

# Tableaux et boucles

<https://github.com/heig-vd-progserv-course/heig-vd-progserv1-course>

Visualiser le contenu complet sur GitHub [à cette adresse](#).

L. Delafontaine, avec l'aide de [GitHub Copilot](#).

Ce travail est sous licence [CC BY-SA 4.0](#).

# Retrouvez le contenu complet de cette présentation sur GitHub

*Cette présentation est un résumé du contenu complet disponible  
sur GitHub.*

*Pour plus de détails, retrouvez le [contenu complet sur GitHub](#) ou en  
cliquant sur l'en-tête de ce document.*

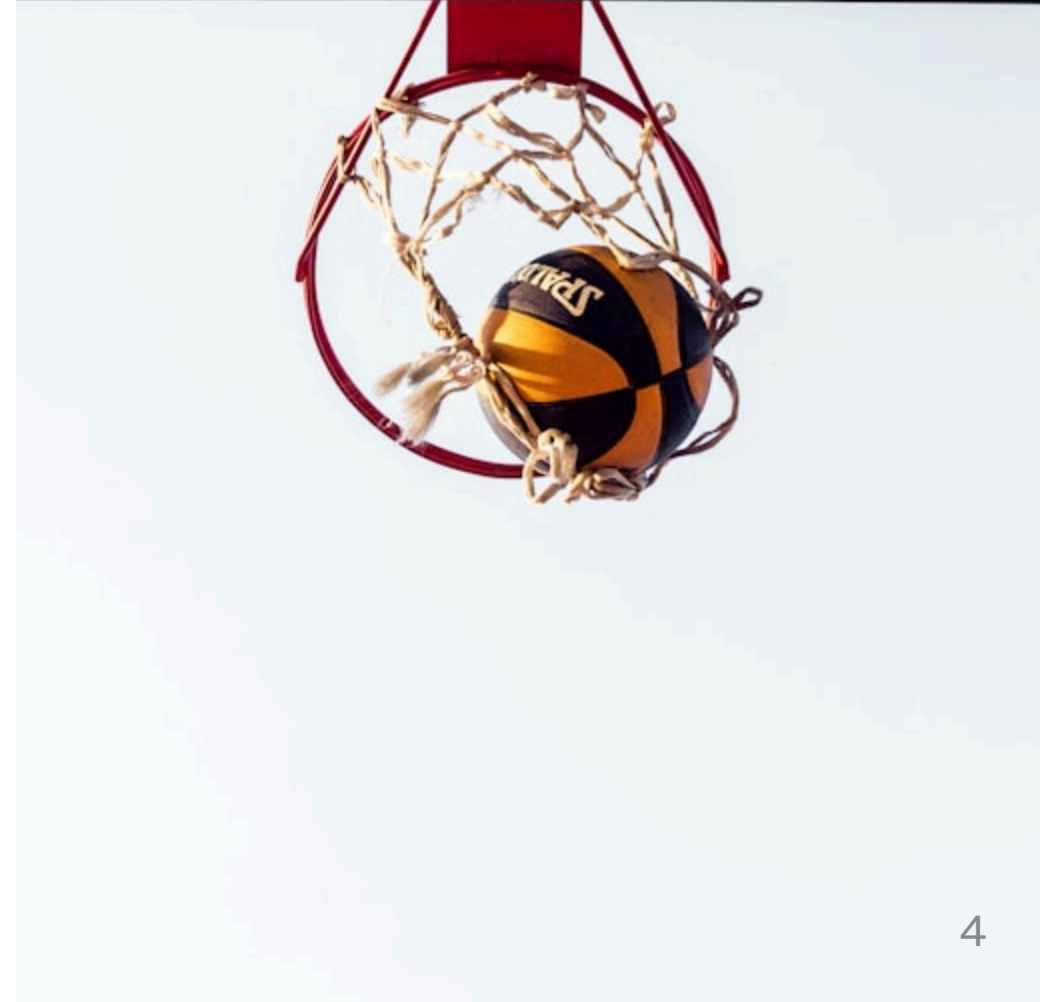
# Objectifs (1)

- Décrire les tableaux et leurs caractéristiques.
- Décrire la différence entre les tableaux indexés, les tableaux associatifs et les tableaux multidimensionnels.
- Utiliser et manipuler des tableaux (indexés, associatifs et multidimensionnels).



## Objectifs (2)

- Décrire les boucles et leurs caractéristiques.
- Décrire la différence entre les boucles `for`, `while`, `do...while` et `foreach`.
- Utiliser les boucles pour parcourir des tableaux et des collections de données.



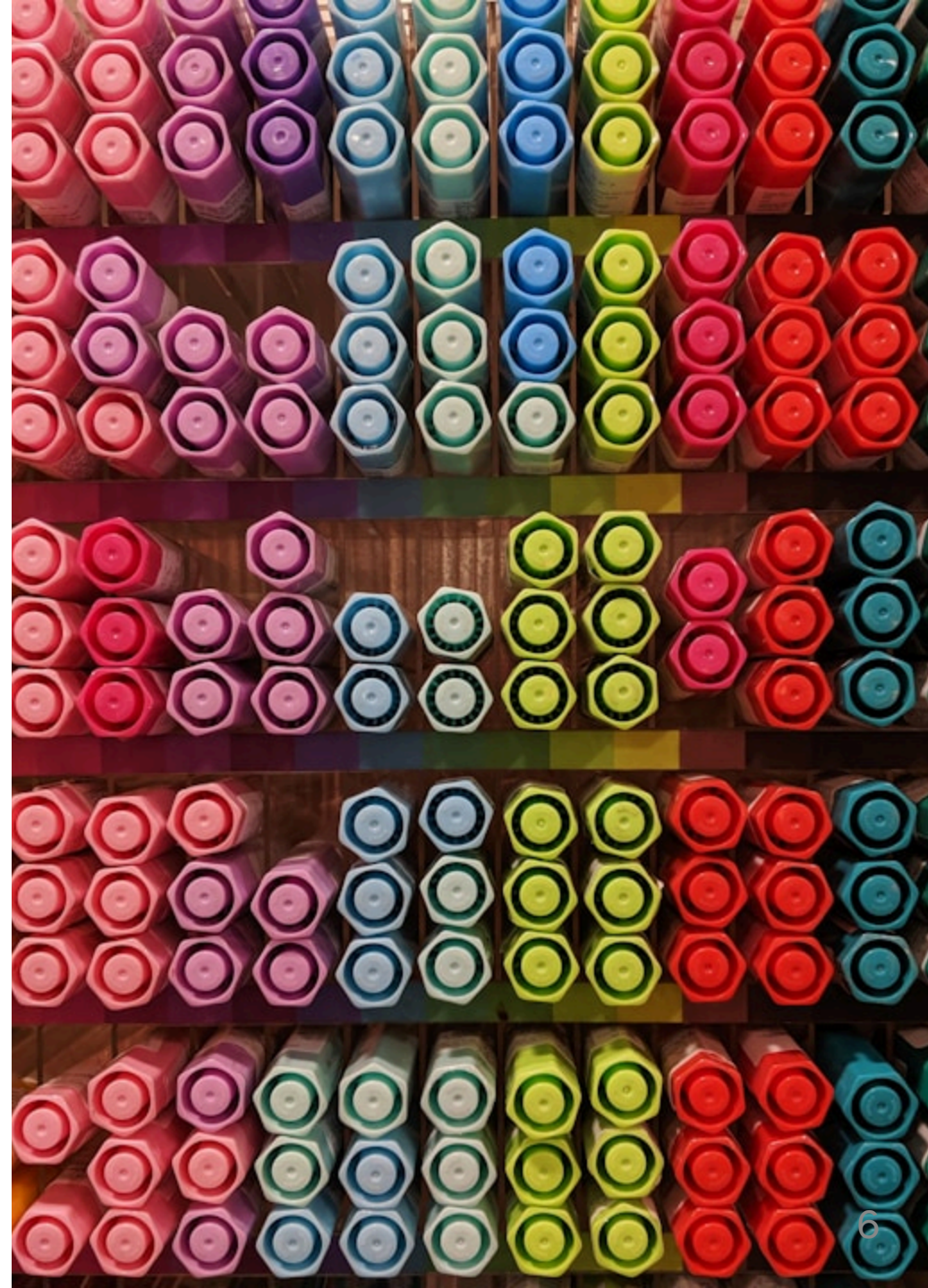
## Objectifs (3)

- Utiliser quelques fonctions utiles pour travailler avec les tableaux et les boucles.



# Les tableaux

- Les tableaux sont des collections de valeurs.
- Les tableaux sont déclarés entre des crochets ( `[]` ) ou avec la fonction `array()`.
- Les valeurs peuvent être de n'importe quel type.
- Il existe trois types de tableaux en PHP : indexés, associatifs et multidimensionnels.



# Tableaux indexés (1)

- Forme la plus simple de tableau.
- Les valeurs sont indexées par des entiers.
- Les index commencent à 0.
- Les valeurs peuvent être de n'importe quel type (entier, chaîne de caractères, tableau, etc.).



## Tableaux indexés (2)

```
<?php
$fruits = ['apple', 'banana', 'orange', 'kiwi'];

echo $fruits[0] . "<br>"; // Affiche 'apple'
echo $fruits[1] . "<br>"; // Affiche 'banana'
echo $fruits[2] . "<br>"; // Affiche 'orange'
echo $fruits[3] . "<br>"; // Affiche 'kiwi'
```

# Tableaux indexés (3)

```
public class Main {  
    public static void main(String[] args) {  
        String[] fruits = {"apple", "banana", "orange", "kiwi"};  
  
        System.out.println(fruits[0]); // Affiche 'apple'  
        System.out.println(fruits[1]); // Affiche 'banana'  
        System.out.println(fruits[2]); // Affiche 'orange'  
        System.out.println(fruits[3]); // Affiche 'kiwi'  
    }  
}
```

# Tableaux indexés (4)

Ce tableau indexé peut être représenté sous la forme d'une table, composée de paires de clé-valeur :

Index	Valeur
0	'apple'
1	'banana'
2	'orange'
3	'kiwi'

# Tableaux indexés (5)

```
<?php
$mixed = ['apple', 123, true, 3.14];

echo $mixed[0] . "<br>"; // Affiche 'apple'
echo $mixed[1] . "<br>"; // Affiche 123
echo $mixed[2] . "<br>"; // Affiche true
echo $mixed[3] . "<br>"; // Affiche 3.14
```

```
// Équivalent en Java

// Il n'est pas possible de créer un tableau
// contenant des types différents en Java.
```

# Tableaux indexés (6)

Imaginons maintenant que nous souhaitons représenter une personne à l'aide d'un tableau indexé. Nous pourrions créer un tableau `$person` qui contient le nom, l'âge et la ville de la personne :

```
<?php
$person = ['John Doe', 30, 'New York'];

echo $person[0] . "<br>"; // Affiche le nom de la personne
echo $person[1] . "<br>"; // Affiche l'âge de la personne
echo $person[2] . "<br>"; // Affiche la ville de la personne
```

## Tableaux indexés (7)

Ce tableau indexé peut être représenté sous la forme d'une table, composée de paires de clé-valeur :

Index	Valeur
0	' John Doe '
1	30
2	' New York '

Ce n'est pas très intuitif... Solution : les tableaux associatifs.

# Tableaux associatifs (1)

- Les valeurs sont indexées par des chaînes de caractères, appelées *clés*.
- Les clés peuvent être de n'importe quel type et peuvent être complètement arbitraires.
- Utilisés pour stocker des données sous forme de paires clé-valeur.



## Tableaux associatifs (2)

```
<?php
$person = [
    // Les caractères `=>` sont utilisés pour associer
    // une clé à une valeur
    'name' => 'John Doe',
    'age' => 30,
    'city' => 'New York',
];

echo $person['name'] . "<br>"; // Affiche 'John Doe'
echo $person['age'] . "<br>"; // Affiche 30
echo $person['city'] . "<br>"; // Affiche 'New York'
```

## Tableaux associatifs (3)

Ce tableau associatif peut être représenté sous la forme d'une table, composée de paires de clé-valeur :

Clé	Valeur
name	' John Doe '
age	30
city	' New York '

Plus intuitif que le tableau indexé !

# Tableaux associatifs (4)

```
// Équivalent en Java

// Il n'est pas possible de créer un tableau associatif
// en Java, mais nous pouvons utiliser une `HashMap` pour
// obtenir un résultat similaire (non décrit ici).
```

# Tableaux multidimensionnels (1)

- Les tableaux multidimensionnels sont des tableaux qui contiennent d'autres tableaux.
- Ils peuvent être utilisés pour représenter des données en plusieurs dimensions.
- Indexés par des entiers ou des chaînes de caractères.



# Tableaux multidimensionnels (2)

```
<?php
// Un tableau multidimensionnel contenant des tableaux indexés
$matrix = [
    [1, 2, 3], // Un premier tableau indexé
    [4, 5, 6], // Un deuxième tableau indexé
    [7, 8, 9], // Un troisième tableau indexé
];

echo $matrix[0][0] . "<br>"; // Affiche 1
echo $matrix[1][1] . "<br>"; // Affiche 5
echo $matrix[2][2] . "<br>"; // Affiche 9
```

# Tableaux multidimensionnels (3)

```
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        // Un tableau multidimensionnel contenant des tableaux indexés
        int[][] matrix = {
            {1, 2, 3}, // Un premier tableau indexé
            {4, 5, 6}, // Un deuxième tableau indexé
            {7, 8, 9} // Un troisième tableau indexé
        };

        System.out.println(matrix[0][0]); // Affiche 1
        System.out.println(matrix[1][1]); // Affiche 5
        System.out.println(matrix[2][2]); // Affiche 9
    }
}
```

# Tableaux multidimensionnels (4)

```
<?php
// Un tableau multidimensionnel contenant des tableaux associatifs
$users = [
    // `john` est une clé complètement arbitraire
    // représentant un premier utilisateur
    'john' => [ // Un premier tableau associatif
        'name' => 'John Doe',
        'age' => 30,
        'city' => 'New York',
    ],
],
```

## Tableaux et boucles

```
// `jane` est une clé complètement arbitraire
// représentant un second utilisateur
'jane' => [ // Un deuxième tableau associatif
    'name' => 'Jane Doe',
    'age' => 25,
    'city' => 'Los Angeles',
],
];

echo $users['john']['name'] . "<br>"; // Affiche 'John Doe'
echo $users['jane']['age'] . "<br>"; // Affiche 25
echo $users['john']['city'] . "<br>"; // Affiche 'New York'
```

# Tableaux multidimensionnels (5)

```
// Équivalent en Java  
  
// Il n'est pas possible de créer un tableau associatif  
// en Java, mais nous pouvons utiliser une `HashMap` pour  
// obtenir un résultat similaire (non décrit ici).
```

# Les boucles

- Les boucles sont des structures de contrôle qui permettent d'exécuter un bloc de code plusieurs fois.
- Elles sont utilisées pour parcourir des tableaux ou des collections de données.
- Il existe plusieurs types de boucles en PHP : `for` , `while` , `do...while` et `foreach` .



# Boucle for (1)

- Utilisée pour exécuter un bloc de code un nombre fixe de fois.
- Composée de trois parties :
  1. L'initialisation du compteur
  2. La condition d'arrêt
  3. L'incrémentement du compteur
- Si la condition est vraie, la boucle continue. Sinon, la boucle s'arrête.



## Boucle `for` (2)

```
<?php
// Affiche les nombres de 0 à 9
for ($i = 0; $i < 10; $i++) {
    echo "$i<br>";
}
```

1. L'initialisation du compteur : `$i = 0`
2. La condition d'arrêt : `$i < 10`
3. L'incrémentement du compteur : `$i++`

# Boucle **for** (3)

```
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println(i);
        }
    }
}
```

# Boucle `while` (1)

- Utilisée pour exécuter un bloc de code tant qu'une condition est vraie.
- La condition est vérifiée avant chaque itération.
- Si la condition est vraie, la boucle continue. Sinon, la boucle s'arrête.



# Boucle `while` (2)

```
<?php
$i = 0;

// Affiche les nombres de 0 à 9
while ($i < 10) {
    echo "$i<br>";
    $i++;
}
```

# Boucle `while` (3)

```
public class Main {  
    public static void main(String[] args) {  
        int i = 0;  
  
        while (i < 10) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

## Boucle `do...while` (1)

- Similaire à la boucle `while`, mais avec une différence importante : la condition est vérifiée après chaque itération.
- La boucle s'exécute au moins une fois, même si la condition est fausse dès le départ.
- Si la condition est vraie, la boucle continue. Sinon, la boucle s'arrête.



## Boucle `do...while` (2)

```
<?php
$randomNumber = null;

do {
    // La fonction `rand()` génère un nombre aléatoire entre 1 et 10
    $randomNumber = rand(1, 10);
    echo "The random number is $randomNumber<br>";
} while ($randomNumber < 8);
```

## Boucle `do...while` (3)

```
public class Main {
    public static void main(String[] args) {
        int randomNumber = null;

        do {
            // La fonction `Math.random()` génère un nombre aléatoire
            // entre 1 et 10
            randomNumber = (int) (Math.random() * 10) + 1;
            System.out.println("The random number is " + randomNumber);
        } while (randomNumber < 5);
    }
}
```

# Boucle `foreach` (1)

- Utilisée pour parcourir les éléments d'un tableau ou d'une collection.
- La syntaxe est plus simple que celle des boucles `for` et `while`.
- La boucle `foreach` itère sur chaque élément du tableau ou de la collection.



## Boucle `foreach` (2)

```
<?php
$fruits = ['apple', 'banana', 'orange'];

// L'ordre des champs ici est inversé par rapport à Java !
foreach ($fruits as $fruit) {
    echo "$fruit<br>";
}
```

## Boucle `foreach` (3)

```
import java.util.Arrays;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<String> fruits = Arrays.asList("apple", "banana", "orange");

        // L'ordre des champs ici est inversé par rapport à PHP !
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

# Boucle `foreach` (4)

```
<?php
$users = [
    'john' => [
        'name' => 'John Doe',
        'age' => 30,
        'city' => 'New York',
    ],
    'jane' => [
        'name' => 'Jane Doe',
        'age' => 25,
        'city' => 'Los Angeles',
    ],
];
```

## Tableaux et boucles

```
// `$user` contient la valeur de l'élément du tableau
foreach ($users as $user) {
    echo "Name: {$user['name']}<br>";
    echo "Age: {$user['age']}<br>";
    echo "City: {$user['city']}<br>";
    echo "<br>";
}
```

```
// Équivalent en Java
```

```
// Il n'est pas possible de créer un tableau associatif
// en Java, mais nous pouvons utiliser une `HashMap` pour
// obtenir un résultat similaire (non décrit ici).
```

# Fonctions utiles pour les tableaux et les boucles (1)

- Nous étudierons les fonctions plus en détail dans les prochains contenus, mais voici la base :
  - Une fonction est un ensemble d'instructions qui prend des entrées, effectue une tâche spécifique et retourne un résultat.
  - Inspirées des mathématiques, les fonctions sont utilisées pour effectuer des calculs, manipuler des données, etc.
  - Par ex. :  $f(x) = x^2$ ,  $f(1) = 1$ ,  $f(2) = 4$ ,  $f(3) = 9$ , etc.

# Fonctions utiles pour les tableaux et les boucles (2)

- PHP propose plusieurs fonctions utiles pour travailler avec les tableaux et les boucles.
- Ces fonctions permettent de manipuler les tableaux, de les trier, de les filtrer, de les transformer, etc.
- La liste complète des fonctions possibles sur les tableaux est disponible dans la documentation officielle de PHP (<https://www.php.net/manual/fr/ref.array.php>), mais voici quelques-unes des fonctions les plus courantes.

## Fonctions `print()` et `print_r()` (1)

- La fonction `print()` affiche une chaîne de caractères (équivalent à `echo`).
- La fonction `print_r()` permet d'afficher un tableau et ses éléments de manière lisible.

Pour se souvenir de la différence : le `r` de `print_r()` peut être associé à "recursive" (récuratif en anglais) : affiche les éléments d'un tableau, puis les éléments des tableaux imbriqués à l'intérieur de ce tableau, et ainsi de suite.

# Fonctions `print()` et `print_r()` (2)

```
<?php
$fruits = ['apple', 'banana', 'orange', 'kiwi'];

print_r($fruits);
```

```
Array ( [0] => apple [1] => banana [2] => orange [3] => kiwi )
```

# Fonction `count()`

- La fonction `count()` retourne le nombre d'éléments dans un tableau.
- Elle peut être utilisée pour déterminer la taille d'un tableau.

```
<?php
$fruits = ['apple', 'banana', 'orange', 'kiwi'];

for ($i = 0; $i < count($fruits); $i++) {
    echo "$fruits[$i]<br>";
}
```

## Fonction `array_push()` (1)

- La fonction `array_push()` ajoute un ou plusieurs éléments à la fin d'un tableau.
- Elle modifie le tableau d'origine et retourne le nouveau nombre d'éléments.
- Elle est utile pour ajouter des éléments à un tableau sans avoir à spécifier l'index.

## Fonction `array_push()` (2)

```
<?php
$fruits = ['apple', 'banana', 'orange'];

array_push($fruits, 'kiwi', 'pear');

print_r($fruits);
```

```
Array ( [0] => apple [1] => banana [2] => orange [3] => kiwi [4] => pear )
```

# Conclusion

- Les tableaux permettent de stocker des collections de données : indexés, associatifs et multidimensionnels.
- Les boucles permettent de parcourir ces collections de données : `for`, `while`, `do...while` et `foreach`.
- Les tableaux ont eux aussi des fonctions utiles pour les manipuler.



# Questions

Est-ce que vous avez des questions ?

# À vous de jouer !

- (Re)lire le [support de cours](#).
- Explorer les [exemples de code](#).
- Faire les [exercices](#).
- Réaliser le [mini-projet](#).
- Poser des questions si nécessaire.

**N'allez pas trop vite, vous avez le temps !** N'hésitez pas à vous entraidez si vous avez des difficultés.



# Sources

- [Illustration principale](#) par [Richard Jacobs](#) sur [Unsplash](#)
- [Illustration](#) par [Aline de Nadai](#) sur [Unsplash](#)
- [Illustration](#) par [Faris Mohammed](#) sur [Unsplash](#)
- [Illustration](#) par [Maksym Kaharlytskyi](#) sur [Unsplash](#)
- [Illustration](#) par [Amol Tyagi](#) sur [Unsplash](#)
- [Illustration](#) par [NASA](#) sur [Unsplash](#)
- [Illustration](#) par [Justin](#) sur [Unsplash](#)
- [Illustration](#) par [Nikita Kachanovsky](#) sur [Unsplash](#)