PHP

VivifyAcademy

Internal (built-in) functions

Internal (built-in) functions

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

While you are not expected to learn all of them at once, there are some useful functions that can help in everyday programming and we will start from there.

Overview

- Php.net functions definitions
- Math functions
- String functions
- Array functions
- Function handling functions

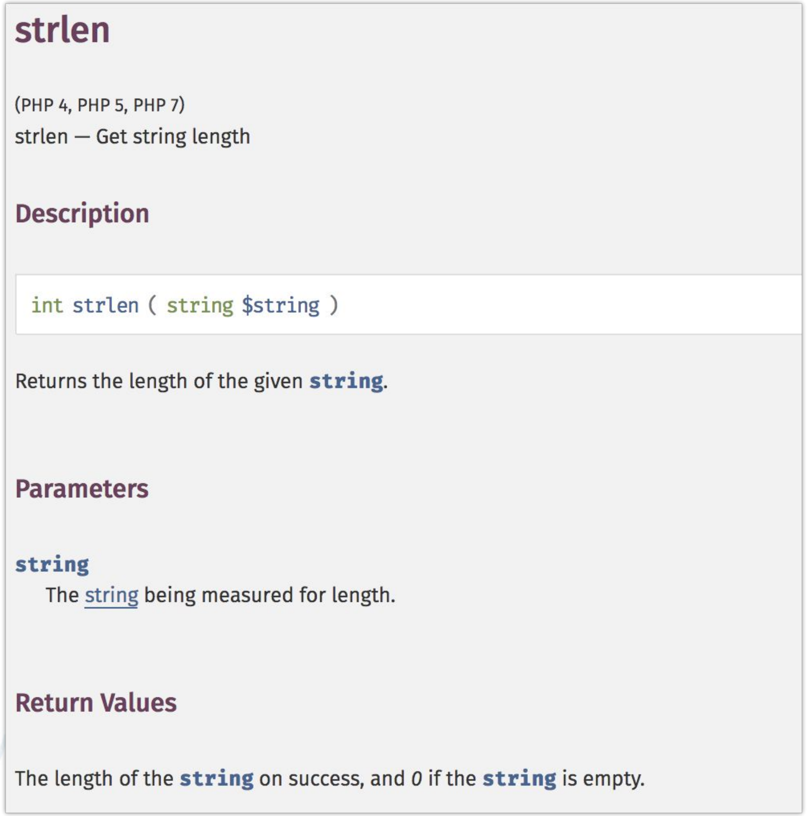
Php.net functions definitions

php.net function definitions

All in-built functions are defined on php.net website and provided with full usage cases and ways to use them.

Understanding how to read these definitions is really important as you will be able to quickly understand how each of in-built function works and how to use them just by looking at the php documentation. Examples are also provided

strlen() definition



strlen

(PHP 4, PHP 5, PHP 7)
strlen — Get string length

Description

```
int strlen ( string $string )
```

Returns the length of the given **string**.

Parameters

string
The string being measured for length.

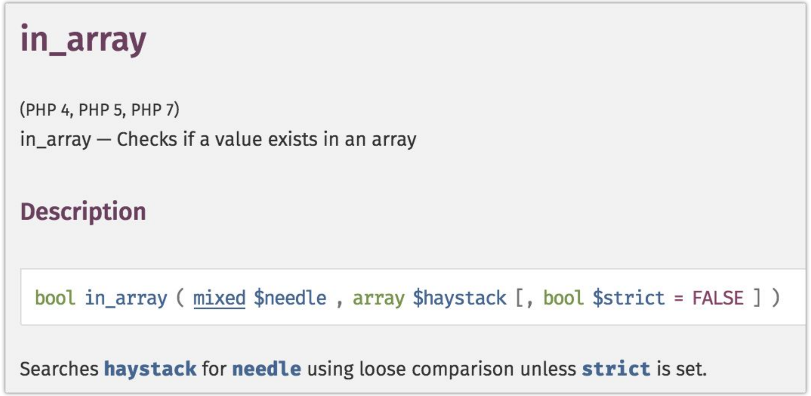
Return Values

The length of the **string** on success, and *0* if the **string** is empty.

- Each function has return type
- Each parameter has type
- Explanation is given for both return value and each parameter

http://php.net/manual/en/function.strlen.php

in_array() and abs() definition



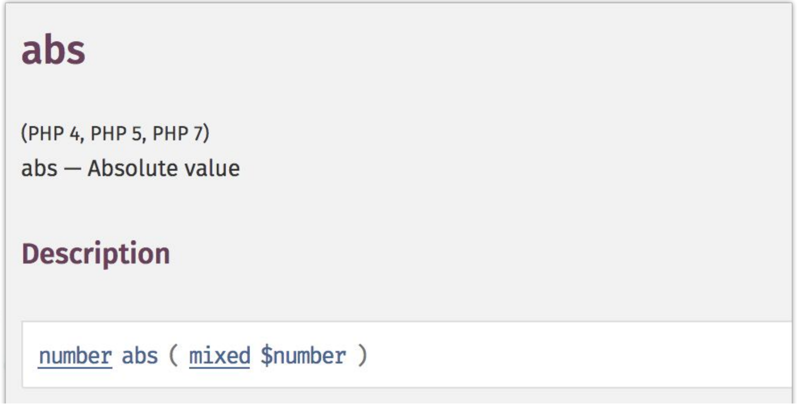
in_array

(PHP 4, PHP 5, PHP 7)
in_array — Checks if a value exists in an array

Description

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Searches **haystack** for **needle** using loose comparison unless **strict** is set.



abs

(PHP 4, PHP 5, PHP 7)
abs — Absolute value

Description

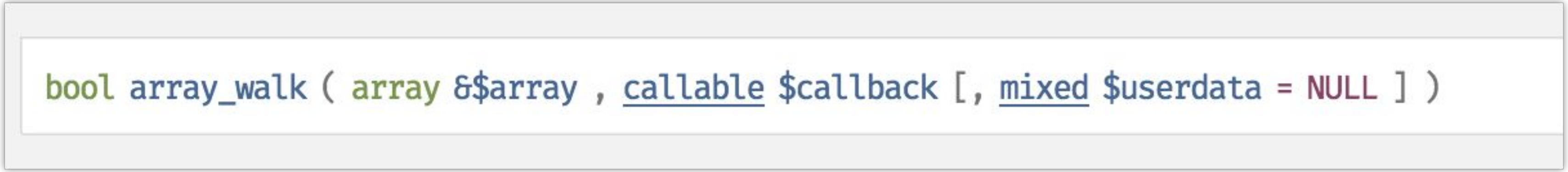
```
number abs ( mixed $number )
```

- **“mixed”** parameters can be of multiple type
- Parameters inside **[ ]** are optional

- **“number”** can be both integer or float

http://php.net/manual/en/function.in-array.php

array_walk definition



```
bool array_walk ( array &$array , callable $callback [, mixed $userdata = NULL ] )
```

- **"callable"** expects the callback function
- **&$array** means that this array is passed by reference (original array that is passed to the function will be changed)

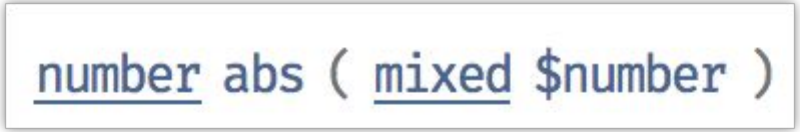
http://php.net/manual/en/function.array-walk.php

Math functions()

Math functions()

PHP includes a number of functions for dealing with math. All of them are a part of PHP core and can be used out of the box.

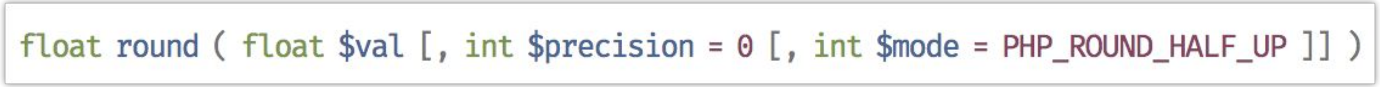
abs()


number abs (mixed $number)

Returns absolute value of the given number.

```
<?php

echo abs(-5); // 5

?>
```

round()

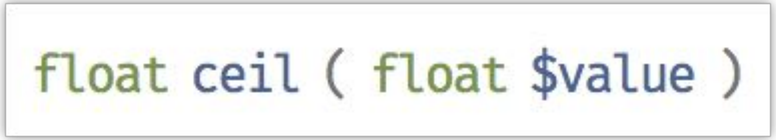


```
float round ( float $val [, int $precision = 0 [, int $mode = PHP_ROUND_HALF_UP ]] )
```

Returns the rounded value of val to specified precision (number of digits after the decimal point).

```
<?php

echo round(3.5); // 4

echo round(1.85583, 2); // 1.86

?>
```

ceil()

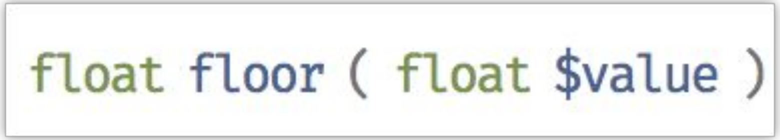


```
float ceil ( float $value )
```

Returns the next highest integer value.

```
<?php

echo ceil(5.2); // 6

?>
```

floor()

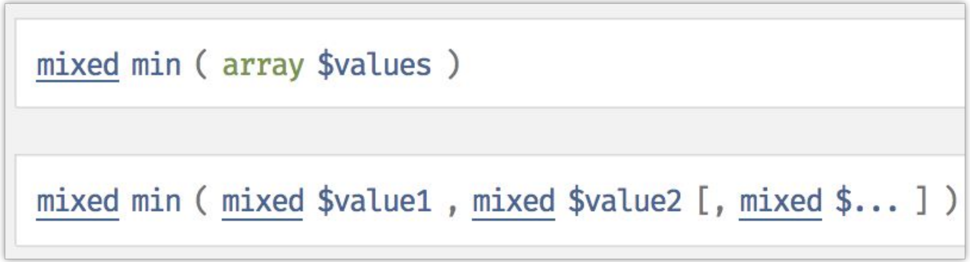


```
float floor ( float $value )
```

Returns the next lowest integer value.

```
<?php

echo floor(5.7); // 5

?>
```

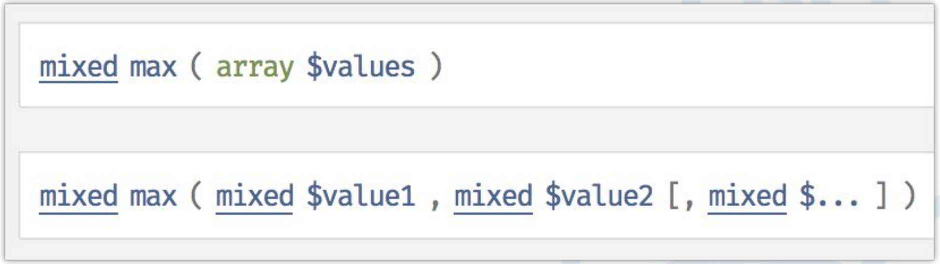
max() and min()



```
mixed min ( array $values )

mixed min ( mixed $value1 , mixed $value2 [, mixed $... ] )
```

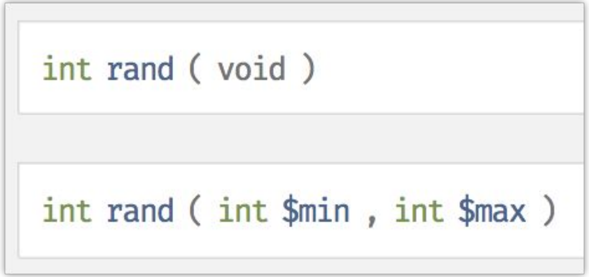
If an array is passed, returns the highest value in the array. If at least two parameters are passed, returns the greater of them.

```php
<?php

echo max(2, 3, 7); // 7

echo min([2, 4, 5]); // 2

?>
```



```
mixed max ( array $values )

mixed max ( mixed $value1 , mixed $value2 [, mixed $... ] )
```

rand()

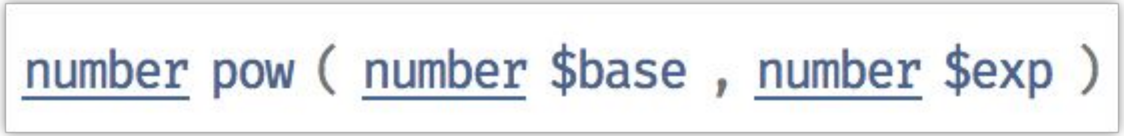


```
int rand ( void )

int rand ( int $min , int $max )
```

Returns a pseudo random value between passed arguments. If called without the optional `min` and `max` arguments, `rand()` returns a pseudo-random integer between 0 and `getrandmax()`.

```
<?php

echo rand(5, 24); // Some random value between 5 and 24

?>
```

pow()

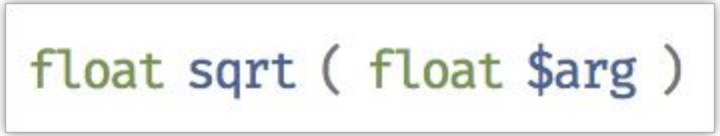

number pow (number $base , number $exp)

Returns the base argument raised to the power of exponent argument.

```
<?php

echo pow(2, 4); // 16

echo pow(-2, 3) // -8

?>
```

```
Note: In PHP 5.6 onwards, you may prefer to use the ** operator.
```

sqrt()



```
float sqrt ( float $arg )
```

Returns the square root of the passed argument or the special value `NAN` for negative numbers.

```
<?php

echo sqrt(9); // 3

?>
```

Useful links

- http://php.net/manual/en/ref.math.php

Zadaci

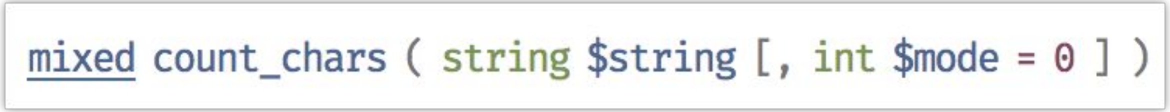
1. Napisati funkciju koja vraca nasumican broj izmedju 5 i 500.
2. Napisati funkciju koja za niz brojeva (45, 12, 1, 100, 6) izracunava kvadratne korene svakog broja, smesta ih u novi niz, i ispisuje na ekranu.
3. Prosiriti prethodni primer tako da se u novi niz smestaju samo cele vrednosti brojeva, i to zaokruzene na gornju vrednost.

String functions()

String function()

As with math, PHP also includes a lot of functions for string manipulation.

count_chars()

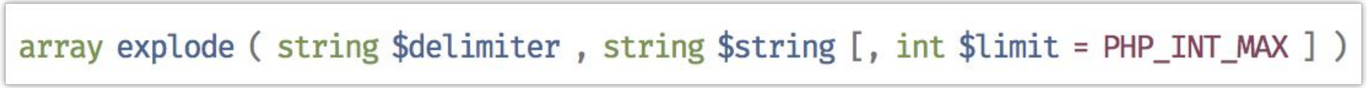


```
mixed count_chars ( string $string [, int $mode = 0 ] )
```

Counts the number of occurrences of every byte-value in a string, and returns it in different ways. Along with the string, you can pass a number from 0 to 5, which represents a mode. For example, if you pass 3 you will get a string containing all unique characters.

```
<php?

$str = "Hello, world!";

echo count_chars($str, 3); // !HWdelor

?>
```

explode()



```
array explode ( string $delimiter , string $string [, int $limit = PHP_INT_MAX ] )
```

Returns an array of strings, each of which is a substring of string formed by splitting it using the string delimiter.

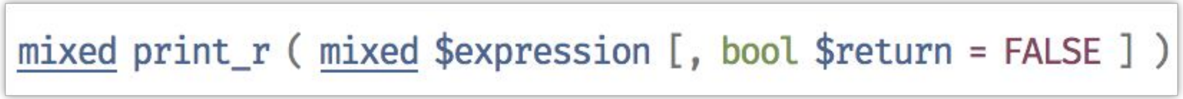
```
<?php

$str = "Hello, world!";

$pieces = explode(" ", $str);

echo $pieces[1]; // world!

?>
```

implode()

Joins array elements with a string.

```
<?php

$pieces = array("12", "3", "2017");

$str = implode("-", $pieces);

echo $str; // 12-3-2017

?>
```

print_r()

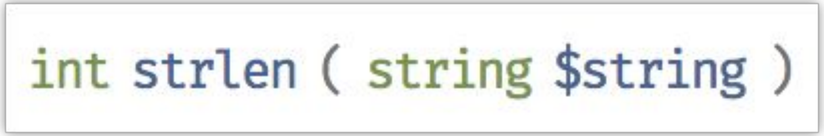


```
mixed print_r ( mixed $expression [, bool $return = FALSE ] )
```

Prints information about a variable in a way that humans can read and understand.

```
<?php

$arr = ["a" => "firstname", "b" => "lastname"];

print_r($arr);

?>
```

strlen()

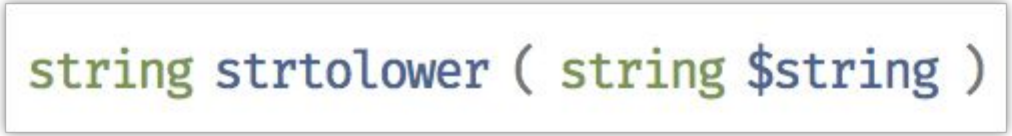


```
int strlen ( string $string )
```

Returns the length of the given string.

```
<?php

$str = "Hello, world!";

echo strlen($str); // 13

?>
```

strtolower()

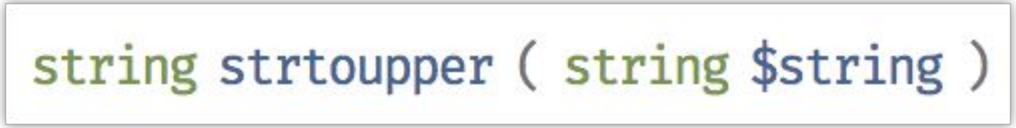


```
string strtolower ( string $string )
```

Returns string with all letters converted to lowercase.

```
<?php

$str = "I love PHP.";

echo strtolower($str); // i love php.

?>
```

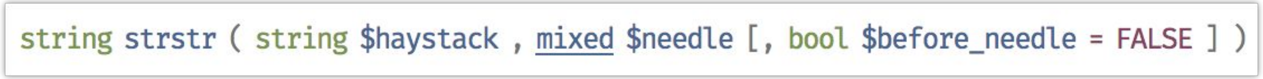
strtoupper()


string strtoupper (string $string)

Returns string with all letters converted to uppercase.

```
<?php

$str = "I love PHP.";

echo strtoupper($str); // I LOVE PHP.

?>
```

strstr()

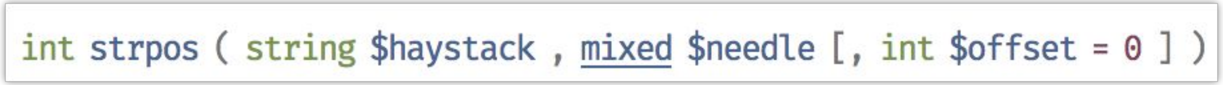


```
string strstr ( string $haystack , mixed $needle [, bool $before_needle = FALSE ] )
```

Returns part of the passed string starting from and including the first occurrence of the delimiter to the end of the passed string.

```
<?php

$email = "example@example.com";

$domain = strstr($email, "@");

echo $domain; // @example.com

?>
```

strpos()

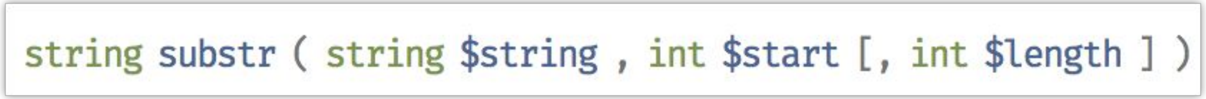


```
int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

Find the position of the first occurrence of a substring in a string. This function is case sensitive.

```
<?php

$str = "I love PHP";

echo strpos($str, "PHP"); // 7

?>
```

substr()

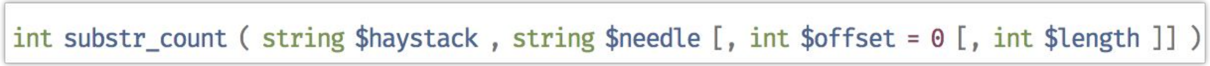


```
string substr ( string $string , int $start [, int $length ] )
```

Returns the part of a string specified by the start and length parameters.

```
<?php

$str = "Hello, world!";

echo substr($str, 0, 4); // Hell

echo substr($str, -3); // ld!

echo substr($str, -3, 1); // l

?>
```

substr_count()

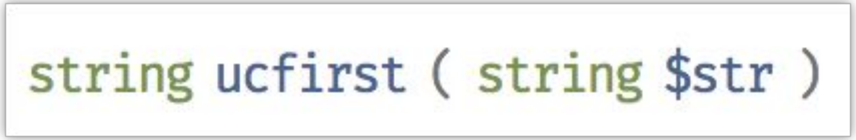


```
int substr_count ( string $haystack , string $needle [, int $offset = 0 [, int $length ]] )
```

Counts the number of substring occurrences.

```
<?php

$text = "This is a test";

echo substr_count($text, "is"); // 2

?>
```

ucfirst()


string ucfirst (string $str)

Returns a string with the first character of passed string capitalized.

```
<?php

$str = "hello, world!";

echo ucfirst($str); // Hello, world!

?>
```

Useful links

- http://php.net/manual/en/ref.strings.php

Zadaci

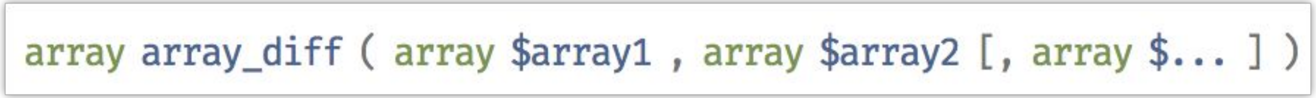
1. Napisati funkciju koja zadatu recenicu transformise:
   a. Svako slovo u zadatoj recenici postaje veliko slovo
   b. Svako slovo u zadatoj recenici postaje malo slovo
   c. Prva slova svake reci postaju veliko slovo
2. Napisati funkciju koja iz korisnikovog emaila, izvlaci njegov username (zatim prepraviti da vraca domen).
3. Napisati funkciju koja vraca najduzu rec iz zadatog stringa.

Array function()

Array function()

The array functions allow you to access and manipulate arrays. Simple and multi-dimensional arrays are supported.

array_diff()

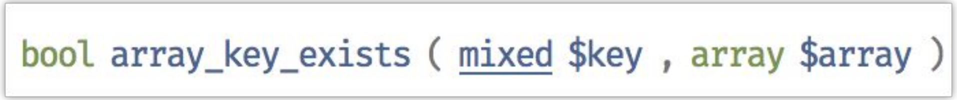


```
array array_diff ( array $array1 , array $array2 [, array $... ] )
```

Compares the passed array against one or more other arrays and returns the values in the passed array that are not present in any of the other arrays.

```
<?php

$array1 = array("a" => "green", "red", "blue", "red");

$array2 = array("b" => "green", "yellow", "red");

$result = array_diff($array1, $array2);

print_r($result); // array(1 => ‘blue’)

?>
```

array_key_exists()

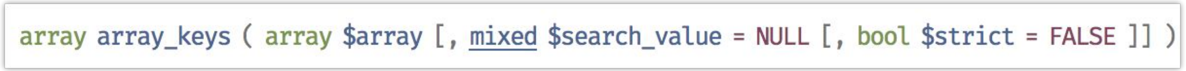


```
bool array_key_exists ( mixed $key , array $array )
```

Return true if the given key or index exists in the array.

```
<?php

$arr = array("Volvo" => "XC90", "BMW" => "X5");

echo array_key_exists("Volvo", $arr); // true

?>
```

array_keys()

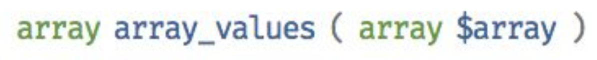


```
array array_keys ( array $array [, mixed $search_value = NULL [, bool $strict = FALSE ]] )
```

Returns the keys from the array.

```
<?php

$array = array(0 => 1, "color" => “green”);

print_r(array_keys($array));

?>
```

array_values()

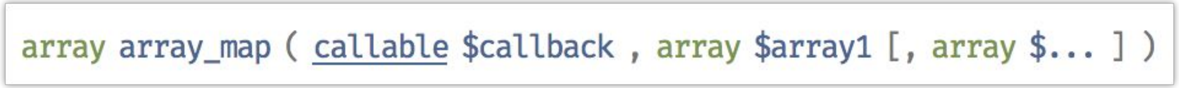


```
array array_values ( array $array )
```

Returns the values from the array.

```
<?php

$array = array(0 => 1, "color" => “green”);

print_r(array_values($array));

?>
```

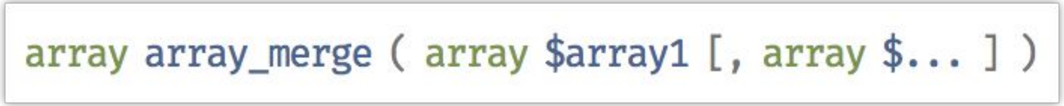
array_map()


array array_map (callable $callback , array $array1 [, array $...])

Applies the callback to the elements of the given arrays and returns a new array with transformed elements.

```php
<?php

function square($n) { return $n * $n; }

$arr = array(1, 2, 3, 4);

$squared = array_map(“square”, $arr);

print_r($squared);

?>
```

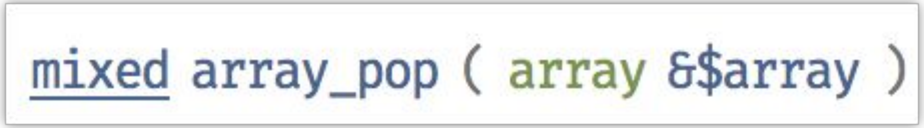
array_merge()


```
array array_merge ( array $array1 [, array $... ] )
```

Merges one or more arrays.

```
<?php

$arr1 = array("color" => "red");

$arr2 = array("color" => "green", "shape" => "square");

$result = array_merge($arr1, $arr2);

print_r($result);    // array("color" => "green", "shape" => "square")

?>
```

array_pop()

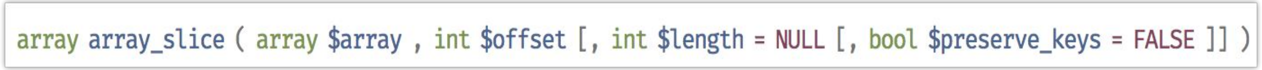


```
mixed array_pop ( array &$array )
```

Pops and returns the last value of the array, shortening the array by one element.

```
<?php

$fruits = array("apple", "banana", "orange");

echo array_pop($fruits);  // orange

print_r($fruits);         // array("apple", "banana")

?>
```

array_slice()

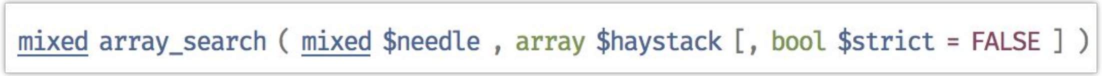


```
array array_slice ( array $array , int $offset [, int $length = NULL [, bool $preserve_keys = FALSE ]] )
```

Returns the sequence of elements from the array as specified by the offset and length parameters.

```
<?php

$arr = array("a", "b", "c", "d", "e");

$output = array_slice($arr, 2);  //  returns "c", "d", and "e"

?>
```

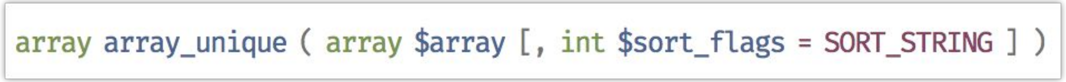
array_search()


```
mixed array_search ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Searches the array for a given value and returns the first corresponding key if successful.

```
<?php

$array = array(0 => 'blue', 1 => 'red', 2 => 'green');

$key = array_search('green', $array);

echo $key; // 2

?>
```

array_unique()

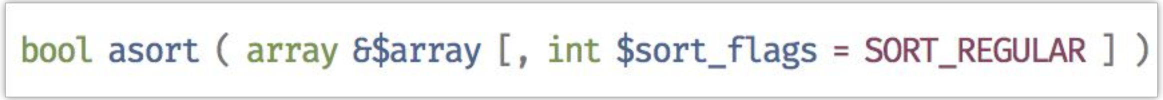


```
array array_unique ( array $array [, int $sort_flags = SORT_STRING ] )
```

Removes duplicate values from an array.

```
<?php

$arr = array("a" => "green", "red", "b" => "green", "blue", "red");

$result = array_unique($arr);

print_r($result);

?>
```

asort()

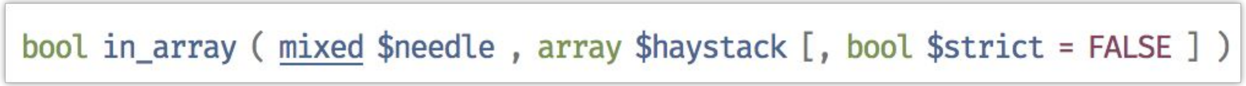


```
bool asort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Sorts an array and maintains index association.

```php
<?php

$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" =>
"apple");

asort($fruits);

?>
```

in_array()

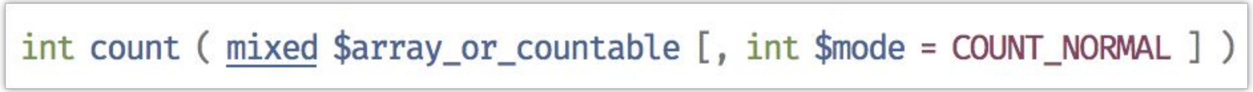


```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Checks if a value exists in an array. This function is case sensitive.

```
<?php

$os = array("Mac", "Linux");

echo in_array("Mac", $os); // true

?>
```

count()



```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Returns the number of elements in an array.

```
<?php

$arr = array("a", "b", "c");

echo count($arr); // 3

?>
```

Useful links

- http://php.net/manual/en/ref.array.php

Zadaci

1. Napisati funkciju koja vraca najduzi indeks u nizu.
2. Napisati funkciju koja za niz brojeva, pronalazi nedostajuce vrednosti (npr [1,2,4,5,7], skripta vraca [3,6]).
3. Napisati funkciju koja ispisuje element niza sa najvecom apsolutnom vrednoscu.
4. Napisati funkciju koja vraca niz prvih 5 reci iz stringa.

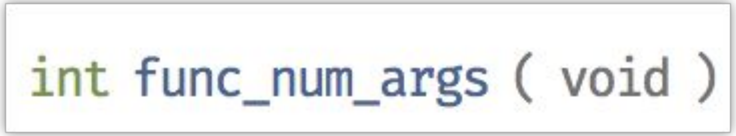
Function handling functions()

VIVIFY ACADEMY

Function handling functions()

These functions all handle various operations involved in working with functions.

func_num_args()

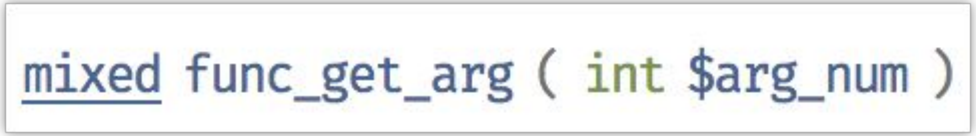


```
int func_num_args ( void )
```

Returns the number of arguments passed to the function.

```
<?php

function foo() {

    $numargs = func_num_args();

    echo "Number of arguments: $numargs\n";

}

foo(1, 2, 3); // Number of arguments: 3

?>
```

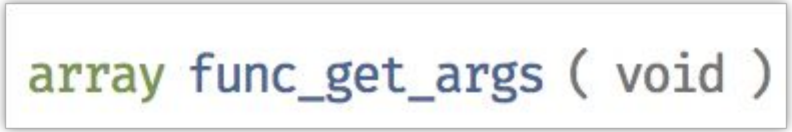
func_get_arg()


mixed func_get_arg (int $arg_num)

Return an item from the argument list

```php
<?php

function foo() {

    echo "Second argument is: " . func_get_arg(1) . "\n";

}

foo(1, 2, 3); // Second argument is: 2

?>
```

func_get_args()



```
array func_get_args ( void )
```

Returns an array comprising a function's argument list.

Example on next slide 👉

func_get_args()

```php
<?php

function foo() {

    $arg_list = func_get_args();

    for ($i = 0; $i < count($arg_list); $i++) {

        echo "Argument $i is: " . $arg_list[$i] . "\n";

    }

}

foo(1, 2, 3);

?>
```

Useful links

- http://php.net/manual/en/ref.funchand.php

Sneak peek - VivifyAcademy Advanced Course

In Laravel (https://laravel.com/), the `Illuminate\Support\Collection` class provides a fluent, convenient wrapper for working with arrays of data. Laravel Collections are PHP arrays on steroids.

The `Collection` class allows you to chain its methods to perform fluent mapping and reducing of the underlying array.

https://laravel.com/docs/5.5/collections

Join us on VivifyAcademy Advanced Course to learn more why is Laravel the best framework for PHP. 😎