영 진 전 문 대 학 교

글로벌시스템융합과 - GSC

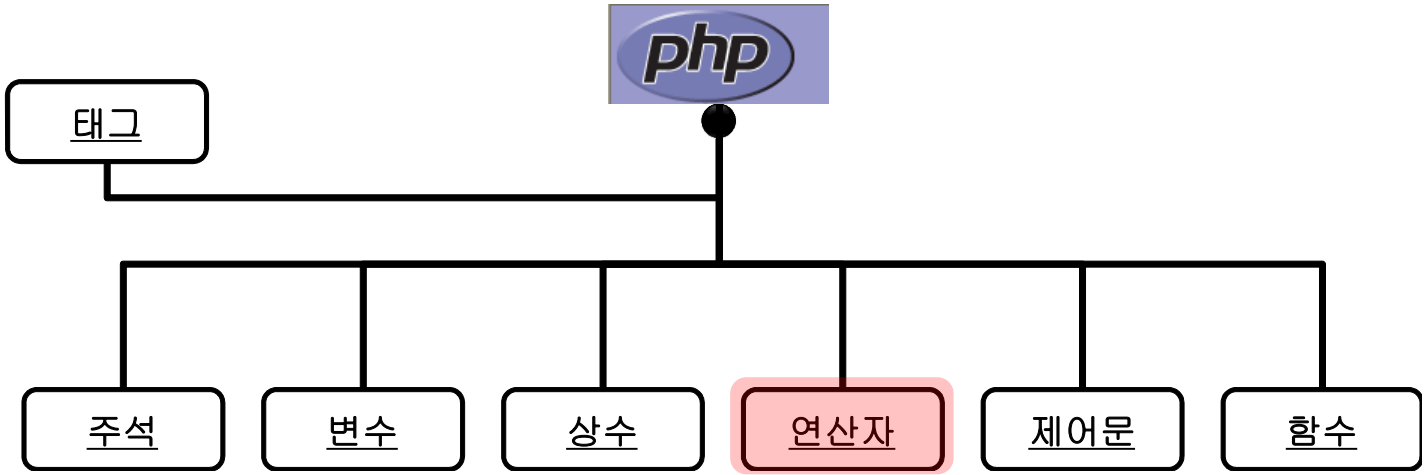
PHP – Operator, Flow control and Function

정영철 교수

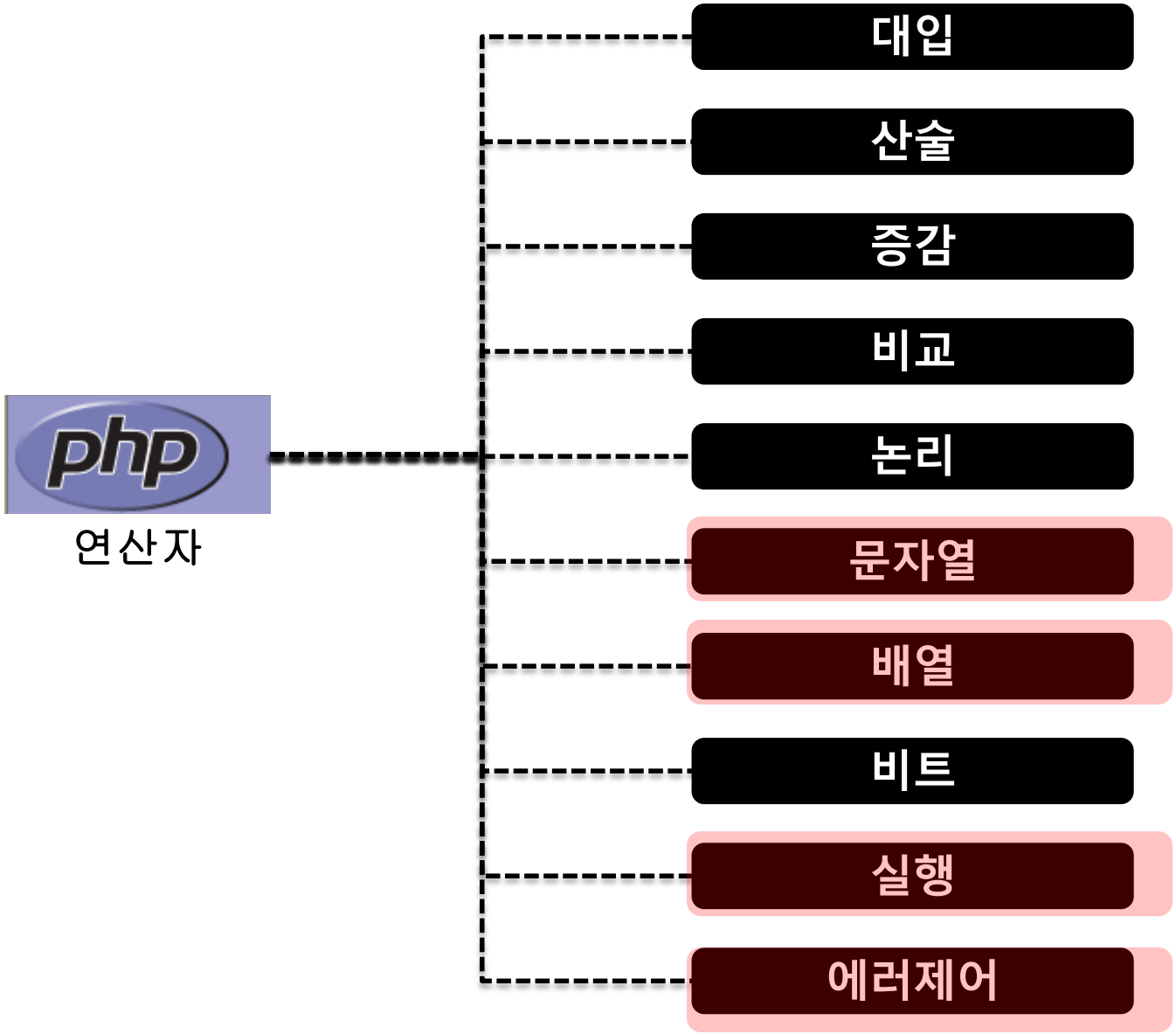
글로벌시스템융합과

A.I WITH BETTER LIFE

연산자 (Operator)



PHP에서 사용되는 연산자 종류



산술 연산자 (Arithmetic Operator) (1)

Arithmetic Operators

Example	Name	Result
+$a	Identity	Conversion of $a to int or float as appropriate.
-$a	Negation	Opposite of $a.
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulo	Remainder of $a divided by $b.
$a ** $b	Exponentiation	Result of raising $a to the $b'th power. Introduced in PHP 5.6.

산술 연산자 (Arithmetic Operator) (2)

- 산술 연산 후 결과 값은 integer or float
- 피연산자가 문자열 인 경우, integer or float으로 형 변환
  - 문자열 숫자로 변경 불가 시 0 리턴, <- 문자열의 첫 번째 글자가 Digit이 아닌 경우

	integer	float	string	boolean	null
integer	integer	float	integer	integer	integer
float	float	float	float	float	float
string	integer	float	integer	integer	integer
boolean	integer	float	integer	integer	integer
null	integer	float	integer	integer	integer

산술 연산자 : 예제

```
8   // integer + float -> float
9   $result = $iValue + $fValue;
10  var_dump($result); // float(2)
11
12  // integer + boolean -> integer, true는 false는 0으로 취급
13  $result = $iValue + $bValue;
14  var_dump($result); // int(2)
15
16  // integer + string -> integer,
17  // 문자열 첫 글자가 숫자일 경우 파싱 가능한 범위 내 번역 후 정수 반환
18  // 문자열 첫 글자가 문자일 경우 0반환
19  $result = $iValue + $strValue;
20  var_dump($result); // int(2)
21  $result = $iValue + "1234god";
22  var_dump($result); // int(1235)
23  $result = $iValue + "god111";
24  var_dump($result); // int(1)
25  $result = $iValue + "1.6";
26  var_dump($result); // float(2.6)
27
28  // integer + null -> integer, null은 0으로 취급
29  $result = $iValue + $nullValue;
30  var_dump($result); // int(1)
```

```
2   $iValue = 1;
3   $fValue = 1.0;
4   $bValue =  true;
5   $strValue =  "1";
6   $nullValue = null;
```

비교 연산자 (Comparison operators)

Operators		
Example	**Name**	**Result**
$a == $b	Equal	TRUE if $a is equal to $b after type juggling.
$a === $b	Identical	TRUE if $a is equal to $b, and they are of the same type.
$a != $b	Not equal	TRUE if $a is not equal to $b after type juggling.
$a <> $b	Not equal	TRUE if $a is not equal to $b after type juggling.
$a !== $b	Not identical	TRUE if $a is not equal to $b, or they are not of the same type.
$a < $b	Less than	TRUE if $a is strictly less than $b.
$a > $b	Greater than	TRUE if $a is strictly greater than $b.
$a <= $b	Less than or equal to	TRUE if $a is less than or equal to $b.
$a >= $b	Greater than or equal to	TRUE if $a is greater than or equal to $b.
$a <=> $b	Spaceship	An integer less than, equal to, or greater than zero when $a is less than, equal to, or greater than $b, respectively. Available as of PHP 7.

Type juggling precedence : float > integer

문자열의 경우 float or integer로 형 변환

비교 연산자 (Comparison operators)

```php
$iValue_1      = 1;
$iValue_2      = 2;
$fValue_3      = 3.0;
$strValue_1    = "1";
$bValue        = true;

echo (($iValue_1 == $strValue_1) ? "true"  : "false")."<br>";   // true
echo (($iValue_1 === $strValue_1) ? "true"  : "false")."<br>"; // false


echo (($iValue_1 <> $fValue_3) ? "true"  : "false")."<br>"; // true
echo (($iValue_1 <> $bValue) ? "true"  : "false")."<br>"; // false

echo (($iValue_1 !== $bValue) ? "true"  : "false")."<br>"; // true
echo (($iValue_1 !== $fValue_3) ? "true"  : "false")."<br>"; // true

echo 1 <=> 1;  // 0
echo 1 <=> 2;  // -1
echo 2 <=> 1;  // 1
```

증감 연산자 (Incrementing/Decrementing operators)

```php
$value = 1;

echo ++$value."<br>"; // 2
echo $value++."<br>"; // 2
echo $value."<br>";     // 3

$value = "2";
echo ++$value."<br>"; // 3
echo --$value."<br>";   // 2

$value = null;
echo --$value;    // No effect, 영향 없음
                  // Array, Object, Boolean의 경우 증감 연산자 실행 안됨.
echo ++$value;  // 1
```

논리 연산자 (Logical operators) (1)

- JAVA와 달리 and, or, xor 추가
- &&, || ↔ and, or 차이 : 기능은 동일 단 우선 순위 다름
  - and, or 연산자가 PHP 전체 연산자 종류 중 우선 순위가 가장 낮음
  - 논리 연산식을 표현하기 위해서 추가됨

Logical Operators		
Example	**Name**	**Result**
$a and $b	And	TRUE if both $a and $b are TRUE.
$a or $b	Or	TRUE if either $a or $b is TRUE.
$a xor $b	Xor	TRUE if either $a or $b is TRUE, but not both.
! $a	Not	TRUE if $a is not TRUE.
$a && $b	And	TRUE if both $a and $b are TRUE.
$a || $b	Or	TRUE if either $a or $b is TRUE.

논리 연산자 (Logical operators) (2)

```php
$value = 1;

if( 2 > 3 && ++$value > 1)
    echo "true"."<br>";
echo $value."<br>";

if( 2 < 3 || ++$value > 1)
    echo "true"."<br>";
echo $value;
// 실행 결과 값?

// xor input 값 둘 중 하나만 참 일때만 true
echo (false xor false) ? "true" : "false";
echo (true xor false) ? "true" : "false";
echo (false xor true) ? "true" : "false";
echo (true xor true) ? "true" : "false";

$g = true && false; // false
$h = true and false; // true
var_dump($g, $h);
```

XOR truth table

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

논리 연산자 (Logical operators) (2)

```php
$result = true and false;
var_dump($result);  // bool(true)
```

```php
$result = true && false;
var_dump($result);  // bool(false)
```

문자열 연산자 (String operator)

- 문자열을 서로 이어주는 역할

```php
<?
    $a = "안녕";
    $b = "하세요";

    $c = $a.$b;

    echo $c;

    echo "<br>";

    $c .= " 여러분~~";

    echo $c;
?>
```

```
안녕하세요
안녕하세요 여러분~~
```

실행 연산자 (Execution Operator) (1)

- 실행연산자(`)을 이용하여 CLI 명령을 실행할 수 있다.

```php
<?
   $output = `ls `;

   echo $output;
?>
```

```
1.1.php 1_id.php 2.php 3.php 4.1.php 4.2.php 4.3.php 4.php 5.php index.html phpinfo.php
```

실행 연산자 (Execution Operator) (2)

```php
$result = `dir`; // 시스템 명령어 실행 "dir"
// 문자열 내 HTML 태그 -> HTML 기호로 변경
$result = htmlspecialchars_decode($result);
// 개행문자 <br>로 변경
$result = nl2br($result);
$output = <<<"HTMLEND"
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    $result
</body>
</html>
HTMLEND;

echo $output;
```

```
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: F8BE-8031

C:₩AutoSet9₩public_html₩test 디렉터리

2020-05-26 오후 08:20

.
2020-05-26 오후 08:20
..
2020-05-19 오후 09:35 227 1.php
2020-05-25 오후 07:16 364 10.php
2020-05-26 오후 06:07 822 11.php
2020-05-26 오후 07:09 624 12.php
2020-05-26 오후 07:39 498 13.php
```

에러제어 연산자 (1)

- 에러가 발행할 수 있는 곳에 @ 기호를 붙여 에러를 출력하지 않게 한다.

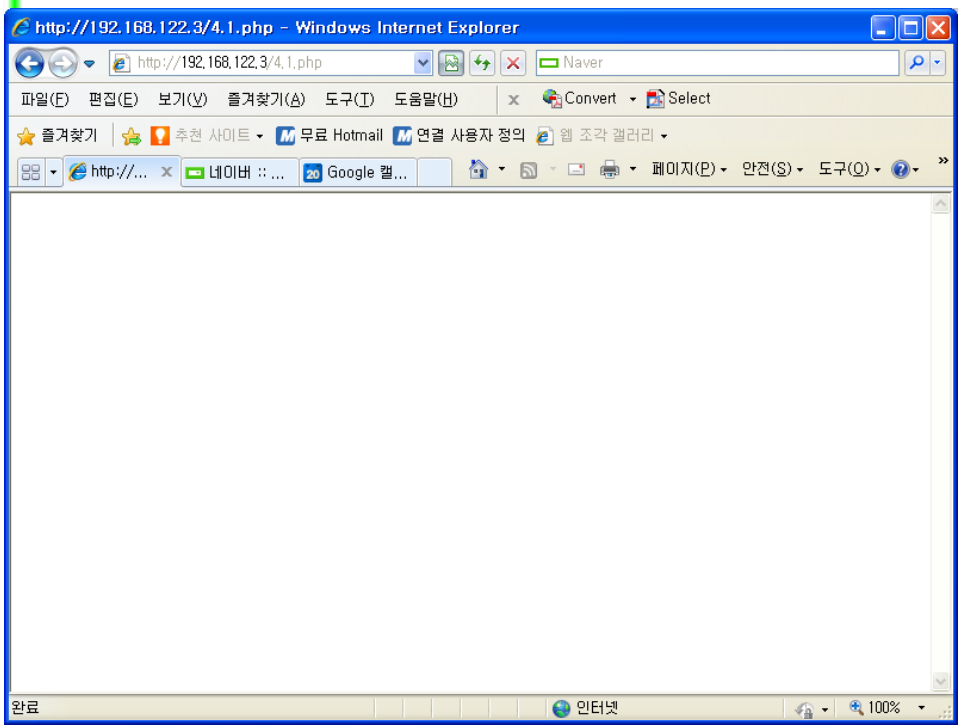
```php
<?
   include "test.php";
?>
```

Warning: include(test.php) [function.include]: failed to open stream: No such file or directory in **/usr/local/server/apache/htdocs/4.1.php** on line 2

Warning: include() [function.include]: Failed opening 'test.php' for inclusion (include_path='.:/usr/local/server/php/lib/php') in **/usr/local/server/apache/htdocs/4.1.php** on line 2

에러제어 연산자 (2)

```
<?
  @include "test.php";
?>
```



Array 연산자 : Array 개념 (1)

- **PHP에서 Array는 (key, value)로 구성: [순서 보장 HashMap]**
  - Key가 JAVA array의 index와 같음
  - JAVA와 달리 index 값이 integer와 문자열 값을 가질 수 있다.
  - Key 값이 생략 될 경우 integer로 할당
  - Key 값과 value 값의 매칭을 위해 화살표 연산자 제공 : =>

- **JAVA와 달리 하나의 배열 내 value 값의 자료형 다형 가능**

- **Array는 리터럴 상수 [ ], array() 함수를 이용해서 생성**

Array 연산자 : Array 개념 (2)

```php
// 배열 생성 : 리터럴 상수 이용
$valueList_1 = [1, 2, 3];
var_dump($valueList_1); //array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }

// 배열 생성 : array 함수 이용
$valueList_2 = array(4, 5, 6);
var_dump($valueList_2); // array(3) { [0]=> int(4) [1]=> int(5) [2]=> int(6) }

// 원소 값들의 다형성
$valueList_3 = [1, "4", true, 2.3];
var_dump($valueList_3); //array(4) { [0]=> int(1) [1]=> string(1) "4" [2]=> bool(true) [3]=> float(2.3) }

$valueList_4 = ["first" => 1, "Second" => 2, 3, 4];
var_dump($valueList_4); // array(4) { ["first"]=> int(1) ["Second"]=> int(2) [0]=> int(3) [1]=> int(4) }

// count() : 배열 내 원소 갯수 반환
echo count($valueList_4);

// 배열 내 원소 순회
for($i = 0 ; $i < count($valueList_1) ; $i++)
    echo $valueList_1[$i]; // 1, 2, 3

foreach($valueList_4 as $key => $value)
    echo $key."=>".$value."<br>"; //first=>1, Second=>2, 0=>3, 1=>4
```

Array 연산자 : 연산자 종류

Operators		
Example	**Name**	**Result**
$a + $b	Union	Union of $a and $b.
$a == $b	Equality	TRUE if $a and $b have the same key/value pairs.
$a === $b	Identity	TRUE if $a and $b have the same key/value pairs in the same order and of the same types.
$a != $b	Inequality	TRUE if $a is not equal to $b.
$a <> $b	Inequality	TRUE if $a is not equal to $b.
$a !== $b	Non-identity	TRUE if $a is not identical to $b.

Array 연산자 : +

```php
$value_1 = [1, 2, 3];
$value_2 = [3, 4, 5];

// 배열 + 배열 = 배열의 Key(키) 값을 기준, 좌항에 우항을 더한 배열 생성
// 중복 Key 값이 있을 경우 좌항 값 선택
$result = $value_1 + $value_2;
var_dump($result); // array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }

$value_3 = [1, 2, 3];
$value_4 = [3, 4, 5, 6];
$result = $value_3 + $value_4;
var_dump($result);// array(4) { [0]=> int(1) [1]=> int(2) [2]=> int(3) [3]=> int(6) }

$value_5 = [1, 3=>2, 3];
$value_6 = [3, 4, 5, 6];
$result = $value_5 + $value_6;
var_dump($result); // array(5) { [0]=> int(1) [3]=> int(2) [4]=> int(3) [1]=> int(4) [2]=> int(5) }
```

Array 연산자 : ==, ===

```php
$value_1 = [1, 2, 3];                    // array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
$value_2 = [1, 2, 3];                    // array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
$value_3 = [3, 2, 1];                    // array(3) { [0]=> int(3) [1]=> int(2) [2]=> int(1) }
$value_4 = [2 => 3,  1 => 2, 0 => 1]; //  array(3) { [2]=> int(3) [1]=> int(2) [0]=> int(1) }

// == 같은 key/value 값을 가지고 있을 경우 true
echo ($value_1 == $value_2) ? "true" : "false"; // true
echo ($value_1 == $value_3) ? "true" : "false"; // false
echo ($value_1 == $value_4) ? "true" : "false"; // true

// == 같은 key/value 값을 가지고 있고, 입력 순서도 동일 할 때 true
echo ($value_1 === $value_2) ? "true" : "false"; // true
echo ($value_1 === $value_3) ? "true" : "false"; // false
echo ($value_1 === $value_4) ? "true" : "false"; // false
```

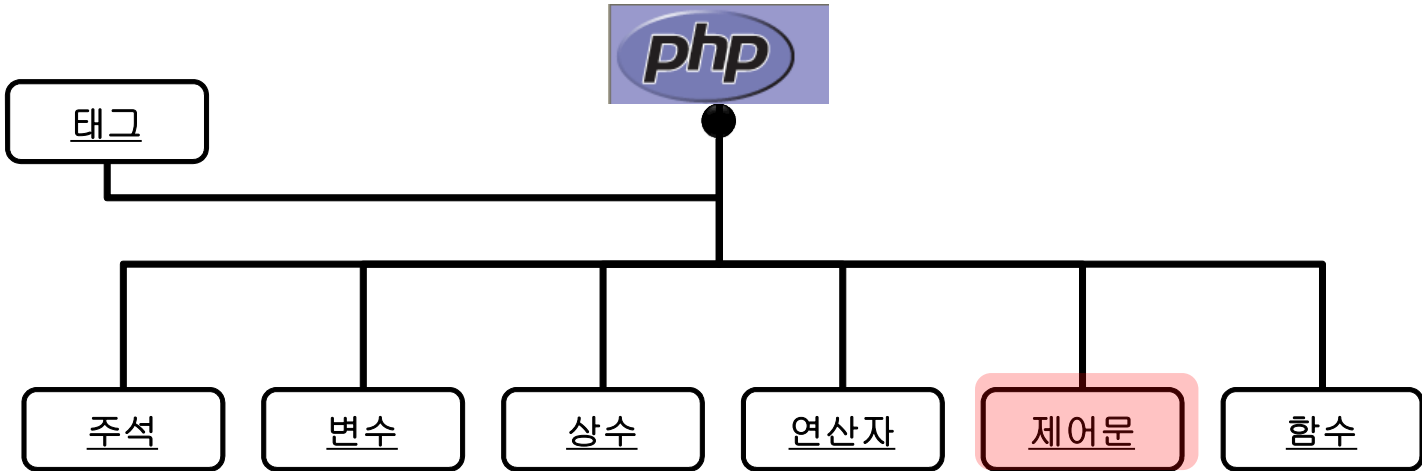
Array 연산자 : !=, !==

```php
$value_1 = [1, 2, 3];                    // array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
$value_2 = [1, 2, 3];                    // array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
$value_3 = [3, 2, 1];                    // array(3) { [0]=> int(3) [1]=> int(2) [2]=> int(1) }
$value_4 = [2 => 3,  1 => 2, 0 => 1]; //  array(3) { [2]=> int(3) [1]=> int(2) [0]=> int(1) }

// != 좌항, 우항 key/value 값 기준 원소값이 틀릴 경우
echo ($value_1 != $value_2) ? "true" : "false"; // false
echo ($value_1 != $value_3) ? "true" : "false"; // true
echo ($value_1 != $value_4) ? "true" : "false"; // false

// !== 좌항, 우항 key/value 값 기준 원소값이 틀리거나, 입력 순서가 일치하지 않을 때 true
echo ($value_1 !== $value_2) ? "true" : "false"; //false
echo ($value_1 !== $value_3) ? "true" : "false"; // true
echo ($value_1 !== $value_4) ? "true" : "false"; // true
```

흐름제어 (Flow control)



흐름제어

흐름제어

- 기본
- 선택문
  - if
  - switch
- 반복문
  - for
  - while
  - do while
  - foreach
- Alternative syntax
- include, require
  include_once, require_once

foreach (1)

- **Array (배열)의 원소, Object의 멤버를 순회할 때 사용**
- **Associated key 값을 가지는 PHP 배열에 적합**

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

```
3   $myList = ["first" => 10, "second" => 20, 30, 40];
4
5   foreach ($myList as $key => $value)
6       echo "myList[{$key}] : {$value} <br>";
```

myList[first] : 10
myList[second] : 20
myList[0] : 30
myList[1] : 40

foreach (2)

```php
class Triangle {
    public $width;
    private $height;
    public function __construct($argWidth,  $argHeight)
    {
        $this->width = $argWidth;
        $this->height = $argHeight;
    }
    public function getDimenstion() {
        return ($this->width * $this->height) / 2;
    }
}

$myTObj = new Triangle( argWidth: 4,  argHeight: 3);

foreach ($myTObj as $member => $value) {
    echo $member." : ".$value."<br>";
} // width : 4
```

Alternative syntax (1)

- 흐름 제어에 따른 HTML 문장 출력 시 효율적
- if, while, for, foreach, switch에 사용가능
  - If, while, for, foreach, switch ":" 로 시작
  - *endif;, endwhile;, endfor;, endforeach;, endswitch; 로 종료*

```
1   <?php
2   $inputValue = rand(0, 1); // 0 ~ 1사이 정수 난수 발생, 즉 0, 1 난수 발생
3   ?>
4
5   <?php if($inputValue == 1): ?>
6       <p>1번 선택 </p>
7       <h1> 안녕하세요, 만나서 반갑습니다. </h1>
8   <?php else: ?>
9       <p>The value excepting 1 is selected </p>
10      <h1> Hello, Nice to meet you!!. </h1>
11  <?php endif; ?>
12
13
```

Alternative syntax (2)

```
11  <table>
12      <tr>
13          <td>순번</td>
14          <td>학번</td>
15          <td>이름</td>
16      </tr>
17  <?php
18      $stdList = [10 => "최채흥", 20 => "원태인", 30 => "오승환"];
19  ?>
20  <?php
21  $i = 1; foreach($stdList as $key => $value):?>
22      <tr>
23          <th><?php echo $i; ?></th>
24          <th><?php echo $key; ?></th>
25          <th><?php echo $value ?></th>
26      </tr>
27  <?php $i++; endforeach; ?>
28  </table>
```

순번	학번	이름
1	10	최채흥
2	20	원태인
3	30	오승환

include, require, include_once, require_once

- 자주 사용되는 코드들을 하나의 독립 파일(모듈)로 관리
- 특정 파일을 현 코드 내 삽입 할 경우 아래 명령어 사용

순번	명령어	사용법	설명
1	include	include 'test.php'; include ('test.php');	• test.php 파일을 현재 위치에 삽입 • 파일을 찾지 못할 경우 Warning
2	require	require 'test.php'; require ('test.php');	• test.php 파일을 현재 위치에 삽입 • 파일을 찾지 못할 경우 Error
3	include_once	include_once 'test.php'; include_once ('test.php');	• 1번과 동일 • 중복 삽입 방지, 이전 코드 라인에 "test.php"가 호출되었을 경우, Skip
4	require_once	require_once 'test.php'; require_once ('test.php');	• 2번과 동일 • 중복 삽입 방지, 이전 코드 라인에 "test.php"가 호출되었을 경우, Skip

구분	include	require
파일 없을 경우	• 경고(E_WARNING), 계속 실행	• 치명적 오류(E_ERROR), 즉시 종료
사용 예시	• 선택적 모듈	• 핵심 구성요소
_once 버전	• 중복 포함 방지용 (include_once, require_once)	

include 예제

my_util.php

```php
<?php
function sum ($argA, $argB) {
    return $argA + $argB;
}
?>
```

main.php

```php
<?php
include ('my_util.php');

echo sum( argA: 2, argB: 3);
?>
```

main.php – Error case

```php
<?php
include ('my_util.php_error');

echo "Check point";

echo sum( argA: 2, argB: 3);
?>
```

Warning: include(my_util

Warning: include(): Failec
Check point
Fatal error: Call to undef

require 예제

my_util.php

```php
<?php
function sum ($argA, $argB) {
    return $argA + $argB;
}
?>
```

main.php

```php
<?php
require ('my_util.php');

echo sum( argA: 2, argB: 3);
?>
```

main.php – Error case

```php
<?php
require ('my_util.php_error');

echo "Check point";

echo sum( argA: 2, argB: 3);
?>
```

Warning: require(my_util.php

Fatal error: require(): Failed o

require_once 예제

my_util.php

```php
<?php
function sum ($argA, $argB) {
    return $argA + $argB;
}
?>
```

main.php

```php
<?php
require_once ('my_util.php');
require_once ('my_util.php');

echo sum( argA: 2, argB: 3);
?>
```

main.php – Error case

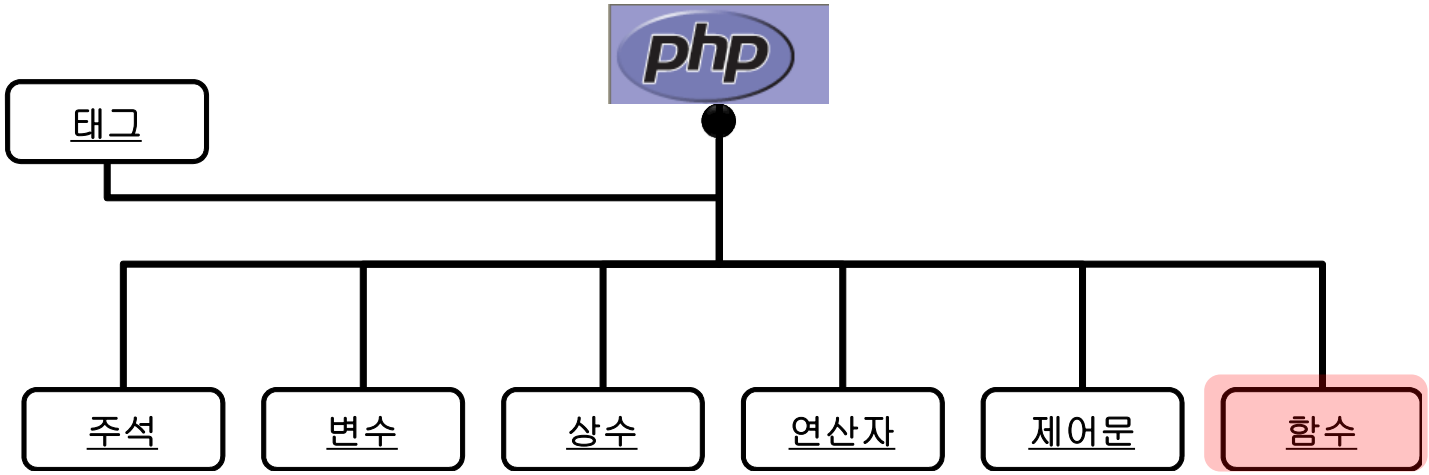
```php
<?php
require ('my_util.php');
require ('my_util.php');

echo sum( argA: 2, argB: 3);
?>
```

Fatal error: Cannot redeclare sum()

include_once도 require_once 와 동일
단 Warning 메시지만 발생

함수 (Function)



함수의 선언

- PHP 함수 정의 법
  - Function declaration
  - Function expression (Anonymous Function)

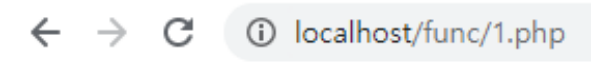
```php
// 함수 선언식 (Function declaration)
function println($msg) {
    echo $msg."<br>";
}

// 함수 표현식(Function expression), 익명함수(Anonymous function)
$print = function ($msg) {
    echo $msg;
};

println( msg: "Youngchul Jung");   // Youngchul Jung
print("Richard Jung");             // Richard Jung
```

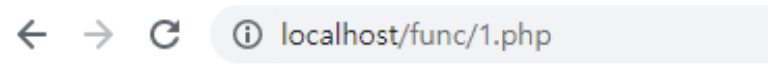
실습 : 아래 함수를 완성 하시오 (4)

```php
63  $myList     = setRandIntNum(4, 1, 10., false);
64  $sum        = sum($myList);
65  $average    = average($myList);
66
67  echo "발생된 난수 값 : ";
68  foreach ($myList as $value)
69      echo $value."  ";
70  echo "<br>";
71  echo " 난수 총 합 : $sum <br>";
72  echo " 난수 평균  : $average <br>";
```



발생된 난수 값 : 4 2 9 3
난수 총 합 : 18
난수 평균 : 4.5

```php
63  $myList     = setRandIntNum(7, 5, 10, true);
64  $sum        = sum($myList);
65  $average    = average($myList);
66
67  echo "발생된 난수 값 : ";
68  foreach ($myList as $value)
69      echo $value."  ";
70  echo "<br>";
71  echo " 난수 총 합 : $sum <br>";
72  echo " 난수 평균  : $average <br>";
```



발생된 난수 값 : 7 7 10 8 10 7 9
난수 총 합 : 58
난수 평균 : 8.2857142857143

Parameter : Call-By-Value and Call-By-Reference (1)

- **함수/메서드 호출 시 매개변수에 값을 전달하는 방식**
  - Call-By-Reference : 매개변수에 주소 값이 전달 됨
  - Call-By-Value : 매개변수에 값이 복사 되어 전달 됨

```php
class Foo {
    public $value = 2;
}
$myObj1     = new Foo();
$value1     = 4;

function changeObjValue($argObj, $argValue) {
    $argObj->value = $argValue;
    $argValue++;
}

// $myObj1 : Call-by-reference
// $value1 : Call-by-value
changeObjValue($myObj1, $value1);

echo $myObj1->value." : ".$value1."<br>"; // 4 : 4
```

Parameter : Call-By-Value and Call-By-Reference (2)

- Primitive variable : call-by-value
- Object : call-by-reference
- Array : call-by-value

```
27  $myArray = [10, 1, 2, 3];
28
29  function changeArrayValue($argArray, $argIndex, $argValue) {
30      // $argArray는 복사된 배열 값이다.
31      $argArray[$argIndex] = $argValue;
32  }
33  // PHP의 함수 특징 : 배열의 경우 Call-By-Value로 전달된다.
34  changeArrayValue($myArray, 0, 0);
35
36  var_dump($myArray); // 10, 1, 2, 3
```

Parameter : & Operator (1)

- 단항 연산자, 우항(원시 변수 또는 객체)의 메모리 주소(*정확히 메모리 주소는 아님*) 값 반환
- call-by-value에서 call-by-reference로 전환 가능

```php
$value_1 = 1;

function increment($argValue) {
    $argValue++;
}

//  $value1 : Call-By-Value
increment($value_1);

echo $value_1."<br>"; // 1
```

```php
$value_1 = 1;

// $argValue변수에 & 연산자 추가
// $argValue변수에 인자 값의 메모리 주소값 저장
function increment(&$argValue) {
    $argValue++;
}

//  $value1 : Call-By-reference
increment($value_1);

echo $value_1."<br>"; // 2
```

Parameter : & Operator (2)

```php
$myArray = [10, 1, 2, 3];

// $argArray : Call-By-Reference
function changeArrayValue(&$argArray, $argIndex, $argValue) {
    $argArray[$argIndex] = $argValue;
}
changeArrayValue($myArray, 0, 0);

var_dump($myArray); // 0, 1, 2, 3
```

PHP 함수 특징

- **PHP 함수 특징**
  - ✓ First-class citizen
  - ✓ Anonymous function 지원
  - ✓ Hoisting 지원
  - ✓ Overloading 지원 안함
  - ✓ Inner function (?)지원
  - ✓ Variable functions 지원
  - ✓ Arrow function & Closure 지원

함수 특징 : First-Class Citizen

```php
// 함수의 주소 값을 변수에 저장
$sum = function ($argA, $argB) {
    echo    "$argA + $argB = ".($argA + $argB)."<br>";
};

// 함수를 매개 변수로 전달
function foo($argFunc) {
    // sum(2,3) 실행
    $argFunc(2, 3);

    // 함수를 반환 값으로 사용
    return $argFunc;
}

$sum_d = foo($sum);

// sum(2,3) 실행
$sum_d(5, 10);
```

2 + 3 = 5
5 + 10 = 15

함수 특징 : Anonymous Function 지원

```
3   // 익명함수 선언 후 $println 변수에 저장
4   $println = function ($argMsg) { echo $argMsg."<br>";};
5
6   function foo ($argFunc, $argA, $argB) {
7       global $println;
8       $println( $argFunc($argA, $argB) );
9   }
10
11  foo(
12      // 익명함수 선언 후 foo 함수 매개변수로 전달
13      function($argA, $argB) {
14          return $argA + $argB;
15          }
16          , 2, 3); // --> 5
```

함수 특징 : Hoisting 지원

```php
// Function hoisting 지원
foo(); // --> hello;

function foo() {
    echo "hello<br>";
}

// 주의!! 변수는 Hoisting을 지원하지 않는다.

echo $value; // Warning! Undefined variable

$value = 3;
```

함수 특징 : Overloading 미 지원 (1)

```php
function foo($argA) {
    echo $argA;
}

// Fatal error: Cannot redeclare foo()
function foo($argA, $argB) {
    echo $argA.":".$argB;
}

foo("Youngchul");
foo("Youngchul", "Jung");
```

함수 특징 : Overloading 미 지원 (2)

```php
function sum()
{
    $argNum = func_num_args(); // 현 실행 함수의 매개변수 갯수 반환
    echo "매개변수 갯수 : " . $argNum . "<br>";

    $argList = func_get_args(); // 현 실행 함수의 매개변수를 배열로 반환
    $result = 0;
    foreach($argList as $value)
        $result += $value;

    return $result;
}

$result_1 = sum(1, 2);      // 매개변수 갯수 : 2
$result_2 = sum(1, 2, 3);   // 매개변수 갯수 : 3
$result_3 = sum(1, 2, 3, 4);// 매개변수 갯수 : 4

echo $result_1."<br>"; // 3
echo $result_2."<br>"; // 6
echo $result_3."<br>"; // 10
```

https://www.php.net/manual/en/ref.funchand.php

함수 특징 : Inner function (?)지원

```
3   bar(); // Error
4
5   foo(); // 1
6
7   // foor() 함수 실행 시 bar 함수가 정의 되고,
8   // 함수는 정의 시 전역 Scope으로 올라가게 된다.
9   bar(); // 2
10
11  function foo() {
12      $foo_value = 1;
13      echo $foo_value;
14
15      function bar() {
16          // 함수 내의 접근 가능한 변수는 함수 내 선언 된 변수만 가능
17          // 고찰 : global, GLOBAL 키워드의 사용 이유를 생각해볼 것!!
18          //echo $foo_value."<br>";  // Unvisible
19
20          $bar_value = 2;
21          echo $bar_value;
22      }
23  }
```

함수 특징 : Variable function 지원

```php
function foo() {
    echo "Hello YC Jung. You are doing great job!";
}

$func_name = "foo";

// 변수에 () 연산자가 붙을 경우, 해당 변수명을 가지는 함수를 실행한다.
$func_name();

class Bar {
    function prtSomething() {
        echo "Hello Richard Jung";
    }
}

$obj = new Bar();

// 객체 메서드에도 적용가능
$method_name = "prtSomething";
$obj->$method_name();
```

함수 특징 : Arrow & Closure function 지원

- **Arrow function , As of PHP 7.4**

```php
<?php

$y = 1;

$fn1 = fn($x) => $x + $y;
// equivalent to using $y by value:
$fn2 = function ($x) use ($y) {
    return $x + $y;
};

var_export($fn1(3));
?>
```

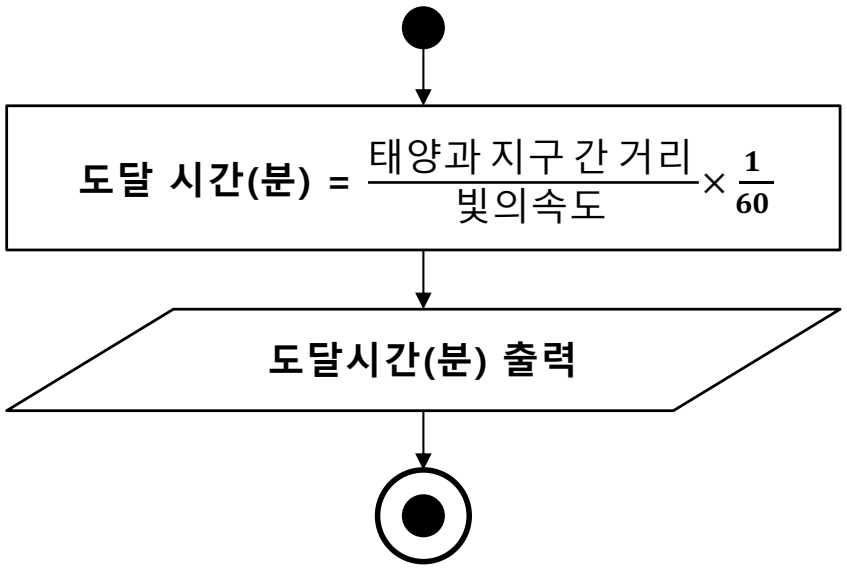
The above example will output:

```
4
```

Extra Slides

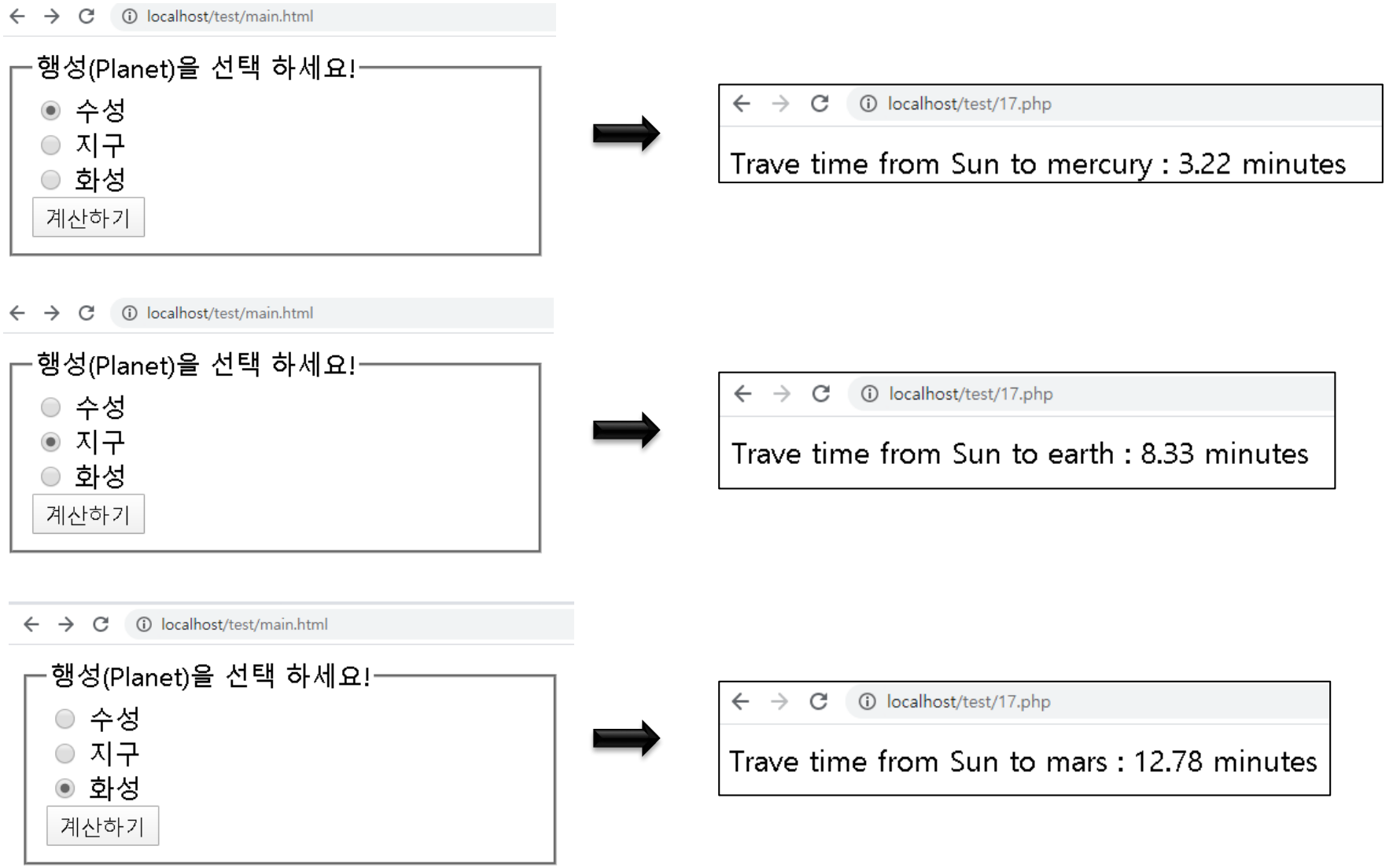
문제1: 빛이 태양에서 출발해서 지구에 도달하는 시간 계산하기 (1)

- 아래 행성의 이름을 입력 받고, 입력 받은 행성에서 태양까지 빛이 도달하는 시간(분)을 계산하라
  - 태양에서 수성까지 거리: **약 5,790만 km**
  - 태양에서 지구까지 거리: **약 1억 5천만 km**
  - 태양에서 화성까지 거리: **약 2억 3천만 km**
  - 빛의 속도 (Speed of Light): 약 **30만 km/s**
- 출력 결과는 소수점 둘째 자리 반올림
  - **www.php.net에서 "Math" 키워드로 검색, round() 함수 이용**



문제1: 빛이 태양에서 출발해서 지구에 도달하는 시간 계산하기 (2)

✓ 결과 값



문제 2: 표준편차 계산 프로그램 (1)

- 사용자로부터 5개의 정수 값을 입력 받아, 평균, 분산, 표준 편차 값을 계산하라
- 출력 값은 소수점 둘째 자리까지.

$$\textbf{표본분}산, V(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

$$\textbf{표본표}준편차\ \sigma(X) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$$

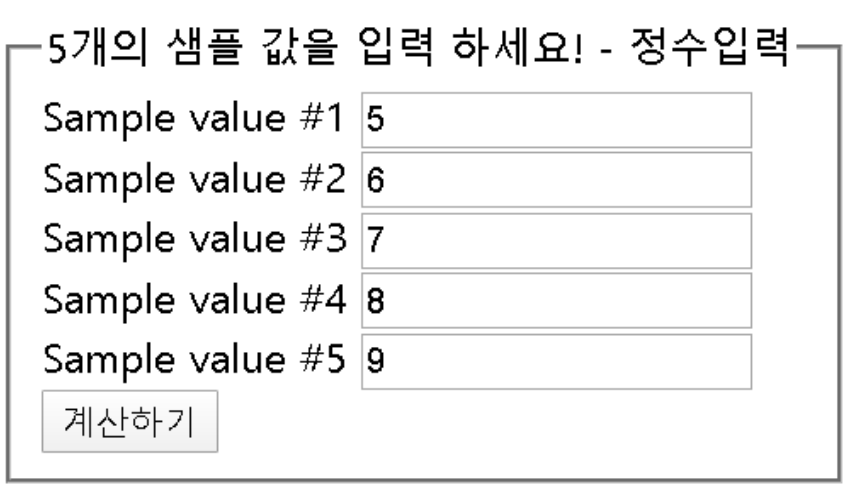
n = 총 샘플 수

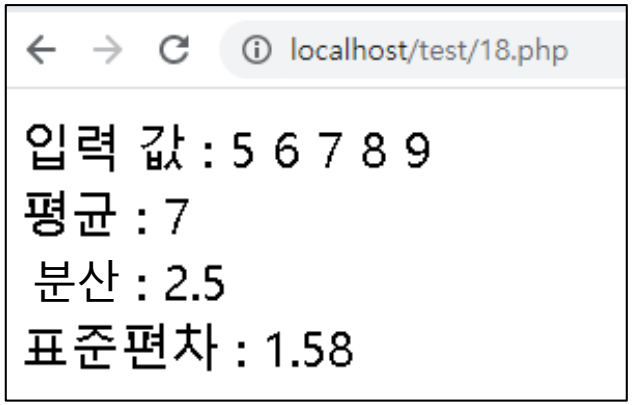
$\bar{X}$ = 평균

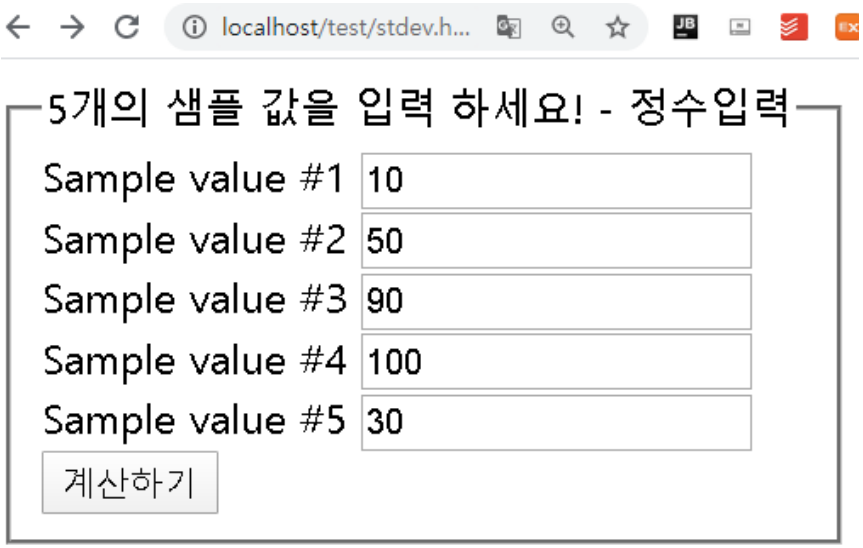
- 제곱근은 pow ()함수, 루트는 sqrt()함수 이용.
- www.php.net에서 찾아 볼 것

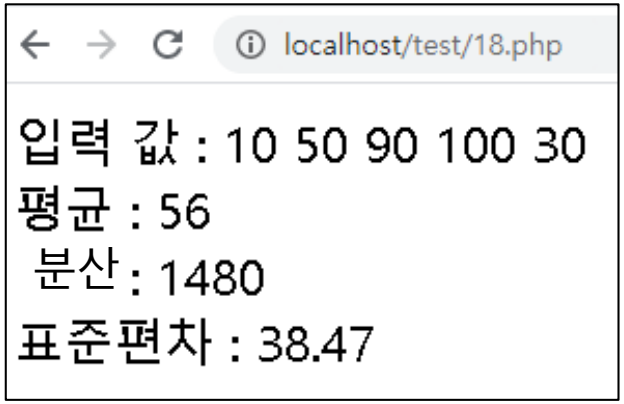
문제 2: 표준편차 계산 프로그램 (2)

✓ 결과 값









Q/A

감사합니다

주문식교육의 산실

영진전문대학교