7. 배열

배열은 동일한 변수명으로 여러 개의 값들을 표현하기 위해 사용되며 일종의 데이터 타입니다. 여러 개의 값들은 인덱스라고 불리는 키 값으로 구분하게 되며, 인덱스에는 연속적인 숫자 인덱스와 문자열 인덱스가 있습니다. 인덱스를 문자열로 구성한 배열을 연관 배열(associative array)라고 합니다.

7.1 1차원 배열

배열의 생성 방법은 다음과 같이 array 키워드를 이용하는 방법과 array 키워드 없이 [ ] 기호를 이용한 방법 있습니다.

연관 배열에서는 Key와 Value 사이에는 => 기호를 이용하여 연결합니다.

숫자 인덱스	`$ary_name = array(value_1, . . . ., value_n);`
	`$ary_name = [value_1, . . . ., value_n];`
연관 배열	`$ary_name = array("key" => value_1, . . . ., "key" => value_n);`
	`$ary_name = ["key" => value_1, . . . ., "key" => value_n];`

(1) 숫자 인덱스 배열 생성 및 출력

인덱스의 번호는 다른 프로그래밍 언어와 마찬가지로 0부터 시작합니다.

다음 코드는 배열을 생성하여 출력하는 예제입니다.

array_1.php		결과
1	`$ary1 = array(12, 34, 56);`	
2	`$ary2 = [12, 34, 56];`	
3	`$ary3 = array("Kim", "Lee", "Park");`	
4	`$ary4 = ["Kim", "Lee", "Park"];`	
5	`print_r ($ary1[0]);`	12
6	`print_r ($ary2);`	Array([0] => 12 [1] => 34 [2] => 56)
7	`echo ($ary3[2]);`	Park
8	`echo ($ary4);`	Warning: Array to string conversion in

- 5번 라인 : print_r 함수로 $ary1 배열 변수의 첫 번째(인덱스 0) 원소 출력
- 6번 라인 : $ary2 배열 변수의 모든 원소 출력
- 7번 라인 : echo 함수로 $ary3 배열 변수의 세 번째(인덱스 2) 원소 출력.
- 8번 라인 : echo 함수는 배열의 전체 내용을 출력하지 못함.

(2) 연관 배열의 생성

연관 배열의 인덱스는 문자열이며 일반적으로 원소의 값과 연관된 단어로 표기합니다. 인덱스와 값은 => 기호를 이용하여 연결합니다. 또한 인덱스 단어는 인용부호를 이용하여 표기하며 단일, 이중 모두 가능합니다.

array_2.php		결과
1	$ary1 = array("Kor"=>89, "Eng"=>86, "Math"=>91);	
2	$ary2 = ['Kor'=>89, 'Eng'=>86, 'Math'=>91];	
3	$ary3 = array("Company"=>"Kia", "Model"=>"Niro", "Year"=>2022);	86
4	$ary4 = ['Company'=>"Kia", 'Model'=>"Niro", 'Year'=>2022];	Array ([Kor] => 89 [Eng] => 86 [Mat] => 91)
5	print_r ($ary1["Eng"]);	Niro
6	print_r ($ary2);	Array ([Company] => Kia [Model] => Niro [Year] => 2022)
7	print_r ($ary3["Model"]);	
8	print_r ($ary4);	

- 1~4번 라인 : 인용부호를 이용하여 연관 배열의 인덱스 표기
- 5번 라인 : print_r 함수로 $ary1 배열 변수의 Eng 인덱스 원소 출력
- 6, 9번 라인 : $ary2 배열 변수의 모든 원소 출력
- 7번 라인 : echo 함수로 $ary3 배열 변수의 Model 인덱스 원소 출력.

7.2 2차원 배열

1차원 배열이 여러 줄 모여서 구성된 것이 2차원 배열입니다. 1차원 배열과 마찬가지로 array 키워드를 이용하는 방법과 array 키워드 없이 [ ] 기호를 이용한 방법 있습니다.

숫자 인덱스	$ary_name = array(array (value_1, ..., value_n), array (value_1, ..., value_n), : : array (value_1, ..., value_n));
	$ary_name = [[value_1, ..., value_n], [value_1, ..., value_n], : : [value_1, ..., value_n]];
연관 배열	$ary_name = array("key1" => array (key11 => value_1, . .value_n), "key2" => array (key11 => value_1, . .value_n), : : "keym" => array (key11 => value_1, . .value_n));
	$ary_name = ["key1" => [key11 => value_1, . . , key1n => value_n], ["key2" => [key11 => value_1, . . , key1n => value_n], : : ["keym" => [key11 => value_1, . . , key1n => value_n]];

예제를 통해 2차원 배열을 살펴보겠습니다.
다음은 학생 1명의 성적표를 표로 나타낸 것입니다. 이렇게 학생이름과 과목명 3개가 있다고 가정합니다.

과목 / 이름	Kor	Eng	Math
student_1	89	86	91

이 표를 $student_1 배열명으로 선언하여 1차원 배열로 표현하면 다음과 같습니다.

```
$student_1 = array("Kor"=>89, "Eng"=>86, "Math"=>91);
```

많은 학생들이 성적표가 있을대 다음과 같은 표 형태로 표현할 수 있는데, student_1 부터 student_n 까지 1차원 배열명이 n개가 필요합니다.

과목 / 이름	Kor	Eng	Math
student_1	89	86	91
student_2	86	79	81
:	:	:	:
student_n	74	69	86

이 표를 1차원 배열로 각각 선언하면 다음과 같습니다.

```
$student_1 = array("Kor"=>89, "Eng"=>86, "Math"=>91);
$student_2 = array("Kor"=>86, "Eng"=>79, "Math"=>81);
    :                    :          :           :
$student_n = array("Kor"=>89, "Eng"=>86, "Math"=>91);
```

그런데 변수명이 n개까지 많아져서 프로그래밍하는데 복잡해집니다. 따라서 배열명 1개로 2차원 배열로 재구성하면 다음과 같이 선언할 수 있습니다.

```
$student = array( "student_1" => array ("Kor"=>89, "Eng"=>86, "Math"=>91),
                  "student_2" => array ("Kor"=>86, "Eng"=>79, "Math"=>81),
                                 :         :          :
                  "student_n" => array ("Kor"=>74, "Eng"=>69, "Math"=>86)
                );
또는
$student = [ "student_1" => ["Kor"=>89, "Eng"=>86, "Math"=>91],
             "student_2" => ["Kor"=>86, "Eng"=>79, "Math"=>81],
                    :         :           :
             "student_n" => ["Kor"=>89, "Eng"=>86, "Math"=>91]
           ];
```

$student는 배열명이고 student_1, student_2 등은 학생 이름으로 구성됩니다.

다음 코드는 연관 배열을 $student 배열명을 선언하고 foreach 구문으로 출력하는 예제입니다.

array_std_1.php

```
01 $student = array("student_1" => array ("Kor"=>89, "Eng"=>86, "Math"=>91),
02                  "student_2" => array ("Kor"=>86, "Eng"=>79, "Math"=>81),
03                  "student_3" => array ("Kor"=>74, "Eng"=>69, "Math"=>86)
04                 );
05 echo "<table border=1><tr><th>Name<th>Kor<th>Eng<th>Math";
06 foreach ($student as $key1 => $value1) {
07     echo "<tr><td>{$key1}";
08         foreach ($value1 as $key2 => $value2) {
09                 echo "<td>{$value2}";
11         }
12         echo "<br>";
13 }
```

위 코드의 수행 결과는 다음과 같습니다.

Name	Kor	Eng	Math
student_1	89	86	91
student_2	86	79	81
student_n	74	69	86

- 1~4번 라인 : 3명 학생의 성적을 연관 2차원 배열로 구성
- 6번 라인 : $student 배열명을 기준으로 각 행에 대한 키와 값을 추출

$key1	$value1
student_1	array ("Kor"=>89, "Eng"=>86, "Math"=>91)
student_2	array ("Kor"=>86, "Eng"=>79, "Math"=>81)
student_3	array ("Kor"=>89, "Eng"=>86, "Math"=>91)

- 7번 라인 : 학생 이름에 해당되는 $student 배열의 $key 값 출력
- 8번 라인 : student_1, student_2, student_3 각각을 기준으로 키와 값을 추출.

위 예제에서 중요한 부분은 8번 라인입니다. 2차원 배열이기 때문에 다음 표와 같이 student 배열명의 $value1 값은 $key2와 $value2의 배열명 역할을 합니다.

student_1	
$key2	$value2
Kor	89
Eng	86
Math	91

student_2	
$key2	$value2
Kor	86
Eng	79
Math	81

student_3	
$key2	$value2
Kor	74
Eng	69
Math	86

배열이 연관 배열이 아닐 경우에는 for 구문과 숫자 인덱스를 이용하여 동일한 결과를 수행합니다. 학생 이름 값인 $student_1 등을 값으로 지정하여 각 행을 선언합니다.

array_std_2.php

```
01 $student = array(array ("student_1", 89, 86, 91),
02                  array ("student_2", 86, 79, 81),
03                  array ("student_3", 74, 69, 86)
04                 );
05 echo "<table border=1><tr><th>이름<th>국어<th>영어<th>수학";
06 for ($i=0; $i<count($student); $i++) {
07         echo "<tr>";
08         for ($j=0; $j<count($student[0]); $j++) {
09                 echo "<td> {$student[$i][$j]} ";
11         }
12         echo "<br>";
13 }
```

- 6번 라인 : count() 함수를 이용하여 $student 배열의 1차 원소 수(3개)만큼 for 문 수행
- 8번 라인 : count() 함수를 이용하여 $student[0] 배열의 원소 수(4개)만큼 for 문 수행

다음 코드는 숫자 인덱스 배열을 foreach 구문을 이용하여 동일한 작업을 수행하는 예제입니다.

array_std_3.php

```
01 $student = array(array ("student_1", 89, 86, 91),
02                  array ("student_2", 86, 79, 81),
03                  array ("student_3", 74, 69, 86)
04                 );
05 echo "<table border=1><tr><th>이름<th>국어<th>영어<th>수학";
06 foreach ($student as $key1 => $value1) {
07     echo "<tr>";
08         foreach ($value1 as $key2 => $value2) {
09                 echo "<td>{$value2}";
11         }
12         echo "<br>";
13 }
```

- 6번, 8번 라인 : 배열 원소의 수를 계산하지 않고 foreach 구문으로 항목이 없을 때까지 값 추출

8. 객체

객체는 자체적으로 속성(변수)과 기능(메서드, 함수)들로 구성된 단위 프로그램입니다. 함수는 하나의 기능만을 수행하고, 객체는 여러 개의 메서드들로 구성될 수 있는 때문에 함수보다 많은 기능을 수행할 수 있습니다.
다른 프로그래밍 언어와 마찬가지로 PHP 언어에도 객체에 대해 많은 개념들이 있습니다. 본 도서에서는 PHP 객체의 기본적인 동작과 사용법에 대해서만 살펴봅니다.

8.1 클래스

클래스는 객체를 만들어 내기 위한 설계도입니다. 이 설계도에는 속성(변수)과 기능(메서드, 함수)들이 정의되며, 이 설계도를 만들기 위해 필요한 키워드가 class입니다.
다음과 같이 class 키워드를 통해 클래스를 만들려면 클래스 이름, 속성, 기능을 정의합니다. 속성은 일종의 변수이며 필드, 멤버변수로 불리기도 합니다. 기능은 함수의 역할을 하며 객체지향에서는 메서드라고 합니다.
속성은 초기값 설정도 가능하며 메서드의 param1 등 파라메터는 선택 항목입니다. public 키워드로 선언된 속성은 클래스의 외부에서도 접근할 수 있습니다.

```
class cls_name {
  public $property_1;
  public $property_2;
  function method_1() {
          처리 코드
  }
  function method_2() {
          처리 코드
  }
}
```

```
class object_name {
  public $property_1 = "초기값";
  public $property_2 = "초기값";
  function method_1(param1, param2, ..) {
          처리 코드
  }
  function method_2(param1, param2, ..) {
          처리 코드
  }
}
```

8.2 객체의 생성

정의된 클래스를 프로그램 코드에서 사용하려면 객체를 생성해야 합니다. 클래스로부터 객체를 만드는 과정을 클래스의 인스턴스화라고 하며, 클래스로부터 만들어진 객체를 그 클래스의 인스턴스라고 합니다. 따라서 객체와 인스턴스는 같은 의미로 사용되기도 합니다.
인스턴스는 일종의 변수이며 new 키워드를 사용해 생성합니다.

```
$instance_name = new class_name;
```

클래스를 정의할 때 속성에 지정된 초기값은 수정되지 않습니다. 이 초기값은 인스턴스에 의해 변경될 수 있지만, 클래스에 설정된 초기값은 그대로 유지됩니다.

8.3 객체의 속성과 메서드 사용

객체(인스턴스)의 속성과 메서드를 사용하기 위해서는 -> 기호를 사용합니다.

```
$instance_name->property_1;
$instance_name->method_1();
```

주의할 점은 클래스에서 속성을 정의할 때는 $ 기호를 표기하지만 인스턴스에서 사용할 때는 $ 기호를 표기하지 않습니다.

다음 코드는 속성의 초기값과 메서드로 구성된 baseball 클래스를 정의하는 예제입니다. baseball 클래스의 인스턴스를 생성하여 초기값을 출력합니다.

object_1.php		결과
01	class baseball {	
02	public $name = "홍길동";	
03	public $team = "LG Twins";	
04	function hello() {	
05	echo "KBO 기록실 ";	
06	}	
07	}	
08	$player = new baseball();	
09	echo $player->hello();	KBO 기록실
10	echo $player->name;	홍길동
11	echo $player->team;	LG Twins

- 1~7번 라인 : baseball 클래스 생성
- 2~3번 라인 : $name, $team 속성에 초기값 지정
- 8번 라인 : baseball 클래스의 인스턴스 변수인 $player 생성
- 9~11번 라인 : $player의 hello() 메서드 실행 결과 출력
- 10~11번 라인 : $player 인스턴스에 저장된 객체의 초기값 출력

다음 코드는 2개의 메서드로 구성된 player 클래스를 활용하는 예제입니다. 2개의 인스턴스를 생성하여 상황에 적절한 메서드를 호출합니다.

object_2.php	
01	class baseball {
02	public $name;
03	public $team;
04	function hit_avg($atbat, $hit) {
05	return $hit/$atbat;
06	}
07	function er_avg($inning, $er) {
08	return ($er*9)/$inning;
09	}
10	}
11	$hitter = new baseball;
12	$pitcher = new baseball;
13	$hitter->name = "홍길동 타자";
14	$hitter->team = "LG Twins";
15	$avg = $hitter->hit_avg(257, 97);
16	$pitcher->name = "이순신 투수";
17	$pitcher->team = "두산 베어스";
18	$er = $hitter->er_avg(155, 58);
19	printf("{$hitter->name}({$hitter->team}) 타율 : %0.3f", $avg);
20	printf("{$pitcher->name}({$pitcher->team}) 방어율 : %2.2f", $er);
결과	홍길동 타자(LG Twins) 타율 : 0.377 이순신 투수(두산 베어스) 방어율 : 3.37

- 1~10번 라인 : baseball 클래스 생성
- 2~3번 라인 : 속성 선언
- 4~6번 라인 : 2개의 인자를 가지는 hit_avg() 메서드 정의
- 7~9번 라인 : 2개의 인자를 가지는 er_avg() 메서드 정의
- 11~12번 라인 : baseball 클래스의 객체 변수(인스턴스)인 $hitter, $pitcher 생성
- 13~15번 라인 : $hitter 객체의 속성 값을 설정하고 hit_avg() 메서드 호출
- 16~18번 라인 : $pitcher 객체의 속성 값을 설정하고 er_avg() 메서드 호출
- 19~20번 라인 : $hitter, $pitcher 객체들의 속성과 메서드의 실행결과 출력