Chapter 07 함수

함수는 자주 사용되는 코드 블록을 미리 만들어 놓은 작은 단위의 프로그램입니다. 이렇게 만들어 놓은 함수들을 다른 프로그램들에서 함수 이름만 호출하여 손쉽게 활용할 수 있습니다.
함수에는 프로그램 개발자가 만드는 사용자 정의 함수(User Defined Function)와 PHP 시스템에서 제공하는 내장함수(Built-in Function)으로 나뉩니다.
PHP 언어의 함수 설명에 앞서 프로그래밍 언어에서 사용되는 용어인 Formal Parameter, Actual Argument, Call by value, Call by reference 등에 살펴봅니다.

1. Formal Parameter vs Actual Argument

1.1 Parameter vs Argument

parameter(매개변수)와 argument(실제값)의 명칭을 구분하지 않고 사용하지만, parameter는 함수의 정의 부분에 나열되어 있는 변수들을 의미하며, argument는 함수를 호출할 때 전달되는 실제 값을 의미합니다.

parameter : 함수정의 (a, b)	argument : 함수호출 (a, b)

function add (a, b)
{
  return a + b;
}
	sum(a, b);

add() 함수의 정의 부분에 있는 a, b는 parameter라고 표현하며, 함수 호출 부분에 있는 a, b는 argument라고 표현하는 것이 일반적입니다.

1.2 Formal vs Actual

또한 parament, argument 단어의 의미를 보다 명확히 하기 위해 각각 formal, actual 단어를 추가하기도 합니다. 즉 formal parameter와 actual argument로 표현합니다.
formal parameter는 함수를 정의하는 형식(formal)에 사용되는 변수(parameter)이고, actual argument는 함수를 호출할 때 실제(actual)로 전달되는 값(argument)라고 이해하면 됩니다.

영어 단어	한글 단어	의미
Formal Parameter	매개변수, 형식인자	함수 정의에 사용되는 변수명
Actual Argument	실인자(인수), 전달인자(인수), 전달값	함수 호출에 사용되는 실제값

※ F와 P 발음이 유사하다고 생각한다면 이해하기 편리합니다.

2. 값 전달(call by value) vs 주소 전달(call by address)

chapter 05의 2절 변수에 대한 고찰에서도 설명했듯이 변수가 생성되면 = 기호를 기준으로 왼쪽 변수명(Left_value)은 주소(address) 또는 참조(Reference)이며, 오른쪽은 변수에 저장되는 값이라고 했습니다.

Left_value = Right_value	$score = 99

함수를 호출할 때도 이 개념이 적용됩니다. 함수의 argument로 변수명이 사용될 때 2가지 방식으로 전달되는데, 『값(Right_value)에 의한 호출』과 『주소(address, reference)에 의한 호출』입니다.

『값(Right_value)에 의한 호출』을 call by value라고 하며, 『주소(address, reference)에 의한 호출』을 call by address(reference)라고 하며, 프로그래밍 언어에서는 2가지 방식을 모두 지원합니다.

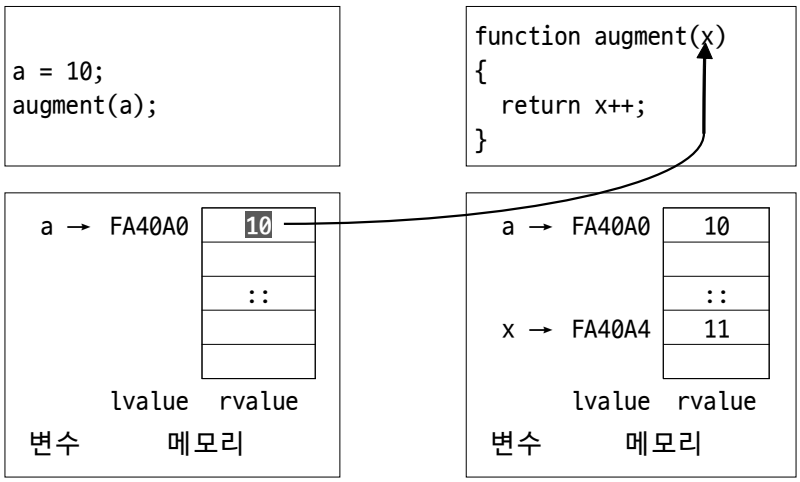
다음 코드를 이용하여 call by value, call by address 방식을 비교하여 설명합니다.

함수 정의	함수 호출

function augment(x)
{
  return x++;
}
	a = 10;
	augment(a);

2.1 값에 의한 호출(call by value)

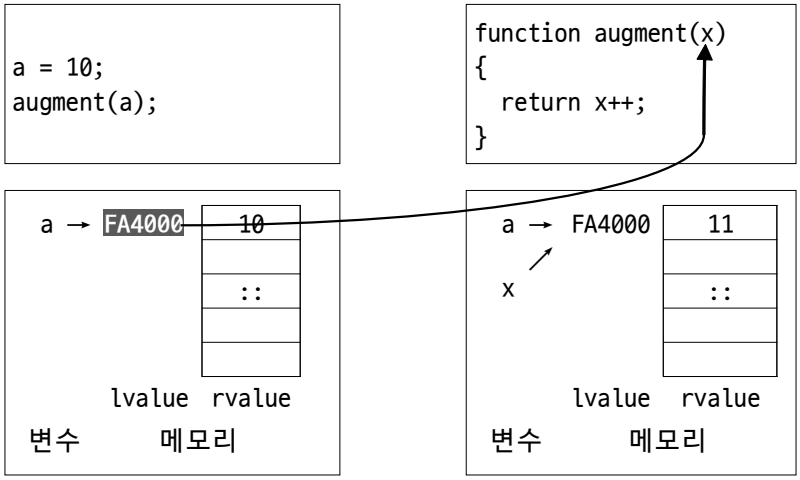
call by value 방식은 함수를 호출할 때 argument 변수에 저장된 실제 값을 parameter 변수에 전달하여 함수를 실행하는 방식입니다.



위의 그림에서 augment() 함수를 호출할 때 argument 변수로 a를 이용하면 a에 저장된 10이라는 실제 값이 parameter 변수 x에 전달됩니다. x의 값을 1로 증가시킨 후 메모리의 변수 값의 상황을 살펴보면 a 변수의 값은 변화가 없습니다. 즉 argument 변수 a와 parameter 변수 x는 다른 변수입니다.

2.2 주소에 의한 호출(call by address)

call by value 방식은 함수를 호출할 때 argument 변수의 주소 값을 parameter 변수에 전달하여 함수를 실행하는 방식입니다.



위의 그림에서 augment() 함수를 호출할 때 argument 변수로 a를 이용하면 a 변수의 주소 값이 parameter 변수 x에 전달됩니다. 이렇게 되면 argument 변수 a와 parameter 변수 x는 같은 주소를 공유하게 됩니다. 따라서 x의 값을 1로 증가시킨 후 메모리의 변수 값의 상황을 살펴보면 a 변수의 값도 변화가 있습니다. 즉 argument 변수 a와 parameter 변수 x는 같은 메모리를 사용하는 변수입니다.

Call By Value	Call By Reference
- argument의 실제 값을 parameter에 전달 - 함수 내부에서 paramenter 값이 변경되도 외부 값은 바뀌지 않음 - 장점 : argument 값에 영향을 주지 않기 때문에 안전함 - 단점 : 메모리 사용량이 늘어남	- argument의 주소 값을 parameter에 전달 - 함수 내부에서 paramenter 값이 변경되면 외부의 argument 값도 바뀜. - 장점 : 값 복사가 없으므로 빠름 - 단점 : argument 값에 영향을 주기 때문에 감안하여 코딩해야 함

3. 사용자(개발자) 정의 함수(User-Defined Function)

3.1 사용자 정의 함수 생성

PHP 언어에서는 function 키워드를 이용하여 함수명을 정의합니다. 함수명은 변수명을 만드는 방식과 동일하게 영문자와 _ 기호를 활용하여 만듭니다.
다음과 같이 함수 이름으로 user_func로 지정하고, user_func 함수 내부에서 사용될 변수(매개변수, parameter)도 지정할 수 있습니다.

```
function user_func($param_1, $param_2, ... $param_n)
{
     코드 블록;
}
```

3.2 함수의 사용(호출)

미리 정의된 함수는 함수명을 호출하여 사용할 수 있고, 필요한 경우 매개변수에 전달될 값을 지정할 수 있습니다.
user_func 함수에 전달인수(실인수, argument) 값을 이용하여 다음과 같이 함수를 호출합니다.

```
user_func($arg_1, $arg_2, ... $arg_n);
```

3.3 매개변수(parameter)와 초깃값

함수를 정의할 때 매개변수는 없을 수도 있으며, 매개변수에 초깃값을 미리 지정할 수 있습니다.
다음 코드에서 sum 함수의 매개변수는 2개이고 2번째 매개변수는 초깃값이 할당되어 있습니다.

	param_init.php	결과
1	function sum($param1, $param2=10)	
2	{	
3	$result = $param1 + $param2;	
4	return $result;	
5	}	
6	echo sum(10);	20
7	echo sum(10,20);	30

- 6번 라인 : 실인수 1개로 sum 함수를 호출하면 초깃값으로 함수를 수행
- 7번 라인 : 실인수 2개로 sum 함수를 호출하면 초깃값은 무시되고 실인수 값으로 함수를 수행

3.4 함수 실행과 return 문

return 문은 함수 실행의 결과를 호출한 코드에 전달하는 기능을 하지만 return 문없이 실행 결과를 전달할 수도 있습니다. 주의할 점은 return 문은 함수를 종료하는 것이기 때문에 return 문 이하의 코드들은 무시된다는 점입니다.

다음 코드에서 sumB 함수는 return 문 없이 결과값을 전달하고 있습니다. sumC 함수는 11번 라인의 return 문으로 함수는 종료됩니다. 따라서 곱하기 연산을 하는 12번 라인의 return 문은 실행되지 않습니다.

	return.php	결과
01	function sumA($param1, $param2)	
02	{	
03	return($param1 + $param2);	
04	}	
05	function sumB($param1, $param2)	
06	{	
07	echo($param1 + $param2);	
08	}	
09	function sumC($param1, $param2)	
10	{	
11	return($param1 + $param2);	
12	return($param1 * $param2);	
13	}	
14	echo sumA(10,20);	30
15	$result = sumB(10,20);	30
16	echo sumC(10,20);	30

3.5 call by value

PHP 언어에서는 함수의 값 전달 방식은 기본적으로 call by value 방식을 사용합니다.
다음의 코드를 보겠습니다. 5번 라인에서 $arg 변수 값을 1로 설정하고, 6번 라인에서 addFive() 함수를 call by value 방식으로 호출합니다. $arg 변수의 1 값이 $param 매개변수에 전달되어 함수가 수행됩니다. 또한 7번 라인의 $arg 변수 값에는 변화가 없습니다.
당연하게도 $arg 변수와 $param 변수는 전혀 다른 변수입니다. call by value 방식으로 함수가 수행되기 때문이지요.

	call_value.php	결과
1	function addFive($param)	
2	{	
3	return($param = $param + 5);	
4	}	
5	$arg=1;	
6	echo addFive($arg);	6
7	echo $arg;	1

3.6 call by refrence

다음 코드를 보겠습니다. call_value.php 코드와 동일지만 1번 라인의 매개변수 앞에 & 기호가 추가됩니다. & 기호는 주소를 나타낸다고 설명한 바 있습니다. 함수를 정의할 때 &$param 형태로 매개변수를 지정하게 되면 $arg 변수의 주소 값을 전달받게 됩니다. 이렇게 되면 $param 변수와 $arg 변수는 메모리 주소를 공유하기 때문에 각 변수들의 값 변경은 다른 변수에도 영향을 주게 됩니다. 따라서 7번 라인의 $arg 변수 값은 3번 라인에서 변경된 $param 변수 값과 동일하게 됩니다.

	call_address.php	결과
1	function addFive(&$param)	
2	{	
3	return($param = $param + 5);	
4	}	
5	$arg=1;	
6	echo addFive($arg);	6
7	echo $arg;	6

이렇게 PHP 언어에서는 함수 호출시 call by reference 방식으로 수행하려면 매개변수에 & 기호를 이용합니다.

비록 call_address.php 코드처럼 함수가 수행되기를 의도하지는 않았겠지만, call by address 방식과 call by value 방식의 이해는 매우 중요합니다.

4. 내장 함수(Built-in Function)

PHP 언어에서 제공하는 내장 함수는 매우 방대하여 본 도서에서는 자주 사용되는 함수를 위주로 설명합니다.
모든 내장 함수는 매개변수를 가지게 되며, 수행된 결과를 반환합니다. 이때 매개변수와 반환 값의 데이터 타입이 정의되어 있습니다.
예를 들어 ceil 함수의 경우 매개변수는 int 또는 float(int¦float)의 숫자($num)형입니다. 또한 ceil 함수의 반환값은 float 형입니다.

```
float ceil(int¦float $num)
```

다른 예로 ord 함수의 경우에는 매개변수는 $character(문자열중 하나의 문자)이며 반환값은 int 형입니다.

```
int ord(string $character)
```

앞으로 설명될 내장 함수의 기본 형식은 위의 예와 같이 표현합니다.

5. 수치 함수

5.1 수학 함수

수학 함수는 삼각 함수, 지수 함수, 로그 함수 등 복잡한 수학 공식을 간편하게 처리할 수 있는 함수입니다.
대표적인 수학 함수를 다음 표에 제시하였으며, 더 많은 함수들은 다른 자료를 활용하기 바랍니다.

함수	설명
float sin(float $num)	$num의 사인값 반환
float cos(float $num)	$num의 코사인값 반환
float tan(float $num)	$num의 탄젠트값 반환
float exp(float $num)	$num의 e 지수값 반환
float log(float $num)	$num의 자연 로그값 반환
float log10(float $num)	$num의 상용 로그값 반환
int¦floatabs(int¦float $num)	$num의 절대값 반환
float pi()	Pi 값 반환
float rad2deg(float $num)	$num 라디언의 각도값 반환
float deg2rad(float $num)	$num 각도의 라디언값 반환
float sqrt(float $num)	$num의 제곱근 반환

다음 코드는 대표적인 수학 함수들의 사용 사례입니다.

	math_1.php	결과
01	echo sin(30);	-0.30481062110222
02	echo cos(60);	-0.95241298041516
03	echo tan(45);	1.6197751905439
04	echo exp(10);	22026.465794807
05	echo log(2);	0.69314718055995
06	echo log10(100);	2
07	echo abs(-11);	11
08	echo pi();	3.1415926535898
09	echo rad2deg(1);	57.295779513082
10	echo deg2rad(57.295);	0.99998639493015
11	echo pow(3, 7);	2187
12	echo sqrt(5);	2.2360679774998

5.2 소수점 함수

소수점 처리를 위한 함수이며 ceil 함수는 소수점이 포함된 정수의 상위 정수값을 계산하고, floor 함수는 하위 정수값을 계산합니다. round 함수는 일반적인 반올림 함수입니다.

함수	설명
float ceil(int¦float $num)	$num의 상위 올림 정수값 반환
float floor(int¦float $num)	$num의 하위 내림 정수값 반환
float round(int¦float $num, int $precision)	$num의 반올림 정수값 반환

다음 코드에서 round 함수는 소수점 자리수 단위 반올림이 가능합니다.

	math_2.php	결과
1	echo ceil(5.11);	5
2	echo floor(5.95);	5
3	echo round(5.57);	6
4	echo round(5.54, 1);	5.5

5.3 최소값, 최대값

함수의 인수로 나열된 숫자중에서 최소값, 최대값을 찾아냅니다.

함수	설명
mixed min(mixed $value, mixed ...$values)	인수 목록의 최소값 반환
mixed max(mixed $value, mixed ...$values)	인수 목록의 최대값 반환

다음 코드에서 배열 원소의 최소값, 최대값을 찾아내기 위해 배열을 함수의 인수로 사용 가능합니다.

	math_3.php	결과
01	echo min(3, 1, 6, 4, 7, 2);	1
02	echo min(array(3, 1, 6, 4, 7, 2));	1
03	echo max(3, 1, 6, 4, 7, 2);	7
04	echo max(array(3, 1, 6, 4, 7, 2));	7

5.4 진수 함수

10진수(decimal)를 2진수(binary), 8진수(octal), 16진수(hexdecimal)로 변환하기 위한 함수입니다.

함수	설명
string decbin(int $num)	$num(10진수)을 2진수로 반환
string decoct(int $num)	$num(10진수)을 8진수로 반환
string dechex(int $num)	$num(10진수)을 16진수로 반환
int¦float bindec(string $binary_string)	$binary_string을 10진수로 반환
int¦float octdec(string $octal_string)	$octal_string을 10진수로 반환
int¦float hexdec(string $hex_string)	$hex_string을 10진수로 반환

10진수로 변환하는 함수들의 인수는 기본적으로 문자열로 전달해야 합니다. 다만 2진수와 8진수는 문자열이 아니어도 처리되지만 16진수는 반드시 문자열로 전달해야 합니다.

math_4.php		결과
01	echo decbin(13)." ";	1101
02	echo decoct(13)." ";	15
03	echo dechex(13)." ";	d
04	echo bindec("101011")." ";	43
05	echo octdec("714")." ";	460
06	echo hexdec("ad7")." ";	2775

5.5 숫자의 출력

number_format() 함수는 일련의 숫자에 소수점, 콤마 등을 삽입하기 위해 사용되는 함수입니다.

```
string number_format(float $num, 소수점자리수, 소수점기호, 천단위기호):
```

다음 코드에서 number_format() 함수는 기본적으로 천단위 콤마가 표시됩니다. 4번 라인에서 2번째(반드시 정수 표기), 3번째 인수를 이용하여 소수점의 위치와 모양을 달리 표현했습니다. 5번 라인은 천단위 기호를 @로 변경하여 출력했습니다.

math_5.php		결과
01	$num = 123456798;	
02	echo number_format($num);	123,456,798
03	echo number_format($num, "2");	123,456,798.00
04	echo number_format($num, "3", "_");	123,456,798_000
05	echo number_format($num, "0", "", "@");	123@456@798

6. 문자열 함수

문자열을 다양한 형태로 재구성하는데 사용하는 함수입니다. 대표적인 문자열 함수는 다음과 같습니다.

함수	설명
string chr($ascii_num)	$ascii_num(10진수) 값을 문자로 반환
int ord(string $character)	$character 문자를 ASCII(10진수) 값으로 반환
string nl2br(string $string)	$string 문자열의 ₩n 을 태그로 반환
array explode(string $separator, string $string)	$string 문자열을 $separator 기호로 분리하여 분리된 분리된 문자열들은 배열로 반환
string implode(string $separator, array $array)	$array 배열의 문자열을 $seperator 기호로 결합하여 문자열로 반환
int strlen(string $string)	$string 문자열의 길이를 바이트 단위로 반환

6.1 nl2br() 함수 : 웹브라우저에서 줄바꿈

다음 코드에서 3번 라인 문자열 5번 라인의 nl2br(newline to br) 함수를 이용하여 \n 기호를
 태그로 변환하여 문자열을 재구성합니다.

string_1.php		결과
01	echo chr(65);	A
02	echo ord("a");	97
03	$string = "PHP \n Web Site";	
04	echo $string;	PHP Web Site
05	echo nl2br($string);	PHP Web Site

6.2 explode() 함수 : 문자열 분리

다음 코드에서 3번 라인의 explode 함수는 공백을 기준으로 $string1 변수의 문자열을 분리하여 $ary 배열에 저장합니다. 6번 라인에서는 / 기호를 기준으로 $string2 변수의 문자열을 분리하여 배열 형태로 저장합니다.

string_2.php		결과
01	$string1 = "PHP JSP ASP";	
02	$string2 = "PHP/JSP/ASP";	
03	$ary = explode(" ", $string1);	$ary[0]=PHP,$ary[1]=JSP, $ary[2]=ASP
04	echo $ary[0], $ary[1], $ary[2];	array(3) { [0]=> string(3) "PHP" [1]=>
05	var_dump (explode("/", $string2));	string(3) "JSP" [2]=> string(3) "ASP" }

6.3 implode() 함수 : 문자열 결합

다음 코드에서 2번 라인의 implode 함수는 $ary 배열의 원소들을 <= 기호를 이용하여 하나의 문자열로 반환합니다. 또한 strlen 함수를 이용하여 $language 변수에 저장된 문자의 개수를 반환합니다.

string_3.php		결과
01	$ary = ['PHP', 'JSP', 'ASP'];	
02	$language = implode("<=", $ary);	
03	echo $language;	PHP<=JSP<=ASP
04	echo strlen($language);	13

7. 날짜/시간 함수

날짜와 시간 정보를 반환하는 함수입니다. 이런 정보들은 timestamp를 계산하여 나타내는데, timestamp 값은 Unix 시스템의 원년인 『1970년 1월 1일 0시 0분 0초』에서 1초 단위로 계산된 값입니다. Unix 시스템의 원년부터 프로그램이 실행되는 시점까지의 초단위 값은 time() 함수를 통해 확인됩니다.

date() 함수는 timestamp 값을 이용하여 PHP 프로그래밍이 실행되는 시점의 날짜를 다양한 형식으로 출력합니다. date() 함수의 기본 형식은 다음과 같습니다.

```
string date(string $format)
```

data() 함수로 출력되는 형식($format)은 다음과 같습니다. 년, 월, 시, 분, 초 등을 다양한 형태를 출력하기 위해 형식 문자를 이용하는데 인용부호로 감싸서 표현합니다.

구분	형식 문자	설명	출력 형태
날짜 요일	d	날짜를 2자리 숫자로 표현(leading 0 포함)	01 ~ 31
	j	날짜를 숫자로 표현	1 ~ 31
	D	요일을 3개 알파벳으로 표현	Mon ~ Sun
	l	요일을 영어 단어로 표현	Monday ~ Sunday
	N	요일을 숫자로 표현(1은 월요일)	1 ~ 7
	w	요일을 숫자로 표현(0은 일요일)	0 ~ 6
월	F	월을 영어 단어로 표현	January ~ December
	m	월을 2자리 숫자로 표현(leading 0 포함)	01 ~ 12
	n	월을 숫자로 표현	1 ~ 12
	M	월을 3개 알파벳으로 표현	Jan ~ Dec
	t	월의 일수를 표현	28 ~ 31
연도	Y	연도를 4자리 숫자로 표현	1999, 2022
	y	연도를 2자리 숫자로 표현	01, 22
시간	a	오전, 오후를 소문자로 표현	am, pm
	A	오전, 오후를 대문자로 표현	AM, PM
	g	시간을 12시간 단위로 표현	1 ~ 12
	G	시간을 24시간 단위로 표현	0 ~ 23
	h	시간을 12시간 단위로 표현(leading 0 포함)	01 ~ 12
	H	시간을 24시간 단위로 표현(leading 0 포함)	00 ~ 23
분/초	i	분을 표현(leading 0 포함)	00 ~ 59
	s	초를 표현(leading 0 포함)	00 ~ 59
전체 일시	c	2022-07-18T12:17:31+02:00	
	r	Mon, 18 Jul 2022 12:17:31 +0200	
time()		1970년 1월 1일 0시 0분 0초부터 누적된 초단위 값 출력	

다음 코드는 date() 함수와 형식 문자를 이용하여 날짜, 시간 등을 다양한 형태로 출력한 예제입니다.

	datetime.php	결과
01	echo time()." ";	1658140587
02	echo date("Y년 n월 j일(D)")." ";	2022년 7월 18일(Mon)
03	echo date("d-m-Y")." ";	18-07-2022
04	echo date("A h시 i분 s초")." ";	PM 12시 36분 27초
05	echo date("A h:i:s")." ";	PM 12:36:27
06	echo date("Y-F-d h:i:s(A)")." ";	2022-July-18 12:36:27(PM)
07	echo date("c")." ";	2022-07-18T12:36:27+02:00
08	echo date("r");	Mon, 18 Jul 2022 12:36:27 +0200

지금까지 자주 사용되는 내장 함수들의 기본적인 내용에 대해서 살펴봤습니다. 이외에도 파일/디렉토리 함수, 데이터베이스 함수, 네트워크 함수, 세션 함수 등 다양한 분야에 많은 함수들이 있습니다. 이들 함수들은 각 분야가 설명되는 장/절에서 소개합니다.

8. 디렉터리(폴더) 함수

디렉터리 함수는 디렉터리의 내부에 있는 파일들의 목록을 보거나, 디렉터리 변경, 생성, 삭제하는 하는 함수입니다.

8.1 readdir() : 파일 목록 추출

디렉터리의 파일 목록을 추출하기 위해서는 디렉터리 열기, 읽기, 닫는 과정이 기본적으로 필요합니다.

디렉토리 열기		파일 목록 추출		디렉토리 닫기
opendir()	➡	readdir()	➡	closedir()

이와 관련된 함수들의 설명은 다음과 같습니다.

함수	설명
resource opendir($directory)	$directory의 디렉터리를 열어서 시스템 자원으로 반환
bool closedir($dir_handle)	$dir_handle에 지정된 시스템 자원의 디렉터리를 닫음
bool is_dir($directory)	$directory에 저장된 이름이 디렉터리인지 점검
string readdir($dir_handle)	$dir_handle에 지정된 시스템 자원에서 디렉터리(파일) 목록을 문자열로 반환

다음 코드는 h:/xampp/htdocs 디렉터리를 열어서 readdir() 함수를 이용하여 파일 목록을 출력하는 예제입니다(드라이브 h는 사용자에 맞게 설정하세요).

readdir.php		브라우저 출력 결과

01	$dir_name = "h:/xampp/htdocs";
	$dir_handle = opendir($dir_name);
	echo "{$dir_handle}
	";
	if(is_dir($dir_name)){
	  echo "{$dir_name}는 디렉터리
	";
	}
	else { echo "{$dir_name}는 파일"; }
	while($dir_output = readdir($dir_handle))
	{ echo "{$dir_output}
	"; }
	closedir($dir_handle);
		Resource id #3
		h:/xampp/htdocs는 디렉터리
		.
		..
		applications.html
		bitnami.css
		dashboard
		: : :
		sqldata
		webalizer
		xampp

- 1번 라인 : h:/xampp/htdocs 디렉터리 문자열을 $dir_name 변수에 저장
- 2번 라인 : $dir_name 디렉터리를 opendir() 함수로 열고 그 시스템 자원을 $dir_handle 변수에 저장
- 4번~7번 라인 : $dir_name 변수에 저장된 문자열이 디렉터리인지를 점검
- 8번 라인 : $dir_handle 시스템 자원에서 readdir() 함수로 한줄씩 읽어서 $dir_output 변수에 저장
- 9번 라인 : $dir_output 변수에 저장된 디렉터리명 또는 파일명을 출력
- 10번 라인 : $dir_handle 시스템 자원을 종료(닫기)

8.2 scandir() 함수 : 파일 목록 배열형태 추출

scandir() 함수는 파일 목록을 배열형태로 추출하는 함수이며, 파일 열기/닫기 하지 않습니다.

함수	설명
array scandir($directory, 1)	$directory에 지정된 시스템 자원에서 디렉터리(파일) 목록을 배열로 반환 opendir(), closedir() 함수 사용하지 않음 정렬 옵션인 1 값을 사용하여 내림차순 정렬

다음 코드는 h:/xampp/htdocs 디렉터리를 scandir() 함수를 이용하여 파일 목록을 배열 형태로 출력하는 예제입니다.

scandir.php		브라우저 출력 결과	
		2번 라인 결과	4번라인 결과
01	$dir_name = "h:/xampp/htdocs";	Array ([0] => . [1] => .. [3] => bitnami.css : : : [13] => webalizer [14] => xampp)	Array ([0] => xampp [1] => webalizer [2] => sqldat : : : [13] => .. [14] => .)
02	$dir_handle = scandir($dir_name);		
03	print_r($dir_handle);		
04	$dir_handle = scandir($dir_name,1);		
05	print_r($dir_handle);		

- 1번 라인 : h:/xampp/htdocs 디렉터리 문자열을 $dir_name 변수에 저장
- 2번 라인 : $dir_name 디렉터리를 scandir() 함수로 열고 결과를 $dir_handle 배열 변수에 저장
- 3번 라인 : print_r() 함수로 $dir_handle 배열 변수의 내용 출력
- 4번 라인 : $dir_name 디렉터리를 scandir() 함수의 내림차순 옵션으로 열고 결과를 $dir_handle 배열 변수에 저장
- 5번 라인 : print_r() 함수로 $dir_handle 배열 변수의 내용 출력

8.3 디렉터리 생성과 삭제

디렉터리의 생성은 mkdir(), 삭제는 rmdir() 함수를 이용하여 수행합니다.

함수	설명
bool mkdir($directory)	$directory에 지정된 디렉터리 생성
bool rmdir($directory)	$directory로 지정된 디렉터리 삭제

디렉터리의 생성 또는 삭제는 is_dir() 함수로 해당 디렉터리의 존재 여부를 확인하고 수행합니다. is_dir() 함수에 지정된 디렉터리가 False면 존재하지 않는다는 의미이기 때문에 mkdir() 함수로 생성할 수 있고, True면 존재한다는 의미이기 때문에 rmdir() 함수로 삭제할 수 있습니다.

is_dir(dir_name)	False	➡	mkdir()
	True	➡	rmdir()

8.3.1 mkdir() 함수 : 디렉터리 생성

mkdir은 『make directory』의 약어이며 지정한 이름으로 디렉터리를 생성합니다. 디렉터리가 생성되는 경로는 mkdir() 함수가 수행되는 파일의 위치와 동일합니다. 다른 경로에 생성하려면 경로명을 정확히 지정해야 합니다.
다음 코드는 mkdir() 함수를 이용하여 myfolder 디렉터리를 생성하는 예제입니다.

mkdir.php		브라우저 출력 결과
01	$dir_name = "myfolder";	myfolder 디렉터리 생성됨
02	if(is_dir($dir_name)){	
03	echo "{$dir_name} 디렉터리가 이미 존재함";	
04	}	
05	else {	
06	mkdir($dir_name);	
07	echo "{$dir_name} 디렉터리 생성됨";	
08	}	

- 1번 라인 : 생성할 myfolder 디렉터리명을 $dir_name 변수에 저장
- 2~4번 라인 : is_dir() 함수로 생성하고자 하는 디렉터리의 존재 여부를 확인
- 5~8번 라인 : 생성하고자 하는 디렉터리가 없으면 mkdir() 함수로 디렉터리 생성되며, 생성되는 경로는 mkdir.php 함수의 경로와 동일

8.3.2 rmdir() 함수 : 디렉터리 삭제
rmdir은 『remove directory』의 약어이며 지정한 이름의 디렉터리를 삭제합니다. 삭제되는 디렉터리는 기본적으로 rmdir() 함수가 수행되는 파일과 동일한 경로에 있어야 합니다. 다른 경로의 디렉터리를 삭제하려면 경로명을 정확히 지정해야 합니다.
다음 코드는 rmdir() 함수를 이용하여 myfolder 디렉터리를 삭제하는 예제입니다.

rmdir.php		브라우저 출력 결과
01	$dir_name = "myfolder";	myfolder 디렉터리 삭제됨
02	if(is_dir($dir_name)){	
03	rmdir($dir_name);	
04	echo "{$dir_name} 디렉터리 삭제됨";	
05	}	
06	else {	
07	echo "삭제할 {$dir_name} 디렉터리 없음";	
08	}	

- 1번 라인 : 삭제할 myfolder 디렉터리명을 $dir_name 변수에 저장
- 2~5번 라인 : 삭제하고자 하는 디렉터리의 존재 여부를 확인하고 존재하면 rmdir() 함수로 삭제되며, 삭제되는 경로는 rmdir.php 함수의 경로와 동일
- 6~8번 라인 : 삭제하고자 하는 디렉터리가 없으면 문자 출력

9. 파일처리 함수

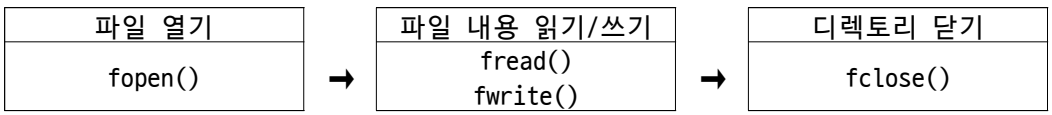
파일처리는 파일의 생성, 읽기, 쓰기, 삭제 등의 수행입니다. 파일처리 함수들이 수행되려면 파일을 열고, 닫는 과정이 기본적으로 필요합니다.
fopen() 함수의 반환 값은 resource 즉, 시스템 자원입니다. 이 시스템 자원을 변수에 저장하여 파일 읽기, 쓰기, 추가, 닫기 등을 수행합니다.
$filesize() 함수는 파일의 크기를 알아 내기 위한 함수이며, 바이트 수를 반환합니다.
처리하려는 파일이 기존에 존재하는 지를 점검하기 위해 is_file(), file_exists() 함수를 사용합니다.

함수	설명
resource fopen($filename, $mode)	$filename의 파일을 열어서 시스템 자원으로 반환
bool fclose($file_handle)	$file_handle에 지정된 시스템 자원의 파일을 닫음
int filesize($filename)	$filename 파일의 크기를 바이트 수로 반환
bool is_file($filename)	$filename에 저장된 이름이 파일인지 점검
bool file_exists($filename)	$filename에 저장된 이름이 파일 또는 디렉터리인지 점검

fopen() 함수의 $mode 옵션의 설명을 다음과 같습니다.

$mode	설명
r	읽기 전용. 읽기 위치(파일포인터)는 파일의 처음. 파일이 없으면 에러
r+	읽기/쓰기. 읽기/쓰기 위치(파일포인터)는 파일의 처음. 파일이 없으면 에러
w	쓰기 전용. 파일이 있으면 내용을 삭제하고, 없으면 새 파일 생성
w+	읽기/쓰기. 파일이 있으면 내용을 삭제하고, 없으면 새 파일 생성
a	추가 전용. 추가 위치(파일포인터)는 파일의 끝. 파일이 없으면 새 파일 생성
a+	읽기/추가. 읽기/추가 위치(파일포인터)는 파일의 끝. 파일이 없으면 새 파일 생성
x	쓰기 전용으로 생성. 파일이 있으면 에러, 없을 때 새 파일 생성
x+	읽기/쓰기용으로 생성. 파일이 있으면 에러, 없을 때 새 파일 생성

파일의 내용을 처리하기 위해서는 파일 열기, 파일 처리(읽기/쓰기), 파일 닫기 과정이 기본적으로 필요합니다.



9.1 파일내용 읽기

파일 내용을 읽기 위해서는 fopen() 함수의 모드를 r 또는 r+로 설정해야 합니다.

함수	설명
string fread($file_handle, $length)	$file_handle 정보를 이용하여 $length 바이트 수만큼 문자를 읽음
string fgetc($file_handle)	파일 포인터에서 문자 단위로 읽음
string fgets($file_handle)	파일 포인터에서 라인 단위로 읽음

다음 파일들은 fread(), fgetc(), fgets() 각 함수 예제에 사용되는 텍스트 파일입니다.

fread.txt	fgetc.txt	fgets.txt
fread line-1 fread line-2 	getc line-1 getc line-2	gets line-1 gets line-2

9.1.1 fread() 함수 : 파일 읽기

다음 코드는 fread() 함수를 이용하여 fread.txt 파일의 내용을 읽어서 그대로 출력하는 예제입니다.

fread.php		브라우저 출력 결과
1	$file_name = "fread.txt";	
2	$file_handle = fopen($file_name, "r");	Resource id #3
3	echo "{$file_handle} ";	fread.txt 파일의 내용
4	$contents = fread($file_handle, filesize($file_name));	fread line-1
5	echo "{$file_name} 파일의 내용 {$contents}";	fread line-2
6	fclose($file_handle);	

- 1번 라인 : $file_name 변수에 파일 이름 지정
- 2번 라인 : $file_name 파일을 r(읽기) 모드로 열고 그 시스템 자원을 $file_handle 변수에 저장
- 3번 라인 : $file_handle 시스템 자원 출력(시스템 자원이 어떤 형태인지 확인해 봄)
- 4번 라인 : $file_handle 정보를 이용하여 파일의 바이트 수 만큼 읽어서 $contents 변수에 저장
- 5번 라인 : $contents 변수에 저장된 내용을 출력
- 6번 라인 : 시스템 자원을 닫음(파일 이름을 닫지 않고 시스템 자원을 닫음)

9.1.2 fgetc() 함수 : 문자 단위 추출

다음 코드는 fgetc() 함수를 이용하여 getc.txt 파일의 내용을 읽어서 문자 단위로 출력하는 예제입니다.

fgetc.php		브라우저 출력 결과
1	$file_name = "fgetc.txt";	g
2	$file_handle = fopen($file_name, "r");	e
3	while ($char = fgetc($file_handle)){	t
4	echo "$char ";	c
5	}	:

- 1번 라인 : $file_name 변수에 파일 이름 지정
- 2번 라인 : $file_name 파일을 r(읽기) 모드로 열고 그 시스템 자원을 $file_handle 변수에 저장
- 3번 라인 : $file_handle 정보에서 fgetc() 함수를 이용하여 1개 문자를 읽어서 $char 변수에 저장. while 구문으로 파일 끝까지 읽음.
- 4번 라인 : 읽은 1개 문자 출력

9.1.3 fgets() 함수 : 라인 단위 추출

다음 코드는 fgets() 함수를 이용하여 gets.txt 파일의 내용을 읽어서 라인 단위로 출력하는 예제입니다.

fgets.php		브라우저 출력 결과
1	$file_name = "fgets.txt";	gets line-1 gets line-2 gets line-3
2	$file_handle = fopen($file_name, "r");	
3	while ($str = fgets($file_handle)){	
4	echo "$str ";	
5	}	

- 1번 라인 : $file_name 변수에 파일 이름 지정
- 2번 라인 : $file_name 파일을 r(읽기) 모드로 열고 그 시스템 자원을 $file_handle 변수에 저장
- 3번 라인 : $file_handle 정보에서 fgets() 함수를 이용하여 1개 라인을 읽어서 $str 변수에 저장. while 구문으로 파일 끝까지 읽음.
- 4번 라인 : 읽은 1개 라인 출력

9.2 파일에 내용 쓰기

파일에 내용을 쓰기 위해서는 fopen() 함수의 모드를 w, w+, a, a+, x, x+ 중 하나로 설정해야 합니다.

함수	설명
int fwrite($file_handle, $string)	$file_handle 정보를 이용하여 $string 문자열을 쓰기

다음 파일들은 w, a+, x 각 모드로 fwrite() 함수의 예제에 사용되는 텍스트 파일입니다.

fwrite_w.txt	fwrite_aplus.txt	fwrite_x.txt
write w-mode line-1 write w-mode line-2	write a-mode line-1 write a-mode line-2	write x-mode line-1 write x-mode line-2

9.2.1 fwrite() 함수 - w 모드

다음 코드는 w 모드와 fwrite() 함수를 이용하여 기존 파일에 문자열을 쓰기하는 예제입니다. 기존 파일이 없으면 지정된 파일명으로 생성합니다.

fwrite_w.php		fwrite_w.txt 파일 내용

1
2
3
4
	$file_name = "fwrite_w.txt";
	$file_handle = fopen($file_name, "w");
	fwrite($file_handle, "new text write-w");
	fclose($file_handle);
		--실행 전
		write w-mode line-1
		write w-mode line-2
		--실행 후
		new text write-w

- 1번 라인 : $file_name 변수에 파일 이름 지정
- 2번 라인 : $file_name 파일을 w(쓰기) 모드로 열고 그 시스템 자원을 $file_handle 변수에 저장
- 3번 라인 : $file_handle 정보에서 fwrite() 함수를 이용하여 기존 파일에 새로운 문자열을 쓰기(기존 파일 내용은 삭제됨)

9.2.2 fwrite() 함수 - a+ 모드

다음 코드는 a+ 모드와 fwrite() 함수를 이용하여 기존 파일의 끝에 문자열을 추가하는 예제입니다. a, a+ 모드의 파일 포인터는 파일 끝에 있으므로 읽기를 하면 파일의 내용이 표시되지 않습니다. 기존 파일이 없으면 지정된 파일명으로 생성합니다.

fwrite_aplus.php		fwrite_aplus.txt 파일 내용

1
2
3
4
5
6
	$file_name = "fwrite_aplus.txt";
	$file_handle = fopen($file_name, "a+");
	fwrite($file_handle, "new text write-aplus");
	$contents = fgets($file_handle);
	echo "{$contents}";
	fclose($file_handle);
		--실행 전
		write aplus-mode line-1
		write aplus-mode line-2
		--실행 후
		write aplus-mode line-1
		write aplus-mode line-2
		new text write-aplus

- 1번 라인 : $file_name 변수에 파일 이름 지정
- 2번 라인 : $file_name 파일을 a+(읽기/추가) 모드로 열고 그 시스템 자원을 $file_handle 변수에 저장
- 3번 라인 : $file_handle 정보에서 fwrite() 함수를 이용하여 기존 파일의 끝에 새로운 문자열을 추가
- 4번 라인 : fgets() 함수를 이용하여 파일의 내용을 읽어서 $contents 변수에 저장
- 5번 라인 : $contents 변수의 값을 출력. 파일 포인터가 파일의 끝에 있으므로 출력할 내용은 없음

9.2.3 fwrite() 함수 - x 모드

다음 코드는 x 모드와 fwrite() 함수를 이용하여 새로운 파일을 생성하고 문자열을 쓰기하는 예제입니다. x 모드는 새로운 파일만 생성하는 방식으므로 기존 파일(fwrite_x.txt)이 있으면 에러가 발생합니다.

fwrite_x.php		브라우저 출력내용

01
02
03
04
	$file_name = "fwrite_x.txt";
	$file_handle = fopen($file_name, "x");
	fwrite($file_handle, "new text write-x");
	fclose($file_handle);
		Warning: fopen(fwrite_x.txt): Failed to open stream: File exists .....
		Fatal error: Uncaught TypeError: fwrite(): Argument #1 ....