าง หรือ มีค่าเป็น NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool(true) หากเป็นเท็จจะแสดงคำว่า bool(false)

บรรทัดที่ 5 ขึ้นบรรทัดใหม่

บรรทัดที่ 6 แสดงผลการตรวจสอบตัวแปร $value2 เป็นค่าว่าง หรือ มีค่าเป็น NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool(true) หากเป็นเท็จจะแสดงคำว่า bool(false)

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

3.4.3 unset ( )

เป็นฟังก์ชันที่ใช้สำหรับยกเลิกการใช้ตัวแปรที่ระบุ เมื่อไม่ต้องการใช้ตัวแปรตัวนั้นอีกต่อไป เพื่อคืนทรัพยากรให้แก่ระบบ มีรูปแบบดังนี้

```
void unset ( mixed $var [, mixed $... ] )
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการยกเลิก

ตัวอย่างที่ 3.17 การใช้ฟังก์ชัน unset ( ) สำหรับยกเลิกการใช้ตัวแปร

```
1    <?php
2        $value = " Parinya";
3        var_dump (isset ($value) );
4        echo "<br>";
```

```
5           unset ($value);
6           var_dump (isset ($value1) );
7       ?>
```

ผลลัพธ์

```
bool (true)
bool (false)
```

จากตัวอย่างที่ 3.17 ตัวอย่างการใช้ฟังก์ชัน unsset ( ) สำหรับยกเลิกตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $value มีค่าเท่ากับ " Parinya"

บรรทัดที่ 3 แสดงผลการตรวจสอบตัวแปร $value ว่าได้ประกาศใช้ และ ไม่เป็นค่า NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool (true) หากเป็นเท็จจะแสดงคำว่า bool (false) จากตัวอย่างจะเห็นว่าบรรทัดแรกที่แสดงผล คือ bool (true) อันเนื่องมาจากตัวแปร $value มีการประกาศใช้ และ ไม่เป็นค่า NULL

บรรทัดที่ 4 ขึ้นบรรทัดใหม่

บรรทัดที่ 5 ยกเลิกการใช้งานตัวแปร $value

บรรทัดที่ 6 แสดงผลการตรวจสอบตัวแปร $value ว่าได้ประกาศใช้ และ ไม่เป็นค่า NULL จริงหรือเท็จ หากเป็นจริงจะแสดงคำว่า bool (true) หากเป็นเท็จจะแสดงคำว่า bool (false) จากตัวอย่างจะเห็นว่าบรรทัดที่สองผลลัพธ์ คือ bool (false) อันเนื่องมาจากตัวแปร $value ถูกยกเลิกด้วยฟังก์ชัน unset ( )

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

3.5 ตัวดำเนินการ และนิพจน์

ตัวดำเนินการ (Operators) คือ สัญลักษณ์ที่ใช้กำหนดรูปแบบการประมวลผลข้อมูล

นิพจน์ (Expressions) คือ การกระทำเพื่อให้ได้ผลลัพธ์ค่าหนึ่งค่า ประกอบไปด้วยตัวถูกกระทำ (Operands) และตัวดำเนินการ เขียนเรียงกันไป เช่น 3 * 2 - 1 + 7 หรือ a * 5 เป็นต้น

ลำดับความสำคัญของตัวดำเนินการ มีความสำคัญมากในการเขียนโปรแกรม ถ้าจัดลำดับผิดก็อาจจะทำให้โปรแกรมของเกิดข้อผิดพลาด (Bug) ได้ และบางทีจะทำให้หาจุดผิดพลาดนั้นๆ ยากด้วย ลำดับความสำคัญของตัวดำเนินการแต่ละตัวจะมีความสำคัญแตกต่างกัน โดยลำดับความสำคัญสูงสุดจะถูกดำเนินการก่อนลำดับความสำคัญต่ำ ตัวอย่าง ดังนี้

ตัวอย่างที่ 3.18 แสดงนิพจน์และตัวดำเนินการ

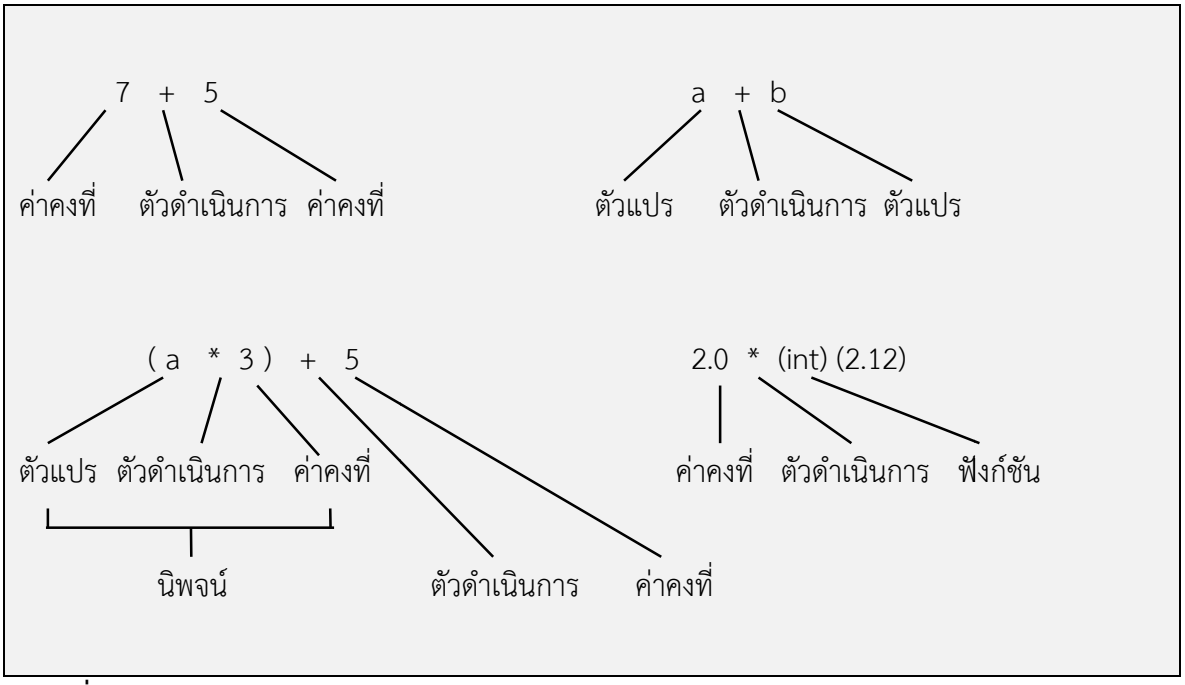
```
2 + 4 * 3
```

จากตัวอย่างที่ 3.18 แสดงนิพจน์และตัวดำเนินการ ถ้าลองมาคำนวณกันเอง ก็คงจะได้หลายๆ ค่าแตกต่างกันไปเนื่องจากว่าไม่รู้ว่าจะนำตัวไหนมาคำนวณกันก่อน จากตัวอย่างเครื่องหมายที่มีความสำคัญสูงคือเครื่องหมายคูณ รองลงมา คือ บวก หมายความว่าจะดำเนินการคำนวณ 4 * 3 ก่อน แล้วค่อยนำผลลัพธ์นี้ไปบวกกับ 2 ผลลัพธ์จึงมีค่าเท่ากับ 14

แต่ถ้าไม่ต้องการให้เกิดความยุ่งยากในการจัดลำดับความสำคัญ ขอแนะนำให้ใช้เครื่องหมายวงเล็บ ( ) ในการแบ่งการคำนวณออกจากกัน ตัวอย่างดังนี้

ตัวอย่างที่ 3.19 แสดงนิพจน์และดำลับความสำคัญของตัวดำเนินการ

```
(2 + 4) * 3
หรือ ขึ้นอยู่กับว่าจะให้นิพจน์ใดคำนวณก่อน-หลัง การคำนวณจะเกิดขึ้นในวงเล็บก่อนเสมอ
2 + (4 * 3)
```



ภาพที่ 3.2 แสดงองค์ประกอบของการคำนวณทางคณิตศาสตร์

3.5.1 ประเภทของตัวดำเนินการ

ประเภทของตัวดำเนินการ สามารถแบ่งได้ 7 ประเภท ประกอบด้วย 1) ตัวดำเนินการทางคณิตศาสตร์ 2) ตัวดำเนินการเชื่อมต่อข้อความ 3) ตัวดำเนินการกำหนดค่า 4) ตัวดำเนินการเพิ่ม และลดค่า 5) ตัวดำเนินการเปรียบเทียบ 6) ตัวดำเนินการเปรียบเทียบทางตรรกะ และ 7) ตัวดำเนินการอื่นๆ มีรายละเอียด ดังต่อไปนี้

1) ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) ประกอบด้วย

ตารางที่ 3.2 ตัวดำเนินการทางคณิตศาสตร์

ลำดับความสำคัญ	สัญลักษณ์	ความหมาย
1	*	คูณ (Multiplication)
	/	หาร (Division)
	%	เศษจากการหาร (Modulus)
2	+	บวก (Addition)
	-	ลบ (Subtraction)

จากตารางที่ 3.2 ตัวดำเนินการทางคณิตศาสตร์ จะมีลำดับความสำคัญอยู่ 2 ระดับ คือ ระดับที่ 1 ที่มีความสำคัญมากสุด ประกอบด้วย *, / และ % และระดับที่ 2 ประกอบด้วย + และ – ในกรณีที่นิพจน์ใดๆ มีตัวดำเนินการระดับเดียวกันปรากฏอยู่ในนิพจน์ นิพจน์จะเริ่มคำนวณจากซ้ายไปขวาเสมอ รูปแบบการ

ตัวอย่างที่ 3.20 แสดงตัวอย่างการคำนวณทางคณิตศาสตร์และกำหนดค่าให้กับตัวแปร

```
1    <?php
2        $a = 2*100/10;
3        $b = 100/10%2;
4        $c = 100%80*2;
5        $d = 100/10+30-20;
6    ?>
```

จากตัวอย่างที่ 3.20 ตัวอย่างการคำนวณทางคณิตศาสตร์และลำดับความสำคัญ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $a มีค่าเท่ากับ 20 (เพราะลำดับความสำคัญของตัวดำเนินการคูณและหารมีลำดับความสำคัญเท่ากัน การคำนวณจะเริ่มจากซ้ายไปขวา คือ 2 คูณกับ 100 มีค่าเท่ากับ 200 แล้วนำไปหารด้วย 10 จึงมีค่าเท่ากับ 20)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $b มีค่าเท่ากับ 0 (เพราะลำดับความสำคัญของตัวดำเนินการหารและเศษจากการหาร มีลำดับความสำคัญเท่ากัน การคำนวณจะเริ่มจากซ้ายไปขวา คือ 100 หารกับ 10 มีค่าเท่ากับ 10 แล้วนำไปหาผลเศษจากการหารด้วย 10 จึงมีค่าเท่ากับ 0)

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $c มีค่าเท่ากับ 40 (เพราะลำดับความสำคัญของตัวดำเนินการเศษจากการหารและคูณ มีลำดับความสำคัญเท่ากัน การคำนวณจะเริ่มจากซ้ายไปขวา คือ 100 หารหาผลเศษจากการหารกับ 80 มีค่าเท่ากับ 20 แล้วนำไปคูณด้วย 2 จึงมีค่าเท่ากับ 40)

บรรทัดที่ 5 กำหนดค่าให้กับตัวแปร $d มีค่าเท่ากับ 20 (เพราะการคำนวณจะเริ่มต้นจากตัวดำเนินการหารก่อนเนื่องจากตัวดำเนินการหารมีความสำคัญมากกว่าบวกและลบ ดังนั้นจะเริ่มต้นจาก 100 หารด้วย 10 มีค่าเท่ากับ 10 ลำดับถัดไปบวกและลบมีลำดับความสำคัญเท่ากัน จะเริ่มจากซ้ายไปขวา คือ นำผลลัพธ์ที่ได้จากการหาร คือ 10 ไปบวกด้วย 30 มีค่าเท่ากับ 40 แล้วจึงลบด้วย 20 ผลลัพธ์ที่ได้จึงมีค่าเท่ากับ 20)

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

ในทางปฏิบัติเพื่อหลีกเลี่ยงข้อผิดพลาดจากลำดับความสำคัญของตัวดำเนินการ ควรใช้วงเล็บในการจัดแบ่งกลุ่มให้ชัดเจน

2) ตัวดำเนินการเชื่อมต่อข้อความ (String Concatenation)

สำหรับในภาษา PHP จะใช้เครื่องหมายจุด (.) ในการเชื่อมต่อข้อความ ตัวอย่างตัวดำเนินการสำหรับเชื่อมต่อข้อความ ดังตัวอย่างที่ 3.21

ตัวอย่างที่ 3.21 การใช้เครื่องหมายจุด (.) ในการเชื่อมต่อข้อความ

```
1   <?php
2       $str = "My" . "SQL";                    // $str = “MySQL”
3       $fname = "Parinya";
4       $lname = "Noidonprai";
5       $fullname = $fname ." ". $lname
6       echo $fullname;
7   ?>
```

ผลลัพธ์

```
Parinya Noidonprai
```

จากตัวอย่างที่ 3.21 ตัวอย่างการเชื่อมข้อความ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $str มีค่าเท่ากับข้อความ "MySQL"

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $fname มีค่าเท่ากับข้อความ "Parinya"

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $lname มีค่าเท่ากับข้อความ "Noidonprai"

บรรทัดที่ 5 กำหนดค่าให้กับตัวแปร $fullname มีค่าเท่ากับตัวแปร $fname เชื่อมด้วยช่องว่างและค่าของตัวแปร $lname คือ มีค่าเท่ากับข้อความ "Parinya Noidonprai"

บรรทัดที่ 6 แสดงผลค่าที่เก็บไว้ในตัวแปร $fname

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

3) ตัวดำเนินการกำหนดค่า (Assignment)

ตัวดำเนินการกำหนดค่า คือ การกำหนดค่าให้กับตัวแปรที่อยู่ทางด้านซ้ายของตัวดำเนินการกำหนดค่า ด้วยค่าที่อยู่ทางขวาโดยประกอบด้วยตัวดำเนินการดังต่อไปนี้

ตารางที่ 3.3 ตัวดำเนินการกำหนดค่า

สัญลักษณ์	ความหมาย	ตัวอย่าง
=	กำหนดค่าปกติ	$x = 10;
+=	นำค่าที่กำหนดไปบวกเพิ่มจากค่าเดิมตัวแปร แล้วนำผลลัพธ์ที่ได้จัดเก็บไว้ในตัวแปรเดิม	$x = 10; $x += 8; // $x = 18
-=	ลดค่าตัวแปรลงเท่ากับค่าที่ระบุ	$x = 10; $x -= 8; // $x = 2
*=	คูณค่าเดิมของตัวแปรด้วยค่าที่ระบุ	$x = 10; $x *= 8; // $x = 80
/=	หารค่าเดิมของตัวแปรด้วยค่าที่ระบุ	$x = 16; $x /= 8; // $x = 2
%=	นำค่าที่ระบุไปหารค่าเดิมของตัวแปร แต่จะเอาเฉพาะเศษจากการหารเท่านั้น หรือเรียกว่าการหารแบบ Modulus	$x = 10; $x %= 3; // $x = 1
.=	ใช้ในการเชื่อมต่อข้อความโดยนำข้อความด้านขวามือไปต่อท้ายข้อความด้านซ้ายมือ	$x = "PHP"; $x .= "/MySQL"; // $x = "PHP/MySQL"

4) ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

ประกอบด้วยตัวดำเนินการดังต่อไปนี้

ตารางที่ 3.4 ตัวดำเนินการสำหรับเพิ่ม และลดค่า

สัญลักษณ์	ความหมาย	ตัวอย่าง
++	เพิ่มค่าในตัวแปรอีก 1 ค่า	$x = 10; $x++; // $x = 11
--	ลดค่าในตัวแปรลง 1 ค่า	$x = 10; $x--; // $x = 9

ข้อควรระวังเกี่ยวกับการใช้ตัวดำเนินการ ++ และ --

การวางตัวดำเนินการ ++ หรือ -- ไว้ด้านหน้า หรือหลังตัวแปร หากตัวแปรนั้นอยู่เดี่ยวๆ ค่าที่ได้จะไม่ต่างกัน เช่น กรณีตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.22 ข้อควรระวังการวางตัวดำเนินการ ++ หรือ -- ไว้ด้านหน้า หรือหลังตัวแปร

```
<?php
    $x = 10;
    $y = 10;
    $x++;           // ผลลัพธ์ คือ $x = 11
    ++$y;           // ผลลัพธ์ คือ $y = 11
?>
```

แต่หากนำไปใช้ในรูปแบบของนิพจน์ หรือกระทำกับค่าอื่นๆ ด้วย ค่าที่ได้อาจแตกต่างกันไป เช่น สองกรณีต่อไปนี้

ตัวอย่างที่ 3.23 การเปรียบเทียบรูปแบบของการใช้นิพจน์สองกรณี

$x = 10; $y = 20; $y += ++$x; // $x = 11, $y = 31 หากกำหนดแบบนี้ y จะมีค่าเท่ากับ (20+1)+10=31 นั่นคือ จะเพิ่มค่า x ขึ้นไปอีก 1 ก่อนแล้วค่อยนำไปบวกกับ y	$x = 10; $y = 20; $y += $x++; // $x = 11, $y = 30 หากกำหนดแบบนี้ y จะมีค่าเท่ากับ 20+10=30 นั่นคือ จะนำค่า x เดิมไปบวกกับ y ก่อน แล้วค่อยเพิ่มค่า x ขึ้นไปอีก 1

5) ตัวดำเนินการเปรียบเทียบ (Comparison)

ตัวดำเนินการเปรียบเทียบใช้ในการเปรียบเทียบหาค่าความจริงระหว่าง 2 นิพจน์ โดยผลลัพธ์ที่ได้จะเป็นได้เพียง true หรือ false อย่างใดอย่างหนึ่งเท่านั้น ตัวดำเนินการในกลุ่มนี้ มีดังนี้

ตารางที่ 3.5 ตัวดำเนินการเปรียบเทียบ

สัญลักษณ์	ความหมาย	สัญลักษณ์	ความหมาย
<	น้อยกว่า	==	เท่ากัน
<=	น้อยกว่า หรือเท่ากับ	===	เท่ากันทั้งหมดทั้งค่า และชนิดข้อมูล
>	มากกว่า	!=	ไม่เท่ากัน
=>	มากกว่า หรือเท่ากับ		

ตัวอย่างที่ 3.24 ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ

```
1   <?php
2       $a = (10 <= 9);
3       $b = (10 == 10);
4       $c = (10 == "10");
5       $d = (10 === "10");
6       $e = ("php" == "PHP");
7   ?>
```

จากตัวอย่างที่ 3.24 ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $a มีค่าเท่ากับ false (เพราะผลของการเปรียบเทียบค่า คือ 10 มีค่าน้อยกว่าหรือเท่ากับ 9 ไม่เป็นความจริง)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $b มีค่าเท่ากับ true (เพราะผลของการเปรียบเทียบค่า คือ 10 มีค่าเท่ากับ 10 เป็นความจริง)

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $c มีค่าเท่ากับ true (เพราะผลของการเปรียบเทียบเฉพาะค่า คือ 10 มีค่าเท่ากับ "10" เป็นความจริง)

บรรทัดที่ 5 กำหนดค่าให้กับตัวแปร $d มีค่าเท่ากับ false (เพราะผลของการเปรียบเทียบค่าและชนิด คือ 10 มีค่าเท่ากับ "10" ไม่เป็นความจริง คือ 10 ตัวแรกเป็นชนิดเลขจำนวนเต็ม ส่วน "10" เป็นข้อความ)

บรรทัดที่ 6 กำหนดค่าให้กับตัวแปร $e มีค่าเท่ากับ false (เพราะผลของการเปรียบเทียบชนิดของตัวพิมพ์ไม่เหมือนกัน จึงทำให้ไม่เป็นความจริง)

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

6) ตัวดำเนินการเปรียบเทียบทางตรรกะ (Bitwise Operators)

ตัวดำเนินการเปรียบเทียบทางตรรกะ เป็นการเปรียบเพื่อหาค่าความจริงระหว่าง 2 นิพจน์ เช่น หากพิสูจน์นิพจน์แรกเป็นจริง และนิพจน์ที่สองเป็นเท็จ แล้วผลลัพธ์จะออกมาเป็นอย่างไร เป็นต้น ตัวดำเนินการในการเปรียบเทียบทางตรรกะมีดังต่อไปนี้

ตารางที่ 3.6 ตัวดำเนินการเปรียบเทียบทางตรรกะ

ตัวดำเนินการ	การเปรียบเทียบ	ผลลัพธ์
! หรือ not	!(true)	true
	!(false)	false

ตารางที่ 3.6 (ต่อ)

ตัวดำเนินการ	การเปรียบเทียบ	ผลลัพธ์
&& หรือ and	true && true	true
	true && false	false
	false && false	false
|| หรือ or	true || true	true
	true || false	true
	false || false	false
^ หรือ xor	true ^ true	false
	true ^ false	true
	false ^ false	false

ตัวอย่างที่ 3.25 ตัวอย่างผลลักษณะการเปรียบเทียบระหว่าง 2 นิพจน์

```
1       <?php
2           $a = ! (1 == 2);
3           $b = (1 != 2) && (1>0);
4           $c = (1 == 2) && (1>0);
5           $d = (1 == 2) || (1>0);
6       ?>
```

จากตัวอย่างที่ 3.25 ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $a มีค่าเท่ากับ true เมื่อ $a = ! (false)

บรรทัดที่ 3 กำหนดให้ตัวแปร $b มีค่าเท่ากับ true เมื่อ $b = (true) && (true))

บรรทัดที่ 4 กำหนดให้ตัวแปร $c มีค่าเท่ากับ false เมื่อ $c = (false) && (true)

บรรทัดที่ 5 กำหนดให้กับตัวแปร $d มีค่าเท่ากับ true เมื่อ $d = (false) || (true)

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

7) ตัวดำเนินการอื่นๆ

ในภาษา PHP ยังมีตัวดำเนินการอื่นๆ ที่ไม่ถูกจัดให้อยู่ในประเภทใดเลย ประกอบด้วย 1) ตัวดำเนินการใช้เงื่อนไขในการกำหนดค่า (Ternary Operator) 2) ตัวดำเนินการระงับการเกิดข้อผิดพลาด (Error suppression Operator) และ 3) ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ (Execute Command Operator) และตัวดำเนินการในการเลือกค่า มีรายละเอียด ดังนี้

7.1) ตัวดำเนินการใช้เงื่อนไขในการกำหนดค่า

ตัวดำเนินการใช้เงื่อนไขในการกำหนดค่า คือ ตัวดำเนินการในการเปรียบเทียบเงื่อนไข ถ้าตรงกับเงื่อนไขผลลัพธ์จะเป็นอย่างไร และกรณีที่ไม่ตรงตามเงื่อนไขผลลัพธ์จะเป็นอย่างไร รูปแบบการใช้งาน ดังนี้

รูปแบบ

```
เงื่อนไข? x : y.
```

เมื่อ เงื่อนไข หมายถึง เงื่อนไขสำหรับการกำหนดค่า

x หมายถึง หากเงื่อนไขเป็นจริงจะมีค่าเท่ากับค่าที่กำหนดใน x

y หมายถึง หากเงื่อนไขเป็นเท็จจะมีค่าเท่ากับค่าที่กำหนดใน y

ตัวอย่างที่ 3.26 ตัวดำเนินการใช้เงื่อนไขในการกำหนดค่า

```
1   <?php
2       $a = (1 > 0) ? true : false;
3       $b = (10 % 2 == 0) ? "เลขคู่" : "เลขคี่";
4   ?>
```

จากตัวอย่างที่ 3.26 ตัวอย่างการใช้ตัวดำเนินการใช้เงื่อนไขในการกำหนดค่า อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $a มีค่าเท่ากับ true เพราะ (1 > 0) เป็นจริง

บรรทัดที่ 3 กำหนดให้ตัวแปร $b มีค่าเท่ากับ "เลขคู่" เมื่อ นำ 2 ไปหาเศษของผลหารเลข 10 มีค่าเท่ากับ 0

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

7.2) ตัวดำเนินการระงับการเกิดข้อผิดพลาด

ตัวดำเนินการระงับการเกิดข้อผิดพลาด แทนด้วยสัญลักษณ์ @ นำหน้าประโยคคำสั่ง หรือส่วนอื่นๆ ที่ต้องการเพื่อให้โปรแกรมมองข้ามข้อผิดพลาด (error) ที่อาจจะเกิดขึ้น มีตัวอย่างการใช้งานตัวดำเนินการ ดังนี้

ตัวอย่างที่ 3.27 ตัวอย่างการใช้ตัวดำเนินการระงับการเกิดข้อผิดพลาด

```
1   <?PHP
2           @include ("file.php");
3           echo "Suratthani Rajabhat University";
4   ?>
```

จากตัวอย่างที่ 3.27 ตัวอย่างการใช้ตัวดำเนินการระงับการเกิดข้อผิดพลาด อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 นำเข้าไฟล์ file.php โดยใช้ร่วมกับตัวดำเนินการระงับการเกิดข้อผิดพลาด ในกรณีที่ไม่มีไฟล์ file.php ที่จะนำเข้าจะแสดงข้อความดังนี้

```
Warning: include(file.php) [function.include]: failed to open stream: No such file or directory in C:\AppServ\www\temp\test2.php on line 2
Warning: include() [function.include]: Failed opening 'file.php' for inclusion (include_path='.;C:\php5\pear') in C:\AppServ\www\temp\test2.php on line 2
Suratthani Rajabhat University
```

จากข้อความด้านบนระบบจะแจ้งข้อความเตือนเมื่อไม่พบไฟล์ file.php เพื่อนำเข้ามาประมวลผลร่วม ในกรณีใช้ร่วมกับตัวดำเนินการระงับการเกิดข้อผิดพลาด จะแสดงผลตามปกติในบรรทัดที่ 3 โดยไม่ปรากฏข้อผิดพลาด

บรรทัดที่ 3 แสดงข้อความ Suratthani Rajabhat University

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

7.3) ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ

ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ โดยใช้สัญลักษณ์ ` (Grave Accent) แล้วตามด้วยคำสั่งภายนอกของระบบปฏิบัติการที่ต้องการใช้งาน (ทั้งนี้ขึ้นอยู่กับระบบปฏิบัติการที่ประมวลผลภาษา PHP) ตัวอย่างดังนี้

ตัวอย่างที่ 3.28 ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ MS-Windows

```
1   <?php
2       $listing = `dir /a c:`;
3       echo $listing;
4   ?>
```

จากตัวอย่างที่ 3.28 ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ MS-Windows เพื่อเรียกดูไฟล์ข้อมูล ผลลัพธ์ที่ได้อาจแตกต่างกันไปขึ้นอยู่กับโครงสร้างไฟล์และไดเร็กทอรีภายในเครื่องนั้นๆ

ตัวอย่างที่ 3.29 ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ Unix เรียกดูไฟล์ข้อมูล

```
1   <?php
2       $listing = `ls -al`;
3       echo $listing;
4   ?>
```

จากตัวอย่างที่ 3.29 ตัวดำเนินการเรียกใช้คำสั่งของระบบปฏิบัติการ Unix เพื่อเรียกดูไฟล์ข้อมูล ซึ่งผลลัพธ์จะคล้ายๆ กับตัวอย่างที่ 3.28 ทั้งนี้สามารถประยุกต์ใช้งานได้กับทุกระบบปฏิบัติการ

สรุป

ตัวแปรในภาษา PHP ไม่จำเป็นต้องกำหนดชนิดให้กับตัวแปรเนื่องจากชนิดของตัวแปรสามารถปรับเปลี่ยนได้โดยอัตโนมัติขึ้นอยู่กับการกำหนดค่าให้กับตัวแปร การกำหนดค่าให้กับตัวแปรสามารถกำหนดค่าได้หลายรูปแบบ เช่น กำหนดค่าเริ่มต้น กำหนดค่าคงที่ กำหนดจากการคำนวณ กำหนดจากผลการทำงานของฟังก์ชัน เป็นต้น ซึ่งการเขียนสคริปต์ PHP ส่วนใหญ่จะนิยมให้ระบบที่พัฒนามีความยืดหยุ่น ดังนั้นการกำหนดค่าให้กับตัวแปรมักจะมาจากการคำนวณ เพื่อให้ได้ผลลัพธ์ที่หลากหลายใช้กำหนดทางเลือกในการแสดงผลที่แตกต่างกันไปตามค่าที่ได้จากการคำนวณ สำหรับการคำนวณนั้นจำเป็นต้องทราบถึงสัญลักษณ์ของเครื่องหมายดำเนินการ ลำดับความสำคัญ และหน้าที่ แล้วนำมาดำเนินการระหว่างตัวแปรกับตัวแปร หรือ ตัวแปรกับค่าใดๆ แล้วนำผลลัพธ์ที่ได้จากการดำเนินการกำหนดค่าให้กับตัวแปร เพื่อนำตัวแปรนั้นๆ ไปประมวลผลในส่วนอื่นๆ ของระบบต่อไป

คำถามท้ายบท

1. จงอธิบายข้อกำหนดสำหรับประกาศใช้ตัวแปร พร้อมยกตัวอย่างประกอบ
2. จงอธิบายวิธีการกำหนดให้กับตัวแปรที่สำคัญ มีกี่วิธี ประกอบด้วยอะไรบ้าง พร้อมยกตัวอย่างประกอบในแต่ละวิธี
3. จงอธิบายขอบเขตของตัวแปร มีกี่ชนิด ประกอบด้วยชนิดอะไรบ้าง พร้อมยกตัวอย่างประกอบในแต่ละชนิด
4. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1   <?php
2       function number ($value=10) {
3           $value=$value*2;
4           return $value;
5       }
6       $num = 5;
7       $num = number ($num);
8       echo $num;
9   ?>
```

5. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1       <?php
2               $a = ! (1 == "1");
3               $b = (1 != 2) && (1>=0);
4               $c = (1 <= 2) && (1>=0);
5               $d = (1 === "1") || (1>0);
6       ?>
```