Chapter 05 PHP 기본 문법

1. PHP 코드 작성의 구조

1.1 PHP 코드의 시작과 끝 : <?php ... ?>

PHP 프로그래밍 언어의 기본 구조는 <?php 기호로 시작해서 ?> 기호로 마무리됩니다. PHP 문법으로 작성된 모든 소스코드들은 <?php 기호와 ?> 기호 사이에 놓이게 되며, PHP 엔진은 이들 소스코드들을 해석하여 처리합니다.

```
<?php
  PHP 문법 소스코드
?>
```

```
<?
  PHP 문법 소스코드
?>
```

PHP 코드의 시작 기호를 <? 기호도 사용할 수 있습니다. 권장하는 표현 방법은 아니지만 이렇게 짧게 표현할 수 있도록 설정하는 방법을 살펴보겠습니다.

Apache 웹서버의 환경 설정처럼 PHP 언어도 환경 설정을 할 수 있는데 이와 관련된 파일은 『php.ini』 이며 \xampp\php 폴더에 저장되어 있습니다. 이 파일도 매우 중요한 파일이기 때문에 원본은 반드시 복사해서 보관해야 합니다.

php.ini 파일을 편집기로 불러와서 short_open_tag 단어로 검색하면 다음과 같이 147번 라인과 198번 라인에서 찾아집니다. 147번 라인은 주석이므로 제외하고 198번 라인을 변경하면 재설정됩니다.

198번 라인의 short_open_tag 초기값은 Off 값으로 설정되어 있는 이를 On 값으로 재설정합니다.

/xampp/php/php.ini

```
147  ; short_open_tag
148  ;   Default Value: On
 :      ... <생략> ...
197  ; https://php.net/short-open-tag
198  short_open_tag=Off
```

/xampp/php/php.ini

```
147  ; short_open_tag
148  ;   Default Value: On
 :      ... <생략> ...
197  ; https://php.net/short-open-tag
198  short_open_tag=On
```

주의할 점은 이렇게 php.ini 파일을 변경하면 저장해야 하며, 동시에 Apache 웹서버를 재시작해야 재설정 내용이 반영됩니다.

이렇게 설정하면 <?php 기호와 <? 기호를 이용하여 PHP 코드의 시작을 알릴 수 있습니다. 다만 <? 시작 기호는 다른 언어들의 기호들과 혼동이 있을 수 있기 때문에 원래 기본값인 Off으로 유지하는 것이 좋습니다.

1.2 문장 구분

PHP 코드에서 한 문장의 끝은 세미콜론(;) 기호로 구분됩니다. 각 문장들의 끝을 ; 기호로 종료하면 한 줄에 이어서 작성해도 정상적으로 처리됩니다.

다음의 2개의 파일들은 동일하게 처리되지만, 코드의 가독성을 위해 각 라인을 분리해서 작성하는 것이 좋습니다.

separate_line.php	one_line.php

<?php
  echo " Hello-1 ";
  echo " Hello-2 ";
?>
	<?php
	  echo " Hello-1 "; echo " Hello-2 ";
	?>

1.3 공백과 Enter(개행)

들여쓰기 등을 위한 여러 개의 공백은 한 개와 동일합니다. 또한 각 문장들은 ; 기호로 구분되므로 개행도 무시됩니다. 이러한 공백, 개행 등은 코드의 가독성을 위해 사용되는 것으로 코드의 실행에는 영향을 주지 않습니다.
다음의 2개의 파일들은 동일하게 처리됩니다.

space_enter_1.php	space_enter_2.php

<?php
  $age = 21;
  $nation="Korea";
 	
  echo $age, $nation;
?>
	<?php $age=21;$nation="Korea";echo $age,$nation;?>

1.4 주석(Comment)

주석은 프로그램을 개발할 때 코드들에 대한 설명을 기록하기 위해 사용되며, 주석 처리된 문장들은 코드로 인식되지 않습니다. 1줄 주석(//, #), 2줄 이상 주석(/* */)을 사용할 수 있습니다.

- // 주석내용 : 1줄 주석
- # 주석내용 : 1줄 주석
- /* 주석내용 */ : 2줄 이상 주석

comment.php	결과
// echo 문 테스트-1 echo " Hello PHP-1 "; # echo 문 테스트-2 echo " Hello PHP-2 "; /* echo 문 테스트-3 echo " Hello PHP-3 "; */	Hello PHP-1 Hello PHP-2

3번째 echo 문은 2중 이상 주석 기호 내부에 있기 때문에 실행되지 않습니다.
2줄 이상 주석 기호를 이중으로 사용하면 에러가 발생하니 주의해야 합니다.

comment_error.php	결과
/* echo 문 테스트-3 /* echo " Hello PHP-3 "; */ */	Parse error: syntax error, unexpected token "*"

1.5 대문자와 소문자

알파벳의 대문자와 소문자는 같은 경우도 있고 다른 경우도 있습니다. PHP 키워드인 <?php, echo, if, print 등은 <?PHP, ECHO, IF, PRINT와 같이 사용해도 됩니다. 그러나 사용자가 정의한 변수명, 상수명은 대소문자를 구별합니다(case sensitive). 예를 들어 $Name과 $name은 다른 변수입니다. 그런데 사용자가 정의한 함수명은 대소문자를 구별하지 않습니다(case insensitive). add() 함수와 ADD() 함수는 동일합니다. 좀 헷갈리죠? 어쨌든 잘 구분해야 합니다.

1.6 PHP 변수명

PHP 변수명은 다른 언어와 달리 $ 기호를 이용하여 변수명을 만듭니다. 변수명에 대한 자세한 내용은 3절의 PHP 변수를 참고하세요.

1.7 처리 결과의 출력

Php 프로그램의 처리 결과는 웹브라우저에 출력됩니다. 출력을 위해 주로 echo, print, printf 등을 이용합니다.

1.7.1 echo 문

문자열이나, 변수, 상수에 저장된 값들을 출력하기 위해 사용되며 HTML 태그를 포함하여 웹브라우저에 전송하는 함수입니다. echo 문은 여러 개의 인수를 출력할 수 있습니다.

(1) 문자열 출력

출력되는 문자열은 인용부호를 이용하여 출력합니다. 단일, 이중 인용부호 모두 사용 가능합니다.

echo_1.php	결과
echo "PHP 학습"; echo 'HTML과 echo함수'; echo "웹브라우저 출력";	PHP 학습HTML과 echo함수웹브라우저 출력

(2) 변수 출력

변수명으로만 출력이 가능하며, 이중 인용부호, 괄호 등을 이용하여 출력합니다.

echo_2.php	결과
$str1 = "PHP 학습"; $str2 = "HTML과 echo함수"; $str3 = "웹브라우저 출력"; echo $str1; echo "$str2"; echo ($str3);	PHP 학습HTML과 echo함수웹브라우저 출력

또한 변수는 다른 문자열과 함께 출력할 수 있습니다. 변수와 문자열 공백없이 출력하기 위해 {} 기호를 사용합니다.

echo_3.php	결과
$str1 = "PHP 학습"; echo "$str1 HTML과 echo 함수"; echo " "; echo "{$str1}HTML과 echo 함수";	PHP 학습 HTML과 echo 함수 PHP 학습HTML과 echo 함수

(3) HTML 태그 출력

인용부호를 이용하여 문자열이나, HTML 태그를 포함하여 웹브라우저에게 전송하면, 웹브라우저는 태그의 기능을 해석하여 출력합니다.

echo_4.php	결과
echo "PHP 학습 "; echo "HTML과 echo함수 "; echo "웹브라우저 출력";	PHP 학습 HTML과 echo함수 웹브라우저 출력

(4) 여러 개의 인수 출력

여러 개의 인수를 콤마(, : 인수 나열) 연산자 또는 점(. : 문자열 결합) 연산자로 연결하여 출력합니다.   기호는 개수만큼 웹브라우저에 공백을 표시합니다.

echo_5.php	결과
$str1 = "PHP 학습 "; echo $str1, "HTML과 echo함수 "; echo $str1. "HTML과 echo함수 "; echo "HTML과 echo함수", "웹브라우저 출력";	PHP 학습 HTML과 echo함수 PHP 학습 HTML과 echo함수 HTML과 echo함수웹브라우저 출력

1.7.2 print 문

echo 문과 동일한 방법으로 사용하지만, 인수는 1개만 가능하며 괄호는 사용하지 않습니다.

print_1.php	결과
$str1 = "PHP 학습"; echo "$str1 HTML과 echo 함수"; echo " "; echo "{$str1}HTML과 echo 함수";	PHP 학습 HTML과 echo 함수 PHP 학습HTML과 echo 함수

print_2.php	결과
$str1 = "PHP 학습 "; print $str1, "HTML과 echo함수 ";	Parse error: syntax error, unexpected token "," in

1.6.3 printf() 함수

printf() 함수는 정수, 실수, 문자열을 다양한 형태로 출력하는 함수입니다. % 기호와 함께 각 형태를 나타내는 지시 문자를 통해 출력을 설정합니다.

다음 표는 데이터 타입에 따라 사용 가능한 지시 문자를 나타냅니다.

데이터 타입	지시 문자
int	d, u, c, o, x, X, b
float	e, E, f, F, g, G, h, H
string	s

지시 문자	설명
b	정수를 이진수로 표현
c	정수를 아스키 문자로 표현
d	정수를 10진수로 표현
e	정수를 e지수로 표현
u	정수를 부호없는 10진수로 표현
f	정수를 부동소수점으로 표현
o	정수를 8진수로 표현
s	문자열로 표현
x	정수를 16진수(소문자)로 표현
X	정수를 16진수(대문자)로 표현
+, -	정수의 부호(+, -) 표현

다음 코드는 각 지시 문자에 따른 출력

printf.php	결과

`$a =  54321;  $b = -54321;  $c = 72;`
`printf("%%b = '%b'", $a);`
`printf("%%c = '%c'", $c);`
`printf("%%d = '%d'", $a);`
`printf("%%e = '%e'", $a);`
`printf("%%u = '%u'", $a);`
`printf("%%f = '%f'", $a);`
`printf("%%o = '%o'", $a);`
`printf("%%s = '%s'", $a);`
`printf("%%x = '%x'", $a);`
`printf("%%X = '%X'", $a);`
`printf("%%+d = '%+d'", $a);`
`printf("%%+d = '%+d'", $b);`
	
	`%b = '1101010000110001'`
	`%c = 'H'`
	`%d = '54321'`
	`%e = '5.432100e+4'`
	`%u = '54321'`
	`%f = '54321.000000'`
	`%o = '152061'`
	`%s = '54321'`
	`%x = 'd431'`
	`%X = 'D431'`
	`%+d = '+54321'`
	`%+d = '-54321'`

2. 변수

2.1 변수와 메모리 주소

변수와 상수는 수치나 문자열 같은 값들을 저장하는 기억장소입니다. 변수는 값들이 변경될 수 있는 기억장소이며, 상수는 한번 지정한 값을 변경할 수 없는 기억장소입니다.
이러한 기억장소를 메모리(RAM : 주기억장치)의 물리적 주소라고 하며, 변수의 이름은 곧 메모리 주소입니다. 이러한 메모리 주소로 프로그래밍에 사용하는 것은 표기하기도 어렵고 기억하기도 불가능합니다. 따라서 메모리 주소를 사람이 기억하기 편하도록 만든 것이 변수명입니다.

2.1.1 변수없는 프로그래밍

변수없이 프로그래밍을 한다면 다음과 같이 메모리의 물리적 주소에 값을 할당하고, 연산을 수행해야 합니다. 1개의 메모리 주소에 저장되는 값의 크기는 1바이트이며, 여기서 저장되는 값은 2바이트 정수형이라고 가정합니다.
코드에서의 & 기호는 메모리 주소라는 것을 표기하기 위해 사용되는 일반적인 표기법입니다.

코드
&AF4C11 = 23 &AF4C13 = 45 &AF4C15 = &AF4C11 + &AF4C13

메모리 주소	값 저장공간 (1Byte)
:	
AF4C11	23
AF4C12	
AF4C13	45
AF4C14	
AF4C15	68
AF4C16	
:	

2.1.2 변수있는 프로그래밍

하지만 변수를 이용하여 프로그래밍하면 다음과 같이 간단하게 작성할 수 있습니다.(메모리의 시작주소가 변수명에 할당됩니다.)

코드
a = 23 b = 45 c = a + b

변수명		메모리 주소	값 저장공간 (1Byte)
		:	
a	→	AF4C11	23
		AF4C12	
b	→	AF4C13	45
		AF4C14	
c	→	AF4C15	68
		AF4C16	
		:	

따라서 프로그램 코드에서 변수가 만들어지면 컴퓨터 시스템에서는 각 변수의 메모리 주소와 값 저장 공간이 확보됩니다. 저장 공간의 크기는 변수의 데이터 타입에 따라 정해지기도

하고 변하기도 합니다.
흔히 얘기하는 메모리 용량은 메모리 주소의 개수이며, 따라서 메모리 용량이 1MB 라면 약 백만개의 메모리 주소가 있다고 보면 됩니다.

변수에 값을 할당하는 일반적인 방법은 다음과 같습니다. = 기호를 기준으로 왼쪽에 있는 것을 Left_value, 오른쪽을 있는 것을 Right_value라고 부릅니다.

```
Left_value = Right_value
```

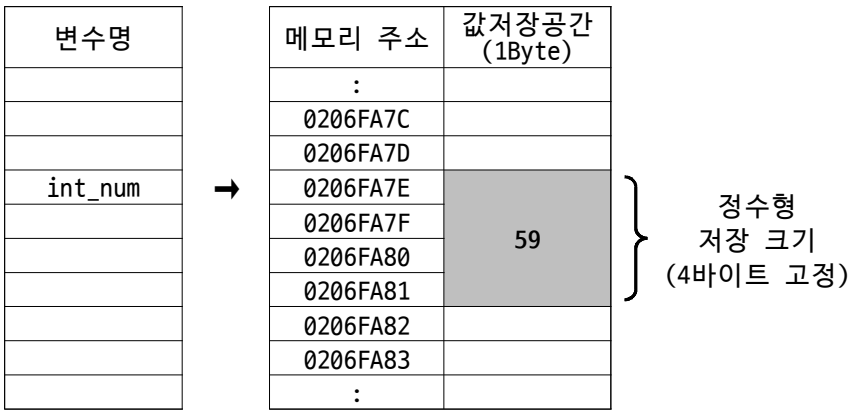
그래서 변수명을 lvalue 또는 메모리 주소라고도 합니다. Refernce(참조) 단어도 같은 의미로 사용되는 용어입니다.

2.2 기억장소와 데이터 타입

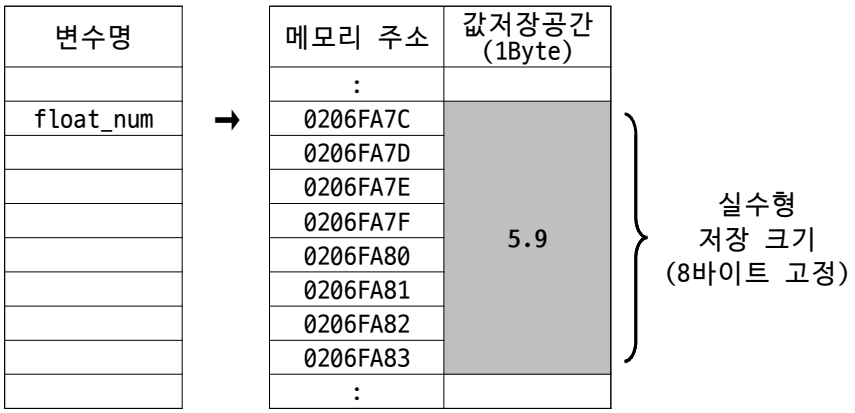
하나의 메모리 주소에는 1바이트 크기의 값이 저장됩니다. 메모리에 저장되는 데이터는 그 타입에 따라 다양한 크기를 가지게 되는데 시스템 또는 프로그래밍 언어마다 차이는 있지만 정수형은 4 바이트, 실수형은 8바이트, 문자열은 저장한 글자 수 만큼의 바이트 길이를 가진다고 가정합니다.
따라서 정수형 4바이트를 저장하기 위해서는 메모리 주소 4개가 필요한 셈이죠.

2.2.1 데이터 타입의 크기

예를 들어 정수형이 4바이트일 경우 int_num 변수에 59 값을 저장하면 다음과 같이 기억장소가 할당됩니다. int_num 변수의 메모리 주소 0206FA7E 이며, 저장되는 값은 비록 작은 정수이지만 4바이트 공간이 할당됩니다.

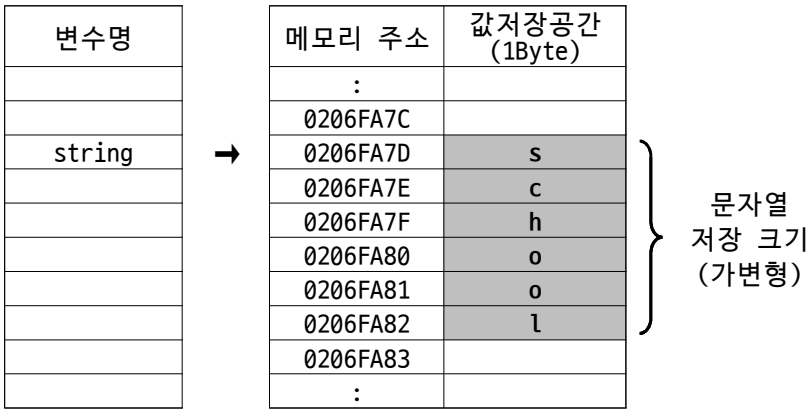


마찬가지로 실수형이 8바이트일 경우 float_num 변수에 5.9 값을 저장하면 다음과 같이 기억장소가 할당됩니다. float_num 변수의 메모리 주소 0206FA7C 이며, 저장되는 값은 비록 작은 실수이지만 8바이트 공간이 할당됩니다.



문자형일 경우에는 문자열의 길이가 정해져 있지 않기 때문에 몇 바이트가 할당될지는 알수 없지만 1개의 문자는 1바이트입니다. 따라서 1개의 메모리 주소에 1개의 문자가 저정됩니다.

string 변수에 'school' 값을 저장하면 저장하면 다음과 같이 기억장소가 할당됩니다. string 변수의 메모리 주소 0206FA7D 이며, 저장되는 값은 6바이트 공간이 할당됩니다. string 변수에 다른 문자열이 할당되면 당연히 그 만큼의 크기를 가집니다.



2.2.2 메모리 주소와 용량

위에서 살펴본 메모리 주소는 편의상 『0206FA7C』형태의 16진수로 표현되어 있는데, 이것을 통해서 메모리 용량을 알 수 있습니다.

우선 이 주소는 16진수 8자리로 구성되어 있으며, 16진수 1자리는 2의 4승에 해당됩니다. 16진수 2자리는 당연히 2의 8승 이겠죠? 만약 메모리 주소가 16진수 2자리만으로 표현되면 256개의 주소를 가지게 됩니다. 그리고 메모리 주소 1개에는 1바이트를 저장할 수 있으므로 16진수 2자리의 메모리 용량은 256바이트가 됩니다.

이런 방식으로 계산하면 아래와 같이 16진수 8자리 메모리 주소의 용량을 알아낼 수 있습니다.

```
16진수 1자리 : 0 ~ F ⇒ 2^4 ⇒ 16B
16진수 2자리 : 00 ~ FF ⇒ 2^8 ⇒ 256B
16진수 4자리 : 0000~FFFF  ⇒ 2^16 ⇒ 65,536 ⇒ 64KB
16진수 8자리 : 00000000~FFFFFFFF ⇒ 2^32 ⇒ 4,294,967,295 ⇒ 4MB
```

2.3 기본형 변수와 참조형 변수

2.3.1 기본형 변수

메모리 주소와 관련되어 지금까지 살펴본 변수의 데이터 타입은 정수형, 실수형, 문자형 등이며 이를 기본형 변수라고 합니다. 기본형 변수는 메모리 저장공간에 변수의 값 자체가 직접 보관됩니다. 앞서 살펴봤던 기본형 변수의 메모리 구조를 스택(Stack) 이라고 합니다.

변수명
int_num

➡

스택	
메모리 주소	저장공간
:	
0206FA7E	59
0206FA7F	
:	

2.3.2 참조형 변수

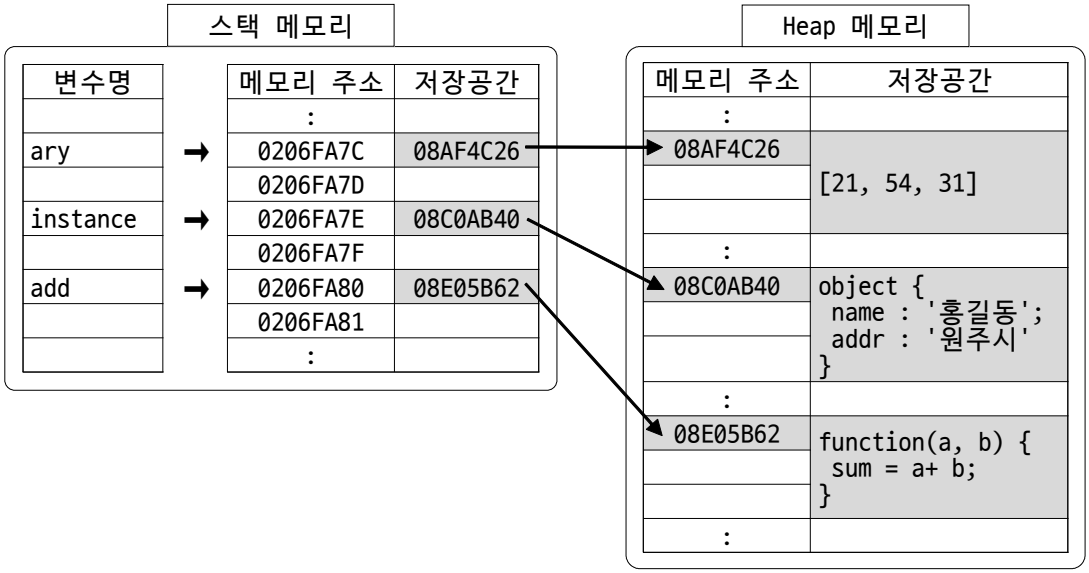
이러한 기본형 변수와 달리 참조형(주소형) 변수는 저장공간에 보관되는 값의 형태가 다릅니다. 참조형 변수는 배열, 객체, 함수와 같이 저장되는 값의 형태가 복합적인 변수를 말합니다.

다음과 같이 배열명 ary, 객체명 instance, 함수명 add 으로 선언된 코드가 있다고 가정합니다. 보다시피 기본형 변수의 값들보다 복잡하죠.

```
ary = [21, 54, 31]; // 배열
instance = object {     // 객체
  name : '홍길동',
  addr : '원주시'
}
add = function(a, b) {  // 함수
sum = a+ b;
}
```

그렇다면 참조형 변수의 값들은 메모리 주소가 어떻게 배정될까요? 참조형 변수의 값들은 복잡하기 때문에 스택 메모리에서 처리할 수 없고 Heap 메모리라는 공간을 확보하여 관리됩니다.

다음 그림을 보겠습니다.



우선 ary, instance, add 참조형 변수에 기본형 변수와 마찬가지로 스택 메모리 주소가 배정되고, 저장공간에는 값이 아닌 별도로 확보한 Heap 메모리 주소를 보관합니다.
따라서 참조형 변수가 호출되면 스택의 저장공간에 보관된 Heap 메모리의 주소 이동하여 Heap 메모리에서 처리됩니다.
이렇게 참조형 변수의 저장공간에 값이 아닌 Heap 메모리의 주소가 보관되기 때문에 참조형 변수라고 불립니다. 앞에서 참조와 주소는 같은 의미라고 설명했습니다.

2.4 변수, 주소, 값

할당 연산자 = 기호의 왼쪽에 있는 변수을 lvalue라고 표현했고, 이는 메모리 주소와 동일하다고 설명한 바 있습니다.

2.4.1 일반형 변수, 주소, 값

아래 코드에서처럼 a, b 변수에 값을 할당해서 실행하면 시스템에서는 각 변수에 메모리 주소를 자동적으로 배정하고 그 메모리 주소에 값을 저장합니다. 시스템에서는 a 변수에 456ABF 메모리 주소를 배정하고 10을 할당합니다. b = a 문장에 의해 b 변수에 456AC0 메모리 주소를 배정하고 10 값을 b 변수에 할당하게 됩니다. a, b 변수는 다르기 때문에 이후에 b 변수의 값이 변경되더라도 a 변수와는 관계가 없습니다.

<table>
<tr><th>코드</th><th>기억장소(메모리) 상황</th></tr>
<tr><td>a = 10;
b = a;
b = 20;</td><td>
<table>
<tr><th>변수</th><th>메모리주소</th><th>값</th></tr>
<tr><td>a</td><td>456ABF</td><td>10</td></tr>
<tr><td>b</td><td>456AC0</td><td>20</td></tr>
</table>
➡
<table>
<tr><th>변수</th><th>메모리주소</th><th>값</th></tr>
<tr><td>a</td><td>456ABF</td><td>10</td></tr>
<tr><td>b</td><td>456AC0</td><td>20</td></tr>
</table>
</td></tr>
</table>

2.4.2 변수에 주소 할당

변수에 할당되는 값을 메모리 주소로도 지정할 수 있습니다. 변수의 주소는 & 기호로 나타내는데 아래 코드에서 b=&a; 문장은 a 변수의 주소값을 b 변수에 할당합니다. 이렇게 되면

a, b 변수는 같은 메모리 주소을 공유하여 사용하기 때문에 어떤 하나의 변수 값을 변경하게 되면 다른 변수도 변경되게 됩니다.

<table>
<tr><th>코드</th><th>기억장소(메모리) 상황</th></tr>
<tr><td>a=10;
b=&a;
b=20;</td><td>
<table>
<tr><th>변수</th><th>메모리주소</th><th>값</th></tr>
<tr><td>a</td><td>456ABF</td><td rowspan="2">10</td></tr>
<tr><td>b</td><td>456ABF</td></tr>
</table>
➡
<table>
<tr><th>변수</th><th>메모리주소</th><th>값</th></tr>
<tr><td>a</td><td rowspan="2">456ABF</td><td rowspan="2">20</td></tr>
<tr><td>b</td></tr>
</table>
</td></tr>
</table>

다음 PHP 코드의 4번 라인에서 $b 변수 출력은 10인데 이것은 $b 변수에 $a 변수 값인 10 값이 할당되서가 아니라, $b 변수와 $a 변수가 메모리 주소를 공유하기 때문입니다.
5번 라인에서 $b 변수에 20을 할당하면 $a 변수의 값도 20을 변경됩니다. 이것 역시 $a 변수와 $b 변수가 메모리 주소를 공유하기 때문이지요.

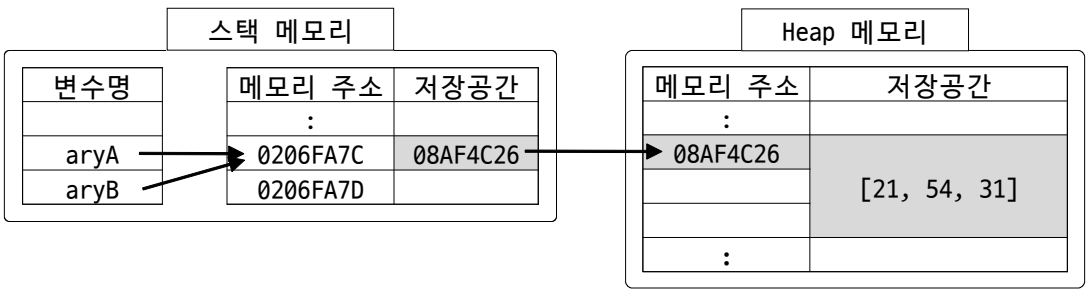
	assign.php	결과
1	$a=10;	
2	$b=&$a;	
3	echo "\$a 변수값:", $a, " ";	$a 변수값:10
4	echo "\$b 변수값:", $b, " ";	$b 변수값:10
5	$b=20;	
6	echo "\$a 변수값:", $a, " ";	$a 변수값:20
7	echo "\$b 변수값:", $b, " ";	$b 변수값:20

사실 이런 코드는 실제로 많이 사용되는 것은 아니지만 변수와 메모리 시스템과의 관계 및 참조형 변수를 이해하는데 매우 중요합니다.

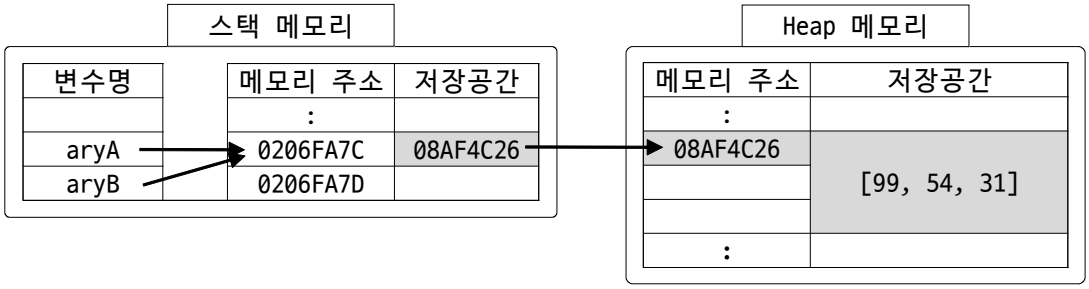
2.4.3 참조형 변수, 주소, 값
참조형 변수의 주소와 값 할당에 대해 구체적으로 살펴보겠습니다.
(1) 배열
다음과 같이 배열이 선언되었다고 가정합니다.

```
1  aryA = [21, 54, 31]; // aryA[0]=21, aryA[1]=54, aryA[2]=31
2  aryB = aryB; // aryB[0]=21, aryB[1]=54, aryB[2]=31
3  aryA[0] = 99;
4  print aryA, aryB;
```

1~2번 라인의 문장에 의해 aryA, aryB 배열은 동일한 메모리 주소를 가지게 됩니다.



3번 라인에서 aryA[0] 원소의 값을 수정하게 되면 aryB 배열에 영향을 주게 됩니다. 따라서 aryA, aryB 배열은 동일한 값을 출력하게 됩니다.



다만, 이러한 처리는 참조형 변수의 일반적인 경우이며, 프로그래밍 언어에 따라 다를 수 있습니다.

특히 **PHP 언어에서는 배열의 복사는 기본적으로 값을 복사합니다. 배열의 복사를 참조방식으로 하려면 명시적으로 & 기호를 표기해야 합니다.**

다음의 PHP 코드를 살펴봅니다.

2번 라인의 $aryB = $aryA; 문장으로 배열 복사가 수행되고, $aryA[0] 원소의 값 변경이 있지만 $aryB[0] 값은 변경되지 않습니다.

array_reference_1.php		결과
1	$aryA = [21, 54, 31];	
2	$aryB = $aryA;	
3	$aryA[0] = 99;	
4	echo "\$aryA[0] : {$aryA[0]} ";	aryA[0] : 99
5	echo "\$aryB[0] : {$aryB[0]} ";	aryB[0] : 21

다음의 PHP 코드는 배열의 복사를 참조방식으로 하는 경우입니다.

2번 라인의 $aryB = &$aryA; 문장처럼 & 주소 기호로 명시적인 배열 참조 복사를 수행합니다. $aryA, $aryB 배열의 주소가 동일하므로 $aryA[0] 원소 값의 변경은 $aryB[0] 원소에도 영향을 주게 됩니다.

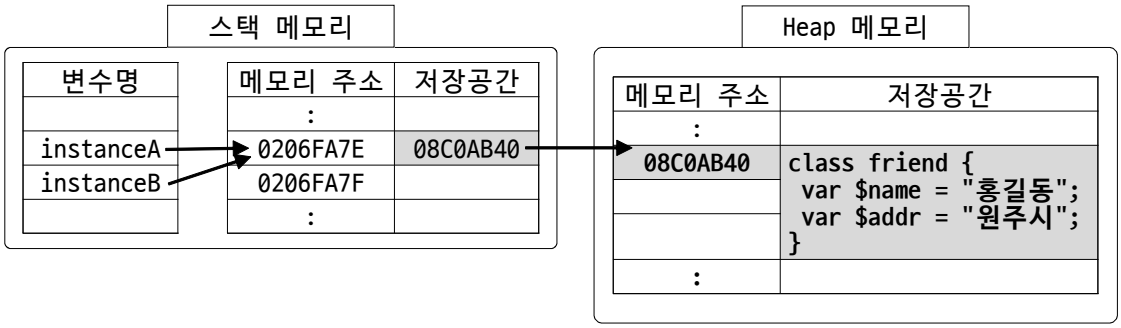
array_reference_2.php		결과
1	$aryA = [21, 54, 31];	
2	$aryB = &$aryA ;	
3	$aryA[0] = 99;	
4	echo "\$aryA[0] : {$aryA[0]} ";	aryA[0] : 99
5	echo "\$aryB[0] : {$aryB[0]} ";	aryB[0] : 99

(2) 객체

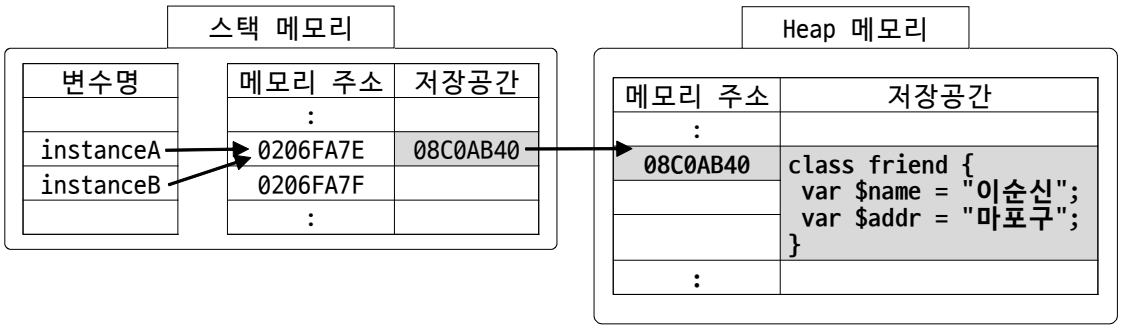
PHP 언어에서 객체의 복사는 참조방식으로 수행되기 때문에 다음과 같은 PHP 코드로 설명하겠습니다.

object_reference_1.php		결과
1	class friend {	
2	var $name = "홍길동";	
3	var $addr = "원주시";	
4	}	friend Object
5	$instanceA = new friend();	([name] => 이순신 [addr] => 마포구)
6	$instanceB = $instanceA;	friend Object
7	$instanceA->name = "이순신";	([name] => 이순신 [addr] => 마포구)
8	$instanceA->addr = "마포구";	

1~6번 friend 객체의 $instanceA 인스턴스를 생성하고 이것을 $instanceB에 복사합니다. 객체는 참조형 변수이기 때문에 $instancA와 $instanceB는 주소를 공유합니다. 따라서 다음 그림과 같이 수행됩니다.



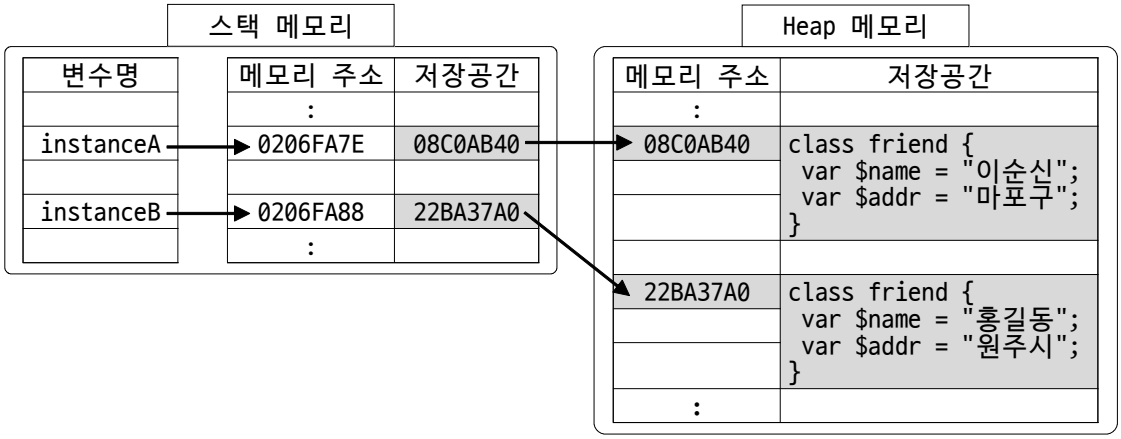
7~8번 라인에서 $instanceA의 프로퍼티인 name, addr 값이 변경되면 $instanceB에도 영향을 주게 되어 동일한 값을 출력합니다.



만약 다음 코드의 6번 라인처럼 객체를 복사하지 않고 $instanceB 인스턴스를 생성하면 어떻게 될까요?

	object_reference_2.php	결과
1	class friend {	
2	var $name = "홍길동";	
3	var $addr = "원주시";	
4	}	friend Object
5	$instanceA = new friend();	([name] => 이순신 [addr] => 마포구)
6	$instanceA = new friend();	friend Object
7	$instanceA->name = "이순신";	([name] => 이순신 [addr] => 마포구)
8	$instanceA->addr = "마포구";	

두 개의 $instanceA와 $instanceB는 당연히 다른 인스턴스이므로 다음 그림처럼 메모리 주소도 다르며, 서로에게 영향을 주지 않습니다.


스택 메모리
변수명
메모리 주소
저장공간
:
instanceA
0206FA7E
08C0AB40
instanceB
0206FA88
22BA37A0
:
Heap 메모리
메모리 주소
저장공간
:
08C0AB40
class friend {
var $name = "이순신";
var $addr = "마포구";
}
22BA37A0
class friend {
var $name = "홍길동";
var $addr = "원주시";
}
:

3. PHP 변수

3.1 변수명 생성 규칙

PHP 언어에서 변수명을 만들기 위해서는 지켜야 할 규칙이 있습니다.

- 변수명 앞에는 반드시 $ 기호를 붙여야 합니다.
- 변수명은 영문자 또는 _ (밑줄) 기호로만 시작해야 합니다.
- 변수명의 나머지 부분은 영문자, 숫자, _ 기호로 구성해야 합니다.
- 변수명은 대소문자를 구분합니다(Case Sensitive). 즉 $name과 $Name은 다른 변수명입니다.

가능한 변수명		불가능한 변수명	
$Name	$_ID	name	$#name
$salaryYear	$manOfYear	$88_age	$ name
$salary_2022	$age_of_people	$age-of-people	$age of people

3.2 PHP 변수의 유연성

C와 같은 언어에서와 달리 PHP 변수는 미리 선언할 필요가 없으며, 데이터 타입도 정의할 필요가 없습니다. 이것을 느슨한(loosly) 데이터 타입이라고 합니다. 또한 변수에 정수 값를 저장한 후에 실수 값를 저장할 수 있고 문자열 값을 저장할 수도 있습니다. 이를 데이터에 기반한 데이터 타입의 자동 설정이라고 하는데 즉, 수시로 저장되는 데이터 타입에 따라 변수의 데이터 타입도 결정됩니다.

datatype_1.php		설명
1	$a = "Hello PHP";	$a 변수는 문자형 데이터 타입
2	$b = 10;	$b 변수는 정수형 데이터 타입
3	$a = 10.1;	$a 변수는 실수형 데이터 타입
4	$b = "Web Server";	$b 변수는 문자형 데이터 타입

3.3 변수에 저장되는 데이터 값

변수에 저장되는 데이터 값의 형태는 2절에서 살펴본 바와 같이 논리값, 정수, 실수, 문자열, 배열, 객체, 함수 등으로 PHP 언어에서 사용되는 데이터 타입들이 모두 가능합니다.

변수에 저장된 값들이 어떤 데이터 타입인지를 확인하기 위해서는 is_ 키워드로 시작하는 함수를 사용합니다. 즉 is_int, is_float, is_string 등의 함수를 정수형 변수인지, 실수형 변수인지를 확인할 수 있습니다. is_ 관련 함수는 인수에 해당되는 변수나 값이 참 또는 거짓인지를 판정하여 true 또는 false 값으로 반환하는 함수입니다.

다음 예제는 is_int 함수 등에 변수명을 인수로 하여 그 변수의 데이터 타입이 무엇인지를 확인합니다.

	datatype_2.php	결과
1	$var_1 = "Hello PHP";	
2	var_dump(is_int($var_1));	bool(false)
3	var_dump(is_string($var_1));	bool(true)
4	$var_2 = 10;	
5	var_dump(is_int($var_2));	bool(true)
6	var_dump(is_float($var_2));	bool(false)
7	$var_3 = 10.1;	
8	var_dump(is_int($var_3));	bool(false)
9	var_dump(is_float($var_3));	bool(true)

3.4 미리 정의된 변수(Predifined Variables)

미리 정의된 변수는 PHP 프로그램의 처리과정에서 발생하는 데이터 값이 보관되는 변수들을 참조하는 변수입니다. PHP 시스템에서 미리 만들어 놓고 모든 PHP 프로그램에서 사용할 수 있는 일종의 내장 변수명입니다. 자세한 설명을 사용하는 장/절에서 설명합니다.

미리 정의된 변수에 슈퍼 전역 변수(Super Global Varaibles)와 일반 미리 정의된 변수로 구분되는데, 슈퍼 전역 변수는 대문자로 표현되는 특징이 있습니다.

미리 정의된 변수에 대한 목록은 다음 표에 제시합니다.

미리 정의된 변수		역할 및 기능
슈퍼 전역 변수	$GLOBALS	전역 범위에서 사용되는 모든 변수들을 참조
	$_SERVER	서버 정보 및 실행시 발생되는 환경 정보들을 참조
	$_GET	HTTP GET 방식에 사용되는 변수들을 참조
	$_POST	HTTP POST 방식에 사용되는 변수들을 참조
	$_FILES	HTTP 업로드시 파일 변수들을 참조
	$_REQUEST	HTTP 요청시 발생되는 정보들을 참조
	$_SESSION	세션 처리시 사용되는 변수들을 참조
	$_ENV	PHP 파서 동작시 발생하는 환경변수들을 참조
	$_COOKIE	쿠키 처리시 사용되는 변수들을 참조
일반 기정의 변수	$http_response_header	HTTP 응답 헤더 정보들을 참조
	$argc	커맨드 라인 환경에서 argument 개수 반환
	$argv	커맨트 라인 환경에서 argument 명칭을 배열로 반환

4. 상수

상수는 값을 한번 할당하면 나중에 변경할 수 없습니다. 상수명은 영문자나 _(밑줄) 기호로 시작해야 하며, 이어서 영문자, 숫자, _ 기호가 나올 수 있습니다. 상수명은 대소문자가 구별(case sensitive)되며 대문자로 만드는 것이 관례입니다. 상수에는 정수, 실수, 문자열, 배열이 저장될 수 있습니다.

4.1 const 키워드로 상수 만들기

다음 코드는 실수형 상수와 배열형 상수를 정의하는 예제입니다. var_dump() 함수를 이용하여 상수의 데이터 타입을 확인할 수 있습니다.

constant_1.php		결과
1	`const TAX = 0.15;`	
2	`const FRUITS = array('apple', 'grape');`	`float(0.15)`
3	`echo var_dump(TAX);`	`array(2) { [0]=> string(5) "apple"`
4	`echo var_dump(FRUITS);`	`[1]=> string(5) "grape" }`
5	`echo FRUITS[1];`	`grape`

4.2 define() 함수로 상수 만들기

define() 함수의 첫 번째 argument로 상수명을 표기하는데 반드시 인용부호를 이용해야 합니다.

constant_2.php		결과
1	`define("TAX", 0.15);`	
2	`define("HELLO", "Good Moring");`	
3	`echo var_dump(TAX), " ";`	`float(0.15)`
4	`echo var_dump(HELLO);`	`string(12) "Good Morning"`

4.3 미리 정의된 상수

PHP 언어에서는 코드가 실행되면서 많은 값들이 발생하게 되는데 이값들은 미리 정의된 상수에 저장됩니다.

4.3.1 핵심 상수

PHP 엔진이 설치되면서 시스템과 연관된 정보들을 값으로 보관하는 PHP 모듈 상수입니다. 다음은 미리 정의된 핵심 상수들중 일부입니다.

핵심 상수	기능
PHP_VERSION	설치된 PHP 버전을 나타내는 상수
PHP_OS	운영체제를 나타내는 상수
PHP_OS_FAMILY	세부 운영제체를 나타내는 상수
PHP_MAXPATHLEN	경로가 포함된 파일이름의 최대 길이를 나타내는 상수
PHP_INT_SIZE	정수의 크기를 바이트 단위로 나타내는 상수

다음 코드는 핵심 모듈 상수들의 값을 나타내는 예제입니다.

constant_3.php		결과
1	echo PHP_VERSION, " ";	8.1.6
2	echo PHP_OS, " ";	WINNT
3	echo PHP_OS_FAMILY, " ";	Windows
4	echo PHP_MAXPATHLEN, " ";	2048
5	echo PHP_INT_SIZE, " ";	8

4.3.2 매직 상수

매직 상수는 컴파일 또는 실행할 때 발생되는 값이며, 어떤 코드를 언제 실행했는냐에 따라 다른 값을 나타냅니다. 상수명 앞뒤에 _(밑줄) 기호 2개씩 포함됩니다.

다음은 미리 정의된 매직 상수들중 일부입니다.

매직 상수	기능
__LINE__	파일 내에서 현재 라인번호를 나타내는 상수
__DIR__	파일명을 포함하고 있는 폴더명을 나타내는 상수
__FILE__	폴더를 포함한 파일명을 나타내는 상수

다음 코드는 핵심 모듈 상수들의 값을 나타내는 예제입니다.

constant_4.php		결과
1	<?php	
2	echo __LINE__, " ";	2
3	echo __DIR__, " ";	H:\xampp\htdocs\source\05
4	echo __FILE__, " ";	H:\xampp\htdocs\source\05\constant-4.php
5	echo __LINE__, " ";	5
6	?>	

5. 데이터 타입

변수(Variable)와 상수(Constant)는 수치나 문자열같은 값들을 저장하는 기억장소입니다. 변수는 값들이 변경될 수 있는 기억장소이며, 상수는 한번 지정한 값을 변경할 수 없는 기억장소입니다. 데이터 타입은 이들 변수와 상수에 저장되는 값들의 형태(정수, 실수, 문자열, Bool 등)를 의미합니다.

PHP 언어는 느슨한(loosly) 데이터 타입이므로 int, string 등과 같은 선언을 하지 않으며, 타입 변환도 자유롭습니다.

5.1 데이터 타입

데이터 타입은 기본형과 혼합형으로 나뉘는데, 기본형에는 논리형(boolean), 정수(integer), 실수(float), 문자열(string)들이 있으며, 혼합형에는 배열(arary), 객체(object), 함수(function) 등이 있습니다.

5.1.1 데이터 타입 관련 함수

(1) var_dump 함수

변수의 데이터 타입과 값에 관련된 정보를 출력하는 함수입니다. 다음 코드는 변수에 할당된 값의 데이터 타입과 값을 출력합니다.

dump.php	결과
`$a = 123;` `$b = 1.23;` `$c = "name";` `$d = array("a", "b", "c");` `var_dump($a); echo " ";` `var_dump($b); echo " ";` `var_dump($c); echo " ";` `var_dump($d);`	`int(123)` `float(1.23)` `string(4) "name"` `array(3) { [0]=> string(1) "a" [1]=>` `string(1) "b" [2]=> string(1) "c" }`

(2) gettype() 함수

변수의 데이터 타입을 반환하는 함수입니다. 반환하는 형태는 boolean, integer, double, string, array, object, NULL, unknown type 등의 문자열입니다.

gettype.php	결과
`$a = 123;` `$b = 1.23;` `$c = "name";` `$d = array("a", "b", "c");` `echo gettype($a), " ";` `echo gettype($b), " ";` `echo gettype($c), " ";` `echo gettype($d), " ";`	`integer` `double` `string` `array`

(3) settype() 함수

변수의 타입을 변환하는 함수입니다. 변수의 값과 변환되는 타입에 따라 변환이 적용되지

않는 경우도 있으며, 정상적으로 변환되면 값도 변경됩니다.

setype.php	결과

$a = 123;
$b = 1.23;
$c = "name";
$d = array("a", "b", "c");
settype($a, "string"); var_dump($a);
settype($b, "integer");var_dump($b);
settype($c, "array");var_dump($c);
settype($d, "string");var_dump($d);
	string(3) "123"
	int(1)
	array(1) { [0]=> string(4) "name" }
	
	Warning: Array to string conversion in on line 9
	string(5) "Array

(4) 타입 자동 변환

PHP 언어의 변수는 원래와 다른 데이터 타입의 값을 저장하면 자동으로 변환됩니다.

setype.php	결과
$a = 123; echo gettype($a), " "; $a = 1.23; echo gettype($a), " "; $a = "name"; echo gettype($a), " "; $a = array("a", "b", "c"); echo gettype($a), " ";	integer double string array

5.2 기본형 데이터 타입

5.2.1 논리형 : Booleans

논리형은 참과 거짓을 판정하기 위해 사용되는 true(참), false(거짓) 2가지 값만을 저장할 수 있습니다. 대소문자를 구별하지 않기 때문에 TRUE, False 값도 같은 의미를 가집니다. 주의할 점은 true, false는 키워드이기 때문에 “true”, “false”와 같이 인용부호 속에 표기하면 참, 거짓의 의미가 사라지고 단순히 문자열로 인식된다는 점입니다.

bool_1.php	결과

$var_1 = false;
$var_2 = "false";
var_dump((bool) $var_1);
var_dump((bool) $var_2);
echo gettype($var_1);
echo gettype($var_2);
	
	bool(false)
	string(5) "false"
	boolean
	string

데이터 타입에 따라 참, 거짓의 의미는 다르게 표현됩니다. 아래 표시된 값들만 false를 의미하는데 이외의 모든 값들은 true입니다.

데이터 타입	flase 값	데이터 타입	false 값
논리형	false	문자형	"", "0"
정수형	0	배열	원소없는 배열
실수형	0.0	NULL	NULL

아래 코드에서 false 값으로 처리되는 라인은 1, 3, 5, 7, 8, 9번입니다.

bool_2.php		결과
1	var_dump((bool) false);	bool(false)
2	var_dump((bool) "false");	bool(true)
3	var_dump((bool) 0);	bool(false)
4	var_dump((bool) -1);	bool(true)
5	var_dump((bool) 0.0);	bool(false)
6	var_dump((bool) 0.1);	bool(true)
7	var_dump((bool) "");	bool(false)
8	var_dump((bool) "0");	bool(false)
9	var_dump((bool) array());	bool(false)
10	var_dump((bool) array(5));	bool(true)

이런 논리형 값은 다음 코드의 제어문(if, while) 등에서 true, false 값을 확인하는데 사용되기도 합니다. 1번 라인 if 조건식의 false 값은 거짓이므로 2번라인의 echo 문을 처리하고, 3번 라인 if 조건식의 “a” 값은 참이므로 3번 라인의 echo 문을 처리합니다.

bool_3.php		결과

1
2
3
4
	if (false) { echo "true
	"; }
	else { echo "false
	"; }
	if ("a") { echo "true
	"; }
	else { echo "false
	"; }
		false
		true

5.2.2 정수형 : Interger

음수가 포함된 정수값을 표현하는 데이터 타입으로 중요한건 정수값의 범위입니다. MS-Windows와 같은 운영체제가 32bit, 64bit 인지에 따라 정수값의 범위에 차이가 납니다. 10진수로 900경에 해당되는 숫자네요.

운영체제	범위(10진수)
32비트	-2,147,483,648 ~ 2,147,483,647
64비트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

아래 코드에서 PHP의 기정의 상수(시스템 상수)인 PHP_INT_MIN, PHP_INT_MAX 등을 이용하여 현재 운영체제에서의 정수의 최솟값, 최대값을 알아낼 수 있습니다.

다음의 코드에서 PHP_INT_SIZE 시스템 상수의 값은 현재 테스트되는 시스템의 운영체제가 8Byte 즉 64bit 시스템임을 알려줍니다.

integer_1.php		결과
1 2 3	echo PHP_INT_SIZE; echo PHP_INT_MIN; echo number_format(PHP_INT_MAX);	8 -9223372036854775808 9223372036854775807

정수는 10진수 외에도 2진수, 8진수, 16진수로도 표현할 수 있습니다. 2진수, 8진수, 16진수의 표기는 맨앞에 숫자 0이 위치하고 2번째에 각각 b(Binary), o(Octal), x(heXadec) 알파벳으로 몇 진수 인지를 구분합니다. 8진수의 경우에는 o 알파벳을 사용하지 않아도 됩니다. 다음 코드는 10진수 4567 값에 해당되는 각 진수의 표기법입니다.

integer_2.php		결과
1	$int_dec = 4567;	4567
2	$int_octal = 0o10727;	4567
3	$int_hexa = 0x11D7;	4567
4	$int_bin = 0b0001000111010111;	4567

5.2.3 실수형 : float

실수값 표현을 위한 데이터 타입으로 부동소수점 방식으로 표기됩니다. 실수를 표현하기 위해서 8Byte의 공간을 사용하며, 소수점 이하 14자리까지의 정밀도를 가지면서 최대 1.8e308 값을 표현할 수 있습니다. 실수는 매우 큰수를 표현하기 때문에 지수표기법을 사용하면 간단하게 표현할 수 있습니다.

운영체제	범위(10진수)
32비트	2.2250738585072E-308 ~ 1.7976931348623E+308
64비트	2.2250738585072E-308 ~ 1.7976931348623E+308

float.php		결과
1	$float_1 = 1.234;	1.234
2	$float_2 = 1.2e3;	1200
3	$float_3 = 7E-10;	7.0E-10

5.2.4 문자열 : string

문자열을 저장하기 위한 데이터 타입입니다. 문자 1개는 1바이트에 해당되며, 문자열의 최대 길이는 32bit 시스템에서 2,147,483,647 개이며 64bit 시스템에서는 제한이 없습니다.
문자열 데이터는 작은 따옴표, 큰 따옴표 쌍 내부에 작성됩니다.

작은 따옴표 속의 문자열에서 작은 따옴표를 표현하거나, 큰 따옴표 속의 문자열에서 큰 따옴표를 표현하려면 \ (백슬래시) 기호를 이용하면 됩니다.

string.php		결과
1	echo '작은 따옴표 문자열 ';	작은 따옴표 문자열
2	echo "큰 따옴표 문자열 ";	큰 따옴표 문자열
3	echo '작은 따옴표 \' 표기 ';	작은 따옴표 ' 표기
4	echo "큰 따옴표 \" 표기 ";	큰 따옴표 " 표기

\ 기호는 이스케이프 시퀀스(Escape Sequence) 기능을 알리는 것으로 표현이 안되는 문자나 기능을 표현하기 위해 큰 따옴표 속에서 사용합니다. 단 이 기능들은 콘솔 등 텍스트 형태로 문자열을 출력할 때 사용되며, 웹브라우저에는 적용되지 않습니다.

시퀀스	기능
\n	줄바꿈(Newline, 아스키코드 10번)
\r	줄바꿈(carriage Return, 아스키코드 13번)
\t	수직탭(horizontal Tab, 아스키코드 9번)
\v	수평탭(Vertical tab, 아스키코드 11번)
\e	ESC 키(아스키코드 27번)
\f	페이지바꿈(Form feed 아스키코드 12번)
\\	백슬래시
\$	달러 기호
\"	double-quote

5.3 혼합형 데이터 타입

배열, 객체, 함수 등은 혼합형 데이터 타입에 해당됩니다.
배열과 객체는 Chapter 07, 함수는 Chapter 07에서 설명합니다.

6. 연산자(Operators)

연산자는 값들을 연산하는데 사용하는 기호입니다. 연산 대상 즉 피연산자(operand)의 형태나, 연산된 결과값의 형태에 따라 산술연산자, 비교연산자, 논리연산자 등으로 구분하기도 합니다.

6.1 연산자 우선순위

6.2 할당 연산자(Assignment Operators)

변수나 상수 등에 값을 할당하는 연산자로 = 기호를 사용합니다.

연산자	사용예	설명
=	$age = 10 $count = $i	$age 변수에 10을 대입 $count 변수에 $i 변수의 값을

3절의 변수 설명에서 할당 연산자 = 기호의 왼쪽에 있는 변수을 lvalue라고 표현했고, 이는 메모리 주소와 동일하다고 설명한 바 있습니다.

6.3 산술 연산자(Arithmetic Operators)

산술 연산자는 수치 계산에 사용되는 연산자입니다. 일상에서 사용되는 계산과 동일합니다.

연산자	사용예	의미	연산자	사용예	의미
+	+$a	양수 연산자	*	$a * $b	곱하기 연산자
-	-$a	음수 연산자	/	$a / $b	나누기 연산자
+	$a + $b	더하기 연산자	%	$a % $b	나머지 연산자
-	$a - $b	빼기 연산자	**	$a ** $b	거듭제곱 연산자

%(Modulo) 연산자는 정수만을 처리하기 때문에 소수점 부분은 버리고 정수만으로 계산합니다. 또한 첫 번째 피연산자가 음수일 경우에만 결과값을 음수로 처리합니다.

	arithmetic.php	결과
1	$a=7;	
2	echo (-$a), " ";	-7
3	echo ($a**5), " ";	16807
4	echo ($a % 4.9), " ";	3
5	echo ($a % 4), " ";	3
6	echo (-$a % 4), " ";	-3
7	echo ($a % -4), " ";	3
8	echo (-$a % -4), " ";	-3

6.4 증감 연산자(Incrementing/Decrementing Operators)

어떤 변수를 1로 증가하거나, 1로 감소하는 연산을 수행하는 연산자입니다. 예를 들어 ++$a 연산은 $a 변수의 값을 1로 증가시키는 것이며 $a=$a+1 문장과 동일합니다. 증감 연산자는 변수의 뒤에 있을 수도 있습니다. $a++ 연산은 $a 변수값으로 먼저 처리후 1 증가시킵니다. 증감 연산자는 문자열에도 사용할 수 있습니다. “” 인용부호에 안에 있는 문자열을 1 증감할 수 있는데 영문자의 경우 알파벳 순서에서 1을 증감합니다.

연산자	사용예	의미
++	$b = ++$a	$a 변수 값을 1 증가하여 $b 변수에 저장 ($b=$a+1)
	$d = $c++	$c 변수 값을 $d 변수에 저장한 후 $c 변수값 1 증가 ($d=$c; $c=$c+1)
--	$b = --$a	$a 변수 값을 1 감소하여 $b 변수에 저장 ($b=$a+1)
	$d = --$c	$c 변수 값을 $d 변수에 저장한 후 $c 변수값 1 감소 ($d=$c; $c=$c-1)

incre_decre.php		결과
01	$a = 10;	
02	$b = ++$a;	
03	echo "a 변수 값 : $a", " ";	a 변수 값 : 11
04	echo "b 변수 값 : $b", " ";	b 변수 값 : 11
05	$c = 20;	
06	$d = $c++;	
07	echo "c 변수 값 : $c", " ";	c 변수 값 : 21
08	echo "d 변수 값 : $d", " ";	d 변수 값 : 20
09	$e = "30";	
10	$e--;	
11	echo "e 변수 값 : $e", " ";	e 변수 값 : 29
12	$f = "abc";	
13	++$f;	
14	echo "f 변수 값 : $f", " ";	f 변수 값 : abd

6.5 문자열 연결 연산자(String Operators)

여러 문자열들을 합치는 연산자로 .(dot) 기호를 사용합니다. 다음 코드는 $nation 변수와 $city 변수의 문자열을 하나의 문자열로 연결합니다.

dot.php		결과
01	$nation = "대한민국";	
02	$city = "원주시";	대한민국원주시
03	echo $nation . $city;	

6.6 비트 연산자(Bitwise Operators)

피연산자들의 값을 2진수로 바꿔서 각 비트끼리 계산하여 10진수 값으로 나타내는 연산자입니다.

연산자	사용예	의미
¦	$a ¦ $b	$a, $b 변수의 2진수 값을 각 비트끼리 or 연산
&	$a & $b	$a, $b 변수의 2진수 값을 각 비트끼리 and 연산
^	$a ^ $b	$a, $b 변수의 2진수 값을 각 비트끼리 xor 연산 (각 비트 값이 서로 다를 경우(배타적)에만 1로 연산)
~	~$a	$a 변수의 2진수 값을 0은 1로, 1은 0으로 연산
<<	$a << $b	$a 변수의 2진수 값을 $b 변수의 숫자만큼 왼쪽으로 이동하고 오른쪽을 0으로 채우고 연산
>>	$a >> $b	$a 변수의 2진수 값을 $b 변수의 숫자만큼 오른쪽 비트들을 버리고 연산

다음 코드에서 11~13번 라인은 변수의 값을 2진수로 지정하여 ^(xor) 연산을 수행한 결과입니다.

	bit_operator_1.php	결과
01	$a = 45; //이진수 00101101	
02	$b = 74; //이진수 01001010	
03	$or = $a ¦ $b;	
04	$and = $a & $b;	
05	$xor = $a ^ $b;	
06	$not = ~$a;	
07	echo "or 변수 : ".$or." ";	or 변수 : 111
08	echo "and 변수 : ".$and." ";	and 변수 : 8
09	echo "xor 변수 : ".$xor." ";	xor 변수 : 103
10	echo "not 변수 : ".$not." ";	not 변수 : -46
11	$d = 0b00101101;// 10진수 45	
12	$e = 0b01001010;// 10진수 74	
13	$f = $d ^ $e;	
14	echo $f;	103

<<, >> 연산자는 비트 이동 연산자(shift operator)라고 합니다.

	bit-operator-2.php	결과
01	$a = 45; //이진수 00101101	
02	$b = 3;	
03	$left_shift = $a << $b; // 이진수 00101101000	
04	$right_shift = $a >> $b; // 이진수 00101	360
05	echo $left_shift." ";//왼쪽으로 $b 비트 이동후 값	5
06	echo $right_shift." ";//왼쪽으로 $b 비트 이동후 값	

비트 연산자에 의한 2진수 계산을 위해서 Windows에서 제공하는 계산기 프로그램을 활용해 보세요. 이 계산기 프로그램을 프로그래머용으로 선택하면 2진수뿐만 아니라 8진수, 16진수, 비트 시프트 계산도 가능하며 각 진수들 간의 변환도 쉽게 할 수 있습니다.	

6.7 비교 연산자

피연산자들의 크기를 비교하는 연산자입니다. 연산의 결과는 반드시 참(true, 1) 또는 거짓(false, 0) 2가지 형태로만 나타납니다. 주로 if 문이나 while 문과 같이 논리적으로 판단해야 하는 문장에서 사용됩니다.

연산자	사용예	설명
==	$a == $b	$a와 $b 변수의 값이 같으면 true
===	$a === $b	$a와 $b 변수의 값과 데이터 타입이 같으면 true
!=	$a != $b	$a와 $b 변수의 값이 다르면 true
<>	$a <> $b	$a와 $b 변수의 값이 다르면 true
!==	$a !== $b	$a와 $b 변수의 값과 데이터 타입이 다르면 true
<	$a < $b	$a 변수 값이 $b 변수 값보다 작으면 true
>	$a > $b	$a 변수 값이 $b 변수 값보다 크면 true
<=	$a <= $b	$a 변수 값이 $b 변수 값보다 같거나 작으면 true
>=	$a >= $b	$a 변수 값이 $b 변수 값보다 같거나 크면 true
<=>	$a <=> $b	$a가 $b보다 크면 1. 같으면 0. 작으면 -1

PHP 언어에서는 데이터 타입 변환이 자동으로 이루어지기 때문에 비교 대상이 되는 수치 문자열은 수치로 변환되어 연산됩니다. 다만 연산할 때만 적용되며 변수의 원래 데이터 타입은 그대로 유지됩니다.

다음 코드에서 echo 문은 1 값으로 참으로 확인할 수 있고, 거짓 값은 출력되진 않지만 비교 연산의 결과값은 거짓입니다. var_dump() 함수를 이용하여 비교 연산의 참, 거짓을 true, false 값으로 확인할 수 있습니다.

comparison_operator.php		결과
01	$a = 10;	
02	$b = "10";	
03	echo ($a == $b)." ";	1
04	echo ($a != $b)." ";//false 면 출력되지 않음	
05	echo ($a === $b)." ";//false 면 출력되지 않음	
06	echo ($a !== $b)." ";	1
07	var_dump($a == $b);	bool(true)
08	var_dump($a != $b);	bool(false)
09	var_dump($a === $b);	bool(false)
10	var_dump($a !== $b);	bool(true)
11	var_dump(1 <=> 1);	int(0)
12	var_dump(1 <=> 2);	int(-1)
13	var_dump(2 <=> 1);	int(1)
14	echo(1 <=> 1);	0
15	echo(1 <=> 2);	-1
16	echo(2 <=> 1);	1

6.8 논리 연산자

피연산자의 논리값(참, 거짓)로 연산을 하며, 결과값도 논리값이 됩니다. 피연산자들은 변수뿐만 아니라, 수식도 가능합니다.

연산자	사용예	설명
and	$a and $b	$a, $b 변수의 논리값이 모두 true면 true (논리곱)
or	$a or $b	$a, $b 변수의 논리값중 하나만 true면 true (논리합)
xor	$a xor $b	$a와 $b 변수의 논리값이 서로 다르면 true (배타적 논리합)
!	!$a	$a 변수의 논리값을 역
&&	$a && $b	$a, $b 변수의 논리값이 모두 true면 true (논리곱)
¦¦	$a ¦¦ $b	$a, $b 변수의 논리값중 하나만 true면 true (논리합)

6.8.1 and와 &&, or와 ¦¦

and/or 연산자와 &&/¦¦ 연산자는 각각 동일한 기능을 하는 연산자입니다. 차이점은 연산 우선순위에 있습니다.

다음 코드에서 3번 라인 $c 변수의 논리값은 false가 되어야 하지만 true로 연산됩니다. and 연산자의 연산우선순위는 = 연산자보다 낮기 때문에 $c=$a 코드가 먼저 실행되고 끝나버립니다. 따라서 and 연산자를 사용하려면 4번 라인과 같이 ( ) 기호로 감싸서 우선순위를 높여야 합니다. 반면에 5번 라인에서와 같이 &&, ¦¦ 연산자들은 = 연산자보다 우선순위가 높기 때문에 $a&&$b 코드가 먼저 수행되어 결과값을 $e 변수에 할당합니다.

logical_operator.php		결과
1	$a = true;	
2	$b = false;	
3	$c = $a and $b;	
4	$d = ($a and $b);	
5	$e = $a && $b;	
6	var_dump($c);	bool(true)
7	var_dump($d);	bool(false)
8	var_dump($e);	bool(false)

6.9 실행 연산자(Execution Operators)

실행 연산자는 ~ 키에 해당되는 백틱 기호입니다. 실행 연산자는 쉘 명령어를 웹브라우저에서 실행되도록 하며, shell_exec() 함수와 동일한 기능을 합니다.

다음 코드는 윈도우 쉘 명령어의 실행 결과를 웹 브라우저에 출력하는 예제입니다.

excute.php		결과
1 2	$output = `dir`; echo "<pre>$output</pre>";	H 드라이브의 볼륨: Education H:\xampp\htdocs\source\05 디렉터리 2022-09-22 오후 02:32 . 2022-09-22 오후 02:32 .. 2022-07-13 오전 11:24 188 arithmetic.php 2022-09-21 오전 11:15 178 autotype.php 2022-07-14 오전 11:25 387 bit-operator.php 2022-07-09 오전 11:07 49 bool.php : : :
1 2	$output = shell_exec(`dir`;); echo "<pre>$output</pre>";	

6.10 3항 연산자(Ternary Operator)

3항 연산자는 if else 문이 1회만 사용될 경우와 동일한 수행을 하며 코드를 간결하게 작성하기 위해 사용됩니다.

3항 연산자	if 문
(조건문) ? (문장_A) : (문장_B)	if (조건문) { 문장_A } else { 문장_B }

다음 코드는 $point 변수의 값이 90 이상일 경우와 아닐 경우 각각에 대한 문자열을 출력하는 예제이며, 동일한 처리를 하는 if else 문의 코드입니다.

ternary.php	if_else_1.php	결과

$point = 88;
echo ($point >= 90)? "A 학점": "B 학점";
	$point = 88;
	if ($point >= 90) {
	echo "A 학점";
	} else {
	echo "B 학점";
	}
		B 학점