

A decorative graphic on the left side of the slide, consisting of a series of vertical and horizontal lines of varying lengths, some ending in small circles, resembling a circuit board or a stylized tree structure.

PHP2: FUNCTIONS

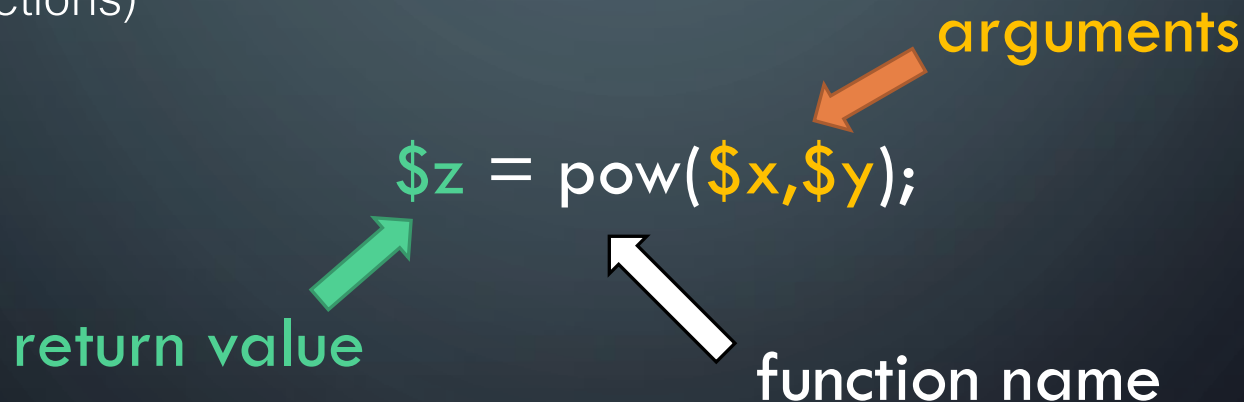
WRITTEN BY THAPANAPONG RUKKANCHANUNT

SOLUTION TO LAB 1

- Refer to DEMO

WHAT IS FUNCTION?

- คำสั่ง `pow(x,y)` ถือเป็นฟังก์ชันที่มากับภาษา PHP ซึ่งเราจะเรียกฟังก์ชันแบบนี้ว่าเป็น Built-in Functions (ภาษา PHP มีมากกว่า 1000 Built-in Functions)

The diagram shows the code `$z = pow($x,$y);` with three annotations. A green arrow points from the text 'return value' to the variable `$z`. A white arrow points from the text 'function name' to the function `pow`. An orange arrow points from the text 'arguments' to the variables `$x` and `$y` inside the parentheses.

```
$z = pow($x,$y);
```

return value

function name

arguments

USEFUL MATH FUNCTIONS

Function	Description
<code>abs(\$x)</code>	ฟังก์ชันหาค่าสัมบูรณ์
<code>sin(\$x)</code> <code>cos(\$x)</code> <code>tan(\$x)</code>	ฟังก์ชันตรีโกณมิติ
<code>ceil(\$x)</code> <code>floor(\$x)</code>	ฟังก์ชันปัดขึ้น ปัดลง
<code>exp(\$x)</code>	ฟังก์ชันหาเลขยกกำลังฐาน e (e^x)
<code>log(\$x)</code>	ฟังก์ชันหาลอการิทึมฐาน e ($\ln x$)
<code>pi()</code>	ฟังก์ชันที่ให้ค่าพาย (π)
<code>rand(\$min,\$max)</code>	ฟังก์ชันสุ่มจำนวนเต็มที่มีค่าเป็นไปได้อย่างตั้งแต่ $\$min$ ถึง $\$max$
<code>sqrt(\$x)</code>	ฟังก์ชันหารากที่สอง
<code>pow(\$x,\$y)</code>	ฟังก์ชันหาเลขยกกำลัง (x^y)

USER-DEFINED FUNCTIONS

- เราสามารถสร้าง function มาใช้งานเองได้ โดยจะต้องมีการประกาศฟังก์ชันตามโครงสร้างด้านล่าง

```
function functionName ()  
{  
    code to executed;  
}
```

EXAMPLE OF FUNCTION

```
<?php
```

```
function hello() {
```

```
    echo "Hello World";
```

```
}
```

```
hello();      // จะแสดงผลออกทางหน้าจอเป็นคำว่า Hello World
```

```
?>
```

FUNCTION ARGUMENTS

- เราสามารถส่งข้อมูลไปยังฟังก์ชันได้ โดยข้อมูลที่ส่งไปจะเรียกว่า argument ซึ่งก็คือตัวแปรนั่นเอง โดยข้อมูลหนึ่งอย่างจะใช้ argument หนึ่งตัวในการส่ง หากต้องการส่งข้อมูลหลายอย่างให้ใช้ comma คั่นระหว่าง argument

EXAMPLE OF FUNCTION (2)

```
<?php  
function hello($name) {  
    echo "Hello $name";  
}  
  
hello("Joe"); // จะแสดงผลออกทางหน้าจอเป็นคำว่า Hello Joe  
?>
```


EXAMPLE OF FUNCTION (3)

```
<?php
```

```
function hello($name, $age) {
```

```
    echo "Hello $name. You are $age years old.";
```

```
}
```

```
hello("Joe",56);
```

```
// จะแสดงผลออกทางหน้าจอเป็นคำว่า Hello Joe. You are 56 years old.
```

```
?>
```

FUNCTIONS WITH DEFAULT ARGUMENT VALUE

- เราสามารถกำหนดค่าเริ่มต้น (default value) ให้กับ argument ได้ โดยถ้าผู้ใช้ไม่ได้ระบุค่าของ argument ฟังก์ชันก็จะใช้ค่า default value

```
function hello($name = "Joe") {  
    echo "Hello $name";  
}
```

```
hello();           //      Hello Joe
```

```
hello("Christ");   //      Hello Christ
```

RETURN VALUE

- นอกจากเราจะส่งข้อมูลให้ฟังก์ชันแล้ว เรายังสามารถให้ฟังก์ชันส่งข้อมูลกลับออกมาได้ โดยใช้คำสั่ง `return`

```
function functionName ()  
{  
    code to executed;  
    return $value;  
}
```

EXAMPLE OF FUNCTION (4)

```
<?php
function square($x) {
    $z = $x * $x;
    return $z;
}
echo "5^2 = " . square(5);           // 5^2 = 25
echo "7^4 = " . square(square(7));  // 7^4 = 2401
?>
```

VARIABLE SCOPE

- ข้อควรระวังในการประกาศใช้ฟังก์ชันคือขอบเขตของตัวแปร ตัวแปรที่ใช้ในฟังก์ชันจะถูกทำลายหลังจากที่เสร็จสิ้นการทำงานของฟังก์ชันนั้น ๆ ดังนั้น เราไม่สามารถนำตัวแปรนั้นมาใช้ใหม่ได้

EXAMPLE OF FUNCTION (5)

```
<?php
function square($x) {
    $z = $x * $x;
    return $z;
}
echo "5^2 = " . square(5);      // 5^2 = 25
echo "5^2 = $z";               // ERROR
?>
```

EXAMPLE OF FUNCTION (6)

```
<?php
```

```
function square($x) {
```

```
    $z = $x * $x;
```

```
    return $z;
```

```
}
```

```
$z = square(5);
```

```
echo "5^2 = $z";
```

```
?>
```

// ตัวแปร \$z ตัวนี้เป็นคนละตัวกับในฟังก์ชัน

// 5^2 = 25

AREA 51

- Refer to Lab Sheet 2
- Hand-in **ONLINE** before 2nd September 23:59