Fonctions

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.

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 ce qu'est une fonction en programmation.
- Déclarer une fonction en PHP.
- Appeler une fonction en PHP.
- Passer des paramètres à une fonction en PHP.
- Utiliser une valeur de retour.
- Expliquer ce qu'est une portée de variable.

Objectifs (2)

- Utiliser des variables globales.
- Savoir où trouver la documentation sur les fonctions prédéfinies en PHP.
- Utiliser des fonctions prédéfinies en PHP.
- Réutiliser du code avec des fonctions.

Qu'est-ce qu'une fonction ? (1)

- Ensemble d'instructions pour effectuer une tâche spécifique.
- Inspirée des fonctions mathématiques :
  - Exemple : $f(x) = x^2$.
  - x est un paramètre.
  - $f(2) = 4$, $f(3) = 9$, etc.

Qu'est-ce qu'une fonction ? (2)

- En programmation :
  - Définie par un nom.
  - Peut accepter des paramètres.
  - Peut retourner une valeur.
- Permet de structurer le code.
- Peut être réutilisé à plusieurs endroits.

Déclarer une fonction en PHP (1)

- En PHP, une fonction est déclarée avec le mot-clé `function`.
- Suivi du nom de la fonction.
- Suivi des paramètres entre parenthèses (`()`).
- Suivi du corps de la fonction entre accolades (`{}`).

Déclarer une fonction en PHP (2)

```php
<?php
function hello() {
    echo "Hello, world!<br>";
}
```

```java
// Équivalent en Java
public class Main {
    public static void hello() {
        System.out.println("Hello, world!");
    }
}
```

Appeler une fonction en PHP (1)

- Pour appeler une fonction, il suffit d'écrire son nom suivi de parenthèses (`()`).
- Les paramètres peuvent être passés entre les parenthèses.
- Une fonction peut être appelée plusieurs fois dans le code.

Appeler une fonction en PHP (2)

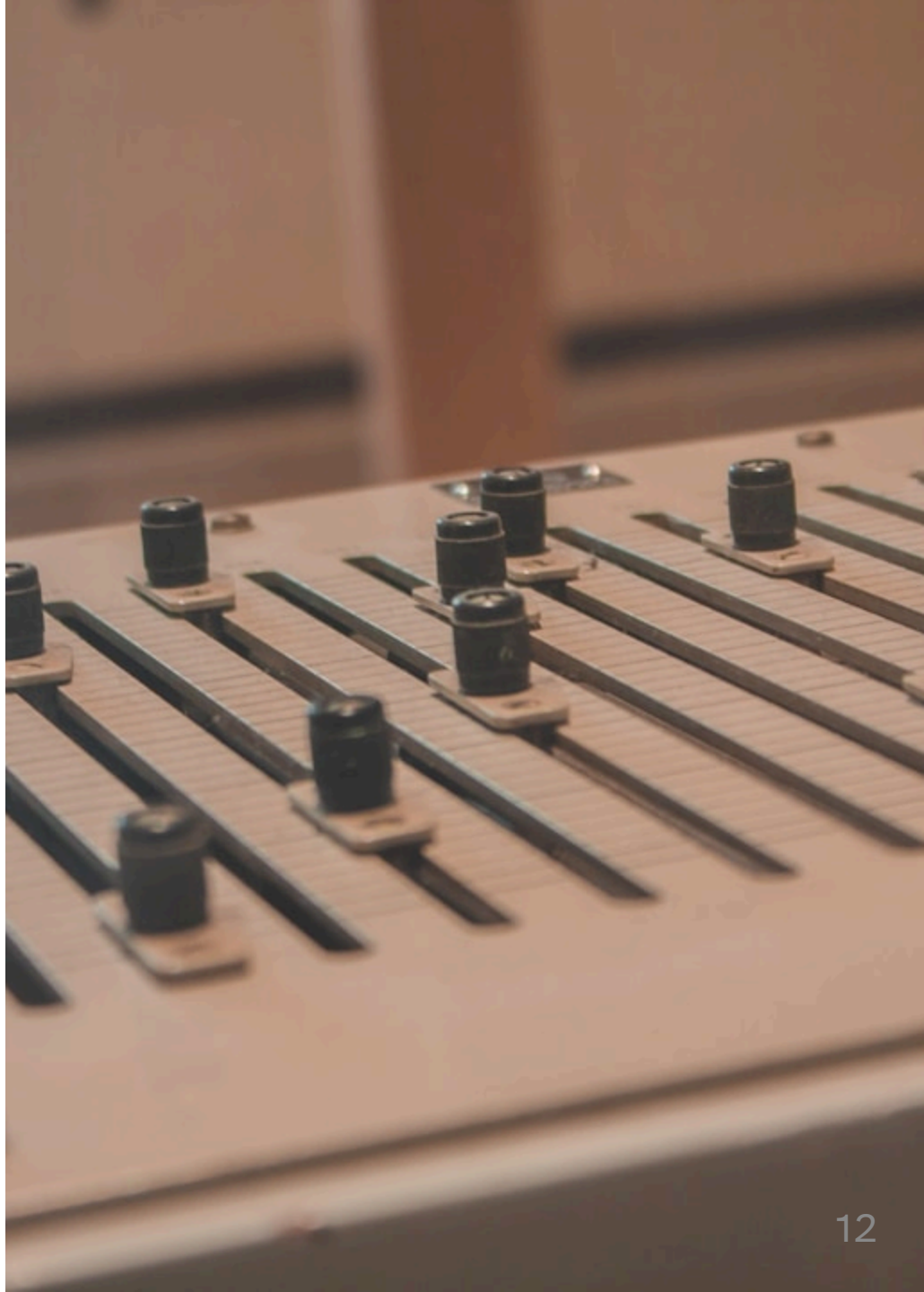
```php
<?php
function hello() {
    echo "Hello, world!<br>";
}

hello(); // Affiche "Hello, world!"
hello(); // Affiche "Hello, world!"
hello(); // Affiche "Hello, world!"
```

Appeler une fonction en PHP (3)

```
// Équivalent en Java
public class Main {
    public static void hello() {
        System.out.println("Hello, world!");
    }

    public static void main(String[] args) {
        hello(); // Affiche "Hello, world!"
        hello(); // Affiche "Hello, world!"
        hello(); // Affiche "Hello, world!"
    }
}
```

Passer des paramètres à une fonction (1)

- Une fonction peut accepter des paramètres.
- Les paramètres sont des valeurs que la fonction peut utiliser pour effectuer une tâche.
- Les paramètres sont déclarés entre les parenthèses de la fonction, séparés par des virgules (`,`).



Passer des paramètres à une fonction (2)

```php
<?php
function hello($name) {
    echo "Hello, $name!<br>";
}

hello("Alice"); // Affiche "Hello, Alice!"
hello("Bob"); // Affiche "Hello, Bob!"
```

Passer des paramètres à une fonction (3)

```
public class Main {
    public static void hello(String name) {
        System.out.println("Hello, " + name + "!");
    }

    public static void main(String[] args) {
        hello("Alice"); // Affiche "Hello, Alice!"
        hello("Bob"); // Affiche "Hello, Bob!"
    }
}
```

Retourner une valeur depuis une fonction (1)

- Une fonction peut retourner une valeur.
- La valeur retournée peut être utilisée dans le code appelant (= le code qui appelle la fonction).
- La valeur retournée (unique) est définie avec le mot-clé `return`.

Retourner une valeur depuis une fonction (2)

```php
<?php
function square($x) {
    return $x * $x;
}

$result = square(3);

echo $result; // Affiche 9
```

Retourner une valeur depuis une fonction (3)

```java
// Équivalent en Java
public class Main {
    public static int square(int x) {
        return x * x;
    }

    public static void main(String[] args) {
        int result = square(3);

        System.out.println(result); // Affiche 9
    }
}
```

Paramètres optionnels (1)

- En PHP, il est possible de définir des paramètres optionnels.
- Les paramètres optionnels ont une valeur par défaut.
- Les paramètres optionnels doivent être définis après les paramètres obligatoires.

Paramètres optionnels (2)

```php
<?php
function hello($name = "world") {
    echo "Hello, $name!<br>";
}

hello(); // Affiche "Hello, world!"
hello("Alice"); // Affiche "Hello, Alice!"
```

```php
// Équivalent en Java

// Il n'est pas possible de définir des paramètres optionnels en Java.
// Ceci est spécifique à PHP.
```

Passer plusieurs paramètres à une fonction (1)

- Une fonction peut accepter plusieurs paramètres.
- Les paramètres sont séparés par des virgules (`,`).
- Les paramètres sont passés dans le même ordre que leur déclaration.

Passer plusieurs paramètres à une fonction (2)

```php
<?php
function add($x, $y) {
    return $x + $y;
}

$result = add(3, 5);

echo $result; // Affiche 8
```

Passer plusieurs paramètres à une fonction (3)

```java
// Équivalent en Java
public class Main {
    public static int add(int x, int y) {
        return x + y;
    }

    public static void main(String[] args) {
        int result = add(3, 5);

        System.out.println(result); // Affiche 8
    }
}
```

Typer les paramètres et la valeur de retour d'une fonction (1)

- Depuis PHP 7, il est possible de typer les paramètres et la valeur de retour d'une fonction.
- Cela permet de s'assurer que les fonctions sont utilisées correctement et que les valeurs retournées sont du type attendu.
- Permet de limiter les erreurs et de rendre le code plus facile à comprendre.
- Le typage est facultatif, mais il est recommandé de l'utiliser autant que possible.

Typer les paramètres et la valeur de retour d'une fonction (2)

```php
<?php
function add(float $x, float $y): float {
    return $x + $y;
}
```

```php
<?php
$result = add(3, "Hello");

// Uncaught TypeError: add():
// Argument #2 ($y) must be of type float, string
```

Typer les paramètres et la valeur de retour d'une fonction (3)

Il est possible de passer plusieurs types de données en utilisant des types union :

```php
<?php
function add(int|float $x, int|float $y): float {
    return $x + $y;
}
```

Types de base : `int`, `float`, `string`, `bool`, `array`, `mixed`, `null` et `void`.

Typer les paramètres et la valeur de retour d'une fonction (4)

```php
<?php
function add(int $x, ?int $y): int {
    if ($y === null) {
        return $x;
    }

    return $x + $y;
}

add(3, null); // Retourne 3
add(3, 5); // Retourne 8
```

Typer les paramètres et la valeur de retour d'une fonction (5)

La syntaxe `?int` est une abréviation pour `int|null`.

```php
<?php
function find(array $numbers, int $id): ?int { // ou : int|null
    foreach ($numbers as $number) {
        if ($number === $id) {
            return $id;
        }
    }

    return null;
}
```

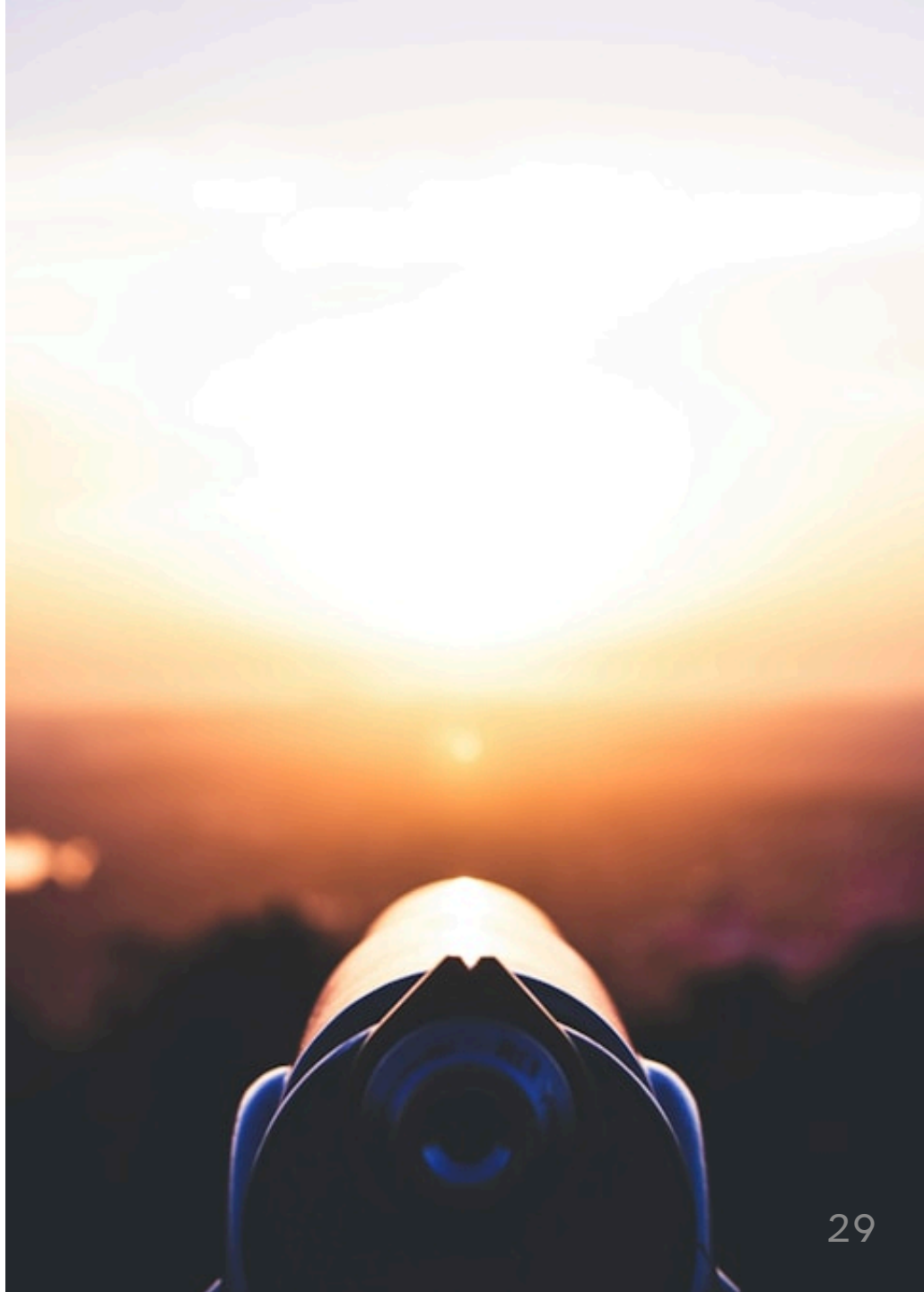
```php
$numbers = [1, 2, 3];

find($numbers, 2); // Retourne 2
find($numbers, 4); // Retourne null car 4 n'est pas dans le tableau $numbers
find([2, 4, 6], 4); // Retourne 4
find([2, 4, 6], 5); // Retourne null car 5 n'est pas dans le tableau [2, 4, 6]
```

De cette manière, la fonction `find` peut retourner soit un `int` (l'identifiant trouvé), soit `null` (si l'identifiant n'est pas trouvé dans le tableau).

Si une fonction est censée ne rien retourner, il est recommandé de typer la valeur de retour avec `void`.

Portée des variables (1)

- La portée d'une variable est l'endroit où elle peut être utilisée.
- Une variable déclarée à l'intérieur d'une fonction ne peut être utilisée qu'à l'intérieur de cette fonction.
- Une erreur survient si une variable est utilisée en dehors de sa portée.



Portée des variables (2)

```php
<?php
function square($x) {
    return $x * $x;
}

echo $x; // Erreur : variable $x non définie
```

Il n'est pas possible d'accéder à la variable `$x` en dehors de la fonction `square`, car elle a une portée locale à cette fonction.

Portée des variables (3)

```
// Équivalent en Java
public class Main {
    public static int square(int x) {
        return x * x;
    }

    public static void main(String[] args) {
        System.out.println(x); // Erreur : variable x non définie
    }
}
```

Variables globales (1)

- Une variable globale est une variable déclarée en dehors de toute fonction.
- Une variable globale peut être utilisée à l'intérieur d'une fonction à l'aide du mot-clé `global`.
- À éviter autant que possible, car cela rend le code difficile à comprendre et à maintenir.

Variables globales (2)

```php
<?php
$x = 42;

function square() {
    global $x;

    $x = $x * $x;
}

square();

echo $x; // Affiche 1764
```

Variables globales (3)

```
// Équivalent en Java
public class Main {
    public static int x = 42;

    public static int square() {
        x = x * x;
    }

    public static void main(String[] args) {
        square();

        System.out.println(x); // Affiche 1764
    }
}
```

Passage par valeur et par référence (1)

En programmation, les paramètres d'une fonction peuvent être de deux manières différentes :

- Passage par valeur.
- Passage par référence.

Passage par valeur et par référence (2)

Passage par valeur

Une copie de la variable est créée et utilisée dans la fonction.

Les modifications apportées à la variable dans la fonction n'affectent pas la variable d'origine.

Il s'agit du comportement par défaut en PHP mais il est aussi possible de passer des paramètres par référence.

Passage par valeur et par référence (3)

Passage par référence

Une référence à la variable d'origine est utilisée dans la fonction.

Les modifications apportées à la variable dans la fonction affectent la variable d'origine.

En Java, les types primitifs (int, float, etc.) sont passés par valeur, tandis que les objets sont passés par référence.

Peu utilisé, mais très pratique dans certains cas, par ex. les boucles.

Passage par valeur en PHP

Une copie de la variable est créée et utilisée dans la fonction.

```php
<?php
function increment($x) {
    $x++;
}

$value = 5;

increment($value);

echo $value; // Affiche 5, car $value n'est pas modifié par la fonction increment
```

Passage par référence en PHP

Une référence à la variable d'origine est utilisée dans la fonction.

```php
<?php
function increment(&$x) {
    $x++;
}

$value = 5;

increment($value);

echo $value; // Affiche 6, car $value est modifié par la fonction increment
```

Fonctions prédéfinies en PHP (1)

- PHP fournit de nombreuses fonctions prédéfinies.
- Ces fonctions permettent de réaliser des tâches courantes.
- La documentation officielle de PHP est une ressource précieuse pour trouver des fonctions prédéfinies.

Fonctions prédéfinies en PHP (2)

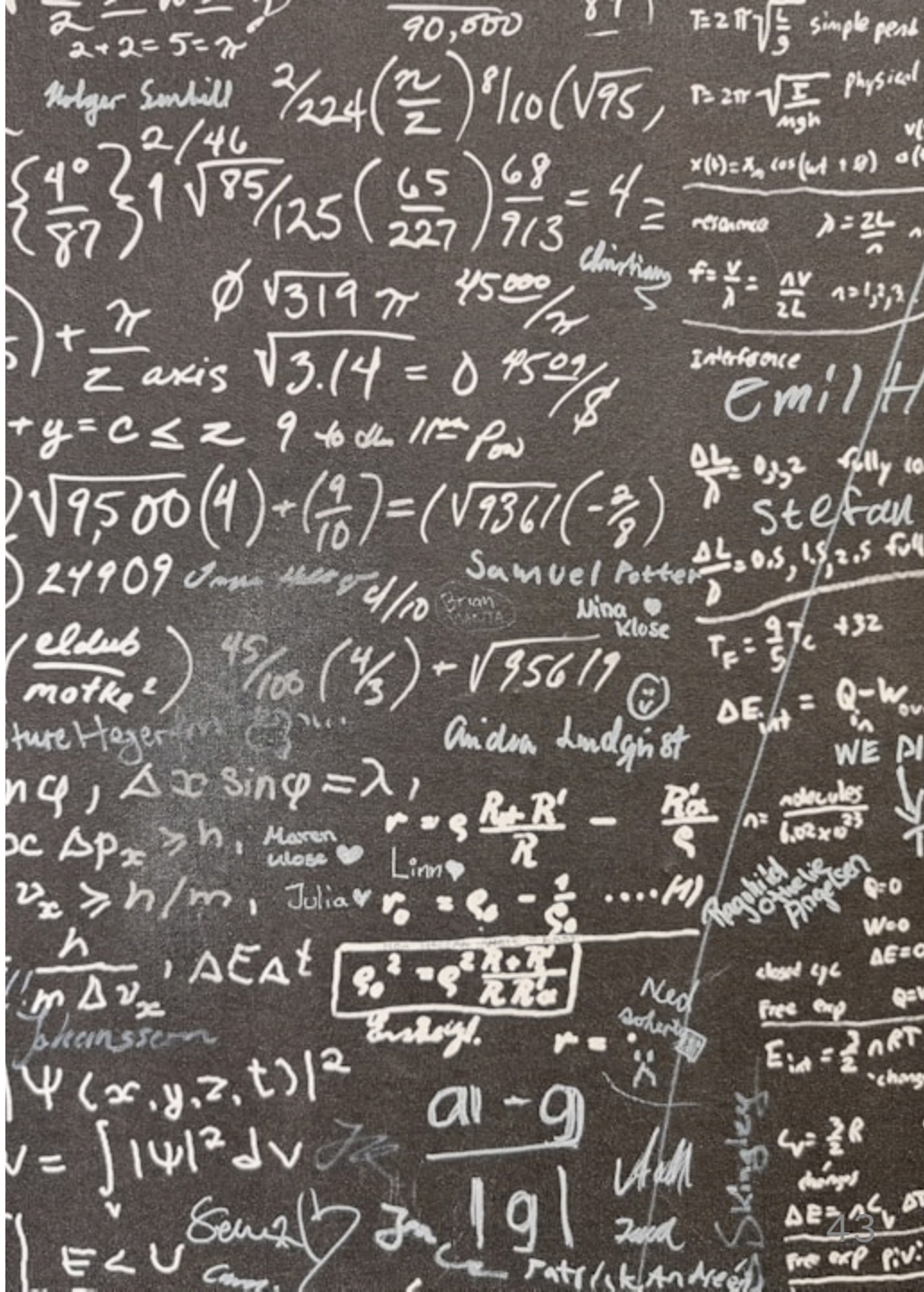
```php
<?php
$length = strlen("Hello, world!");

echo $length; // Affiche 13
```

Fonctions prédéfinies en PHP (3)

```
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        String s = "Hello, world!";
        int length = s.length();

        System.out.println(length); // Affiche 13
    }
}
```

Fonctions mathématiques (1)

- PHP propose de nombreuses fonctions mathématiques.
- Par exemple, `abs`, `sqrt`, `pow`, `min`, `max`, etc.
- Documentation complète.



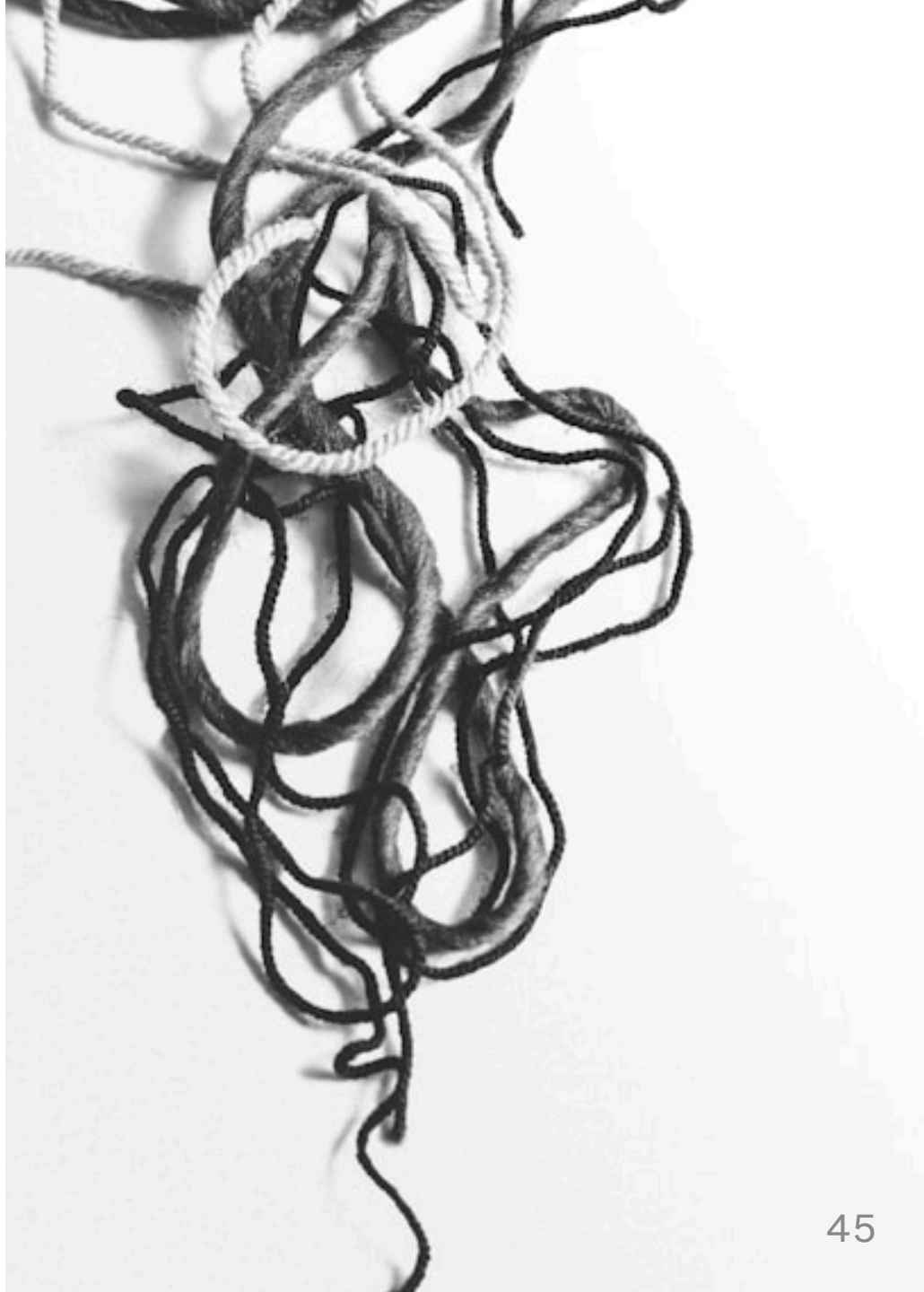
Fonctions mathématiques (2)

```php
<?php
$result = sqrt(16);

echo $result; // Affiche 4
```

```java
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        double result = Math.sqrt(16);
        System.out.println(result); // Affiche 4.0
    }
}
```

Fonctions sur les chaînes de caractères (1)

- PHP propose de nombreuses fonctions pour manipuler des chaînes de caractères.
- Par exemple, `strlen`, `substr`, `str_replace`, `strtolower`, `strtoupper`, etc.
- Documentation complète.



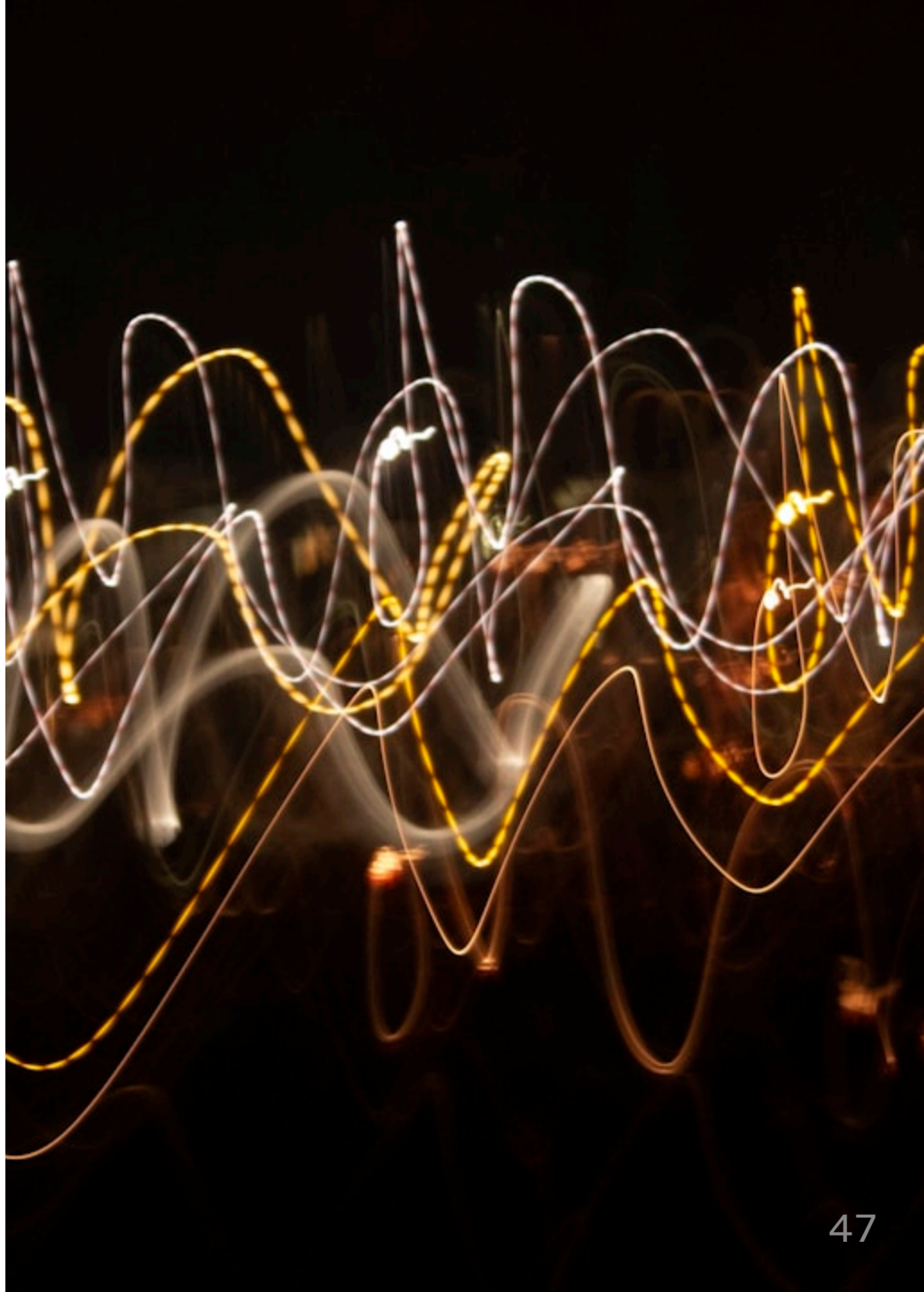
Fonctions sur les chaînes de caractères (2)

```php
$result = strtoupper("hello, world!");

echo $result; // Affiche "HELLO, WORLD!"
```

```java
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        String result = "hello, world!".toUpperCase();

        System.out.println(result); // Affiche "HELLO, WORLD!"
    }
}
```

Fonctions sur les variables (1)

- PHP propose de nombreuses fonctions pour manipuler des variables.
- Par exemple, `isset`, `empty`, `unset`, `is_array`, `is_string`, etc.
- Documentation complète.



Fonctions sur les variables (2)

```php
<?php
$var = 42;

if (isset($var)) {
    echo "The variable is defined.";
} else {
    echo "The variable is not defined.";
}

echo "<br>"; // Retour à la ligne HTML
```

```php
if (isset($undefined)) {
    echo "The variable is defined.";
} else {
    echo "The variable is not defined.";
}
```

Fonctions sur les variables (3)

```
// Équivalent en Java
public class Main {
    public static void main(String[] args) {
        int var = 42;

        if (var != null) {
            System.out.println("The variable is defined.");
        } else {
            System.out.println("The variable is not defined.");
        }

        System.out.println(); // Retour à la ligne
```

```
        int undefined;

        if (undefined != null) {
            System.out.println("The variable is defined.");
        } else {
            System.out.println("The variable is not defined.");
        }
    }
}
```

Réutiliser du code avec des fonctions (1)

- Les fonctions permettent de réutiliser du code.
- Le code est plus facile à lire et à maintenir.
- Il est possible d'importer des fonctions définies dans d'autres fichiers avec la directive `require_once`.

Réutiliser du code avec des fonctions (2)

```php
<?php
// Fichier `functions.php`
function hello(string $name): string {
    return "Hello, $name!<br>";
}
```

```php
<?php
// Fichier `index.php`
require_once __DIR__ . "/functions.php"; // On inclut le fichier

// La fonction `hello` est définie dans le fichier importé
// et peut être utilisée ici
echo hello("Alice");
```

Différence entre `include_once` et `require_once`

Il est possible d'importer des fichiers avec `include_once` et `require_once` :

- `include_once` : si le fichier n'est pas trouvé, un avertissement est émis.
- `require_once` : si le fichier n'est pas trouvé, une erreur fatale est émise.

Nous conseillons de toujours utiliser `require_once`.

Différence entre `require_once` et `require`

Il est également possible d'importer des fichiers avec `include` et `require` au lieu de `*_once` mais des problèmes peuvent survenir :

- `require_once` : le fichier est inclus une seule fois, même si la directive est utilisée plusieurs fois.
- `require` : le fichier est inclus à chaque fois que la directive est utilisée - cela peut entraîner des erreurs.

Il est recommandé d'utiliser `require_once` pour éviter les problèmes liés à l'inclusion multiple du même fichier.

Conclusion

- Les fonctions permettent de structurer et réutiliser du code.
- Les fonctions peuvent accepter des paramètres et retourner des valeurs.
- Fonctions personnelles ou des fonctions prédéfinies.
- La portée des variables est importante à comprendre.

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 (1)

- Illustration principale par Richard Jacobs sur Unsplash
- Illustration par Aline de Nadai sur Unsplash
- Illustration par Birmingham Museums Trust sur Unsplash
- Illustration par Aaron Burden sur Unsplash
- Illustration par Alexander Andrews sur Unsplash
- Illustration par Diane Picchiottino sur Unsplash
- Illustration par Mika Baumeister sur Unsplash
- Illustration par Florian Schmetz sur Unsplash

Sources (2)

- Illustration par Eric Prouzet sur Unsplash
- Illustration par Daniel Christie sur Unsplash
- Illustration par NASA sur Unsplash
- Illustration par Jeriden Villegas sur Unsplash
- Illustration par Thomas T sur Unsplash
- Illustration par Kier in Sight Archives sur Unsplash
- Illustration par Jan Huber sur Unsplash
- Illustration par Jack Church sur Unsplash

Sources (3)

- Illustration par Nikita Kachanovsky sur Unsplash