Übungsaufgabe 15 - Array-Funktionen

Erstellung einer eigenen Funktion `shuffle_assoc`

Aufgabe: es soll eine Funktion `shuffle_assoc` entwickelt werden, die der PHP-internen Funktion `shuffle` entspricht, jedoch bei dem Array insbesondere bei assoziativen Arrays die Key-Zuordnung beibehält.

shuffle — Mischt die Elemente eines Arrays

Beschreibung

```
shuffle(array &$array): bool
```

Diese Funktion mischt die Elemente eines Arrays zufällig (shuffle).

...

Hinweis: Diese Funktion weist den Elementen des Arrays array neue Schlüssel zu. Bestehende Schlüssel, die bereits zugewiesen wurden, werden entfernt statt einfach nur die Schlüssel neu anzuordnen.

Aufgabe: Die zu entwickelnde Funktion `shuffle_assoc` soll die Keyszuordnung jedoch beibehalten, d. h. hatte ein Arrayelement mit z. B. dem Wert '`Oli`' den Key '`name`' (`[..., 'name' => 'Oli', ...]`), dann soll in dem 'gemischtelten' Array der Wert '`Oli`' immer noch den Key '`name`' haben, allerdings an einer anderen, zufälligen Position im Array stehen.
Zu beachten: auch wir wollen mit einem Refernzparameter arbeiten, d. h. mit Parameterübergabe per Referenz

Funktionssignatur:

```
shuffle_assoc(array &$array): void
```

Anmerkung: unsere Funktion hat keinen Rückgabewert (wer will, kann das aber auch wie im Original als `true` machen).

Der Aufruf erfolt natürlich ohne Wertzuweisung oder Weiterverarbeitung in einem Ausdruck

Beispiel Funktionsaufruf:

```
shuffle_assoc($myArray);
```

Beispiel assoziatives Array:

```php
<?php
require_once 'inc/functions.inc.php';

$numbers = [
    'key1' => 1,
    'key2' => 2,
    'key3' => 3,
    'key4' => 4,
    'key5' => 5,
    'key6' => 6,
];

shuffle_assoc($numbers);

print_r($numbers);
```

Ausgabe:

```
Array
(
    [key5] => 5
    [key2] => 2
    [key6] => 6
    [key3] => 3
    [key1] => 1
    [key4] => 4
)
```

Beispiel numerisch indiziertes Array:

```php
<?php
require_once 'inc/functions.inc.php';

$numbers = [1, 2, 3, 4, 5, 6,];

shuffle_assoc($numbers);

print_r($numbers);
```

Ausgabe:

```
Array
(
    [0] => 1
    [4] => 5
    [5] => 6
    [3] => 4
    [1] => 2
    [2] => 3
)
```

Tipps für die Umsetzung

1. Bei der Parameterübergabe per Referenz werden alle Änderungen, die an dem Parameter in der Funktion vorgenommen werden auch an dem Array im globalen Scope vorgenommen, d. h. die Änderung wird am Original vorgenommen (in unserem Fall destruktiv bezüglich der Reihenfolge).

Beispiel:

```php
<?php
function tuewas(array &$array) {
    $array = [];
}

$numbers = [1, 2, 3, 4, 5, 6,];

tuewas($numbers);

var_dump($numbers);
```

Ausgabe:

```
array(0) {
}
```

⇒ das Original-Array wird durch den Funktionsaufruf von `tuewas()` geleert und zwar instantan (unmittelbar, sobald das in der Funktion mit dem Parameter gemacht wird).

Demnach kann das Original-Array geleert werden, aber natürlich auch 'neue' Elemente in gewünschter Reihenfolge wieder eingefügt werden.

Beispiel:

```php
<?php
function tuewas(array &$array) {
    $array = [];
    $array[] = 42;
    $array[] = 15;
    $array[] = 27;
}

$numbers = [1, 2, 3, 4, 5, 6,];

tuewas($numbers);

print_r($numbers);
```

Ausgabe:

```
Array
(
    [0] => 42
    [1] => 15
    [2] => 27
)
```

2. Mittels der PHP-Funktion `rand()` können Zufallswerte generiert werden.
   Dabei kann auch Wertebereich festgelegt werden, aus dem diese Zufallszahl zurückgegeben wird.

rand — Erzeugt eine zufällige Zahl

Beschreibung

```
rand(): int
```

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

Gibt eine Pseudozufallszahl zwischen min und max (inklusive) zurück oder zwischen 0 und getrandmax(), falls keine Parameter angegeben wurden. Wenn z. B. ein Zufallswert zwischen 5 und 15 benötigt wird, so wäre der Aufruf dafür rand(5, 15).

php.net

Beispiel:

```
<?php
echo rand(), PHP_EOL; // beliegige positive Integerzahl zwischen 0 und maximum
integer
echo rand(5,15), PHP_EOL; // Integerzahl zwischen 5 und 15 (einschließlich 5 und
15)

echo rand(0, 6), PHP_EOL; // Integerzahl zwischen 0 und 6 (einschließlich 0 und 6)
```

Ausgabe:

```
176292222
10
4
```

3. Tipps für die Programmlogik

- in der Funktion kannst Du dir eine Kopie des Parameter erzeugen
  ⇒ Änderungen an der Kopie wirken sich nicht auf das Original aus
- in der Funktion kannst Du dir ein Array mit den Keys des Parameter erzeugen
  ⇒ Änderungen an diesem Array wirken sich nicht auf das Original aus
- in der Funktion kannst Du dir ein Array mit den Values des Parameter erzeugen
  ⇒ Änderungen an diesem Array wirken sich nicht auf das Original aus
- in der Funktion kannst Du natürlich auch mit Schleifen arbeiten
- einzelne Array-Element lassen sich mit `unset($array[$key])` löschen
- eine lückenbehaftetes numerisch indiziertes Array kann mit der Funktion `array_values()` neu indiziert werden

Wie Du das umsetzt bleibt dir überlassen - hier gibt es sicherlich mehrere Lösungswege. Wenn möglich verzichte jedoch auf die Verwendung der Funktion `shuffle()`.