

Les conditions

Table des matières

I. Contexte	3
II. Les conditions	3
A. Qu'est-ce qu'une condition ?	3
B. L'instruction conditionnelle if...else	4
C. IF...ELSE.....	5
D. Exercice : Quiz.....	6
III. Switch et Match	7
A. Introduction.....	7
B. L'instruction conditionnelle switch	8
C. L'expression match.....	9
D. La différence entre switch et match	9
E. Exercice : Quiz	10
IV. Inclure des fichiers avec Require et include	12
A. Require et include	12
B. Include_once et require_once	12
C. Exercice : Quiz.....	13
V. Essentiel	14
VI. Auto-évaluation	14
A. Exercice	14
B. Test.....	15
Solutions des exercices	16

I. Contexte

Durée : 1 h

Prérequis : aucun

- Ordinateur avec Windows, macOS ou Linux
- Un navigateur récent : Firefox, Chrome, Safari, Opera
- Éditeur de code (Visual Code est conseillé)
- PHP > 8

Contexte

Les conditions en PHP sont utilisées pour prendre des décisions dans le code, en fonction de conditions spécifiques. Elles permettent de poser des questions à propos de certaines choses et les réponses orientent l'exécution selon des voies différentes.

Par exemple, dans un site de vente en ligne de vêtements, selon les pages du site nous pourrions avoir besoin d'afficher uniquement les articles de telle ou telle catégorie. Les conditions vont nous y aider. Les conditions sont utiles dans tous les langages informatiques et énormément de situations.

Nous verrons tout d'abord comment se construit une condition en PHP avec **if**, **else**, et **else if**, puis nous verrons les instructions conditionnelles **switch** et **match**. Enfin, nous aborderons l'inclusion d'un fichier dans un autre fichier grâce aux instructions **include** et **require**, ce qui vous permettra de gagner beaucoup de temps.

II. Les conditions

A. Qu'est-ce qu'une condition ?

Définition

En informatique, une condition est une instruction qui permet de prendre une décision dans l'exécution d'un programme en fonction de certaines données. Elle est utilisée pour tester si une condition est vraie ou fausse.

Les conditions sont utilisées dans de nombreux programmes informatiques, notamment en langages de programmation, en bases de données, dans les systèmes d'exploitation, etc.

En PHP comme dans tout langage informatique, les conditions sont un outil essentiel pour permettre aux programmes de prendre des décisions et d'adapter leur comportement en fonction des données qu'ils manipulent. Elles sont utilisées par exemple pour vérifier si une variable contient une certaine valeur, si une opération a réussi ou non, ou bien encore pour vérifier si un utilisateur a entré une information valide dans un formulaire.

Une instruction conditionnelle permet d'exécuter une ou plusieurs instructions selon la valeur d'une condition.

Méthode L'instruction conditionnelle IF

L'instruction conditionnelle **if** permet d'exécuter certaines parties du code seulement si une condition donnée est vérifiée.

- Si la condition est vraie (**true**), le code associé à l'instruction **if** est exécuté,
- Si la condition est fausse (**false**), le code est ignoré et le programme passe à la suite.

Ce type de condition est très utile pour rendre les programmes plus flexibles et adaptatifs, car il permet au programme de prendre des décisions en fonction des données d'entrée ou de l'état du programme.

Exemple

Prenons un exemple concret, permettant d'exécuter une instruction si l'utilisateur est majeur :

```
1 <?php
2 $age = 21;
3 if($age >= 18) {
4     echo 'Vous êtes majeur';
5 }
6 ?>
```

B. L'instruction conditionnelle if...else

Méthode

Alors que la condition **if** va uniquement tester si une condition est vraie, la condition **if...else**, va tester si elle est vraie, mais aussi préciser si elle est fausse.

Testons une condition :

- Si c'est vrai, on teste ce qu'il y a à l'intérieur.
- Si c'est faux, on n'exécute pas ce qu'il y a après le **if**, mais on continue jusqu'au **else**, et on exécute ce qu'il y a comme instruction après le **else**.

L'ordre est le suivant :

```
1 if (condition) {
2     // Code à exécuter si la condition est vraie
3 } else {
4     // Code à exécuter si la condition n'est pas vraie
5 }
```

Dans cet exemple, si l'utilisateur est majeur une instruction sera exécutée, tandis que s'il est mineur, une autre instruction le sera.

Exemple

```
1 <?php
2 $age = 32;
3 if($age >= 18) {
4     echo 'Vous êtes majeur';
5 } else {
6     echo 'Vous êtes mineur';
7 }
8 ?>
```

Exemple

Dans cet exemple, la variable **\$age** est définie à 32. Si **\$age** est supérieur ou égale à 18, le bloc de code suivant le **if** sera exécuté et le message « *Vous êtes majeur* » sera affiché. Si **\$age** est strictement inférieur à 18, c'est le bloc de code suivant l'instruction **else**, qui sera exécuté. Dans notre exemple **\$age** est à 32, donc il est supérieur à 18, ainsi le message « *Vous êtes majeur* » sera affiché.

C. IF...ELSE

Méthode Else if

Lorsqu'une seule condition n'est pas suffisante pour couvrir l'ensemble des situations, vous pouvez utiliser la structure si - sinon si - sinon.

Lorsqu'une condition n'est pas vraie (**false**), au lieu de continuer le programme, il est tout à fait possible d'exécuter du code en plus, grâce à l'instruction **else** (sinon).

L'ordre est le suivant :

```
1 <?php
2 $score = 77;
3 if ($score >= 95) {
4     echo "Excellent travail !";
5 } else if ($score >= 85) {
6     echo "Très bien !";
7 } else if ($score >= 70) {
8     echo "Pas mal !";
9 } else {
10    echo "Vous pouvez quand même mieux faire...";
11 }
12 ?>
```

Dans cet exemple, le code vérifie d'abord si la condition 1 (**\$score > 95**) est vraie. Si c'est le cas, le code dans le premier bloc de code est exécuté. Sinon, le code vérifie si la **condition 2 (\$score > 85)** est vraie. Si c'est le cas, le code dans le deuxième bloc de code est exécuté. Et ainsi de suite, jusqu'à ce qu'une des conditions soit vraie. Nous pouvons utiliser plusieurs **else if** à la suite, après un **if** initial, et le premier **else if** qui sera évalué à **true** sera exécuté, et cela sortira automatiquement de la condition.

Si aucune des conditions n'est vraie, le code dans le bloc **else** est exécuté (**echo « Vous pouvez quand même mieux faire... » ;**). Ce bloc est optionnel, mais il est recommandé de l'utiliser pour gérer les cas où aucune des conditions ne correspond.

Remarque

Vous pouvez écrire **else if** ou bien **elseif**, cela reviendra au même, sauf pour la condition ternaire où il faut obligatoirement utiliser l'expression **elseif**.

Exemple

Voici un exemple de l'utilisation du **elseif** :

```
1 <?php
2 $sport = "football";
3 if ($sport == "tennis") {
4     echo "Vous jouez au tennis";
5 } elseif ($sport == "basketball") {
6     echo "Vous jouez au basketball";
7 } elseif ($sport == "football") {
8     echo "Vous jouez au football";
9 } else {
10    echo "Vous ne jouez à aucun de ces 3 sports";
11 }
12 ?>
```

Méthode Opérateur ternaire

Afin de simplifier l'écriture des instructions **if...else**, vous pouvez utiliser l'opérateur ternaire qui permet d'effectuer une opération conditionnelle en une seule ligne afin de simplifier votre code.

```
1 (condition) ? valeur_si_vrai : valeur_si_faux;
```

L'opérateur ternaire se compose de trois parties :

- La condition qui est évaluée (entre parenthèses),
- **?** : L'opérande ternaire, qui est l'équivalent du **if**,
La valeur retournée si la condition est vraie,
- **:** L'équivalent du **else**,
La valeur retournée si la condition est fausse.

Voici un exemple d'utilisation de l'opérateur ternaire en PHP :

```
1 <?php
2 $age = 32;
3 $messageAge = ($age <= 18) ? "Vous êtes mineur." : "Vous êtes majeur.";
4 echo $messageAge;
5 ?>
```

Attention

Il est important d'utiliser l'opérateur ternaire avec parcimonie et de ne pas en abuser, car une utilisation excessive peut rendre le code plus difficile à lire.

Complément

Vous pourrez en apprendre davantage en vous référant à la documentation officielle PHP :

- Sur le `if`¹
- Sur le `else`²
- Et sur le `else if`³

D. Exercice : Quiz

[solution n°1 p.17]

Question 1

À quoi sert une condition ?

- ☐ Elle est utilisée pour tester si une condition est vraie ou fausse
- ☐ Elle sert à identifier une égalité stricte
- ☐ Elle est utilisée pour tester si une condition est vraie

Question 2

Ce morceau de code est correct.

1 <https://www.php.net/manual/fr/control-structures.if.php>

2 <https://www.php.net/manual/fr/control-structures.else.php>

3 <https://www.php.net/manual/fr/control-structures.elseif.php>

```
1 <?php
2 $a = 8;
3 $b = 38;
4 if ($a > $b) {
5     echo "a est plus grand que b";
6 } else if{
7     echo "b est égale ou plus grand que a";
8 }
9 ?>
```

- ☐ Vrai
- ☐ Faux

Question 3

Un opérateur ternaire est à utiliser le plus souvent possible pour simplifier la lecture de votre code par d'autres développeurs.

- ☐ Vrai
- ☐ Faux

Question 4

Qu'est-ce que ce code va afficher ?

```
1 <?php
2 $userType = 'manager';
3 $userName = "Vincent";
4 if ($userType === 'manager') {
5     echo 'Bienvenue sur le site '.$userName. ' !';
6 } else {
7     echo 'N\'étant pas manager du site, vous ne pouvez pas vous connecter.' ;
8 }
9 ?>
```

- ☐ Bienvenue sur le site manager !
- ☐ Bienvenue sur le site manager Vincent !
- ☐ Bienvenue sur le site Vincent !

Question 5

Quel est le code correct pour cette condition ternaire ?

- ☐ \$messageAge = (\$age <=18) ? "Vous êtes mineur." : "Vous êtes majeur.";
- ☐ \$messageAge = (\$age <=18) : "Vous êtes mineur." ? "Vous êtes majeur.";
- ☐ \$messageAge = (\$>18) ? "Vous êtes mineur." ? "Vous êtes majeur.";

III. Switch et Match

A. Introduction

En PHP, **switch** et **match** sont deux structures de contrôle qui permettent de comparer une valeur à une liste de cas possibles et d'exécuter un bloc de code correspondant.

B. L'instruction conditionnelle switch

Méthode

L'instruction **switch** est l'une des structures de contrôle les plus anciennes de PHP (elle existe depuis PHP 4) qui permet de tester une variable par rapport à plusieurs valeurs possibles et d'exécuter un le code correspondant au cas qui correspond à la variable. Cette instruction équivaut à une série d'instructions **if**.

```
1 <?php
2     switch ($variable) {
3     case 'valeur1':
4         // Code à exécuter.
5         break;
6     case 'valeur2':
7         // Code à exécuter.
8         break;
9     default:
10        // Code à exécuter.
11    }
12 ?>
```

Voici la structure d'un **switch** :

- **switch** : le mot-clé indiquant que nous allons utiliser le **switch**.
- **\$variable** : la valeur à comparer dans les **case**.
- **case 'valeur'** : les différentes valeurs de comparaison.
- **break** : l'instruction indiquant de sortir du switch quand on a rencontré un **case**.
- **default** : sert à gérer tous les autres cas et sera exécuté si aucun **case** ne correspond à la valeur de la variable. C'est l'équivalent d'un ELSE.

Exemple

```
1 <?php
2     $fruit = "pomme";
3     switch ($fruit) {
4     case "cerise":
5         echo "Vous avez mangé une cerise";
6         break;
7     case "fraise":
8         echo "Vous avez mangé une orange";
9         break;
10    case "pomme":
11        echo "Vous avez mangé une pomme";
12        break;
13    default:
14        echo "Je ne sais pas quel fruit vous avez mangé";
15    }
16 ?>
```

Dans cet exemple, la variable **\$fruit** est définie à **"pomme"**. L'instruction switch teste la valeur de **\$fruit** et exécute le code correspondant au premier cas qui correspond. Si aucun des cas ne correspond, le code dans la section default est exécuté.

Attention

Pensez à mettre le mot-clé **break** à chaque fin de bloc case, si vous l'oubliez, le programme continuera de comparer les **case** les uns après les autres et tous les **case** seront affichés si vous utilisez un **echo**.

C. L'expression match

Méthode

La principale différence entre les expressions **switch** et **match** est que **match** a été introduit dans la version 8 de PHP et est une structure plus moderne et plus souple que **switch**.

```
1 match ($value) {  
2     cle1 => valeur1,  
3     cle2 => valeur1,  
4     // code  
5     default => valeur n  
6 }
```

Voici le fonctionnement de la structure **match** :

1. La structure **match** commence par prendre une variable comme argument. Cette variable sera comparée à des motifs (patterns) ultérieurs.
2. Ensuite, chaque motif est énuméré dans la structure **match** avec une expression correspondante qui sera évaluée si la variable correspond à ce motif.
3. Si la valeur de la variable correspond à l'un des motifs énumérés, l'expression correspondante est évaluée et le résultat est retourné. Il n'est pas nécessaire d'utiliser un mot-clé **break** pour sortir de la structure **match**.
4. Si la valeur de la variable ne correspond à aucun des motifs énumérés, la section **default** sera évaluée, s'il est spécifié.

Exemple

Voici un exemple avec **match** :

```
1 <?php  
2  
3 $color = "rouge";  
4 $result = match ($color) {  
5     "rouge" => "Le tee-shirt est rouge.",  
6     "vert" => "Le tee-shirt est le vert.",  
7     "bleu" => "Le tee-shirt est le bleu.",  
8     default => "La couleur du tee-shirt est inconnue".  
9 };  
10 echo $result; // Affiche "Le tee-shirt est rouge."  
11 ?>
```

D. La différence entre switch et match

Méthode

À la différence de **switch** qui compare les valeurs avec une égalité faible (**==**), **match** le fait avec un contrôle d'égalité stricte (**===**) en vérifiant si les deux valeurs sont du même type de variable.

Exemple

Pour bien voir la différence entre l'utilisation de **switch** et celle de **match**, voici un code avec **switch** :

```
1 <?php  
2 switch (2) {  
3     case 0:  
4         $result = 'Paul';  
5         break;
```

```

6         case 1:
7             $result = 'Marie';
8             break;
9         case 2:
10            $result = 'Jean';
11            break;
12 }
13 echo $result;
14 // Affiche "Jean"
15 ?>

```

Méthode

Voici comment l'utiliser avec **match**, pour obtenir exactement le même résultat, beaucoup plus rapidement.

```

1 <?php
2     echo match (2) {
3         0 => 'Paul',
4         1 => 'Marie',
5         2 => 'Jean',
6     } ;
7     // Affiche également "Jean"
8 ?>

```

Complément

Vous pourrez en apprendre davantage sur le switch en vous référant à la documentation officielle : switch¹, ainsi que sur le match : match² et sur la différence entre le switch et le match : RFC : match expression v2³.

E. Exercice : Quiz

[solution n°2 p.18]

Question 1

La syntaxe **match** a été introduite en PHP à partir de :

- ☐ La version 6
- ☐ La version 7
- ☐ La version 8

Question 2

L'instruction break est facultative dans un switch :

- ☐ Vrai
- ☐ Faux

Question 3

1 <https://www.php.net/manual/fr/control-structures.switch.php>

2 <https://www.php.net/manual/fr/control-structures.match.php>

3 https://wiki.php.net/rfc/match_expression_v2

Indiquez le bon ordre des instructions dans un switch :

- ☐ \$variable/switch/break/case/default/
- ☐ \$variable/switch/case/break/default
- ☐ switch/\$variable/case/break/default

Question 4

Ce code est correct.

```
1 <?php
2     $day = ' Lundi ' ;
3     switch($day) {
4         case ' Lundi ' :
5             echo ' Aujourd'hui c'est lundi ! ' ;
6             break ;
7         case ' Mardi ' :
8             echo ' Aujourd'hui c'est mardi ! ' ;
9             break ;
10        case ' Mercredi ' :
11            echo ' Aujourd'hui c'est mercredi ! ' ;
12            break ;
13        case ' Jeudi ' :
14            echo ' Aujourd'hui c'est jeudi ! ' ;
15            break ;
16        case ' Vendredi ' :
17            echo ' Aujourd'hui c'est vendredi ! ' ;
18            break ;
19        case ' Samedi ' :
20            echo ' Aujourd'hui c'est samedi ! ' ;
21            break ;
22        case ' Dimanche ' :
23            echo ' Aujourd'hui c'est dimanche ! ' ;
24            break ;
25        default :
26            echo ' Jour invalide ! ' ;
27    }
28 ?>
```

- ☐ Vrai
- ☐ Faux

Question 5

switch et match font exactement la même chose.

- ☐ Vrai
- ☐ Faux

IV. Inclure des fichiers avec Require et include

A. Require et include

Définition **Require et include**

Les instructions **require** et **include** permettent d'inclure un fichier de code à l'intérieur d'un autre fichier PHP. La principale différence entre les deux est la gestion des erreurs. Ces instructions sont utilisées pour charger un fichier externe dans votre script PHP.

Différence entre les instructions require et include

L'instruction **require** est plus stricte que **include**. Si le fichier spécifié dans l'instruction **require** ne peut pas être inclus pour une raison quelconque (par exemple, s'il n'existe pas ou s'il est indisponible), le script PHP s'arrêtera et affichera une erreur fatale. Cela signifie que si le fichier inclus est vital pour le fonctionnement du script, l'utilisation de **require** est plus appropriée.

Exemple

Voici un exemple d'utilisation de **require** pour intégrer le footer dans une page :

```
1 <?php
2     require 'footer.php' ;
3 ?>
```

L'instruction **include**, quant à elle, est plus souple que **require**. Si le fichier spécifié dans l'instruction **include** ne peut pas être inclus pour une raison quelconque, le script PHP continuera son exécution et affichera un avertissement non fatal. Cela signifie que si le fichier inclus n'est pas vital pour le fonctionnement du script, l'utilisation de **include** est plus appropriée.

Exemple

Voici un exemple d'utilisation de **include** :

```
1 <?php
2     Include 'footer.php' ;
3 ?>
```

B. Include_once et require_once

Les instructions **require** et **include** permettent d'inclure plusieurs fois un même fichier. Tandis que leurs variantes **require_once** et **include_once** nous l'interdisent, avec lesquelles on ne pourra inclure un même fichier qu'une seule fois.

Attention

En général, il est recommandé d'utiliser **require_once** pour inclure des fichiers, car cela permet d'éviter des problèmes éventuels si le fichier est inclus plusieurs fois. Cependant, il peut parfois être nécessaire d'utiliser **require** si le fichier doit être inclus à chaque fois que l'instruction est exécutée.

Définition

La fonction **declare()** en PHP permet de définir des options de configuration pour le code à l'intérieur de la portée où elle est définie, ce qui peut affecter le comportement du code de différentes manières.

Complément

Vous pourrez en apprendre davantage sur la façon d'inclure un fichier dans un autre fichier PHP en vous référant à la documentation officielle :

- **Require**¹
- **Include**²
- **Require_once**³
- **Include_once**⁴

De plus, la fonction DECLARE est expliquée dans la documentation officielle : declare⁵

C. Exercice : Quiz

[solution n°3 p.20]

Question 1

L'instruction require_once est recommandée à require pour inclure des fichiers.

- ☐ Vrai
- ☐ Faux

Question 2

L'instruction require est moins stricte que include.

- ☐ Vrai
- ☐ Faux

Question 3

Nous pouvons inclure autant de fois que l'on souhaite l'instruction require_once d'un fichier dans un autre fichier.

- ☐ Vrai
- ☐ Faux

Question 4

Pour inclure un header dans un fichier, quelle instruction faut-il écrire ?

- ☐ <?php require_once (header.php); ?>
- ☐ <?php require_once ("header.php"); ?>
- ☐ <?php require_once "header.php"; ?>

Question 5

1 <https://www.php.net/manual/fr/function.require.php>

2 <https://www.php.net/manual/fr/function.include.php>

3 <https://www.php.net/manual/fr/function.require-once.php>

4 <https://www.php.net/manual/fr/function.include-once.php>

5 <https://www.php.net/manual/fr/control-structures.declare.php>

Je suis une instruction qui inclut un fichier dans le script PHP et interrompt l'exécution du script si celui-ci ne peut pas être inclus. Qui suis-je ?

- ☐ Include_once
- ☐ Require
- ☐ Require_once
- ☐ Include

V. Essentiel

Dans ce cours, nous avons traité des structures conditionnelles qui permettent au programme de suivre un chemin. La structure en **if...else** et **if...else if...else** seront suffisantes. La structure **if** : permet d'exécuter un bloc de code si la condition donnée est vraie. Elle peut être suivie d'une structure **else if** ou **else** pour gérer d'autres cas. La structure **else if** : permet de gérer des cas supplémentaires si la première condition est fausse. Elle peut être utilisée plusieurs fois. Enfin, la structure **else** : permet d'exécuter un bloc de code si aucune des conditions précédentes n'est vraie.

Il est également possible d'utiliser la condition ternaire pour simplifier l'écriture de la structure conditionnelle en une seule ligne. Il ne faut pas non plus en abuser, car cela pourrait rendre illisible le code.

Par contre s'il y a un nombre important d'embranchements, on pourra utiliser soit **switch**, soit **match**, depuis PHP 8.

Il est important de noter que ces structures conditionnelles peuvent être imbriquées les unes dans les autres pour gérer des cas plus complexes. Cependant, il est recommandé de garder une structure simple et lisible pour faciliter la compréhension du code.

Dans le cas pratique, nous allons utiliser tout cela pour déterminer si un individu peut aller en retraite ou devra encore patienter pour cela !

Nous avons également vu comment inclure des fichiers dans d'autres fichiers avec les instructions PHP **include**, **include_once**, **require** et **require_once**, et quand les utiliser suivant les besoins.

VI. Auto-évaluation

A. Exercice

Vous êtes développeur dans une agence Web, et votre patron, Jean-Claude, souhaite savoir si cette année il pourra enfin prendre sa retraite bien méritée, pour enfin vous laisser les rênes de l'agence.

Question 1

[solution n°4 p.21]

Pour cela en utilisant les notions apprises pendant le cours, vous allez écrire un programme qui lui indiquera suivant son âge :

- S'il est encore loin de la retraite s'il a strictement moins de 60 ans,
- S'il s'en approchera s'il a entre 60 ans et strictement moins de 64 ans,
- S'il peut prendre sa retraite cette année s'il a 64 ans,
- Ou bien s'il peut la prendre dès aujourd'hui, car il a 65 ans ou plus.

Indice :

Utilisez la structure **if**, **else if** et **else**.

Question 2

[solution n°5 p.21]

Maintenant, Jean-Claude vous demande de réaliser la même chose, mais cette fois-ci avec un **switch**.

Indice :

Pensez bien à utiliser les **break** à la fin de chaque case.

B. Test**Exercice 1 : Quiz**

[solution n°6 p.21]

Question 1

Parmi ces expressions, sélectionnez les différentes structures conditionnelles.

- ☐ If...else if..else
- ☐ If...else...else if
- ☐ Écriture conditionnelle
- ☐ Écriture ternaire

Question 2

Que va afficher ce programme ?

```
1 <?php
2     $note = 10;
3     if ($note > 10) {
4         echo "L'élève a réussi à son examen.";
5     } else {
6         echo "L'élève a échoué à son examen.";
7     }
8 ?>
```

- ☐ L'élève a échoué à son examen
- ☐ L'élève a réussi son examen
- ☐ Aucune des deux réponses

Question 3

Que va afficher ce programme ?

```
1 <?php
2     $carBrand = "Peugeot";
3     switch ($carBrand) {
4         case "Fiat":
5             echo "La voiture est de la marque Fiat.";
6             break;
7         case "Peugeot":
8             echo "La voiture est de la marque Peugeot.";
9             break;
10        case "Renault":
11            echo "La voiture est de la marque Renault.";
12            break;
13        default:
14            echo "La marque de la voiture est inconnue.";
15    }
16 ?>
```

- ☐ La marque de la voiture est Renault
- ☐ La voiture est de la marque Peugeot.
- ☐ La voiture est de la marque Renault
- ☐ La marque de la voiture est inconnue

Question 4

Imaginons une loterie qui va tirer au sort un numéro parmi 5 numéros. Quel sera le résultat ?

```
1 <?php
2 switch (1) {
3     case 0:
4         $result = 'Hugues';
5         break;
6     case 1:
7         $result = 'Simone';
8         break;
9     case 2:
10        $result = 'Yves';
11        break;
12     case 3:
13        $result = 'Bryan';
14        break;
15     case 4:
16        $result = 'Kimberley';
17        break;
18 }
19 echo 'Le vainqueur du concours est '.$result.', et remporte le voyage à Quiberon';
20 ?>
```

- ☐ Le vainqueur du concours est Hugues, et remporte le voyage à Quiberon
- ☐ Le vainqueur du concours est Simone, et remporte le voyage à Quiberon
- ☐ Le vainqueur du concours est Yves, et remporte le voyage à Quiberon

Question 5

Si le fichier qui est inclus avec `include_once` n'existe pas, PHP émettra une erreur fatale et le script s'arrêtera.


- ☐ Vrai
- ☐ Faux

Solutions des exercices

Exercice p. 6 Solution n°1

Question 1

À quoi sert une condition ?

- ☒ Elle est utilisée pour tester si une condition est vraie ou fausse
- ☐ Elle sert à identifier une égalité stricte
- ☐ Elle est utilisée pour tester si une condition est vraie
-  En langage de programmation, une condition est une expression qui évalue une situation à vrai ou à faux. Elle permet d'exécuter un bloc de code ou une instruction si une condition est remplie, ou bien de passer à une autre instruction si la condition n'est pas remplie.


Question 2

Ce morceau de code est correct.

```
1 <?php
2 $a = 8;
3 $b = 38;
4 if ($a > $b) {
5 echo "a est plus grand que b";
6 } else if{
7 echo "b est égale ou plus grand que a";
8 }
9 ?>
```

☐ Vrai

☒ Faux

 Ce code est incorrect, car la condition ne termine pas ici par un **else**, mais par un **else if**. Le code correct aurait été celui-ci :


```
1 <?php
2 $a = 8;
3 $b = 38;
4 if ($a > $b) {
5 echo "a est plus grand que b";
6 } else {
7 echo "b est plus grand que a";
8 }
9 ?>
```

Question 3

Un opérateur ternaire est à utiliser le plus souvent possible pour simplifier la lecture de votre code par d'autres développeurs.

☐ Vrai


☒ Faux

 C'est faux. Il est important de choisir le bon moment pour l'utiliser et de le faire de manière claire et compréhensible pour les autres développeurs qui liront votre code.

Question 4


Qu'est-ce que ce code va afficher ?

```
1 <?php
2 $userType = 'manager';
3 $userName = "Vincent";
4 if ($userType === 'manager') {
5 echo 'Bienvenue sur le site '.$userName. ' !';
6 } else {
7 echo 'N\'étant pas manager du site, vous ne pouvez pas vous connecter.' ;
8 }
9 ?>
```

- ☐ Bienvenue sur le site manager !
- ☐ Bienvenue sur le site manager Vincent !
- ☒ Bienvenue sur le site Vincent !
-  Bienvenue sur le site Vincent ! En effet, la variable \$userName correspond ici à la chaîne de caractères : "Vincent".

Question 5


Quel est le code correct pour cette condition ternaire ?

- ☒ \$messageAge = (\$age <=18) ? "Vous êtes mineur." : "Vous êtes majeur.";
- ☐ \$messageAge = (\$age <=18) : "Vous êtes mineur." ? "Vous êtes majeur.";
- ☐ \$messageAge = (\$>18) ? "Vous êtes mineur." ? "Vous êtes majeur.";
-  La première réponse est la seule correcte, en effet, une condition ternaire doit s'écrire sous cette forme : (condition) ? valeur_si_vrai : valeur_si_faux;

Exercice p. 10 Solution n°2


Question 1

La syntaxe **match** a été introduite en PHP à partir de :

- ☐ La version 6
- ☐ La version 7
- ☒ La version 8
-  La syntaxe **match** a été introduite depuis la version PHP 8, il ne faut pas l'utiliser sur les versions PHP antérieures.

Question 2


L'instruction break est facultative dans un switch :

- ☐ Vrai
- ☒ Faux
-  L'instruction **break** est obligatoire à la fin de chaque **case**, son oubli provoquerait des erreurs dans les résultats attendus.

Question 3

Indiquez le bon ordre des instructions dans un switch :

- ☐ \$variable/switch/break/case/default/
- ☐ \$variable/switch/case/break/default
- ☒ switch/\$variable/case/break/default


 Voici l'ordre des instructions dans un **switch** :
switch/\$variable/case'valeur'/break/default.

Question 4

Ce code est correct.

```
1 <?php
2     $day = ' Lundi ' ;
3     switch($day) {
4         case ' Lundi ' :
5             echo ' Aujourd'hui c'est lundi ! ' ;
6             break ;
7         case ' Mardi ' :
8             echo ' Aujourd'hui c'est mardi ! ' ;
9             break ;
10        case ' Mercredi ' :
11            echo ' Aujourd'hui c'est mercredi ! ' ;
12            break ;
13        case ' Jeudi ' :
14            echo ' Aujourd'hui c'est jeudi ! ' ;
15            break ;
16        case ' Vendredi ' :
17            echo ' Aujourd'hui c'est vendredi ! ' ;
18            break ;
19        case ' Samedi ' :
20            echo ' Aujourd'hui c'est samedi ! ' ;
21            break ;
22        case ' Dimanche ' :
23            echo ' Aujourd'hui c'est dimanche ! ' ;
24            break ;
25        default :
26            echo ' Jour invalide ! ' ;
27    }
28 ?>
```

- ☒ Vrai
- ☐ Faux

 Ce code est correct, il utilise toutes les conditions requises pour un **switch** : la variable, le **switch**, le **case**, le **break** et le **default**.

Question 5

switch et match font exactement la même chose.

- ☐ Vrai
- ☒ Faux

Q **switch** est une structure de contrôle qui permet de tester une variable ou une expression et d'exécuter différentes actions en fonction de la valeur de cette variable ou de cette expression. L'instruction **match**, quant à elle, est une structure de contrôle introduite dans PHP 8.0. Elle permet de tester une expression et d'exécuter une action en fonction de sa valeur.

Exercice p. 13 Solution n°3

Question 1

L'instruction `require_once` est recommandée à `require` pour inclure des fichiers.

☒ Vrai

☐ Faux

Q Il est fortement recommandé d'utiliser l'instruction **`require_once`** pour inclure des fichiers, afin d'éviter des problèmes éventuels si le fichier est inclus plusieurs fois.

Question 2

L'instruction `require` est moins stricte que `include`.

☐ Vrai

☒ Faux

Q Si le fichier spécifié dans l'instruction **`require`** ne peut pas être inclus, le script PHP s'arrêtera et affichera une erreur fatale. Cela peut être le cas si le fichier requis n'existe pas ou bien est indisponible.

Question 3

Nous pouvons inclure autant de fois que l'on souhaite l'instruction `require_once` d'un fichier dans un autre fichier.

☐ Vrai

☒ Faux

Q Non, il n'est pas possible d'inclure plusieurs fois un fichier avec l'expression **`require_once`** en PHP. Cette fonction a pour but de vérifier si le fichier a déjà été inclus auparavant, et si c'est le cas, elle ne l'inclut pas à nouveau.

Question 4

Pour inclure un header dans un fichier, quelle instruction faut-il écrire ?

☐ `<?php require_once (header.php); ?>`

☐ `<?php require_once ("header.php"); ?>`

☒ `<?php require_once "header.php"; ?>`

Q La syntaxe correcte est **`<?php require_once "header.php"; ?>`**.

En effet, il faut débiter par l'instruction **`require_once`** (ou **`include_once`**, suivant les besoins) puis du nom du fichier que l'on appelle, entre guillemets simples ou bien doubles.

Question 5

Je suis une instruction qui inclut un fichier dans le script PHP et interrompt l'exécution du script si celui-ci ne peut pas être inclus. Qui suis-je ?

- ☐ Include_once
- ☒ Require
- ☒ Require_once
- ☐ Include

Q La fonction est **require_once**, car si le fichier spécifié dans **require_once** ne peut pas être trouvé ou ne peut pas être inclus pour une raison quelconque, PHP émettra une erreur fatale et le script s'arrêtera pour éviter de servir un code incomplet et incohérent, contrairement à **include_once** qui elle ne provoquera pas une erreur fatale si le fichier ne peut pas être inclus.

p. 14 Solution n°4

Voici un exemple de code possible en utilisant **if**, **else if** et **else**. D'autres codes sont tout à fait possibles :

```

1 <?php
2     $age = 59;
3
4     if ( $age === 64 ) {
5         echo "C'est bon, Jean-Claude vous pourrez prendre votre retraite cette année
6     ";
7     } elseif ( $age >= 60 && $age < 64 ) {
8         echo "Dernière ligne droite avant la retraite";
9     } elseif ( $age < 60 && $age > 0 ) {
10        echo "La retraite, c'est pas pour cette année";
11    } else {
12        echo "Jean-Claude, stoppez le travail tout de suite, et en avant la retraite
13    !";
14    }
15 ?>

```

p. 14 Solution n°5

Voici un exemple de code possible en utilisant l'instruction **switch** :

```

1 <?php
2 $age = 59;
3
4 switch ( $age ) {
5     case 64:
6         echo "C'est bon, Jean-Claude, vous pourrez prendre votre retraite cette année.";
7         break;
8     case ( $age >= 60 && $age < 64 ):
9         echo "Dernière ligne droite avant la retraite.";
10        break;
11    case ( $age < 60 && $age > 0 ):
12        echo "La retraite, ce n'est pas pour cette année.";
13        break;
14    default:
15        echo "Jean-Claude, stoppez le travail tout de suite, et en avant la retraite !";
16        break;
17    }
18 ?>

```

Exercice p. 15 Solution n°6

Question 1

Parmi ces expressions, sélectionnez les différentes structures conditionnelles.

- ☒ If...else if..else
- ☐ If...else...else if
- ☐ Écriture conditionnelle
- ☒ Écriture ternaire

Q Le **if...else if..else** et l'écriture ternaire sont des structures conditionnelles permettant aux développeurs d'utiliser des conditions en PHP. Par contre **if...else...else if** n'existe pas et l'écriture conditionnelle non plus.

Question 2

Que va afficher ce programme ?

```
1 <?php
2     $note = 10;
3     if ($note > 10) {
4         echo "L'élève a réussi à son examen.";
5     } else {
6         echo "L'élève a échoué à son examen.";
7     }
8 ?>
```

- ☒ L'élève a échoué à son examen
- ☐ L'élève a réussi son examen
- ☐ Aucune des deux réponses

Q La réponse est « *L'élève a échoué à son examen* ». Il faut bien regarder les signes, et dans le if, il faut que l'élève ait une note supérieure (et non égale) à 10 pour réussir son examen, or il n'a eu que 10.

Question 3

Que va afficher ce programme ?

```
1 <?php
2     $carBrand = "Peugeot";
3     switch ($carBrand) {
4         case "Fiat":
5             echo "La voiture est de la marque Fiat.";
6             break;
7         case "Peugeot":
8             echo "La voiture est de la marque Peugeot.";
9             break;
10        case "Renault":
11            echo "La voiture est de la marque Renault.";
12            break;
13        default:
14            echo "La marque de la voiture est inconnue.";
15    }
16 ?>
```

- ☐ La marque de la voiture est Renault
- ☒ La voiture est de la marque Peugeot.
- ☐ La voiture est de la marque Renault
- ☐ La marque de la voiture est inconnue

Q Il manque un break à la fin du case « Peugeot ». Voici à quoi devrait ressembler le code :

```

1 <?php
2     $carBrand = "Peugeot";
3     switch ($carBrand) {
4         case "Fiat":
5             echo "La voiture est de la marque Fiat.";
6             break;
7         case "Peugeot":
8             echo "La voiture est de la marque Peugeot.";
9             break;
10        case "Renault":
11            echo "La voiture est de la marque Renault.";
12            break;
13        default:
14            echo "La marque de la voiture est inconnue.";
15    }
16 ?>

```

Question 4

Imaginons une loterie qui va tirer au sort un numéro parmi 5 numéros. Quel sera le résultat ?

```

1 <?php
2 switch (1) {
3     case 0:
4         $result = 'Hugues';
5         break;
6     case 1:
7         $result = 'Simone';
8         break;
9     case 2:
10        $result = 'Yves';
11        break;
12    case 3:
13        $result = 'Bryan';
14        break;
15    case 4:
16        $result = 'Kimberley';
17        break;
18 }
19 echo 'Le vainqueur du concours est '.$result.', et remporte le voyage à Quiberon';
20 ?>

```


- ☐ Le vainqueur du concours est Hugues, et remporte le voyage à Quiberon
- ☒ Le vainqueur du concours est Simone, et remporte le voyage à Quiberon
- ☐ Le vainqueur du concours est Yves, et remporte le voyage à Quiberon
- Q** Le vainqueur est Simone, car il s'agit du case 1 qui est demandé.

Question 5

Si le fichier qui est inclus avec `include_once` n'existe pas, PHP émettra une erreur fatale et le script s'arrêtera.

☐ Vrai

☒ Faux

 C'est faux, si le fichier qui est inclus avec **`include_once`** n'existe pas, PHP ne provoquera pas d'erreur fatale. Cela serait le cas avec **`require_once`**.