

# **Formation PHP X75-1**

# Table des matières

De quoi traite ce cours ?.....	6
Quels sont les objectifs du cours?.....	6
Introduction au cours.....	7
Définition et rôle du PHP.....	8
Sites statiques et sites dynamiques.....	8
Définition et rôle du MySQL.....	9
MariaDB et MySQL.....	9
Un peu d'histoire :.....	10
Les différences :.....	10
Pourquoi utiliser le PHP et MySQL ?.....	11
Client et serveur : définitions et interactions.....	11
Le fonctionnement d'Internet et du Web.....	12
Langages client-side et server-side.....	13
Mise en place de notre environnement de travail.....	14
Le travail en local et le travail en production.....	14
Recréer une architecture serveur sur son ordinateur.....	15
Où écrire le code PHP ?.....	15
Le modèle MVC ?.....	15
Testons notre serveur.....	16
La balise PHP.....	17
Syntaxe PHP de base.....	18
Les variables Php.....	20
Qu'est-ce qu'une variable ?.....	21
Les règles de déclaration des variables en PHP.....	21
Utilisation concrète des variables en PHP.....	22
Les types de données PHP.....	23
Le type chaîne de caractères ou String.....	24
Les types de données nombre entier (Integer) et nombre décimal (Float ou Double).....	24
Le type de données booléen (Boolean).....	24
Le type de données Null.....	24
Les types de données PHP tableau (Array) et objet (Object).....	25
Le type de données ressource (Resource).....	25
Les opérateurs de chaînes et la concaténation en PHP.....	25
Les opérateurs arithmétiques.....	26
By the way : Petit rappel utile sur l'ordre des opérations en mathématique.....	27
Exercices variables.....	28
Les opérateurs d'affectation et de comparaison (introduction).....	29
Structures de contrôle.....	30
Les conditions et des opérateurs de comparaison.....	31
Présentation des opérateurs de comparaison.....	31
Présentation des conditions en PHP.....	33
Les conditions if, if...else et if...elseif...else PHP.....	33
La condition if en PHP.....	33
La condition if...else en PHP.....	34

La condition if...elseif...else en PHP.....	34
Imbriquer plusieurs conditions.....	36
Les opérateurs logiques.....	36
Exercices.....	37
Projet jeu du pendu V0.1.....	40
L'instruction switch en PHP.....	41
L'instruction match.....	42
« Read the fucking manual » ou l'art de lire la doc :-). .....	43
La variable tableau.....	44
C'est quoi un tableau ?.....	45
Présentation des tableaux en PHP.....	45
Création d'un tableau numéroté ou indexé en PHP.....	46
Les tableaux associatifs PHP.....	47
Les tableaux multidimensionnels.....	48
Tableau associatif ou indexé, comment choisir.....	49
Les boucles et les opérateurs d'incrément et de décrémentation.....	52
C'est quoi une boucle.....	53
Les opérateurs d'incrément et de décrémentation.....	53
La boucle PHP while (rarement utilisée).....	54
La boucle PHP do...while (rarement utilisée).....	55
La boucle for.....	56
La boucle PHP foreach.....	56
Exercices.....	59
Projet jeu du pendu V0.2.....	62
Inclure des fichiers dans d'autres en PHP avec include et require.....	63
Présentation de include et de require et cas d'utilisation.....	63
Les différences entre include, include_once, require et require_once.....	63
Exercice Reprenez le projet Agence de voyage et séparez les divers éléments.....	64
Les fonctions en PHP.....	65
Introduction aux fonctions PHP.....	66
Définition des fonctions et fonctions internes ou prêtes à l'emploi.....	66
Les fonctions définies par l'utilisateur en PHP.....	66
Les paramètres et arguments des fonctions PHP.....	67
Définir des valeurs par défaut pour les paramètres de nos fonctions.....	68
Portées des variables avec les fonctions.....	69
Le retour d'une fonction.....	70
Le PHP, un langage au typage faible.....	70
Le typage des arguments.....	70
Typage strict.....	71
Typage du retour d'une fonction.....	72
Projet jeu du pendu V0.3.....	73
Le Timestamp UNIX et la date en PHP.....	74
Le Timestamp UNIX.....	75
Obtenir le Timestamp relatif à la date actuelle en PHP.....	75
Obtenir le Timestamp d'une date donnée en PHP.....	75
La gestion des fuseaux horaires.....	76
Obtenir un Timestamp à partir d'une chaîne de caractères.....	76

Obtenir une date à partir d'un Timestamp en PHP.....	77
Obtenir et formater une date en PHP.....	77
Formater une date en PHP : la gestion du décalage horaire.....	78
La comparaison de dates en PHP.....	79
Tester la validité d'une date.....	80
Exercices.....	80
Transformer une date en français.....	81
Découverte du Php Object avec DateTime().....	82
C'est quoi un objet, la base.....	82
Découverte de DateTime().....	83
Exercices.....	86
Cookies et Sessions.....	87
Obligations légales.....	87
Cookies.....	88
C'est quoi un cookies.....	88
Créer un cookie en PHP.....	88
Récupérer la valeur d'un cookie.....	90
Modifier la valeur d'un cookie ou supprimer un cookie.....	90
Sessions.....	91
C'est quoi une sessions.....	91
Démarrer une session en PHP.....	91
Définir et récupérer des variables de session.....	92
Terminer une session et détruire les variables de session.....	93
Projet jeu du pendu V0.1.....	95
Manipuler des fichiers en PHP.....	96
Lire un fichier entier.....	96
Ouvrir, lire et fermer un fichier en PHP.....	97
Ouvrir un fichier en PHP.....	97
Lire un fichier partie par partie.....	98
Lire un fichier jusqu'à un certain point avec fread().....	98
Lire un fichier ligne par ligne avec fgets().....	98
Lire un fichier caractère par caractère avec getc().....	98
Trouver la fin d'un fichier de taille inconnue.....	99
La place du curseur interne ou pointeur de fichier.....	99
Déplacer le curseur manuellement.....	100
Fermer un fichier.....	100
Pendu V0.4.....	101
Créer et écrire dans un fichier en PHP.....	101
Créer et écrire dans un fichier en PHP : les méthodes disponibles.....	101
Écrire dans un fichier avec file_put_contents().....	101
Créer un fichier PHP avec fopen().....	102
Écrire dans un fichier avec fwrite().....	102
Tester l'existence d'un fichier.....	103
Effacer un fichier.....	103
Introduction aux permissions des fichiers et au Chmod.....	103
Pour info : Vérifier et modifier les permissions d'un fichier.....	105
Sur le serveur :.....	105

Via PHP.....	105
Pendu V0.5.....	106
Pendu V1.0.....	106
Et maintenant ?.....	106

## **De quoi traite ce cours ?**

Dans ce cours, nous allons explorer et apprendre à utiliser le langage de programmation PHP ainsi que le système de gestion de bases de données MySQL, basé sur le langage SQL. PHP et MySQL forment un duo réputé pour leur puissance et leur simplicité d'utilisation. Ils sont principalement utilisés dans un contexte web, notamment pour manipuler les données envoyées par les utilisateurs et rendre un site dynamique (PHP), ainsi que pour stocker ces données (MySQL).

## **Quels sont les objectifs du cours?**

Ce cours a deux objectifs principaux : présenter les notions et fonctionnalités importantes de PHP et MySQL, afin que vous disposiez des outils et compétences nécessaires pour commencer à programmer dès la fin du cours, et vous rendre le plus autonome possible.

Pour atteindre ces objectifs, nous ferons un tour d'horizon complet des langages, où chaque grande notion sera présentée, étudiée et illustrée. Je vous encouragerai à être proactif. Plutôt que d'éviter les notions complexes, nous les aborderons en profondeur pour que vous compreniez véritablement leur fonctionnement.

De plus, de nombreux exemples et exercices vous seront proposés pour vous confronter aux difficultés et vous assurer de bien comprendre chaque concept. Cette méthode est, selon moi, la meilleure pour vous rendre rapidement autonome. Si vous prenez le temps de refaire les exemples et exercices, vous devriez être capable de réaliser la plupart de vos projets à la fin du cours.

## **Introduction au cours**

## Définition et rôle du PHP

Le terme PHP signifie « PHP : Hypertext Preprocessor ». Le premier « P » de PHP est lui-même l'abréviation de « PHP », une particularité qui n'a pas une grande importance pour nous.

Créé en 1994, PHP est un langage de programmation dont la version stable la plus récente est la 8.3, sortie le 24 octobre 2024. C'est sur cette version que se base ce cours. Si une nouvelle version venait à sortir, cela ne rendrait pas ce cours obsolète, car les changements entre deux versions mineures sont limités et la rétrocompatibilité est assurée pendant plusieurs années.

PHP permet de créer des pages web générées dynamiquement. En d'autres termes, grâce à PHP, nous pouvons afficher des contenus différents sur une même page en fonction de certaines variables, comme l'heure de la journée ou si l'utilisateur est connecté ou non.

Prenons l'exemple d'un espace client sur un site e-commerce. Un utilisateur qui a déjà commandé sur le site et créé un compte dispose d'un formulaire de connexion à son espace client. En entrant ses informations (généralement un pseudonyme et un mot de passe), celles-ci sont traitées et analysées par PHP. Si les informations sont correctes, PHP récupère des données spécifiques à cet utilisateur et génère dynamiquement les pages de son espace client, incluant l'historique des commandes, le profil, les informations de facturation et l'adresse de livraison.

Lorsque l'utilisateur fournit des informations comme une adresse ou passe une commande, ces données sont généralement enregistrées dans une base de données. PHP permet également de récupérer ces données pour les réutiliser.

PHP s'exécute côté serveur, ce qui le classe parmi les langages « server side », contrairement aux langages « client side » qui s'exécutent côté client. Nous expliquerons ces notions en détail dans la prochaine leçon.

## Sites statiques et sites dynamiques

Les langages de programmation web peuvent être classés en deux grandes catégories :

les langages statiques et les langages dynamiques,

Les sites dits statiques sont caractérisés par leur absence d'interaction et d'adaptation aux visiteurs. Le code des différentes pages reste inchangé, quel que soit l'utilisateur ou toute autre variable. Par exemple, un site de type « CV » ou un site informatif sont généralement statiques, car ils ne nécessitent aucune interaction dynamique avec le visiteur. Un site créé uniquement en HTML et CSS sera toujours statique.

En revanche, les sites dynamiques peuvent fournir des pages différentes pour chaque visiteur ou selon diverses contraintes, permettant ainsi des interactions avec l'utilisateur, comme l'envoi de données. De



nombreux langages permettent de créer des sites dynamiques, chacun ayant ses points forts, ses faiblesses et son champ d'application.

Dans ce cours, nous nous concentrons sur le duo le plus connu : PHP, utilisé pour le calcul et le traitement des données, et MySQL, utilisé pour la gestion des bases de données.

## Définition et rôle du MySQL

MySQL est un système de gestion de bases de données relationnelles. Une base de données est un ensemble structuré de données, telles que des informations clients (nom, adresse, mot de passe), des commentaires de blog, ou le texte d'articles.

Les données dans une base de données ne sont pas directement lisibles par un humain. Pour les manipuler, nous utilisons un langage de bases de données, le plus célèbre étant le SQL (Structured Query Language).

MySQL utilise le langage SQL pour manipuler les données des bases de données. Ses avantages incluent sa simplicité d'utilisation, sa fiabilité, et ses performances. MySQL permet également de gérer différents types de bases de données et peut être utilisé conjointement avec PHP.

Prenons un exemple concret : imaginons que nous voulons créer un site où les utilisateurs peuvent s'inscrire et s'identifier. Nous créons des formulaires d'inscription en HTML et récupérons les données en PHP. Ces données doivent être enregistrées dans une base de données, qui n'est rien d'autre qu'un fichier structuré.

Pour créer cette base de données, nous pouvons utiliser une application spécialisée comme phpMyAdmin, dbeaver ou envoyer des requêtes SQL depuis un fichier de code. Nous utilisons une extension PHP (comme PDO ou MySQLi) pour coder en MySQL. Dans notre code, nous écrivons des requêtes SQL pour créer la base de données et y enregistrer les données.

Une base de données créée avec MySQL est une base MySQL(ou mariaDb...), ce qui signifie que ce système de gestion s'occupe de créer le fichier, d'ordonner les données et de les sécuriser. MySQL est un système de gestion de bases de données relationnelles, ce qui signifie que les informations sont stockées dans plusieurs compartiments appelés « tables » qui peuvent communiquer entre elles.

L'idée principale à retenir est que nous ne pouvons pas créer ni manipuler de bases de données sans un système de gestion de bases de données.

## MariaDB et MySQL

Un abus de langage existe souvent. On parle souvent de base de donnée mysql pour des bases de données MariaDB.

## Un peu d'histoire :

L'histoire de MariaDB est étroitement liée à l'acquisition de MySQL par Oracle.

1. **Création de MySQL:** MySQL a été créé en 1995 par Michael "Monty" Widenius, David Axmark et Allan Larsson. Ce système de gestion de bases de données relationnelles est rapidement devenu populaire grâce à sa simplicité et son efficacité.
2. **Acquisition par Sun Microsystems:** En 2008, Sun Microsystems a acquis MySQL pour environ 1 milliard de dollars. Cette acquisition visait à renforcer la position de Sun dans le domaine des logiciels open-source.
3. **Rachat par Oracle:** En 2010, Oracle a racheté Sun Microsystems, incluant MySQL dans la transaction. Cette acquisition a suscité des inquiétudes parmi les utilisateurs et développeurs de MySQL, craignant qu'Oracle, un concurrent direct, ne compromette l'avenir open-source de MySQL.
4. **Création de MariaDB:** En réponse à ces préoccupations, Michael "Monty" Widenius a décidé de créer un fork de MySQL en 2009, qu'il a nommé MariaDB, d'après le prénom de sa fille. MariaDB a été conçu pour rester libre et open-source, tout en offrant une compatibilité maximale avec MySQL.
5. **Évolution de MariaDB:** Depuis sa création, MariaDB a gagné en popularité et est devenu une alternative fiable à MySQL. De nombreuses grandes entreprises et projets, comme Wikipedia et Google, ont migré de MySQL vers MariaDB.

En résumé, MariaDB est né des préoccupations concernant l'avenir de MySQL après son acquisition par Oracle, et il continue de prospérer en tant qu'alternative open-source robuste et compatible.

## Les différences :

MariaDB et MySQL sont tous deux des systèmes de gestion de bases de données relationnelles, mais ils présentent quelques différences clés :

1. **Licence:** MariaDB est entièrement open-source, tandis que MySQL propose des versions open-source et commerciales.
2. **Performance:** MariaDB est souvent considéré comme plus rapide grâce à un plus grand nombre de moteurs de stockage et une meilleure gestion des connexions.
3. **Fonctionnalités:** MySQL supporte des fonctionnalités comme le masquage des données et les colonnes dynamiques, ce qui n'est pas le cas de MariaDB.
4. **Compatibilité:** MariaDB est conçu pour être compatible avec MySQL, facilitant la migration entre les deux.

En résumé, MariaDB offre plus de flexibilité et d'innovation sans verrouillage par le fournisseur, tandis que MySQL est largement utilisé et supporte des fonctionnalités avancées pour la gestion des données.

## **Pourquoi utiliser le PHP et MySQL ?**

Contrairement au HTML et au CSS, qui sont des standards établis, PHP et MariaDB ont de nombreux concurrents : Python, Ruby, JavaScript,... pour PHP, et PostgreSQL, Microsoft SQL Server, SQLite,... pour MySQL, pour ne citer qu'eux.

Pourquoi alors préférer le couple PHP/MySQL aux autres langages ? Il n'y a pas de raison absolue, car les alternatives mentionnées sont également performantes et présentent certains avantages et inconvénients par rapport à PHP et MySQL.

Cependant, le couple PHP/MySQL reste de loin le plus célèbre et le choix de référence pour créer des sites dynamiques et stocker des données, et ce pour de bonnes raisons.

Le premier avantage de PHP réside dans sa structure : c'est un langage très accessible pour les débutants, qui peuvent rapidement en comprendre la syntaxe de base et réaliser leurs premiers scripts, tout en supportant des structures très complexes pour les utilisateurs avancés.

De plus, PHP est un langage open source et donc gratuit, ce qui n'est pas toujours le cas dans le monde du web. Il est universellement reconnu et supporté, fonctionnant sur la majorité des architectures techniques.

PHP se distingue également par ses performances et sa solidité. Étant open source, il bénéficie des contributions de nombreux développeurs, ce qui le rend constamment amélioré et jamais abandonné. Il offre de bonnes performances d'exécution en termes de rapidité et est un langage sûr : les rares failles détectées sont corrigées rapidement.

Les systèmes de gestion de bases de données sont nombreux et la plupart utilisent le SQL standard. J'ai choisi d'utiliser MySQL dans ce cours car c'est le choix le plus populaire parmi les développeurs, et ce pour de bonnes raisons.

MySQL est totalement compatible avec PHP et utilise une syntaxe SQL standard, facilitant les opérations si vous devez changer de système de gestion de bases de données. De plus, MySQL est simple d'utilisation, très robuste et offre d'excellentes performances, que ce soit pour une petite ou une grande structure.

## **Client et serveur : définitions et interactions**

Nous allons définir ce qu'est un « client » et ce qu'est un « serveur » et examiner les grandes différences entre les langages dits « client side » et « server side ».

## Le fonctionnement d'Internet et du Web

Internet est un système conçu pour transporter de l'information. C'est un réseau de réseaux utilisant divers protocoles (règles définissant comment formater, transmettre et recevoir des données) pour envoyer des informations.

Le World Wide Web, ou simplement « Web », est l'un des réseaux d'Internet. Le Web n'est donc qu'une partie d'Internet. Plus précisément, le Web est un réseau de machines interconnectées qui stockent des sites. Lorsqu'une machine est connectée au Web et fournit un accès à un site web, on l'appelle un serveur car elle « sert » le site web.

Un serveur est une sorte de super ordinateur, fonctionnant 24h/24 et 7j/7 (en théorie). Un serveur dispose de logiciels spécifiques et son rôle est de stocker toutes sortes de médias composant les sites (fichiers, images, etc.), et de les rendre accessibles à tout utilisateur à tout moment, où qu'il soit.

Pour pouvoir se comprendre et échanger des données, toutes ces machines doivent parler la même langue, c'est-à-dire utiliser le même protocole. Le Web repose ainsi sur le protocole HTTP (HyperText Transfer Protocol) et sur son frère sécurisé, le HTTPS (Secure HTTP). Pour utiliser ce protocole, nous devons passer par un navigateur web (Chrome, Safari, etc.) que l'on appelle alors un « client HTTP ».

Pour accéder directement à une page web, on passe par un navigateur en utilisant le protocole HTTP. On fournit une adresse au format spécial à notre navigateur : une URL (Uniform Resource Locator) qui sert à identifier une page web de manière unique. Ici, notre navigateur (et nous) sommes des « clients » car nous demandons à accéder à la page web.

Le navigateur va alors chercher où se trouve le serveur hébergeant la page demandée. Pour cela, il utilise un service de DNS (Domain Name Server), des serveurs permettant d'associer un nom de domaine (mon-site-web.com par exemple) à une adresse IP unique. Chaque fournisseur de services Internet fournit une liste d'adresses de DNS à contacter. Si le premier DNS ne reconnaît pas le site, il envoie la demande à d'autres DNS, et ainsi de suite jusqu'à ce qu'un DNS possède le site dans sa liste de noms.

L'adresse IP liée au site est alors renvoyée au serveur. L'IP (Internet Protocol) est une suite de nombres permettant d'identifier de manière unique une machine connectée à Internet. Chaque machine possède sa propre IP, qui change en fonction du réseau sur lequel elle est connectée (l'IP est attribuée par le fournisseur de services Internet).

Le navigateur possède maintenant l'adresse IP de notre site et donc l'adresse de la machine (le fameux « serveur ») sur laquelle il est stocké. Il peut ainsi contacter directement le serveur en utilisant le protocole HTTP pour lui demander de renvoyer la page en question et envoie également notre IP pour que le serveur sache à quelle adresse renvoyer la page demandée.

Lorsque le serveur reçoit la requête, il recherche immédiatement le fichier demandé, effectue éventuellement certaines opérations dessus et le renvoie au navigateur, ou renvoie un code d'erreur si le fichier demandé est introuvable ou ne peut pas être envoyé.

## **Langages client-side et server-side**

Un site Internet est un ensemble de fichiers de code liés entre eux, faisant éventuellement appel à des ressources ou médias comme des images. Le code écrit dans ces fichiers, appelé « script », peut être exécuté soit côté client (client-side), c'est-à-dire directement dans le navigateur de l'utilisateur, soit côté serveur (server-side).

Il est important de comprendre que les navigateurs (côté client) et les serveurs (côté serveur) ne peuvent effectuer que certaines opérations et lire certains langages. La majorité des navigateurs ne comprennent et n'exécutent que du code HTML, CSS et JavaScript. Un navigateur est donc incapable de comprendre du code PHP.

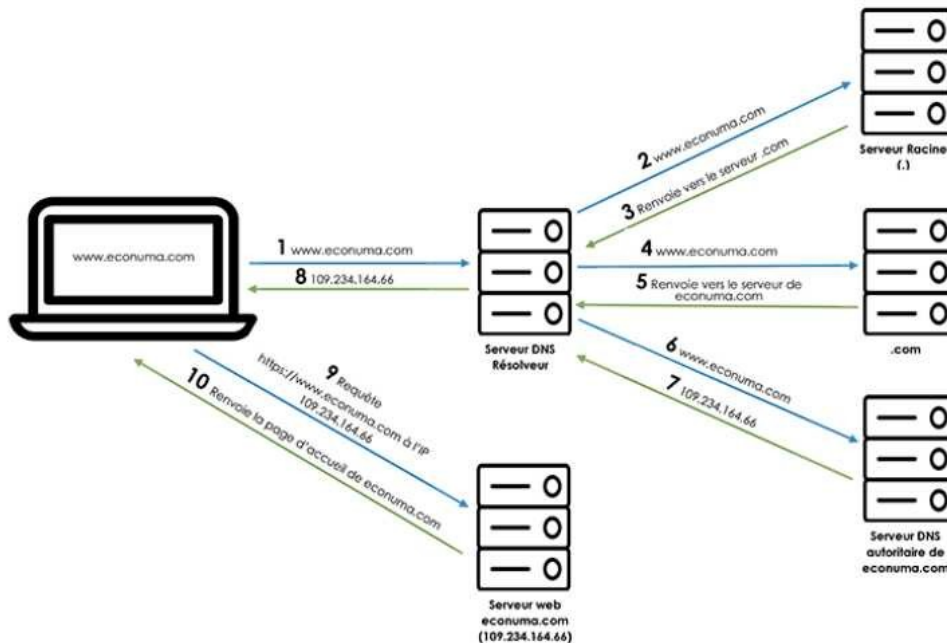
Lorsqu'un navigateur demande à un serveur de lui fournir une page, le serveur exécute tout code incompréhensible pour le navigateur (en utilisant au besoin un interpréteur). Une fois ces opérations effectuées, le serveur renvoie le résultat sous forme de code compréhensible par le navigateur, principalement en HTML. Le navigateur affiche alors la page au visiteur, unique car générée par le serveur pour un utilisateur spécifique en tenant compte de données particulières.

Si la page demandée ne contient aucun code nécessitant l'intervention du serveur, le serveur la renvoie telle quelle au navigateur, qui l'affiche au visiteur.

Les opérations réalisées côté serveur sont transparentes pour le visiteur, qui n'a jamais accès au code exécuté côté serveur. Une fois les opérations terminées, le serveur ne renvoie que du code compréhensible par le navigateur. C'est pourquoi, en analysant le code d'une page, vous ne trouverez jamais d'instructions PHP, mais seulement du code HTML, CSS et JavaScript, des langages exécutés côté client.

## Processus de Résolution DNS et requête de page web

Cas de econuma.com



## Mise en place de notre environnement de travail

Pour coder en HTML et en CSS et afficher le résultat, un simple éditeur de texte et un navigateur suffisent. Pour le PHP et MySQL, cependant, c'est un peu plus complexe car le code s'exécute côté serveur.

Dans cette leçon, nous allons explorer les différentes options pour coder en PHP et MySQL, et voir quels logiciels installer et pourquoi.

## Le travail en local et le travail en production

Lorsque l'on code, on peut travailler soit en local, c'est-à-dire en hébergeant tous les fichiers sur nos propres machines ou hors ligne, soit en production (ou en préproduction), c'est-à-dire sur des fichiers hébergés sur un serveur distant et potentiellement accessibles à tous, en direct.

Lors de la phase de développement d'un site, ou pendant les phases de test ou de débogage, un bon développeur préfère toujours travailler en local ou en préproduction pour ne pas perturber le fonctionnement normal d'un site web.

La préproduction est une copie d'un site également hébergée sur serveur, généralement accessible uniquement aux développeurs pour tester leurs modifications en toute sécurité et en conditions réelles.

Dans cette formation nous travaillerons en local. Cependant, rappelez-vous que PHP et MySQL s'exécutent côté serveur. Il nous faut donc recréer une architecture serveur sur nos machines avec les logiciels adaptés pour tester nos codes PHP. Rassurez-vous, c'est très simple et totalement gratuit.

## **Recréer une architecture serveur sur son ordinateur**

Un serveur dispose de différents programmes lui permettant de lire et de comprendre certains langages informatiques que des ordinateurs « normaux » ne peuvent pas lire. Nous allons donc devoir installer des programmes similaires afin de pouvoir tester nos codes PHP et MySQL.

Vous pouvez installer MAMP, disponible à l'adresse <http://www.mamp.info/>

Mamp est un serveur web de test complet (LAMP = Linux, Apache, Mysql, Php)

Si vous êtes sous Mac Os ou Linux. Il est possible d'installer Apache2, Mysql et Php directement.

Si vous utilisez un autre logiciel LAMP (Wamp, Xamp,..) C'est très bien, inutile d'installer Mamp.

## **Où écrire le code PHP ?**

Nous allons pouvoir écrire nos scripts PHP soit dans des fichiers dédiés, c'est-à-dire des fichiers qui ne vont contenir que du PHP, soit intégrer le PHP au sein de nos fichiers HTML mais qui devront être enregistré avec l'extension .php et pas html.

Soyons clair, mélanger du HTML et du CSS est un non sens aujourd'hui et ne se pratique plus depuis 15ans. Aujourd'hui il n'est plus envisageable de développer un projet en « php procédurale », on ne fait plus que du « php objet » en respectant le modèle MVC.

Malgré cela, nous ferons du procédurale le temps de l'initiation de base.

## ***Le modèle MVC ?***

Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture utilisée pour organiser et structurer le code dans les applications web. Il permet de séparer les préoccupations en trois composants distincts, rendant les applications plus modulaires, plus faciles à maintenir et à étendre. Voici une explication de chaque composant dans le contexte de PHP :

### **1. Modèle (Model):**

- 1.1. Rôle: Le modèle représente la couche des données et la logique métier de l'application. Il gère l'accès aux données et les interactions avec la base de données.
- 1.2. Exemple\*\* : En PHP, le modèle pourrait être un fichier de code qui interagit avec une base de données MySQL pour récupérer, ajouter ou supprimer des données.

### **2. Vue (View):**

- 2.1. Rôle: La vue est responsable de l’affichage des données aux utilisateurs. Elle récupère les données du modèle et les présente dans un format compréhensible, souvent en HTML.
- 2.2. Exemple: Une vue en PHP pourrait être un fichier HTML avec des balises PHP intégrées pour afficher dynamiquement les données récupérées par le modèle.

### 3. Contrôleur (Controller):

- 3.1. Rôle: Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il reçoit les requêtes du client, interagit avec le modèle pour récupérer les données nécessaires, et transmet ces données à la vue pour l’affichage.
- 3.2. Exemple\*\* : Un contrôleur en PHP pourrait être une classe qui traite les requêtes HTTP, appelle les méthodes du modèle pour manipuler les données, et sélectionne la vue appropriée pour afficher les résultats.

Pourquoi utiliser l'architecture MVC en PHP ?

- Séparation des préoccupations: En séparant la logique métier, l’affichage et le contrôle en composants distincts, le MVC améliore la lisibilité et la maintenabilité du code.
- Réutilisation du code: Le MVC permet de réutiliser les modèles et les vues dans différentes parties de l’application.
- Facilité de test: Le MVC permet de tester chaque composant individuellement, simplifiant le débogage et l’amélioration continue de l’application.
- Collaboration efficace: Les développeurs peuvent travailler sur différents composants sans interférer avec le travail des autres, améliorant ainsi la productivité.

## Testons notre serveur


Attention que les fichiers php devront être dans un répertoire dont le parent est le répertoire de Mamp. Il est possible de le connaître en allant dans la configuration de MAMP.

Créons notre premier fichier php. A la racine du dossier de Mamp créons le fichier phpInfo.php

```
<?php
phpinfo();|
```



2 lignes pour un résultat bien long...

PHP Version 8.3.7	
	
System	Linux Osiris 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64
Build Date	May 13 2024 15:27:40
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-mysqld.ini, /etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/php/8.3/apache2/conf.d/10-gd.ini, /etc/php/8.3/apache2/conf.d/15-xml.ini, /etc/php/8.3/apache2/conf.d/20-bcmath.ini, /etc/php/8.3/apache2/conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf.d/20-curl.ini, /etc/php/8.3/apache2/conf.d/20-dom.ini, /etc/php/8.3/apache2/conf.d/20-exif.ini, /etc/php/8.3/apache2/conf.d/20-ffi.ini, /etc/php/8.3/apache2/conf.d/20-fileinfo.ini, /etc/php/8.3/apache2/conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20-gd.ini, /etc/php/8.3/apache2/conf.d/20-gettext.ini, /etc/php/8.3/apache2/conf.d/20-iconv.ini, /etc/php/8.3/apache2/conf.d/20-imageick.ini, /etc/php/8.3/apache2/conf.d/20-intl.ini, /etc/php/8.3/apache2/conf.d/20-mbstring.ini, /etc/php/8.3/apache2/conf.d/20-mysqli.ini, /etc/php/8.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/apache2/conf.d/20-posix.ini, /etc/php/8.3/apache2/conf.d/20-readline.ini, /etc/php/8.3/apache2/conf.d/20-shmop.ini, /etc/php/8.3/apache2/conf.d/20-simplexml.ini, /etc/php/8.3/apache2/conf.d/20-sockets.ini, /etc/php/8.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.3/apache2/conf.d/20-sysvsem.ini, /etc/php/8.3/apache2/conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.d/20-tokenizer.ini, /etc/php/8.3/apache2/conf.d/20-xmlreader.ini, /etc/php/8.3/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.3/apache2/conf.d/20-xsl.ini, /etc/php/8.3/apache2/conf.d/20-zip.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831.NTS
PHP Extension Build	API20230831.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled

la ligne `phpinfo()` ; est une méthode qui vous renvoi la configuration de php sur votre machine. Verifiez la version (8.3 ou suivante). Pour l’instant c’est suffisant. La majorité des données viennent du fichier de configuration de php > `php.ini`. Il vous faudra certainement modifier des choses dedans sur des projets plus complexes.

## La balise PHP

Le serveur, pour être en mesure d’exécuter le code PHP, va devoir le reconnaître. Pour lui indiquer qu’un script ou que telle partie d’un code est écrit en PHP, nous allons entourer ce code avec une balise PHP qui a la forme suivante : `<?php ?>`. Lorsqu’on intègre du PHP dans du code HTML, on va pouvoir placer cette balise et du code PHP à n’importe quel endroit dans notre fichier. On va même pouvoir placer la balise PHP en dehors de notre élément html. De plus, on va pouvoir déclarer plusieurs balises PHP à différents endroits dans un fichier.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Premiere page php</title>
</head>
<body>
<?php

?>
<h1>Ceci est une page dans un fichier .php</h1>
<?php

?>
<p class="relatif gauche">Bonjour à tous,</p>
<p>On ne lâche rien<p>
<hr>
<p>Bonne continuation...</p>
</body>
</html>
```

## Ceci est une page dans un fichier .php

Bonjour à tous,

On ne lâche rien

Bonne continuation...

Évidemment, si vous enlever les 2 balises php, vous aurez le meme résultat. C’est juste du HTML.

## Syntaxe PHP de base

Écrivons une première instruction PHP afin d'analyser et de comprendre la syntaxe générale du PHP.

```
<h1>Ceci est une page dans un fichier .php</h1>

<p>
<!-- ouvrons la balise PHP-->
<?php
// Bonjour à tous, **** On garde cette partie de code au cas ou...
echo "Hello World,";
/*
  Changement du texte pour voir une difference
  Avec l'affichage d'avant
  */
?>
</p>
<p>On ne lâche rien<p>
<hr>
<p>Bonne continuation...</p>
```

### Ceci est une page dans un fichier .php

Hello World,

On ne lâche rien

---

Bonne continuation...

Analysons cela :

A l'affichage, rien n'a fort changé. Un des texte à été modifié mais rien de plus.

- L'instruction ECHO demande à php de renvoyer le contenu à l'explorateur.
- Les instructions PHP sont TOUJOURS suivies par un point virgule ;-)

Notez les 3 commentaires.

- Le premier est en dehors des balises php. C'est un commentaire HTML.
- Le deuxième est un commentaire mono-ligne php
- Le troisième un commentaire multi-lignes php

Demandons à notre explorateur de voir le code source de la page :

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Premiere page php</title>
6 </head>
7 <body>
8
9 <h1>Ceci est une page dans un fichier .php</h1>
10
11 <p>
12 <!-- ouvrons la balise PHP-->
13   Hello World,</p>
14 <p>On ne lâche rien<p>
15 <hr>
16 <p>Bonne continuation...</p>
17 </body>
18 </html>
```

On voit bien le commentaire HTML mais pas les commentaires PHP ! Les commentaires php ne sont pas envoyés à l'explorateur.

(Chrome ne montre pas les commentaires HTML non plus... Mais ils sont bien là!)

## ECHO et PRINT

Les 2 sont fortement identiques. L'usage d'echo est préféré

```
<?php
echo '<p>Bonjour à tous</p>';
echo "<p>On ne lâche rien<p>";
echo 12;
echo " ou ";
echo "13";
echo " personnes sur ", 24, " sont présentes";
echo "<p>On ne lâche rien<p>";
?>
```

Bonjour à tous

On ne lâche rien

12 ou 13 personnes sur 24 sont présentes

On ne lâche rien

Première et deuxième ligne : le contenu peut être entouré d'une simple ou d'un double guillemet.

Troisième et cinquième ligne : un nombre peut être entouré ou pas de guillemets.

Sixième ligne : les contenus peuvent être séparés par des virgules.

## **Les variables Php**

## Qu'est-ce qu'une variable ?

Une variable est un conteneur utilisé pour stocker temporairement des informations, telles qu'une chaîne de caractères (texte) ou un nombre. La particularité d'une variable est qu'elle peut changer, c'est-à-dire qu'elle peut contenir différentes valeurs au fil du temps.

En PHP, les variables ne stockent les informations que temporairement. Plus précisément, une variable n'existe que pendant l'exécution du script qui l'utilise. Ainsi, les variables ne permettent pas de stocker des informations de manière durable. Pour cela, nous utiliserons des fichiers, des cookies ou des bases de données, que nous aborderons plus tard dans ce cours.

Note : Au début de ce cours, nous définirons nous-mêmes les valeurs à stocker dans nos variables, ce qui peut sembler peu utile en pratique. Il est donc normal que vous ne voyiez pas immédiatement l'intérêt d'utiliser des variables. Retenez que les variables deviennent vraiment intéressantes lorsqu'elles servent à stocker des données envoyées par les utilisateurs (via des formulaires, par exemple), pour effectuer les comparaisons, des calculs car nous pourrons ensuite manipuler ces données.

## Les règles de déclaration des variables en PHP

Une variable est donc un conteneur ou un espace de stockage temporaire auquel nous pouvons assigner une valeur. Nous pouvons créer différentes variables dans un script pour stocker diverses valeurs et les réutiliser facilement par la suite. Lorsqu'on crée une variable en PHP, on dit qu'on la « déclare ».

Nous pouvons choisir le nom de nos variables, mais il y a quelques règles à respecter lors de leur déclaration :

- Toute variable en PHP doit commencer par le signe \$ suivi du nom de la variable.
- Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (\_) et ne doit pas commencer par un chiffre.
- Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores, mais pas de caractères spéciaux.
- Le nom d'une variable ne doit pas contenir d'espace.

### Bonnes pratiques :

Une bonne pratique à prendre pour nommer les variables est d'indiquer clairement ce qu'elle représente en nom de variable et d'écrire cela en « camelCase ». Cela consiste à écrire un ensemble de mots sans espaces ni ponctuation, où chaque nouveau mot débute par une majuscule, à l'exception du premier mot qui commence par une minuscule.

Exemple : une variable qui représente l'âge du capitaine : \$ageDuCapitaine. Bien que, en pratique, l'usage de l'anglais est préféré \$ageOfTheCaptain".

es sont sensibles à la casse en PHP. Cela signifie que l'utilisation de majuscules ou de minuscules crée des variables différentes. Par exemple, les variables \$texte, \$TEXTE et \$tEXTe sont distinctes.

Enfin, sachez qu'il existe des noms « réservés » en PHP. Vous ne pouvez pas utiliser ces noms pour vos variables, car le langage PHP les utilise déjà pour désigner différents objets intégrés.

Déclaration de variables en php (à noter que cet exemple n'affiche rien!)

```
<?php
    $prenom = "Pierre";
    $age = 28;
?>
```

Affichons une variable :

```
<?php

$onLacheRien= 'On ne lâche rien';

echo '<p>Bonjour à tous</p>';
echo "<p>";

echo $onLacheRien;

echo "<p>";

?>
```

Bonjour à tous

On ne lâche rien

## Utilisation concrète des variables en PHP

Les variables en PHP sont extrêmement utiles dans de nombreuses situations. Elles permettent de manipuler des données dont les valeurs ne sont pas connues à l'avance. Par exemple, si vous souhaitez manipuler (afficher, stocker, etc.) des données récoltées via un formulaire rempli par vos visiteurs, vous ne connaissez pas les valeurs qui seront envoyées à l'avance.

Pour manipuler ces données, nous utilisons des variables. Nous créons un script qui traite les données envoyées par les utilisateurs. Les valeurs envoyées sont stockées dans des variables, que nous pouvons ensuite manipuler. Ainsi, chaque fois qu'un visiteur envoie le formulaire, notre script se déclenche, les données sont placées dans des variables prédéfinies, et le script manipule ces variables pour les afficher, les stocker, etc.

Les variables sont également essentielles pour dynamiser notre site grâce à leur capacité à stocker différentes valeurs. Par exemple, si nous souhaitons afficher une horloge sur notre site, nous créons un

script qui recalcule l'heure actuelle toutes les secondes. Cette heure est placée dans une variable `$heure`. Le script met à jour cette variable chaque seconde (la variable stocke alors l'heure actuelle) et affiche son contenu.

Nous pouvons utiliser les variables dans de nombreuses autres situations pour créer des scripts plus complexes. Cependant, il serait difficile d'illustrer l'intérêt des variables avec des exemples trop avancés à ce stade.

Pour l'instant, faites confiance au processus et concentrez-vous sur la compréhension de chaque notion présentée. Ces concepts seront très utiles par la suite.

## Les types de données PHP

Les variables PHP vont pouvoir stocker différents types de valeurs, comme du texte ou un nombre par exemple. Par abus de langage, nous parlerons souvent de « types de variables » PHP.

En PHP, contrairement à d'autres langages de programmation, nous n'avons pas besoin de préciser à priori le type de valeur qu'une variable va pouvoir stocker. Le PHP va en effet automatiquement détecter quel est le type de la valeur stockée dans telle ou telle variable, et nous allons ensuite pouvoir performer différentes opérations selon le type de la variable, ce qui va s'avérer très pratique pour nous !

Une conséquence directe de cela est qu'on va pouvoir stocker différents types de valeurs dans une variable au fil du temps sans se préoccuper d'une quelconque compatibilité. Par exemple, une variable va pouvoir stocker une valeur textuelle à un moment dans un script puis un nombre à un autre moment.

Les variables en PHP vont pouvoir stocker 8 grands types de données différents :

- Le type « chaîne de caractères » ou **String** en anglais ;
- Le type « nombre entier » ou **Integer** en anglais ;
- Le type « nombre décimal » ou **Float** en anglais ;
- Le type « booléen » ou **Boolean(bool)** en anglais ;
- Le type « tableau » ou **Array** en anglais ;
- Le type « objet » ou **Object** en anglais ;
- Le type « NULL » qui se dit également **NULL** en anglais ;
- Le type « ressource » ou **Resource** en anglais ;

Nous allons pour le moment nous concentrer sur les types simples de valeurs.

## Le type chaîne de caractères ou String

Le premier type de données qu'une variable va pouvoir stocker est le type String ou chaîne de caractères. Une chaîne de caractères est une séquence de caractères, ou ce qu'on appelle communément un texte.

Notez que toute valeur stockée dans une variable en utilisant des guillemets ou des apostrophes sera considérée comme une chaîne de caractères, et ceci même dans le cas où nos caractères sont à priori des chiffres comme « 28 » par exemple. Il serait impossible de multiplier 28 par deux si 28 est de type string.

```
$string = "texte";  
$string = "10";
```

## Les types de données nombre entier (Integer) et nombre décimal (Float ou Double)

En PHP, on va pouvoir stocker deux types différents de données numériques dans nos variables : le type Integer, qui contient tous les nombres entiers positifs ou négatifs et le type Float ou Double, qui contient les nombres décimaux (nombres à virgule) positifs ou négatifs.

```
$nombre = 10;
```

## Le type de données booléen (Boolean)

Une variable en PHP peut encore stocker une valeur de type booléen (Boolean en anglais). Le type booléen est un type qui ne contient que deux valeurs : les valeurs true (vrai) et false (faux). Ce type n'est pas courant dans la vie de tous les jours mais est très (très) utilisé en informatique.

```
/* SANS GUILEMETS*/  
$bool = true;  
$bool = false;
```

## Le type de données Null

Le type de données Null est un type un peu particulier puisqu'il correspond à l'absence de valeur et sert donc à représenter des variables vides en PHP. Ce type de valeur ne contient qu'une seule valeur : la valeur NULL qui correspond elle-même à l'absence de valeur.

Par exemple cela permet de faire une distinction entre une chaîne de caractère vide et l'absence de valeur.

```
/* SANS GUILEMETS*/  
$null = null;
```



## Les types de données PHP tableau (Array) et objet (Object)

Les types de données Array et Object sont des types de données complexes particuliers qui méritent de faire chacun l'objet de chapitres séparés.

Pour l'instant sachez simplement que l'on va pouvoir stocker plusieurs valeurs d'un coup à l'intérieur d'une variable en lui assignant des valeurs de type Array (tableau) ou Object (objet).

## Le type de données ressource (Resource)

Une ressource est une variable particulière qui contient une référence vers une ressource externe au PHP, comme dans le cas d'une variable qui représente la connexion vers une base de données par exemple.

## Les opérateurs de chaînes et la concaténation en PHP

Concaténer signifie littéralement « mettre bout à bout ». Pour indiquer qu'on souhaite concaténer, c'est-à-dire qu'on souhaite mettre bout à bout deux chaînes de caractères en PHP on utilise le point (.) qu'on appelle également « opérateur de concaténation ».

Pour bien comprendre comment fonctionne l'opérateur de concaténation et son intérêt, il me semble nécessaire de connaître les différences entre l'utilisation des guillemets et des apostrophes lorsqu'on manipule une chaîne de caractères en PHP.

La différence majeure entre l'utilisation des guillemets et des apostrophes est que tout ce qui est entre guillemets va être interprété tandis que quasiment tout ce qui est entre apostrophes va être considéré comme une chaîne de caractères. Ici, « interprété » signifie « être remplacé par sa valeur ». Ainsi, lorsqu'on inclut une variable au sein d'une chaîne de caractères et qu'on cherche à afficher le tout avec un echo par exemple en entourant la chaîne complète avec des guillemets, la variable va être remplacée par sa valeur lors de l'affichage.

En revanche, lorsqu'on utilise des apostrophes, les variables ne vont pas être interprétées mais leur nom va être considéré comme faisant partie de la chaîne de caractères.

Regardez plutôt l'exemple suivant :

```
$nom="Marie";  
$prenom="Dupond";  
echo"Je m'appelle $nom $prenom <br>";  
echo"Je m'appelle {$nom} {$prenom} <br>";  
echo'Je m\'appelle $nom $prenom<br>';  
echo'Je m\'appelle {$nom} {$prenom}';
```

Je m'appelle Marie Dupond  
Je m'appelle Marie Dupond  
Je m'appelle \$nom \$prenom  
Je m'appelle {\$nom} {\$prenom}

Pour concaténer correctement avec l'opérateur de concaténation, la règle est de séparer les différentes variables avec l'opérateur de concaténation (le point) des textes autour.

Chaque texte devra être entouré de guillemets ou d'apostrophes selon ce qu'on a choisi.

L'utilisation de la concaténation peut sembler inutile mais son usage est prioritaire.

```
$nom="Marie";  
$prenom="Dupond";  
echo"Je m'appelle ".$nom." ".$prenom."<br>";  
echo"Je m'appelle '$nom.' '$prenom.'<br>";
```

Je m'appelle Marie Dupond  
Je m'appelle Marie Dupond

## Les opérateurs arithmétiques

Les opérateurs arithmétiques vont nous permettre d'effectuer toutes sortes d'opérations mathématiques entre les valeurs contenues dans différentes variables lorsque ces valeurs sont des nombres.

Le fait de pouvoir réaliser des opérations entre variables va être très utile dans de nombreuses situations. Par exemple, si un utilisateur commande plusieurs produits sur notre site ou plusieurs fois un même produit et utilise un code de réduction, il faudra utiliser des opérations mathématiques pour calculer le prix total de la commande.

En PHP, nous allons pouvoir utiliser les opérateurs arithmétiques suivants :

Opérateur	Nom de l'opération associée
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

Pour info :

- Le modulo correspond au reste entier d'une division euclidienne. Par exemple, lorsqu'on divise 5 par 3, le résultat est 1 et il reste 2 dans le cas d'une division euclidienne. Le reste, 2, correspond justement au modulo.
- L'exponentielle correspond à l'élévation à la puissance d'un nombre par un autre nombre. La puissance d'un nombre est le résultat d'une multiplication répétée de ce nombre par lui-même. Par exemple, lorsqu'on souhaite calculer 2 à la puissance de 3 (qu'on appelle également « 2

exposant 3 »), on cherche en fait le résultat de 2 multiplié 3 fois par lui-même c'est-à-dire  $2*2*2 = 8$ .

### **By the way : Petit rappel utile sur l'ordre des operations en mathématique**

Voici l'ordre de priorité des opérations qu'il faut respecter :

1. Les Parenthèses
2. Les Exposants
3. Les Multiplications et les Divisions (de la gauche vers la droite)
4. Les Additions et les Soustractions (de la gauche vers la droite)

## Exercices variables

Utilisez la fonction `var_dump()` Pour afficher les détails de vos variables

```
$variable1 = 5;
$variable2 = "chaise";
$variable3 = true;
$variable4 = null;
echo '<br> variable1 vaut: ';
var_dump($variable1);
echo '<br> variable2 vaut: ';
var_dump($variable2);
echo '<br> variable3 vaut: ';
var_dump($variable3);
echo '<br> variable4 vaut: ';
var_dump($variable4);
```

variable1 vaut: int(5)  
variable2 vaut: string(6) "chaise"  
variable3 vaut: bool(true)  
variable4 vaut: NULL

variable1 vaut : un entier de valeur 5

variable2 vaut : une chaîne de caractère de 6 caractères et de valeur « chaise »

variable3 vaut : un boolean de valeur « vrai » (ou 1)

variable4 vaut : « rien » pas de type défini

**Exercice 1 :** Créer une variable `name` et l'initialiser avec la valeur de votre choix. Afficher son contenu.

**Exercice 2 :** Créer trois variables `lastname`, `firstname` et `age` et les initialiser avec les valeurs de votre choix. Attention `age` est de type entier. Afficher leur contenu.

**Exercice 3 :** Créer une variable `km`. L'initialiser à 1. Afficher son contenu. Changer sa valeur par 3. Afficher son contenu. Changer sa valeur par 125. Afficher son contenu.

**Exercice 4 :** Créer une variable de type `string`, une variable de type `int`, une variable de type `float`, une variable de type `booléen` et les initialiser avec une valeur de votre choix. Les afficher.

**Exercice 5 :** Créer trois variables `lastname`, `firstname` et `age` et les initialiser avec les valeurs de votre choix. Attention `age` est de type entier. Afficher : "Bonjour" + `lastname` + `firstname` + ",tu as" + `age` + "ans".

**Exercice 6 :** Créer 3 variables. La 1<sup>er</sup> vaut 3, la 2<sup>eme</sup> vaut 4, la 3<sup>eme</sup> vaut 5. Afficher le résultat que vaut `3+4`, `5/4` et `3+5*4`

**Exercice 7 :** Afficher le résultat de  $(8+2\times 2)\div(12\div 4+3)$

**Solutions :** `variables_resolu.php`

## Les opérateurs d'affection et de comparaison (introduction)

Une nouvelle fois, vous devez bien comprendre que le signe égal simple utilisé ci-dessus n'est pas un opérateur de comparaison mais bien un opérateur d'affection (ou d'assignation) : il sert à affecter une valeur à une variable. Cela signifie que l'opérateur = ne représente pas l'égalité d'un point de vue mathématique.

L'égalité en termes de valeurs simples est symbolisée en PHP par le double signe égal : ==. L'égalité en termes de valeurs et de types de données, c'est-à-dire l'identité, va être représentée en PHP par le triple signe égal : ===.

En effet, nous allons voir plus tard que les variables peuvent stocker différents types de données : des chaînes de caractères, des nombres entiers, des nombres décimaux, etc. En utilisant des guillemets ou des apostrophes, on indique que la valeur stockée par la variable est une chaîne de caractères.

Exemple

<code>\$a = « 3 »</code>	Assigne la valeur à « 3 » type string
<code>\$a == 3</code>	Vaut true
<code>\$a === 3</code>	Vaut false (la chaîne de caractère 3 ne vaut pas le chiffre 3)

## Structures de contrôle

# Les conditions et des opérateurs de comparaison

Une structure de contrôle est un ensemble d'instructions qui permet de contrôler l'exécution du code.

Il existe différents types de structures de contrôle. Les deux types les plus connus et les plus utilisés sont les structures de contrôle conditionnelles qui permettent d'exécuter un bloc de code si une certaine condition est vérifiée et les structures de contrôle de boucle qui permettent d'exécuter un bloc de code en boucle tant qu'une condition est vérifiée.

## Présentation des opérateurs de comparaison

Nous allons souvent construire nos conditions autour de variables : selon la valeur d'une variable, nous allons exécuter tel bloc de code ou pas. En pratique, nous allons donc comparer la valeur d'une variable à une certaine autre valeur donnée et selon le résultat de la comparaison exécuter un bloc de code ou pas. Pour comparer des valeurs, nous allons devoir utiliser des opérateurs de comparaison.

Opérateur Définition :

==	Permet de tester l'égalité sur les valeurs
===	Permet de tester l'égalité en termes de valeurs et de types
!=	Permet de tester la différence en valeurs
<>	Permet également de tester la différence en valeurs
!==	Permet de tester la différence en valeurs ou en types
<	Permet de tester si une valeur est strictement inférieure à une autre
>	Permet de tester si une valeur est strictement supérieure à une autre
<=	Permet de tester si une valeur est inférieure ou égale à une autre
>=	Permet de tester si une valeur est supérieure ou égale à une autre

exercices :

### Exercice1

`$a = 3;`

il faut que `$a ... 3` soit true. Trouvez 4 comparaisons renvoyant la valeur true

`var_dump($a ... 3);`

```
var_dump($a ... 3);  
var_dump($a ... 3);  
var_dump($a ... 3);
```

## Exercice 2

```
$a = "3";
```

il faut que \$a ... 3 soit true

```
var_dump($a ... 3);  
var_dump($a ... 3);
```

il faut que \$a .... 3 soit false

```
var_dump($a ... 3);  
var_dump($a ... 3);  
var_dump($a ... 3);  
var_dump($a ... 3);  
var_dump($a ... 3);
```

## Exercice 3

Répondez à la question avant de tester votre réponse avec php

```
$a = true ;  
var_dump($a == true); affiche ...  
var_dump($a == « true »); affiche ...  
var_dump($a < 3); affiche ...  
var_dump($a === true); affiche ...  
var_dump($a == 1); affiche ...  
var_dump($a === 1); affiche ...  
var_dump($a <> 3); affiche ...
```



`var_dump($a == "coucou");` affiche ...

`var_dump($a == "null");` affiche ...

`var_dump($a == 110);` affiche ...

`var_dump($a != 3);` affiche ...

`var_dump($a !== 3);` affiche ...

`var_dump($a != null);` affiche ...

`var_dump($a !== null);` affiche ...

**Solutions** : `opérateur_resolu.php`

## Présentation des conditions en PHP

Les structures de contrôle conditionnelles (ou plus simplement conditions) vont nous permettre d'exécuter différents blocs de code selon qu'une condition spécifique soit vérifiée ou pas.

Par exemple, on va pouvoir utiliser les conditions pour afficher un message de bienvenue différent en PHP sur notre site selon que l'utilisateur soit connu ou un simple visiteur qui ne s'est jamais inscrit sur notre site.

Nous allons très souvent utiliser les conditions avec des variables : selon la valeur stockée dans une variable, nous allons vouloir exécuter un bloc de code plutôt qu'un autre.

Les conditions vont ainsi être un passage incontournable pour rendre un site dynamique puisqu'elles vont nous permettre d'exécuter différents codes et ainsi afficher différents résultats selon le contexte.

## Les conditions if, if...else et if...elseif...else PHP

### La condition if en PHP

La condition `if` est l'une des plus importantes et des plus utilisées dans l'ensemble des langages de programmation utilisant les structures de contrôle et en particulier en PHP.

La condition `if` est également la plus simple, puisqu'elle va nous permettre d'exécuter un bloc de code si et seulement si le résultat d'un test vaut `true`.

Exemple :

```
<?php
$a = 2;
$b = 4;

if ($a < $b) {
    echo '$a est plus petit que b';
}
if ($a > $b) {
    echo '$a est plus grand que b';
}
if ($a == $b) {
    echo '$a est égale à b';
}
```

`$a est plus petit que b`

## La condition if...else en PHP

L'exemple précédent serait parfaitement fonctionnel mais n'est pas efficace. En effet, le programme va tester si \$a est plus petit que \$b, définir que oui et donc afficher la phrase prévue. Mais le programme n'est pas fini... Il va donc tester si \$a > \$b et puis que \$a == \$b. Cela n'a aucun sens... Il n'y a aucun cas de figure où plus d'une phrase serait affichée.

C'est là que vient à notre aide if...else. Pour ne pas embrouiller j'ai supprimé un des tests

```
$a = 2;
$b = 4;

if ($a < $b) {
    echo '$a est plus petit que b';
}
else {
    echo '$a est plus grand que b';
}
//if ($a == $b) {
//    echo '$a est égale à b';
//}
```

L'affichage est bien entendu identique.

Comme vous pouvez le constater, il n'y a pas de condition sur le else. Si la condition du if est false, le else est d'office exécuté. Ce qui crée un bug dans notre programme si \$a == \$b

## La condition if...elseif...else en PHP

La condition if...elseif...else (« si...sinon si...sinon ») est une structure conditionnelle encore plus complète que la condition if...else puisqu'elle va nous permettre cette fois-ci de générer autant de cas que l'on souhaite et d'arranger notre bug.

Notez que contrairement au else, on peut faire suivre autant de elseif que l'on veut dans notre condition if...elseif...else, chacun avec un test différent.

```

if($nom == "Laurent") {echo'coucou Laurent';}
elseif($nom == "Pierre") {echo'coucou Pierre';}
elseif($nom == "Sophie") {echo'coucou Sophie';}
elseif($nom == "...") {echo'coucou ...';}

```

Revenons à notre exemple :

```

if ($a < $b) {
    echo 'a est plus petit que b';
}
elseif ($a > $b) {
    echo 'a est plus grand que b';
}
else {
    echo 'a est égale à b';
}

```

Le else final n'est en rien obligatoire. Nous aurions pu écrire le code cette façon pour toujours le même résultat.

```

$a = 2;
$b = 4;
$txt = 'a est égale à b';

if ($a < $b) {
    $txt = 'a est plus petit que b';
}
elseif ($a > $b) {
    $txt = 'a est plus grand que b';
}
echo $txt;

```

Vous comprendrez que si cela fonctionne ce n'est pas le plus efficace. Le deuxième test ne servant absolument à rien.

Simplifions :

```

$a = 2;
$b = 4;
$txt = 'a est égale à b';

if ($a < $b) {
    $txt = 'a est plus petit que b';
}
else {
    $txt = 'a est plus grand que b';
}
echo $txt;

```

Je vous invite à vous renseigner sur « PHP Shorthand if Statements » qui permet de réécrire de cette façon :

```

$a = 2;
$b = 4;
$txt = 'a est égale à b';

$a < $b ? $txt = 'a est plus petit que b' : $txt = 'a est plus grand que b';

echo $txt;

```

## Imbriquer plusieurs conditions

Bien souvent, il faudra effectuer des tests plus complexe que ce que l'on a vu précédemment. Il est souvent utile d'avoir une sous condition dans une condition.

Exemple : Vérifions si une personne peut acheter une boisson. En Belgique il faut 18 ans pour acheter de l'alcool fort et 16 ans de la bière.

```
$type = "biere"; // valeur biere ou soft ou alcool
$age = 17;
$ok = false; //par défaut on interdit la vente. C'est une bonne pratique |

if ($age < 18) {
    if ($type == "biere") {
        if($age >= 16) $ok = true;
    }
    if($type == 'soft') $ok = true;
}
else $ok = true;

if($ok) echo 'Vous pouvez en avoir';
else echo 'vous ne pouvez pas en avoir';
```

C'est inutilement compliqué dans ce cas... Améliorons la lisibilité avec :

## Les opérateurs logiques

Les opérateurs logiques vont être principalement utilisés avec les conditions puisqu'ils vont nous permettre d'écrire plusieurs comparaisons au sein d'une même condition ou encore d'inverser la valeur logique d'un test.

Opérateur	Définition
AND ou &&	Renvoie true si toutes les comparaisons valent true
OR ou	Renvoie true si une des comparaisons vaut true
!	Renvoie true si la comparaison vaut false (et inversement)

Réécrivons le code précédent :

```
if ($type == 'soft') $ok = true;
elseif ($age >= 18) $ok = true;
elseif ($type == 'biere' && $age >= 16) $ok = true;
```

Déjà plus simple... Mais on peut encore simplifier :

```
if ($age >= 18 || $type == 'soft' || ($type == 'biere' && $age >= 16)) $ok = true;
```

Si le client à au minimum 18ans ou que la boisson est un soft ou que c'est une bière mais que le client à minimum 16 ans alors il peut en acheter...

Notez que par défaut \$ok vaut false. C'est une bonne méthode d'un point de vue sécurité. Pourquoi ?

## Exercices

### Exercice 1 :

Remplissez le if de la conditionnelle suivante qui teste si le résultat atteint la moyenne /20

```
$resultat = 15;
if () {
    echo 'C'est bien vous avez eu la moyenne ou plus au test';
} else {
    echo 'Vous n'avez pas eu la moyenne au test';
}
```

### Exercice 2 :

Déclarer une variable `$budget` qui contient la somme de 1 553,89 €. Déclarer une variable `$achats` qui contient la somme de 1 554,76 €. Afficher si le budget permet de payer les achats.

### Exercice 3 :

Si la variable `age` est inférieur à 18 ans, afficher "Tu es mineur ", sinon "Vous êtes majeur".

### Exercice 4

En fonction de l'heure, afficher ces messages :

avant 12h: Bonne matinée

de 12h à 14h: Bonne appétit

de 14h à 19h: Bonne après midi

sinon Bonne soirée

### Exercice 5 :

Compléter la conditionnelle suivante qui teste l'age et l'origine de la personne.

`$age` est un nombre entier.

`$origin` peut prendre la valeur : "Belge" ou "belge" (ou « français »,...). Attention que l'objectif est de tester les 2 possibilités. Donc pas de fonction pour mettre en minuscule le texte.

Le code ne peut pas changer en dehors des 2 parenthèses.

```
echo '<h2>Exercice 5</h2>';

$age = 15;
$origin = 'français';
if () {
    echo 'Vous avez 18 ans ou plus et êtes Belge';
} elseif () {
    echo 'Vous avez 18 ans ou plus et êtes étranger';
} else {
    echo 'Vous n\'êtes pas majeur';
}
```

**Exercice 6 :** Réécrire le code précédent en ne testant qu'une fois l'âge.

**Exercice 7 :** Rédigez une expression conditionnelle pour tester si un nombre est à la fois un multiple de 3 et de 5. (aide : Modulo - reste d'une division euclidienne)

**Correctif :** conditions.php

## Projet jeu du pendu V0.1

Nous allons créer, petit à petit, un jeu du pendu. Ceci uniquement en PHP, HTML et CSS.

Télécharger le fichier de base 0-fichierBase.php et le fichier CSS style.css

Vous avez à disposition plusieurs choses :

- le code HTML et CSS du pendu
- un bloc « constante d'environnement ». Il contient une constante qui définit le nombre maximum d'échec possible avant le Game Over. Une constante est une sorte de variable mais la valeur ne peut jamais changer contrairement à une variable. Par convention elles sont écrites en majuscules. Pour l'afficher par exemple :

```
echo NBR_CHANCE;
```

- un bloc « variables qui viennent du formulaire ». À terme certaines données seront envoyées par un formulaire html. Le joueur devra par exemple indiquer la lettre à jouer.

Pour l'instant, on « force » la valeur n'ayant pas de formulaire.

- Un bloc « variables externes ». C'est le cas du mot à trouver. Externe car à terme, il viendra d'un service externe au jeu en lui-même.
- Un bloc « variables qui seront calculées ». Dans un premier temps, on force toujours les valeurs. Évidemment le nombre d'erreur devra être calculé en incrémentant la valeur à chaque erreur.

### Objectif de la V0.1

1. Créer un dossier dans votre serveur web « pendu » et y placer les 2 fichiers de base
2. Créer un repository pour le jeu sur github.
3. Afficher les morceaux du pendu suivant le nombre d'erreur (compris entre 0 et NBR\_CHANCE). On commence par la tête
4. Afficher les lettres qui ont déjà été trouvées. (a,b,c)
5. Afficher un message si la lettre proposée est bonne ou pas (il faudra créer une nouvelle variable)
6. Afficher un message quand c'est game over.
7. Afficher un message quand le mot est trouvé (le formulaire contiendra un champ où l'on peut proposer un mot en cours de jeu). Une nouvelle variable devra être définie.

Attention que pour l'instant il ne faut pas vérifier si une lettre est dans le mot mystère ou autre. On force les variables pour tester.

Tester tous les cas en changeant les diverses variables.



## L'instruction switch en PHP

Dans la vie de développeur il faut de temps à autre tester une valeur de nombreuses fois. Cela nous amène à des conditions du type

```
if ($a=...)... elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....
```

```
seif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....elseif($a=...)....
```

switch permet d'être plus lisible et un peu plus rapide. Sa limite est que c'est une comparaison « pauvre » (== et pas ===)

La syntaxe de switch est assez simple. On indique la variable à tester, les valeurs possibles et le code à exécuter si c'est vrai. L'instruction break, permet de quitter le switch. Sans cela le test suivant est effectué.

```
$i=1;
switch ($i) {
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
}

// Équivaut à:

if ($i == 0) {
    echo "i égal 0";
} elseif ($i == 1) {
    echo "i égal 1";
} elseif ($i == 2) {
    echo "i égal 2";
}
```

## L'instruction match

Cette instruction n'est disponible que depuis PHP 8.0. Elle complète switch car la comparaison effectuée est riche (= = et pas ==)

Utilisation de base :

```
$return_value = match ($food) {  
    'apple' => 'This food is an apple',  
    'bar' => 'This food is a bar',  
    'cake' => 'This food is a cake',  
};
```

Utilisation avec des opérations de comparaison :

```
$age = 18;  
  
$output = match (true) {  
    $age < 2 => "Bébé",  
    $age < 13 => "Enfant",  
    $age <= 19 => "Adolescent",  
    $age > 19 => "Jeune adulte",  
    $age >= 99 => "Adulte âgé"  
};
```

## « *Read the fucking manual* » ou *l'art de lire la doc :-)*

Si un jour vous posez une question à un autre développeur et qu'il vous répond RTFM. Le message sera passé...

Comme l'ensemble des langages, php a une documentation exhaustive. Avant de foncer sur un forum ou de demander à une intelligence artificielle cherchez dedans ! Vous aurez accès aux subtilités, aux limites,... de la fonction recherchée.

Php regorge de fonctions pré-établies. Une fonction est un « programme » qui va effectuer un tâche spécifique. Vous connaissez echo, if et d'autres fonctions mais il en existe tant d'autre...

Les fonctions php vont vous permettre d'avoir une solution à un de vos besoins. Les fonctions vont renvoyer un type de réponse et vous devrez lui donner des informations pour que cela fonctionne.

Je vous propose de chercher dans votre moteur de recherche « strtoupper php » Le premier résultat devrait renvoyer vers la doc php.org.

Que voyons nous dans celle-ci

### strtoupper

(PHP 4, PHP 5, PHP 7, PHP 8)

strtoupper — Make a string uppercase

Vous trouvez le nom de la fonction et ce qu'elle fait. Mais également les versions de php compatible !

### Description

```
strtoupper(string $string): string
```

Dans description, cette partie est importante.

Comment utilisons t'on cette fonction. On y indique son nom et entre parenthèse on lui donne la variable qui DOIT être de type string et qui est le mot à mettre en majuscule. Derrière les « : » ce qu'elle répond. Ici un string

Vous trouverez plus bas des exemples d'utilisation et même des commentaires de contributeurs qui explique des subtilités, des trucs et stuces, des limites ou bugs...

## **La variable tableau**

Elle vaut bien un chapitre...

## C'est quoi un tableau ?

Que cela soit en Front ou en back, difficile de passer à coter...

En gros, un tableau est une variable qui contient un ensemble de variable

Imaginez que vous vouliez stocker dans une variable vos compétences en programmation. Si vous n'avez qu'une compétence c'est simple

```
$competences= "php";
```

Mais cela se complique si vous en avez plusieurs...

```
$competences= "php & html & javascript";
```

Cela serait une solution, mais si vous voulez faire un code php qui donnera ceci en html

```
<ul>
    <li>php</li>
    <li>html</li>
    <li>javascript</li>
</ul>
```

C'est pas gagné. Heureusement les tableaux sont là pour ca.

On va pouvoir créer une variable qui pourrait être représentée de la sorte (attention, ce n'est qu'une représentation pour comprendre, cela ne fonctionne pas comme ça)

```
$competences = ($competence1 = « php, $competence2 = «html », $competence3 = « javascript »)
```

Avec un truc du genre nous pouvons faire notre affichage précédent en récupérant les 3 variables compétence.

## Présentation des tableaux en PHP

Les tableaux en PHP sont donc des variables spéciales qui peuvent stocker plusieurs valeurs en même temps.

Dans un tableau, chaque valeur va être associée à une clef unique. Cette clef va nous permettre notamment de récupérer la valeur associée. Nous allons pouvoir définir les différentes clefs ou laisser le PHP générer automatiquement les clefs pour les différentes valeurs d'un tableau.

On va pouvoir créer trois types de tableaux différents en PHP :

- Des tableaux numérotés ou indexés (les clefs vont être des nombres) ;
- Des tableaux associatifs (nous allons définir la valeur que l'on souhaite pour chaque clef) ;
- Des tableaux multidimensionnels (tableaux qui stockent d'autres tableaux en valeur).

Pour créer un tableau, on utilise la syntaxe [], mais vous pouvez croiser encore l'ancienne notation **array()**.

On va pouvoir passer autant d'argument qu'on souhaite stocker de valeurs dans notre tableau. Les arguments vont pouvoir être soit des valeurs simples (auquel cas les clefs seront des entiers générés automatiquement), soit des paires clef => valeur.

## Création d'un tableau numéroté ou indexé en PHP

Les tableaux numérotés sont le type de tableaux le plus simple à créer en PHP puisque les clefs vont être générées automatiquement par le PHP.

Pour créer un tableau numéroté en PHP, il suffit en fait d'indiquer une série de valeurs et le PHP associera automatiquement une clef unique à chaque valeur, en commençant avec la clef 0 pour la première valeur, la clef 1 pour la deuxième valeur, la clef 2 pour la troisième valeur, etc.

```
$competences = ["php", "html", "javascript"];
```

Rien de plus simple. Regardons ce qu'il se cache dans cette variable grâce à `var_dump()`.

```
array(3) {  
    [0]=>  
        string(3) "php"  
    [1]=>  
        string(4) "html"  
    [2]=>  
        string(10) "javascript"  
}
```

Cela nous renvoi bien un tableau de 3 valeurs.

- La première à la clé 0 et est un string de 3 caractères et valeur php
- La deuxième à la clé 1 et est un string de 4 caractères et valeur html
- La troisième à la clé 2 et est un string de 10 caractères et valeur javascript

Les clés ont été définies par php, car elles n'ont pas été indiquées au départ. Nous aurions pu le faire nous même :

Façon 1 :

```
$competences[0] = "php";  
$competences[1] = "html";  
$competences[2] = "javascript";
```

Façon 2 :

```
$competences = [0 => "php", 1 => "html", 2 => "javascript"];
```

Dans tous les cas le `var_dump()` renverra la même chose.

Évidemment, dans le cas présent cela n'a pas de sens de donner les clés, mais cela sera nécessaire dans beaucoup de cas.

Pour afficher les valeurs d'un tableau numéroté une à une, il suffit d'écho notre variable tableau en précisant l'indice (entre crochets) correspondant à la valeur que l'on souhaite afficher.

```
$competences = ["php", "html", "javascript"];  
  
echo $competences[0]; //retournera php  
echo $competences[1]; //retournera html  
echo $competences[2]; //retournera javascript
```

## Les tableaux associatifs PHP

Un tableau associatif est un tableau qui va utiliser des clefs textuelles qu'on va associer à chaque valeur.

Les tableaux associatifs vont s'avérer intéressants lorsqu'on voudra donner du sens à nos clefs, c'est-à-dire créer une association forte entre les clefs et les valeurs d'un tableau.

Les tableaux associatifs vont être différents des tableaux numérotés au sens où nous allons devoir définir chacune des clefs : PHP ne va pas ici pouvoir nommer automatiquement nos clés. Attention que les clés doivent être uniques dans le tableau.

Tout comme pour les tableaux numérotés, on va pouvoir créer notre tableau en une fois en utilisant la syntaxe [] ou le construire clef par clef et valeur par valeur.

"co

Imaginons le tableau indexé suivant qui contient les informations d'une personne.

```
$user = [
    24,
    "médecin",
    "François",
    "Bernard",
    32
];
```

Vous savez que cela comprend la profession, l'âge, le nom, le prénom et le nombre de commande que l'utilisateur a passé sur le site...

A part la profession, impossible de savoir ce que représente quoi... Même si être médecin est rare à 24 ans :-)

Réécrivons cela avec un tableau associatifs

```
$user = [
    "nbrCommande" => 24,
    "profession" => "médecin",
    "prenom" => "François",
    "nom" => "Bernard",
    "age" => 32
];
```

Plus simple à comprendre et à utiliser :

```
e!ho 'Bonjour ' . $user['prenom'] . ' ' . $user['nom'] . ', vous avez ' . $user['age'] . ' ans.  
Vous avez effectué ' . $user["nbrCommande"] . "commandes sur le site";
```

## Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui va lui-même contenir d'autres tableaux en valeurs. C'est donc un tableau de tableaux.

On appelle tableau à deux dimensions un tableau qui contient un ou plusieurs tableaux en valeurs, tableau à trois dimensions un tableau qui contient un ou plusieurs tableaux en valeurs qui contiennent eux-mêmes d'autres tableaux en valeurs et etc.

Les « sous » tableaux vont pouvoir être des tableaux numérotés ou des tableaux associatifs ou un mélange des deux.

Pour illustrer cela, rajoutons un tableau de compétence à notre utilisateur.

```
$user = [  
    "nbrCommande" => 24,  
    "profession" => "médecin",  
    "competences" => [  
        "chirurgie",  
        "ophtalmologie",  
    ],  
    "prenom" => "François",  
    "nom" => "Bernard",  
    "age" => 32  
];
```

Notre tableau associatif comprend maintenant une clé « compétence » qui est un tableau indexé.

```
array(6) {  
    ["nbrCommande"]=>  
    int(24)  
    ["profession"]=>  
    string(8) "médecin"  
    ["competences"]=>  
    array(2) {  
        [0]=>  
        string(9) "chirurgie"  
        [1]=>  
        string(13) "ophtalmologie"  
    }  
    ["prenom"]=>  
    string(9) "François"  
    ["nom"]=>  
    string(7) "Bernard"  
    ["age"]=>  
    int(32)  
}
```

Autre exemple, sur notre site fictif, il n'y a pas qu'un utilisateur mais plusieurs. Imaginons que pour l'instant il y en a deux.



```

$users = [
    [
        "nbrCommande" => 24,
        "profession" => "médecin",
        "competences" => [
            "chirurgie",
            "ophtalmologie",
        ],
        "prenom" => "François",
        "nom" => "Bernard",
        "age" => 32
    ],
    [
        "nbrCommande" => 7,
        "profession" => "ingénieur",
        "competences" => [
            "construction de routes et pont",
            "construction métallique",
            "experte en matériaux durables"
        ],
        "prenom" => "Sophie",
        "nom" => "Plakbala",
        "age" => 41
    ]
];

```

```

array(2) {
  [0]=>
    array(6) {
      ["nbrCommande"]=>
        int(24)
      ["profession"]=>
        string(8) "médecin"
      ["competences"]=>
        array(2) {
          [0]=>
            string(9) "chirurgie"
          [1]=>
            string(13) "ophtalmologie"
        }
      ["prenom"]=>
        string(9) "François"
      ["nom"]=>
        string(7) "Bernard"
      ["age"]=>
        int(32)
    }
  [1]=>
    array(6) {
      ["nbrCommande"]=>
        int(7)
      ["profession"]=>
        string(10) "ingénieur"
      ["competences"]=>
        array(3) {
          [0]=>
            string(30) "construction de routes et pont"
          [1]=>
            string(24) "construction métallique"
          [2]=>
            string(30) "experte en matériaux durables"
        }
      ["prenom"]=>
        string(6) "Sophie"
      ["nom"]=>
        string(8) "Plakbala"
      ["age"]=>
        int(41)
    }
}

```

## Affichage des tableaux multidimensionnels

Nous voilà avec de beaux tableaux, mais comment afficher cela ?

Notre premier tableau dans la hiérarchie est un tableau indexé. On sait donc que le premier utilisateur à la clé 0, le deuxième la clé 1 ...

Donc si nous désirons afficher le prénom du premier utilisateur, il nous suffira d'écrire

```
echo $users[0]["prenom"];
```

De la même façon si on désire la première compétence de cet utilisateur :

```
echo $users[0]["competences"][0];
```

## Tableau associatif ou indexé, comment choisir...

En règle générale on préfère un tableau indexé dans le cas d'une liste dont la longueur n'est pas définie et peu changer. Dans l'exemple des compétences des utilisateurs il serait difficile (et inutile...) de faire un tableau associatif. Des utilisateurs peuvent avoir une ou deux ou 200 compétences.

Dans le même exemple nous pourrions nous dire que c'est la même chose pour les utilisateurs. Il peut y en avoir un ou bien plus. Mais ce n'est pas parfaitement exacte en pratique.

Il est évident que dans un vrai site, la liste des utilisateurs est stockée dans une base de donnée. Cela signifie qu'il existe obligatoirement un identifiant unique par utilisateur. Une sorte de référence qui permet de le trouver facilement dans la base de donnée.

On pourrait imaginer mettre cette référence dans le tableau de valeurs de l'utilisateur comme ceci :

```
$users = [  
  [  
    "referenceClient" => "Ref1",  
    "nbrCommande" => 24,  
    "profession" => "médecin",  
    "competences" => [  
      "chirurgie",  
      "ophtalmologie",  
    ],  
  ],  
  [  
    "referenceClient" => "Ref2",  
    "nbrCommande" => 7,  
    "profession" => "ingénieur",  
  ],  
]
```

Mais une façon bien plus performante serait de mettre cette référence en clé. Comme elle est unique en base de donnée on peut le faire. Cela sera bien plus efficace pour afficher les données d'un utilisateur si on a sa référence.

```
$users = [  
  "ref1" => [  
    "nbrCommande" => 24,  
    "profession" => "médecin",  
    "competences" => [  
      "chirurgie",  
      "ophtalmologie",  
    ],  
    "prenom" => "François",  
    "nom" => "Bernard",  
    "age" => 32  
  ],  
  "ref2" => [  
    "nbrCommande" => 7,  
    "profession" => "ingénieur",  
  ],  
]
```

```
echo $users["ref1"]["prenom"];
```

Exercice :

Transformer le fichier excell (tableau.php) en un tableau php qui doit être exploitable au maximum. Il faut donc pouvoir avoir à la fin une solution pour sortir une information très précise. Pensez à l'ensemble des besoins que vous pourriez avoir après. Avoir une info, faire des calculs, ...

Vous aurez remarqué que les informations sont parfois différentes pour les 2 pays. Soit l'info est manquant, soit elle est incomplète. Malgré cela la structure des 2 tableaux pays doit être la même. Le contenu seul changera.

**Correctif :** tableau\_corrige.php

## **Les boucles et les opérateurs d'incrémentation et de décrémentation**

## C'est quoi une boucle

Les boucles vont nous permettre d'exécuter plusieurs fois un bloc de code, c'est-à-dire d'exécuter un code « en boucle » tant qu'une condition donnée est vérifiée.

Lorsqu'on code, on va en effet souvent devoir exécuter plusieurs fois un même code. Imaginez vous devoir afficher toute les information contenues dans un tableau php. Le meme code sera exploité (un simple echo ici) pour toutes les lignes.

Ou encore récupérer la liste des produits achetés dans une commande, afficher le prénom de tous les utilisateurs de notre site, ...

Nous disposons de quatre boucles différentes en PHP :

- La boucle while (« tant que ») ;
- La boucle do... while (« faire... tant que ») ;
- La boucle for (« pour ») ;
- La boucle foreach (« pour chaque »).

Le fonctionnement général des boucles sera toujours le même : on pose une condition qui sera généralement liée à la valeur d'une variable et on exécute le code de la boucle « en boucle » tant que la condition est vérifiée.

Pour éviter de rester bloqué à l'infini dans une boucle, vous pouvez donc déjà noter qu'il faudra que la condition donnée soit fausse à un moment donné (pour pouvoir sortir de la boucle).

Pour que notre condition devienne fausse à un moment, on incrémentera ou on décrémentera la valeur de notre variable à chaque nouveau passage dans la boucle.

## Les opérateurs d'incrément et de décrémentation

Incrémenter une valeur signifie ajouter 1 à cette valeur tandis que décrémenter signifie enlever 1.

Les opérations d'incrément et de décrémentation vont principalement être utilisées avec les boucles en PHP. Elles vont pouvoir être réalisées grâce aux opérateurs d'incrément ++ et de décrémentation --.

Donc en gros :

$\$x++$  équivaut à  $\$x = \$x + 1$

$\$x--$  équivaut à  $\$x = \$x - 1$

Il existe également les opérateurs  $++\$x$  et  $--\$x$  La seule différence est dans le cas ou vous traitez la valeur de  $\$x$  au moment de l'incrément :

```

$x = 0;
echo ++$x; // affiche 1
$x = 0;
echo $x++; // affiche 0
echo $x; // affiche 1

```

## La boucle PHP while (rarement utilisée)

La boucle while (« tant que » en français) est la boucle PHP la plus simple à appréhender.

La boucle while va nous permettre d'exécuter un certain bloc de code « tant qu'une » condition donnée est vérifiée.

```

$i=0;
while ($i != 10){
    echo '$i vaut: '.$i.'  
';
    $i++;
}

```

```

$i vaut: 0
$i vaut: 1
$i vaut: 2
$i vaut: 3
$i vaut: 4
$i vaut: 5
$i vaut: 6
$i vaut: 7
$i vaut: 8
$i vaut: 9

```

Dans l'exemple ci-dessus, on commence par stocker la valeur 0 dans notre variable \$i. On dit également qu'on « initialise » notre variable.

Ensuite, on crée notre boucle while, qui va réutiliser \$i puisque nous allons boucler sur les valeurs successives stockées dans \$i jusqu'à ce que \$i stocke une certaine valeur.

Que se passe-t-il exactement dans cette boucle ? Dans le cas présent, on demande à notre boucle d'afficher \$i vaut : suivi de la valeur de \$i. Ceci jusqu'à quand la valeur stockée dans \$i soit égale à 10.

A la fin de chaque passage dans la boucle, on ajoute 1 à la valeur précédente contenue dans \$i grâce à l'opérateur d'incrément ++. Cette étape d'incrément est indispensable. Sans celle-ci, \$i contiendrait toujours la valeur 0 (sa valeur de départ) on ne pourrait jamais sortir de la boucle.

## La boucle PHP do...while (rarement utilisée)

La boucle PHP do... while (« faire... tant que » en français) ressemble à priori à la boucle while mais va fonctionner « en sens inverse » par rapport à while.

En effet, la boucle PHP do... while va également nous permettre d'exécuter un code tant qu'une condition donnée est vraie, mais cette fois-ci le code dans la condition sera exécuté avant que la condition soit vérifiée.

Ainsi, même si une condition est fausse dès le départ, on effectuera toujours au moins un passage au sein d'une boucle do...while, ce qui n'est pas le cas avec une boucle while.

```
$i = 0;
$y = 10;
echo '<br>while $i<br>';
while ($i < 10) {
    echo 'Avec while $i vaut: ' . $i . '<br>';
    $i++;
}
echo '<br>while $y<br>';
while ($y < 10) {
    echo 'Avec while $y vaut: ' . $y . '<br>';
    $y++;
}

$i = 0;
$y = 10;
echo '<br>doWhile $i<br>';
do {
    echo 'avec doWhile $i vaut: ' . $i . '<br>';
    $i++;
} while (
    $i < 10
);
echo '<br>doWhile $y<br>';
do {
    echo 'avec doWhile $y vaut: ' . $y . '<br>';
    $y++;
} while (
    $y < 10
);
```

```
while $i
Avec while $i vaut: 0
Avec while $i vaut: 1
Avec while $i vaut: 2
Avec while $i vaut: 3
Avec while $i vaut: 4
Avec while $i vaut: 5
Avec while $i vaut: 6
Avec while $i vaut: 7
Avec while $i vaut: 8
Avec while $i vaut: 9
```

```
while $y
```

```
doWhile $i
avec doWhile $i vaut: 0
avec doWhile $i vaut: 1
avec doWhile $i vaut: 2
avec doWhile $i vaut: 3
avec doWhile $i vaut: 4
avec doWhile $i vaut: 5
avec doWhile $i vaut: 6
avec doWhile $i vaut: 7
avec doWhile $i vaut: 8
avec doWhile $i vaut: 9
```

```
doWhile $y
avec doWhile $y vaut: 10
```

## La boucle for

La boucle PHP for (« pour » en français) est la boucle utilisée majoritairement quand un test est nécessaire.

Nous pouvons décomposer le fonctionnement d'une boucle for selon trois phases :

- Une phase d'initialisation :  
Nous allons tout simplement initialiser la variable que nous allons utiliser dans la boucle.
- Une phase de test :  
Nous allons préciser la condition qui doit être vérifiée pour rester dans la boucle.
- Une phase d'incrémentation : Nous allons pouvoir incrémenter ou décrémenter notre variable afin que notre condition soit fausse à un moment donné.

Concrètement, reprenons l'exemple précédent avec for :

```
for ($i=0; $i<10; $i++){  
    echo 'avec for $i vaut: ' . $i . '<br>';  
}
```

```
$i vaut: 0  
$i vaut: 1  
$i vaut: 2  
$i vaut: 3  
$i vaut: 4  
$i vaut: 5  
$i vaut: 6  
$i vaut: 7  
$i vaut: 8  
$i vaut: 9
```

Meme résultat qu'avec while ou doWhile.

## La boucle PHP foreach

La boucle PHP foreach est un peu particulière puisqu'elle a été créée pour fonctionner avec des variables tableaux . Elles est certainement la plus utilisée aujourd'hui.



La boucle foreach va parcourir un tableau du début à la fin. Il n'y a pas de test pour sortir de la boucle.

Exemple :

```
$array = ['pierre', 'papier', 'ciseau'];  
echo '<br>foreach<br>';  
  
foreach ($array as $value){  
    echo 'avec foreach la valeur vaut: ' . $value . '<br>';  
}
```

```
foreach  
avec foreach la valeur vaut: pierre  
avec foreach la valeur vaut: papier  
avec foreach la valeur vaut: ciseau
```

N'oublions  
pas qu'un

tableau est une association entre une clé et une valeur. La boucle foreach va nous permettre d'avoir accès aux 2.

```
foreach ($array as $key => $value){  
    echo 'avec foreach la valeur de la clé ' . $key . ' vaut: ' . $value . '<br>';  
}
```

```
avec foreach la valeur de la clé 0 vaut: pierre  
avec foreach la valeur de la clé 1 vaut: papier  
avec foreach la valeur de la clé 2 vaut: ciseau
```

## Break et continue, gagnons du temps d'exécution

Imaginez un tableau qui contient le dictionnaire. Le tableau aura donc au moins 60.000 valeurs avec une structure semblable à ceci : `$dico['table'] = "La table est un objet ....."`;

Créons un code qui va trouver la définition d'un mot en particulier :

```
$dico['abris'] = "Un abris est... ..";  
$dico['table'] = "La table est ....." ;  
$dico['croquette'] = "Une croquette est.....";  
$dico['....'] = "....."; //60000 X donc...  
  
$motRecherche = 'table';  
  
foreach ($dico as $key => $value){  
    if($key == $motRecherche) echo 'La définition de ' . $motRecherche . ' est: ' . $value;  
}
```

Cela va fonctionner mais que va faire le programme.

Il va boucler sur le tableau. Très vite il tombera sur le mot « table » et affichera la définition.

Mais le programme ne sera pas fini... Il va continuer sur les milliers d'autres mots à la recherche d'une autre concordance qu'il ne trouvera jamais.

C'est là que l'**instruction break;** est utilisée. Break va simplement arrêter la boucle.

```
foreach ($dico as $key => $value){
    if($key == $motRecherche) {
        echo 'La définition de ' . $motRecherche . ' est: ' . $value;
        break;
    }
}
```

En pratique, dès que la clé vaut table le programme affichera la phrase et terminera de boucler pour continuer

le reste du code éventuellement.

L'**instruction continue;** ne va pas arrêter la boucle mais passer à l'occurrence suivante.

Modifions notre programme pour rechercher 2 définitions :

```
$motRecherche = 'table';
$motRecherche2 = 'panier';

foreach ($dico as $key => $value) {
    if ($key != $motRecherche && $key != $motRecherche2) continue;
    echo 'La définition de ' . $motRecherche . ' est: ' . $value;
}
```

Il faut bien comprendre le fonctionnement. Si \$key ne vaut ni « table », ni « panier », le programme tombe sur l'instruction continue; Il ne va donc pas traiter la suite du code de la boucle et donc le echo dans cet exemple. Il passera au mot suivant. Quand il tombera sur un des 2 mots. Le test sera false, donc l'instruction continue; ne sera pas exécutée contrairement au echo.

Par contre, le script a des problèmes qu'il va falloir arranger...

- Le script affichera 2 lignes avec le mot de \$motRecherche (la clé) dans les deux, et jamais celle de \$motRecherche2
- Le script ne s'arrête pas quand les 2 définitions sont trouvées.

Cela sera votre mission de corriger cela dans les exercices 10 et 11 qui suivent.

## Exercices

### Niveau Bronze

**Exercice 1 :** Affichez une liste de course contenant 10 articles. Pour cela créez un tableau \$list indexé. En utilisant la boucle foreach, affichez la liste sous forme de liste à point <ul><li>

**Exercice 2 :** Réutilisez le code précédent mais affichez le numéro d'ordre de 1 à 10 au début de la ligne. Ceci en récupérant le numéro de la clé.

**Exercice 3 :** Réutilisez le code précédent. N'affichez pas l'item 2 et 3 et n'affichez que 4 produits. Pour cela il faut utiliser exclusivement les instructions continue; et break;

**Exercice 4 :** Réutilisez le code de l'exercice 2. Affichez que les produits impaires. Pour cela créez une \$variable \$afficher = true ; avant le foreach. Changez la valeur de celle ci pour arriver à l'objectif.

**Exercice 5 :** Réutilisez la variable \$list de l'exercice 1. affichez la liste à l'envers (de 10 à 0). N'utilisez plus foreach mais for pour y arriver. Pensez aux clés pour y arriver.

**Exercice 6 :** Déclarez un tableau multidimensionnel, qui contient les 4 saisons, lesquelles contiennent des légumes (2/saison), lesquels contiennent les clés "quantité" et "prix". Profitez des clés pour vous faciliter la vie :-) Les saisons sont uniques et les légumes le sont également (unicité par saison)

Affichez la liste de cette façon :

<ul>

<li>Ete</li>

<li>

<ul>

<li>Fruit 1 : (quantité : 3 , prix : 10€)</li>

<li>Fruit 2 : (quantité : 3 , prix : 10€)</li>

</ul>

</li>

</ul>

.....

### Niveau Argent

**Exercice 7:** Sur base de l'exercice 7 des conditions. Rédigez un programme qui affichera tous les nombres entre 1 et 1000 qui sont multiples de 3 ou de 5. (Modulo)

**Exercice 8 :** Effectuez une suite de tirages de nombres aléatoires (entre 1 et 100) jusqu'à obtenir une suite composée d'un nombre pair suivi de deux nombres impairs.

Coup de pouce : La fonction php rand() permet d'avoir un nombre aléatoire. Utilisez la boucle doWhile (et oui, de temps à autre elle est utile).

**Exercice 9** : Choisissez un nombre de trois chiffres. Effectuez ensuite des tirages aléatoires, et comptez le nombre de tirages nécessaire pour obtenir le nombre initial. Arrêtez les tirages, et affichez le nombre de coups réalisés. Réalisez ce script d'abord avec l'instruction while puis avec l'instruction for.

**Exercice 10** : Corrigez le script dico qui termine la théorie pour corriger le bug.

**Exercice 11** : Corrigez notre script dico qui a servi d'exemple à l'instruction continue; pour que la boucle s'arrête quand les 2 mots sont trouvés.

```
$dico = [.....];
```

```
$motRecherche1 = 'table';
```

```
$motRecherche2 = 'panier';
```

```
$compteur = 2;
```

**Niveau Or :**

**Exercice 12** : Affichez ceci :

```
0000000
1111111
2222222
3333333
4444444
5555555
6666666
7777777
8888888
9999999
```

Vous avez 2 variables :

\$colonnes = 10; \$lignes = 7; et une ou plusieurs boucles for (et donc les variables internes à celles ci).

**Exercice 13** : Affichez ceci :

```
1
22
333
4444
55555
666666
7777777
```

Vous avez une variable \$lignes = 7 ; et une ou plusieurs boucles for(et donc les variables internes à celles ci).

**Exercice 14** : Affichez ceci :

```
00000000
10000000
22000000
33300000
44440000
55555000
66666600
77777777
```

Vous avez 2 variables : \$lignes = 8; \$colonnes = 6; et une ou plusieurs boucles for.(et donc les variables internes à celles ci).

**Exercice 15 :** Affichez ceci :

```
20 19 18
17 16 15
14 13 12
11 10 9
8 7 6
5 4 3
2 1
```

Avec 2 variables et une boucle while

## Projet jeu du pendu V0.2

- Créer un tableau avec Quelques mots « mystères »
- Prendre un mot au hasard dans le tableau créer pour « \$motMystere » (doc php). Du coup, \$motMystere passe en variable calculée.
- Recherche réelle de la lettre proposée dans le mot mystère et réflexion sur les cas possible : si la lettre proposée est A que se passe t'il si le mot mystère est venus, table abandonne.
- Modifier le reste du script pour remplacer ce qui automatisable, les variables \$ok et \$lettres sont donc à déplacer dans la partie variables calculées (encore deux de moins :-))
- Remplacer le contenu de \$lettres qui est actuellement constitué des lettres séparées par des virgules par un code qui afficher par exemple pour « tablette » et « t » comme lettre) T\_\_\_\_TT\_ ;:-)

## **Inclure des fichiers dans d'autres en PHP avec include et require**

Il existe quatre structures de contrôle qui vont se révéler très utiles et pratiques lorsqu'on voudra créer un site : include, require, include\_once et require\_once.

### **Présentation de include et de require et cas d'utilisation**

Les instructions PHP include et require vont nous permettre toutes deux d'inclure des fichiers de code (ou plus exactement le contenu de ces fichiers) à l'intérieur d'autres fichiers de code.

Ces deux instructions sont très puissantes et vont s'avérer extrêmement utiles lors de la création d'un site web.

En effet, imaginions que vous vouliez créer un site personnel de plusieurs pages. Sur votre site, il y a de fortes chances que l'en-tête, le menu et le pied de page soient identiques pour toutes les pages.

Jusqu'à présent, en HTML, nous étions obligés de réécrire le même code correspondant à l'en-tête, au menu et au pied de page sur chacune des pages de notre site. Cela nous fait perdre beaucoup de temps et va poser de sérieux problèmes le jour où l'on souhaite modifier le texte dans notre pied de page par exemple pour tout notre site.

Plutôt que de réécrire tout le code relatif à ces différentes parties sur chacune de nos pages, pourquoi ne pas plutôt enregistrer le code de notre menu dans un fichier séparé que l'on appellera par exemple menu.php, et faire de même pour le code relatif à notre en-tête et à notre pied de page puis ensuite inclure directement ces fichiers sur chacune de nos pages ?

Nous allons pouvoir faire cela grâce aux instructions include et require.

Ainsi, nous n'avons plus qu'à écrire le code relatif à notre menu une bonne fois pour toutes et à l'inclure dans chaque page par exemple. Cela représente une considérable simplification pour la maintenance de notre code puisque nous n'avons qu'à modifier le code de notre menu dans le fichier menu.php plutôt que de le modifier dans toutes les pages si nous avons copié-collé le code du menu dans chaque page.

Notez déjà que pour inclure un fichier dans un autre fichier, il faudra préciser son emplacement par rapport au fichier qui contient l'instruction include ou require de la même façon qu'on pourrait le faire pour faire un lien en ou pour inclure une image en HTML.

### **Les différences entre include, include\_once, require et require\_once**

La seule et unique différence entre les instructions `include` et `require` va se situer dans la réponse du PHP dans le cas où le fichier ne peut pas être inclus pour une quelconque raison (fichier introuvable, indisponible, etc.).

Dans ce cas-là, si l'inclusion a été tentée **avec `include`**, le **PHP renverra un simple avertissement** et le reste du script s'exécutera quand même tandis que si la même chose se produit **avec `require`**, **une erreur fatale sera retournée par PHP** et l'exécution du script s'arrêtera immédiatement.

L'instruction `require` est donc plus « stricte » que `include`.

La différence entre les instructions **`include`** et **`require`** et leurs variantes **`include_once`** et **`require_once`** est qu'on va pouvoir inclure plusieurs fois un même fichier avec `include` et `require` tandis qu'**en utilisant `include_once` et `require_once`** cela ne sera pas possible : **un même fichier ne pourra être inclus qu'une seule fois** dans un autre fichier.

Notez également qu'une variable ou fonction définie dans un fichier sera disponible dans le fichier qui l'inclus (`include` ou `require`) après cette inclusion. Il faut imaginer les inclusions comme des copié collé d'un fichier dans un autre.

Réécrivons le projet chocolat ensemble de cette façon.

C'est une bonne pratique de programmation de spécialiser à outrance les fichiers, les class, les fonctions,...

Dans le cas présent, le découpage du code va permettre d'avoir des fichiers bien plus petits qui seront plus simple à lire et à modifier.

## **Exercice Reprenez le projet Agence de voyage et séparez les divers éléments.**

Dans cet exercice, il sera possible de récupérer le header, le footer, le menu sur toutes les pages mais le découpage du code va permettre également d'avoir des fichiers bien plus petits qui seront plus simple à lire et à modifier.

Télécharger le fichier de base (la structure a changée).

Créer un dossier contents qui contiendra les fichiers dont le contenu est partagé.

Le fichier `index` à la racine est « vide ». Cela va devenir l'unique fichier où aboutissent les requêtes http. Cela signifie qu'il aura le rôle d'un router et va faire un `include/require` du bon fichier dans le dossier « pages » suivant l'url. A vous d'imaginer comment faire.



# Les fonctions en PHP

## Introduction aux fonctions PHP

Les fonctions sont un outil très puissants et pratiques en PHP et un préliminaire à la programmation object. C'est donc incontournable. Cela va encore nous permettre d'avoir un code clair et concis mais également augmenter la possibilité de partager du code entre nos pages.

### Définition des fonctions et fonctions internes ou prêtes à l'emploi

Une fonction correspond à une série cohérente d'instructions qui ont été créées pour effectuer une tâche précise. Pour exécuter le code contenu dans une fonction, il va falloir appeler la fonction.

Nous avons déjà croisé des fonctions dans ce cours avec notamment la fonction `strtoupper()` dont le rôle est de mettre en majuscule une chaîne de caractère.

La fonction **`strtoupper()`** fait partie des fonctions dites « internes » au PHP ou « prêtes à l'emploi » tout simplement car sa définition fait partie du langage PHP.

En effet, vous devez bien comprendre ici que la fonction `strtoupper()` contient une série d'instructions qui permettent de mettre notre chaîne de caractère en majuscule.

Comme cette fonction a été créée et définie par le PHP lui-même, nous n'avons pas à nous soucier des instructions qu'elle contient : nous n'avons qu'à appeler `strtoupper()` en précisant son nom et le nom d'une variable à mettre en couple de parenthèses pour que la série d'instructions qu'elle contient soient exécutées et pour obtenir la chaîne en majuscule.

### Les fonctions définies par l'utilisateur en PHP

En plus des fonctions internes, le PHP nous laisse la possibilité de définir nos propres fonctions. Cela va s'avérer très pratique dans le cas où nous travaillons sur un projet avec des besoins très spécifiques et pour lequel on va souvent devoir répéter les mêmes opérations.

Pour utiliser nos propres fonctions, nous allons déjà devoir les définir, c'est-à-dire préciser une première fois la série d'instructions que chaque fonction devra exécuter lors de son appel.

Pour déclarer une fonction, il faut déjà commencer par préciser le mot clef `function` qui indique à PHP qu'on va définir une fonction personnalisée.

Ensuite, nous allons devoir préciser le nom de notre fonction . Le nom des fonctions va suivre les mêmes règles que celles des variables, à savoir qu'il devra commencer par une lettre ou un underscore et ne devra pas être un nom déjà pris par le langage PHP (comme le nom d'une fonction déjà existante par exemple).

En plus de cela, notez qu'à la différence des variables le nom des fonctions est insensible à la casse. Cela signifie que l'utilisation de majuscules et des minuscules ne servent pas à différencier une fonction d'une autre.

Par exemple, si on crée une fonction `bonjour()`, les notations `BONJOUR()`, `bonJOUR()`,... feront référence à la même fonction.

Après le nom de notre fonction, nous devons obligatoirement mentionner un couple de parenthèses puis ensuite placer les instructions que la fonction devra exécuter lors de son appel entre des crochets.

Créons une première fonction très simple aura comme rôle d'afficher le texte « Bonjour » :

```
function direBonjour(){  
    echo'Bonjour';  
}
```

Si l'on test notre code, cela n'affiche rien. C'est entièrement normal car la fonction n'est pas appelée. Et donc php ne l'utilise pas.

Il ne reste plus qu'à en faire appel pour afficher notre texte à l'écran

```
function direBonjour(){  
    echo'Bonjour';  
    echo'<br>';  
}  
direBonjour();  
direBonjour();
```

Bonjour  
Bonjour

Le texte est visible deux fois car la fonction a été appelé deux fois.

## Les paramètres et arguments des fonctions PHP

Il est évident que l'intérêt de la fonction précédente ne saute pas aux yeux... Par contre, imaginez une fonction qui envoi un mail à l'utilisateur après inscription sur un site. Cette fonction va s'occuper de toutes les tâches utiles à cette mission (en appelant elle même d'autres fonctions spécialisées). Elle devra créer le contenu du mail, rajouter des pièces jointes éventuelles, indiquer l'expéditeur, le destinataire, rajouter les clés de sécurité et enfin l'envoyer...

Pour pouvoir fonctionner cette fonction de mailing aura besoin au minimum de l'adresse du destinataire qui changera à chaque fois évidemment.

En réalité, nous avons déjà vu comment cela fonctionne avec `strtoupper($string)`. Ici on passe une chaîne de caractère à la fonction. Ces informations (la valeur de la chaîne de caractère) sont appelées des **arguments**. Nous allons toujours passer les arguments dans les parenthèses de notre fonction.

Certaines fonctions ne vont pas avoir besoin d'argument pour fonctionner, mais la majorité vont en avoir besoin d'un ou de plusieurs. Par ailleurs, certains arguments vont parfois être facultatifs tandis que d'autres seront obligatoires.

Modifions notre fonction pour que son rôle soit de renvoyer « Bonjour + un prénom ». Cette fonction va avoir besoin qu'on lui passe un prénom pour fonctionner correctement.

Dans sa définition, on va donc indiquer le fait que notre fonction va avoir besoin d'une donnée pour fonctionner en précisant **un paramètre** entre les parenthèses.

Ensuite, lorsqu'on appelle notre fonction, nous allons lui passer un prénom effectif, soit en lui passant une variable qui contient un prénom, soit en lui passant directement une chaîne de caractères. La valeur effective va alors ici remplacer le paramètre de la définition de la fonction c'est **l'argument**.

```
function direBonjour($prenom){  
    echo 'Bonjour '.$prenom;  
    echo '<br>';  
}  
  
direBonjour('Pierre');  
?>
```

Bonjour Pierre

## Définir des valeurs par défaut pour les paramètres de nos fonctions

Le PHP va également nous permettre de définir une valeur par défaut pour un paramètre d'une fonction. Cette valeur sera utilisée si aucun argument n'est fourni lors de l'appel de la fonction.

Si on définit une fonction avec plusieurs paramètres et qu'on choisit de donner des valeurs par défaut à seulement certains d'entre eux, alors il faudra placer les paramètres qui ont une valeur par défaut après ceux qui n'en possèdent pas dans la définition de la fonction.

Dans le cas contraire, le PHP renverra une erreur et notre fonction ne pourra pas être exécutée.

Reprenons l'exemple et rajoutons le rôle de l'utilisateur dans l'affichage. Il est probable que 99 % du temps cela soit un utilisateur, le signifier n'est pas nécessaire.

```
function direBonjour($prenom, $role = "utilisateur"){  
    echo 'Bonjour '.$role.' '.$prenom;  
    echo '<br>';  
}
```

```
direBonjour('Pierre');  
direBonjour("chef", "administrateur");
```

Bonjour utilisateur Pierre  
Bonjour administrateur chef

### Portées des variables avec les fonctions

Retenez que ce qui se passe dans une fonction reste dans la fonction.

Que va afficher le code suivant ?

```
ini_set('display_errors', 1);  
ini_set('display_startup_errors', 1);  
error_reporting(E_ALL);  
$a = 5;  
$b = 10;  
echo '<br>1-> $a vaut: ' . @$a . '<br>';  
echo '1-> $b vaut: ' . @$b . '<br>';  
echo '1-> $x vaut: ' . @$x . '<br><br>';  
function multiplier($x, $b)  
{  
    echo 'Résultat du calcul: ' . $x * $b;  
    echo '<br>';  
    echo '<br>2-> $a vaut: ' . @$a . '<br>';  
    echo '2-> $b vaut: ' . @$b . '<br>';  
    echo '2-> $x vaut: ' . @$x . '<br><br>';  
}  
multiplier($a, $b + 1);  
echo '<br>3-> $a vaut: ' . @$a . '<br>';  
echo '3-> $b vaut: ' . @$b . '<br>';  
echo '3-> $x vaut: ' . @$x . '<br><br>';
```

## Le retour d'une fonction

Notre fonction de multiplication pourrait être utilisée pour faire autre chose que d'imprimer le résultat. Par exemple stocker le résultat en base de donnée. C'est pour cela que très souvent une fonction renvoi un donnée que nous pourrons stocker dans une variable pour être utilisée plus tard.

Pour cela nous aurons besoin de la fonction return, le code suivant fait la même chose que précédemment. Afficher le résultat :

```
function multiplier(float $x, float $y)
{
    return $x * $y;
}
$result = multiplier($a, $b);
echo 'Résultat du calcul: ' . $result;
```

Ou encore plus facilement :

```
echo 'Résultat du calcul: ' . multiplier($a, $b);
```

## Le PHP, un langage au typage faible

A la base le langage PHP est un langage qui ne possède pas un typage fort.

Cela signifie de manière concrète qu'on n'a pas besoin de spécifier le type de données attendues lorsqu'on définit des paramètres pour une fonction car PHP va déterminer lui-même le type des données passées à notre fonction en fonction de leur valeur.

Une conséquence de cela est qu'il va être possible de passer des arguments qui n'ont aucun sens à une fonction sans déclencher d'erreur. On va par exemple pouvoir parfaitement passer deux chaînes de caractères à notre fonction multiplier() sans déclencher d'erreur.

## Le typage des arguments

Depuis la version 7 de PHP il est possible de faire du typage en indiquant le type de données attendues lorsqu'on définit une fonction. Si une donnée passée ne correspond pas au type attendu une erreur sera renvoyée.

Cela va permettre à nos fonctions de ne s'exécuter que si les valeurs passées en argument sont valides et donc d'obtenir toujours un résultat cohérent par rapport à nos attentes.

C'est extrêmement important de prendre très vite l'habitude de la faire. En objet, c'est indispensable pour avoir une chance de déboguer facilement et pour éviter des problèmes.

Les types valides les plus importants sont les suivants :

- string L'argument passé doit être de type string (chaîne de caractères)
- int L'argument passé doit être de type integer (nombre entier)
- float L'argument passé doit être de type float ou double (nombre décimal ou nombre entier...)
- bool L'argument passé doit être de type boolean (booléen)
- array L'argument passé doit être de type array (tableau)
- object L'argument passé doit être de type object (objet)

```
$a = "cinq";  
$b = 10;  
$c = 5;  
function multiplier(float $x, float $y)  
{  
    echo 'Résultat du calcul: ' . $x * $y;  
    echo '<br>';  
}  
multiplier($a, $b);  
multiplier($c, $b);
```

Résultat du calcul: 50

**Fatal error:** Uncaught TypeError: multiplier(): Argument #1 (\$x) must be of type float, string given, called in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 15 and defined in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:9 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(15): multiplier() #1 {main} thrown in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 9

## Typage strict

Essayons ceci :

```
$a = 5;  
$b = 10;  
function multiplier(string $x, float $y)  
{  
    echo 'Résultat du calcul: ' . $x * $y;
```

Résultat du calcul: 50

```
    echo '<br>';  
}  
multiplier($a, $b);
```

Mais pourquoi cela fonctionne... Et bien simplement car PHP va essayer de convertir \$x en string. Comme il y parvient, il n'y a pas d'erreur. Si vous faites un var\_dump de \$x au début de la fonction, cela renvoi `string(1) "5"`

Pour éviter cela, il existe le typage strict. C'est celui qui sera utilisé en Objet. Rajoutons une ligne de paramétrage à notre fichier PHP. Attention elle doit être juste après l'ouverture de la balise <?php

```
<?php  
declare(strict_types=1);
```

Maintenant, notre code précédent renvoi une belle erreur :

**Fatal error:** Uncaught TypeError: multiplier(): Argument #1 (\$x) must be of type string, int given, called in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 19 and defined in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:10 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(19): multiplier() #1 {main} thrown in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 10

## Typage du retour d'une fonction

Comme nous l'avons vu, une fonction renvoi presque toujours une donnée. Celle ci peut également être contrôlée. Ceci en indiquant le type juste après les parenthèses et précédé de deux points

```
function multiplier(float $x, float $y): array
```

**Fatal error:** Uncaught TypeError: multiplier(): Return value must be of type array, float returned in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:12 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(15): multiplier() #1 {main} thrown in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 12



## Projet jeu du pendu V0.3

- Réorganiser le code grâce aux fonctions et placer celles-ci dans un/des fichiers externes.
- Rajouter un formulaire pour la lettre proposée mais également pour proposer un mot
- Le jeu n'est toujours pas jouable car on perd les données précédentes à chaque proposition de lettre. Trouvez une solution pour stocker ce qu'il faut dans le formulaire. Il vous faudra utiliser ceci pour récupérer:

```
$lettres = unserialize($_POST['lettres']);
```

et ceci pour envoyer :

```
htmlspecialchars(serialize($lettres))
```

## **Le Timestamp UNIX et la date en PHP**

# Le Timestamp UNIX

Le format Timestamp UNIX est un format de mesure de date très utilisé en programmation, .

L'heure Unix ou heure Posix est une mesure du temps fondée sur le nombre de secondes écoulées depuis le 1er janvier 1970 00:00:00 UTC. Elle est utilisée principalement dans les systèmes qui respectent la norme POSIX1, dont les systèmes de type Unix, d'où son nom. C'est la représentation POSIX du temps.

Ce format de date est difficile à comprendre pour un humain mais comme le Timestamp UNIX est un nombre on va très facilement pouvoir le manipuler en PHP et l'utiliser pour par exemple comparer deux dates.

Un autre intérêt majeur du Timestamp est qu'il va être le même pour un moment donné quel que soit le fuseau horaire puisque ce nombre représente le nombre de secondes écoulées depuis un point précis dans le temps. Cela va s'avérer très pratique lorsqu'on aura besoin de représenter le temps pour l'ensemble de la planète d'un coup.

## Obtenir le Timestamp relatif à la date actuelle en PHP

Pour obtenir le Timestamp actuel en PHP, c'est-à-dire le nombre de secondes écoulées entre le 1er janvier 1970 à minuit GMT et le moment actuel, nous allons utiliser la fonction `time()`.

```
echo 'temps unix actuel: '. time();
```

temps unix actuel: 1736257650

## Obtenir le Timestamp d'une date donnée en PHP

Pour obtenir le Timestamp UNIX lié à une date donnée, nous allons pouvoir utiliser la fonction `mktime()` qui retourne le Timestamp UNIX d'une date.

Nous allons pouvoir lui passer une série de nombres en arguments. Ces nombres vont constituer une date et vont représenter (dans l'ordre) les parties suivantes de la date dont on souhaite obtenir le Timestamp :

1. L'heure ;
2. Les minutes ;
3. Les secondes ;
4. Le jour ;
5. Le mois ;
6. L'année.

Notez que cette fonction a besoin obligatoirement de l'heure, les autres valeurs peuvent être null.

```
echo 'temps unix actuel avec mktime(): '. mktime(8);  
echo 'temps unix du 19 janvier 2025 08:30:00 avec mktime(): '. mktime(8,30,0,19,1,2025);
```

temps unix actuel avec mktime: 1736240361

temps unix du 19 janvier 2025 08:30:00 avec mktime: 1782894600

## La gestion des fuseaux horaires.

Ici, la fonction `mktime()` calcule le Timestamp d'une date donnée en fonction du fuseau horaire du serveur. Je suis à GMT+1. Quand il est 8h30 il est donc 7h30 sur un fuseau GMT. La fonction va donc transformer mon heure en heure GMT et se baser sur 7h30 pour calculer le Timestamp.

## Obtenir un Timestamp à partir d'une chaîne de caractères

Évidemment dans la relation humain machine, il faudra souvent convertir des dates « humaines » en Timestamp.

La fonction **`strtotime()`** va transformer une chaîne de caractères de format date ou temps en Timestamp.

Pour que cette fonction fonctionne correctement, il va falloir lui passer une chaîne de caractères en argument sous un format qu'elle comprend. Notez déjà que cette fonction se base sur le fuseau horaire de votre serveur.

Il y a de nombreuses façons d'écrire une date en PHP. La doc est votre amie :

<http://php.net/manual/fr/datetime.formats.php>.

On va ensuite pouvoir ajouter à ces dates un temps qu'on va également pouvoir préciser sous différents formats. Le format de loin le plus utilisé pour les temps est le format heures:minutes:secondes comme par exemple 12:30:15.

Notez que la fonction `strtotime()` va également accepter des formats de date relatifs comme :

- Une notation basée sur le jour comme `yesterday` (hier à minuit), `today` (aujourd'hui à minuit), `now` (maintenant), `tomorrow` (demain à minuit), etc. ;
- Une notation ordinale comme « first fri of January 2019 » (premier vendredi de janvier en 2019) ;
- Une notation avec « ago » comme « 2 days ago » (avant-hier)
- Une notation avec des « + » et des « - » comme « +1 day » (demain) ou « - 3 weeks » (il y a 3 semaines).

Nous allons de plus pouvoir combiner ces formats de dates ensemble en respectant certaines règles pour créer des formats de dates complexes.

## Obtenir une date à partir d'un Timestamp en PHP

Pour obtenir une date en PHP, on va pouvoir utiliser la fonction **getdate()**.

Cette fonction va accepter un Timestamp en argument et retourner un tableau associatif contenant les différentes informations relatives à la date liée au Timestamp. Si aucun argument n'est passé à **getdate()**, alors la fonction utilisera le Timestamp relatif à la date courante et retournera donc la date actuelle locale.

```
echo '<pre>';
$now = time();
echo 'temps unix actuel: ' . $now . '<br>';

echo 'tableau complet: ';
var_dump(getdate($now));
echo '<br>Juste l'année: ' . getdate($now)['year'];
```

```
temps unix actuel: 1736259973
tableau complet:array(11) {
    ["seconds"]=>
    int(13)
    ["minutes"]=>
    int(26)
    ["hours"]=>
    int(14)
    ["mday"]=>
    int(7)
    ["wday"]=>
    int(2)
    ["mon"]=>
    int(1)
    ["year"]=>
    int(2025)
    ["yday"]=>
    int(6)
    ["weekday"]=>
    string(7) "Tuesday"
    ["month"]=>
    string(7) "January"
    [0]=>
    int(1736259973)
}
```

Juste l'année:2025

## Obtenir et formater une date en PHP

La fonction PHP **date()** va nous permettre d'obtenir une date selon le format de notre choix.

Cette fonction va pouvoir prendre deux arguments. Le premier argument correspond au format de date souhaité et est obligatoire. Le deuxième argument est facultatif et va être un Timestamp relatif à la date qu'on souhaite retourner.

Si le Timestamp est omis, alors la fonction `date()` se basera sur la date courante du serveur.

Pour indiquer le format de date qu'on souhaite voir renvoyer à la fonction `date()`, nous allons lui passer une série de lettres qui vont avoir une signification spéciale.

La liste > <https://www.php.net/manual/fr/datetime.format.php>

```
echo date('l jS \of F Y h:i:s A');
```

Renvoi quelque chose comme :

Tuesday 7th of January 2025 02:43:05 PM

Remarquez l'ajout du « of » qui est « échappé » par « \ »

## Formater une date en PHP : la gestion du décalage horaire

Notez que la fonction `date()` formate une date localement, ce qui signifie que la date renvoyée va être la date locale (avec le décalage horaire). Si on souhaite retourner une date GMT, alors on utilisera plutôt la fonction `gmdate()` qui va s'utiliser exactement de la même manière.

Par ailleurs, il est également possible que la date renvoyée ne corresponde pas à celle de l'endroit où vous vous trouvez si le serveur qui héberge votre site se situe sur un autre fuseau horaire ou s'il a été paramétré sur un fuseau horaire différent du vôtre il faudra corriger cela.

Pour régler ce problème, on peut utiliser la fonction **`date_default_timezone_set()`** en lui passant un fuseau horaire sous un format valide pour définir le fuseau horaire qui devra être utilisé pour les fonctions relatives à la date utilisées dans le script.

La documentation officielle > <http://php.net/manual/fr/timezones.php>.

```
echo 'date locale: ' . date('d/m/Y H:i:s'). '<br>';  
echo 'date gmt: ' . gmdate('d/m/Y H:i:s'). '<br>';  
  
//on change du fuseau horaire  
date_default_timezone_set('America/New_York');  
echo 'date New York: ' . date('d/m/Y H:i:s'). '<br>';  
echo 'date gmt: ' . gmdate('d/m/Y H:i:s'). '<br>';
```

Renverra quelque chose du genre :

```
date locale: 07/01/2025 17:31:45
date gmt: 07/01/2025 16:31:45
date New York: 07/01/2025 11:31:45
date gmt: 07/01/2025 16:31:45
```

Si votre date locale n'est pas exacte c'est que le timezone indiqué dans votre fichier php.ini est inexacte ou manquant. Trouvez la ligne timezone et remplacez la par

```
date.timezone = 'Europe/Brussels'
```

Il est nécessaire de redémarrer le serveur apache après la modification.

## La comparaison de dates en PHP

La comparaison de dates est une chose difficile en informatique et particulièrement en PHP procédurale.

Comme vous l'avez vu les dates peuvent être formatées de diverses façons : tout en chiffres, tout en lettres, avec un mélange de chiffres et de lettres, avec les mois avant ou après, etc. Le problème c'est qu'une fois la date formatée, cela devient un simple string.

La façon la plus simple est de reconverter les dates en timestamp et de comparer ceux-ci.

On va donc procéder en deux étapes : Récupérer les Timestamp liés aux dates que l'on souhaite comparer avec une fonction comme strtotime() par exemple puis on va comparer les Timestamp.

```
$date1 = "18-01-2025";
$date2 = "19 January 2025";

$date1TS = strtotime($date1);
$date2TS = strtotime($date2);

if ($date1TS == $date2TS) {echo "les dates sont identiques";}
elseif ($date1TS < $date2TS){echo $date1 ." est avant " . $date2;}
else {echo $date1 ." est après " . $date2;}
```

18-01-2025 est avant 19 January 2025

## Tester la validité d'une date

La fonction `checkdate()` a été créée dans ce but précis. Elle sera souvent utilisée pour vérifier qu'une date envoyée via un formulaire est cohérente

Cette fonction va accepter trois chiffres en arguments : le premier représente le mois de la date à tester, le deuxième représente le jour et le troisième représente l'année.

```
//30-02-2025  
if(checkdate(2,30, 2025)) echo "La date est valide";  
else echo "La date n'est pas valide";
```

La date n'est pas valide

## Exercices

1. Calculez l'âge d'une personne née le 27 février 1985. Sa date de naissance est dans la variable suivant : `$date = « 27-02-1985 »` ;
2. Quel jour de la semaine était le 3 mars 1993 ? Affichez le résultat en français. Attention, ne cherchez pas une fonction php de traduction du jour, vous devez le faire vous même.
3. Affichez toutes les années bissextiles comprises entre 2005 et 2052
4. Déterminez quel jour de la semaine seront tous les premier Mai des années comprises entre 2025 et 2035. Si le jour est un samedi ou un dimanche, affichez le message « Désolé ! ». Si le jour est un vendredi ou un lundi affichez « Week end prolongé ! ».
5. L'Ascension est le quarantième jour après Pâques (Pâques compris dans les 40 jours). Calculez les dates de l'Ascension pour les années 2010 à 2015.

Resolu : `date.php`



## Transformer une date en français

Par défaut, les dates vont être renvoyées en anglais. Pour transformer une date en anglais vers du français en php procédurale il existait la fonction `strftime()` qui couplée avec `setlocale()` permettait de la faire. A la lecture de la doc je découvre que la fonction est obsolète depuis php 8.1.

Évidemment il est encore possible de changer la langue en procédurale en utilisant `datefmt_format()`.

Cela semble compliqué mais la raison est que c'est une fonction plus orienté object. On va décortiquer cela.

La fonction `datefmt_format()` attend 2 arguments , le premier est « la configuration de la date voulue » et le deuxième une date au format time (donc un timestamp est valable).

Le premier argument sera construit avec `datefmt_create()` auquel nous lui passons la locale, le type de format de date, le type de format de l'heure, le méridien et enfin le calendrier

```
$date = '2023-02-02 22:22:22';
$fmt = datefmt_create(
    'fr_BE',
    IntlDateFormatter::FULL,
    IntlDateFormatter::FULL,
    'Europe/Brussels',
    IntlDateFormatter::GREGORIAN
);
echo '<br>Le premier format affiché est ' . datefmt_format($fmt,
    strtotime($date));

$fmt = datefmt_create(
    'de_DE',
    IntlDateFormatter::FULL,
    IntlDateFormatter::FULL,
    'Europe/Berlin',
    IntlDateFormatter::GREGORIAN
);
echo '<br>Le deuxième format affiché est ' . datefmt_format($fmt,
    strtotime($date));
```

```

$fmt = datefmt_create(
    'de_DE',
    IntlDateFormatter::FULL,
    IntlDateFormatter::FULL,
    'Europe/Berlin',
    IntlDateFormatter::GREGORIAN,
    'MM/dd/yyyy' // il est possible également d'avoir un format
spécifique
);
echo '<br>Le troisième format affiché est ' . datefmt_format($fmt,
strtotime($date));

```

Le premier format affiché est vendredi 3 février 2023 à 4 h 22 min 22 s heure normale d'Europe centrale

Le deuxième format affiché est Freitag, 3. Februar 2023 um 04:22:22 Mitteleuropäische Normalzeit

Le troisième format affiché est 02/03/2023

## Découverte du Php Object avec DateTime()

Comme vous le voyez c'est assez tortueux de travailler avec les dates en php procédural. Heureusement, en objet, c'est bien plus simple... Une belle occasion de découvrir comment cela marche en découvrant l'objet DateTime(). Comme vous l'avez vu, jusqu'à présent nous avons utilisé les dates comme des chaînes de caractères, des entiers en devant passer d'un état à un autre continuellement.

### C'est quoi un objet, la base

Comme indiqué précédemment une variable peut être de type string, bool, array,..., et object.

L'objet va permettre de représenter des concepts un petit peu plus complexe que les autres types.

Il a l'avantage d'être facilement compréhensible par la machine (comme le Timestamp) mais également pour l'humain comme les strings (02/01/2025,...). Il ne faudra donc plus convertir continuellement nos dates.

Les objets peuvent être existants dans php, c'est le cas de DateTime(), mais il est possible de créer ses propres objets également.

Sans l'approche objet, effectuer des calculs complexes sur les dates est extrêmement complexe.

## Découverte de DateTime()

Comme d'habitude la doc => <https://www.php.net/>

Recherchez DateTime dans le formulaire de recherche. Le premier résultat propose bien DateTime et indique que c'est une class et non une fonction comme date() par exemple.

La page présentant DateTime() est plus complexe que ce que vous avez vu précédemment. Cela s'explique simplement par le fonctionnement d'un objet.

Si vous regardez le synopsis vous voyez que cet objet contient des constantes et des méthodes. Passons pour l'instant les constantes. Vous pouvez voir dans les méthodes que cela ressemble furieusement aux fonctions que l'on a utilisé précédemment. On pourrait dire que cet objet se suffit à lui même, il a en son sein tous les outils pour se modifier (modifier, rajouter un temps, indiquer le fuseau,...). Attention que les fonctions ne peuvent être utilisées que par l'objet DateTime().

La première fonction à regarder est

```
public __construct(string $datetime = "now", ?DateTimeZone $timezone = null)
```

Cette fonction est le constructeur, c'est elle qui va créer notre objet quand on le demande. Elle attend deux paramètres un string qui est la date et l'heure et qui vaut par défaut « maintenant » (le format est identique à ce que l'on a vu avant) et la timezone qui est un objet de type timezone et qui vaut null par défaut.

Construisons cet objet et voyons à quoi il ressemble. Pour cela :

```
$dateObjet = new DateTime();  
var_dump($dateObjet);
```

```
object(DateTime)#1 (3) {  
    ["date"]=>  
    string(26) "2025-01-08 14:53:36.563583"  
    ["timezone_type"]=>  
    int(3)  
    ["timezone"]=>  
    string(15) "Europe/Brussels"  
}
```

Nous avons donc un objet de type DateTime qui contient plusieurs données dont la est la date déjà formatée « humainement ». Cette date contient l'heure dont les millisecondes (donc plus précis que le Timestamp unix).

Si vous essayez de faire :

```
echo $dateObjet["date"];
```

Vous verrez que cela ne fonctionne pas. Comme l'erreur le mentionne, \$dateObjet est un objet pas un tableau ! Meme si le var\_dump l'affiche de la même façon.

Pour pouvoir faire cela nous allons devoir passer par les « fonctions » que l'on appelle méthode dans un objet, bien que cela ne change pas grand-chose... Par contre cela s'utilise différemment. Essayons d'afficher la date en utilisant la méthode format

```
echo $dateObjet->format('d/m/Y');
```

08/01/2025

Imaginons maintenant que nous voulons la date de la semaine prochaine

```
$dateObjet = new DateTime();  
$dateObjet->modify('+1 day');  
echo $dateObjet->format('d/m/Y');
```

Vous remarquez que, à la deuxième ligne je n'ai pas besoin de sauvegardé le résultat dans une variable. Je modifie l'objet directement. Pour cette raison ceci ne fonctionne pas

```
$dateObjet = new DateTime();  
$dateObjet->modify('+1 week');  
$dateObjet->format('d/m/Y');  
echo $dateObjet;
```

Pour la meme raison, mon echo ne sait pas afficher un objet, mais il pouvait afficher le résultat de la méthode modify().

Reprenons l'exercice de calcul de l'âge : Calculez l'âge d'une personne née le 27 février 1985

Pour cela nous allons utiliser la méthode diff() qui va retourner un interval de temps.

Par rapport à maintenant, quel est la difference de temps avec la date de naissance :

```

$dateNaissance = new DateTime("1985-02-27");
$maintenant = new DateTime();
$age = $maintenant->diff($dateNaissance, true);
var_dump($age);
echo $age->y;

```

```

object(DateTime)#3 (10) {
    ["y"]=>
    int(39)
    ["m"]=>
    int(10)
    ["d"]=>
    int(9)
    ["h"]=>
    int(15)
    ["i"]=>
    int(38)
    ["s"]=>
    int(49)
    ["f"]=>
    float(0.682244)
    ["invert"]=>
    int(1)
    ["days"]=>
    int(14560)
    ["from_string"]=>
    bool(false)
}
39

```

Le var\_dump vous montre bien que le résultat est un objet ! Il faut donc aller chercher la valeur demandée avec la petite flèche :-)

procédural :

```

$age = strtotime($maintenant) - strtotime($dateNaissance);
echo "Il a " . floor(abs($age / 60 / 60 / 24 / 365)) . "ans";

```

objet :

```

$age = $maintenant->diff($dateNaissance);
echo "Il a " . $age->y . " ans";

```

Que va afficher ceci ? Pourquoi ?

```
$date1 = new DateTime("1985-02-27");  
$date2 = $date1;  
$date2->modify('+1 year');  
echo $date2->format('d/m/Y');  
echo "<br>".$date1->format('d/m/Y');
```

## Exercices

Créer une fonction permettant de ...

1. ...rajouter des jours à une date.
2. ~~...afficher le jour de la semaine en français pour une date spécifique.~~ Todo : a introduire
3. ...comparer deux dates et indiquer laquelle est la plus récente.
4. ...afficher la date du premier jour du mois suivant à partir de la date actuelle. (réalisable avec une fonction en 2 lignes)
5. ...Afficher le nombre de semaines entre deux dates
6. ...Calculer le nombre de jours ouvrables (excluant les week-ends simplement) entre deux dates.
7. ...trouver la date du dernier vendredi du mois
8. ...ajouter un nombre variable de mois à une date donnée
9. ...calculer le nombre de jours restants jusqu'à la fin de l'année à partir de la date actuelle.
10. ...trouver la date du prochain jour férié à partir de la date actuelle.

# Cookies et Sessions

## Obligations légales

En Europe, les obligations pour un développeur web en matière de cookies sont principalement régies par la directive ePrivacy et le Règlement Général sur la Protection des Données (RGPD). Voici un résumé des principales obligations et législations correspondantes :

1. **Consentement préalable** : Avant de déposer ou lire des cookies sur le terminal d'un utilisateur, il est nécessaire d'obtenir son consentement préalable, sauf pour certains cookies strictement nécessaires au fonctionnement du site[1](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies>).

Il faut en effet différencier un cookie fonctionnel d'un cookie de traçage. Deposier un cookie contenant la langue de l'utilisateur est un cookie fonctionnel et ne nécessite pas de consentement par exemple

2. **Information claire et complète** : Les utilisateurs doivent être informés de manière claire et complète sur l'utilisation des cookies, y compris leur finalité, la durée de conservation des données, et les tiers susceptibles d'y avoir accès[2](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies/que-dit-la-loi>).

3. **Possibilité de retrait du consentement** : Les utilisateurs doivent pouvoir retirer leur consentement à tout moment et de manière aussi simple qu'ils l'ont donné[1](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies>).

4. **Exemptions** : Certains cookies sont exemptés de l'obligation de consentement, notamment ceux nécessaires à la fourniture d'un service expressément demandé par l'utilisateur (par exemple, les cookies de session pour un panier d'achat)[2](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies/que-dit-la-loi>).

5. **Responsabilité** : Les éditeurs de sites web, les régies publicitaires et les réseaux sociaux sont responsables de la conformité de leurs pratiques en matière de cookies[2](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies/que-dit-la-loi>).

Pour plus de détails, vous pouvez consulter les lignes directrices de la CNIL et les recommandations spécifiques sur leur site[1](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies>)[2](<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies/que-dit-la-loi>).

## **Cookies**

### **C'est quoi un cookies**

Un cookie est un petit fichier texte qui ne peut contenir qu'une quantité limitée de données. Les cookies vont être stockés sur les ordinateurs de vos visiteurs. Ainsi, à tout moment, un utilisateur peut lui même supprimer les cookies de son ordinateur.

Du côté serveur, il sera possible de définir la date d'expiration ou de le supprimer.

Généralement, nous allons utiliser les cookies pour faciliter la vie des utilisateurs en préenregistrant des données les concernant comme la langue de l'utilisateur par exemple. Cela permet de lui fournir le contenu dans sa langue lors de sa prochaine visite.

Le fonctionnement est simple, dès qu'un utilisateur demande à accéder à une page du site, les éventuels cookies vont également automatiquement être envoyés dans la requête de l'utilisateur. Bien entendu le cookies est dépendant du domaine du site web. Impossible de récupérer le cookie d'un autre domaine.

On évitera toujours de stocker des informations sensibles dans les cookies comme des mots de passe par exemple car les cookies sont stockés sur l'ordinateur des visiteurs et nous n'avons donc aucune maîtrise ni aucun moyen de les sécuriser facilement après le stockage.

### **Créer un cookie en PHP**

Pour créer un cookie en PHP, nous allons utiliser la fonction `setcookie()`.

les cookies doivent être envoyés avant toute autre sortie (c'est une restriction du protocole HTTP, pas de PHP). Cela vous impose d'appeler cette fonction avant toute balise `<html>` ou `<head>` et aussi des caractères d'espacement blanc.

Cette fonction peut accepter jusqu'à sept valeurs en arguments. Cependant, seul la première (le nom du cookie créé) est obligatoire.

La syntaxe de base de `setcookie()` est la suivante

`setcookie(name, value, expire, path, domain, secure, httponly).`



- **Name** : Le nom du cookie. Le nom d'un cookie est soumis aux mêmes règles que les noms des variables.
- **Value** : La valeur du cookie. Comme cette valeur est stockée sur l'ordinateur d'un utilisateur.
- **Expires** : La date d'expiration du cookie sous forme d'un timestamp UNIX. Si aucune valeur n'est passée en argument, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé).
- **Path** : Le chemin sur le serveur sur lequel le cookie sera disponible. Si la valeur est '/', le cookie sera disponible sur l'ensemble du domaine. Si la valeur est '/cours/', le cookie ne sera disponible que dans le répertoire (le dossier) /cours/ (et dans tous les sous-répertoires qu'il contient).
- **Domain** : Indique le domaine ou le sous domaine pour lequel le cookie est disponible.
- **Secure** : Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Si la valeur passée est true, le cookie ne sera envoyé que si la connexion est sécurisée.
- **Httponly** : Indique si le cookie ne doit être accessible que par le protocole HTTP. Pour que le cookie ne soit accessible que par le protocole http, on indiquera la valeur true. Cela permet d'interdire l'accès au cookie aux langages de scripts comme le JavaScript par exemple, pour se protéger potentiellement d'une attaque de type XSS \*

\*Cross-Site Scripting : est une vulnérabilité de sécurité web qui permet à un attaquant d'injecter des scripts malveillants dans des pages web consultées par d'autres utilisateurs. Voici comment cela fonctionne :

En pratique :

```
setCookie('PHPSESSID', '1234567890');
setCookie('OTHER', 'blabla', time() + 3600, "", "", true, true);
```

Dans la console de l'explorateur on retrouve bien nos cookies :

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
ONE	1234567890	127.0.0.1	/fapme-cours/X...	Session	13		
OTHER	blabla	127.0.0.1	/fapme-cours/X...	2025-01-09T15:...	11	✓	✓

## Récupérer la valeur d'un cookie

Pour récupérer la valeur d'un cookie, nous allons utiliser la variable superglobale `$_COOKIE`.

Cette superglobale est un tableau associatif qui utilise les noms des cookies en clefs et associe leurs valeurs en valeurs du tableau.

On va donc pouvoir accéder à la valeur d'un cookie en particulier en renseignant le nom du cookie en clef de ce tableau.

```
setCookie('PHPSESSID', '1234567890');

if(isset($_COOKIE['PHPSESSID'])) {
    echo "Votre session est: ".$_COOKIE['PHPSESSID'];
}
```

Retournera : Votre session est: 1234567890

## Modifier la valeur d'un cookie ou supprimer un cookie

De façon triviale, pour modifier la valeur d'un cookie, nous allons appeler à nouveau la fonction `setcookie()` en lui passant le nom du cookie dont on souhaite changer la valeur et changer l'argument de type valeur passé à la fonction avec la nouvelle valeur souhaitée.

Pour supprimer un cookie, nous allons encore appeler `setcookie()` en lui passant le nom du cookie qu'on souhaite supprimer et allons cette fois-ci définir une date d'expiration se situant dans le passé pour le cookie en question.

```
setCookie('PHPSESSID', '1234567890');
setCookie('PHPSESSID', '0987654321');
setCookie('PHPSESSID', '1234567890', time() - 3600);
if(isset($_COOKIE['PHPSESSID'])) {
    echo "Votre session est: ".$_COOKIE['PHPSESSID'];
}
else{echo "Votre session n'existe pas";}
```

Retourne : Votre session n'existe pas

## Sessions

### C'est quoi une sessions

Une session en PHP correspond à une façon de stocker des données différentes pour chaque utilisateur en utilisant un identifiant de session unique.

Les identifiants de session vont généralement être envoyés au navigateur via des cookies de session et vont être utilisés pour récupérer les données existantes de la session.

Un des grands intérêts des sessions est qu'on va pouvoir **conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre**. De plus, les informations de session ne vont cette fois-ci pas être stockées sur les ordinateurs de vos visiteurs à la différence des cookies mais plutôt **côté serveur** ce qui fait que les sessions vont pouvoir être beaucoup plus sûres que les cookies. Il est également possible de stocker énormément de valeurs dans une session.

Notez toutefois que le but des sessions n'est pas de conserver des informations indéfiniment mais simplement durant une « session ». Une session démarre dès que la fonction `session_start()` est appelée et se termine en général dès que la fenêtre courante du navigateur est fermée (à moins qu'on appelle une fonction pour terminer la session de manière anticipée ou qu'un cookie de session avec une durée de vie plus longues ait été défini).

La superglobale `$_SESSION` est un tableau associatif qui va contenir toutes les données de session une fois la session démarrée.

### Démarrer une session en PHP

Pour pouvoir utiliser les variables de session, il va avant tout falloir qu'une session soit démarrée à un moment ou à un autre.

Pour démarrer une session en PHP, on va utiliser la fonction `session_start()`. Cette fonction va se charger de vérifier si une session a déjà été démarrée en recherchant la présence d'un identifiant de session et, si ce n'est pas le cas, va démarrer une nouvelle session et générer un identifiant de session unique pour un utilisateur.

Il va falloir appeler `session_start()` avant toute autre opération dans nos pages, c'est-à-dire au début de celles-ci de la même façon qu'on a déjà pu le faire avec la fonction `setcookie()`.

Par ailleurs, notez qu'il va falloir appeler `session_start()` dans chaque page où on souhaite pouvoir accéder aux variables de session. En pratique, on créera généralement une page `header.php` qui va

contenir notre fonction `session_start()` et qu'on va inclure à l'aide de `include` ou `require` dans les pages voulues d'un site.

Lorsqu'une session est démarrée, c'est-à-dire lorsqu'un utilisateur qui ne possède pas encore d'identifiant de session demande à accéder à une page contenant `session_start()`,

cette fonction va générer un identifiant de session unique qui va généralement être envoyé au navigateur sous forme de cookie sous le nom `PHPSESSID`.

Pour être tout à fait précis, le PHP supporte deux méthodes pour garder la trace des sessions : via des cookies ou via l'URL. Si les cookies sont activés, le PHP va préférer leur utilisation. C'est le comportement recommandé. Dans le cas contraire, les informations de session vont être passées via l'URL.

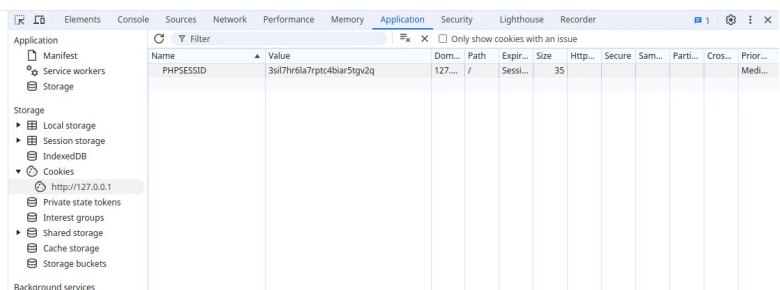
```
session_start();

$id = session_id();

if ($id) echo "id de session (session_id) : " . $id;
if ($cookie = $_COOKIE['PHPSESSID']) echo "<br>id de session (cookie) : " . $cookie;
```

retournera :

```
id de session (session_id)
:3sil7hr6la7rptc4biar5tg2q
id de session (cookie)
:3sil7hr6la7rptc4biar5tg2q
```



The screenshot shows the Chrome DevTools Application tab. On the left, the 'Cookies' section is expanded, showing a cookie for the domain 'http://127.0.0.1'. The main table displays the details of this cookie:

Name	Value	Domain	Path	Expiration	Size	HttpOnly	Secure	SameSite	Partitioned	CrossSite	Prioritized
PHPSESSID	3sil7hr6la7rptc4biar5tg2q	127...	/	Sessi...	35						

## Définir et récupérer des variables de session

Pour définir et récupérer les valeurs des variables de session, nous allons pouvoir utiliser la variable superglobale `$_SESSION`.

Cette superglobale est un tableau associatif qui stocke les différentes variables de sessions avec leurs noms en index du tableau et leurs valeurs en valeurs du tableau.

Nous allons donc très simplement pouvoir à la fois définir de nouvelles variables de session et modifier ou récupérer les valeurs de nos variables de session.

```
session_start();

$id = session_id();
$_SESSION['nom'] = 'Dubois';
$_SESSION['prenom'] = 'Pierre';

echo $_SESSION['nom'] . ' ' . $_SESSION['prenom'];
```

Retournera : Dubois Pierre

Maintenant que les variables de session sont définies, elles sont accessible durant la durée de la session à partir de toutes les pages du site pour lesquelles les sessions ont été activées. Simulons cela de la sorte

```
session_start();

$id = session_id();
//$_SESSION['nom'] = 'Dubois';
//$_SESSION['prenom'] = 'Pierre';

echo $_SESSION['nom'] . ' ' . $_SESSION['prenom'];
```

Cela retourne toujours Dubois Pierre jusqu'à la fermeture de la page du navigateur ou la destruction de la session

## **Terminer une session et détruire les variables de session**

Une session PHP se termine automatiquement lorsqu'un utilisateur ferme la fenêtre de son navigateur.

Souvent un bouton du type logout appellera une destruction de la session ou de certaines variables.

Pour faire cela, nous allons pouvoir utiliser la fonction `session_unset()` qui détruit toutes les variables d'une session.

Pour détruire totalement une session, il va également falloir supprimer l'identifiant de session. Cet identifiant est contenu dans le cookie PHPSESSID qu'on pourra effacer en utilisant setcookie() en définissant une date d'expiration passée pour le cookie.

Notez qu'on va également pouvoir utiliser la fonction session\_unset() (sans lui passer d'argument) pour détruire toutes les variables de la session courante. Cette fonction va également nous permettre de détruire une variable de session en particulier en lui passant sa valeur de la manière suivante : unset(\$\_SESSION['nom-de-la-variable-de-session-a-detruire']).

```
echo "<br><br>id de session:" . $id . "<br>";
echo "<br>Nom: " . $_SESSION['nom'] . "<br>Prenom: " .
$_SESSION['prenom'];

unset($_SESSION['nom']);
echo "<br><br>id de session:" . $id . "<br>";
echo "<br>Nom: " . $_SESSION['nom'] . "<br>Prenom: " .
$_SESSION['prenom'];

session_unset();
echo "<br><br>id de session:" . $id . "<br>";
echo "<br>Nom: " . $_SESSION['nom'] . "<br>Prenom: " .
$_SESSION['prenom'];
```

id de session:3sil7hr6la7rptc4biar5tg2q

**Nom: Dubois**

**Prenom: Pierre**

id de session:3sil7hr6la7rptc4biar5tg2q

**Nom:**

Prenom: Pierre

id de session:3sil7hr6la7rptc4biar5tg2q

**Nom:**

**Prenom:**

## **Projet jeu du pendu V0.1**

Utilisons es cookies, les sessions pour stocker nos valeurs.

# Manipuler des fichiers en PHP

Il est souvent utile de manipuler des fichiers. Importer un listing client, ou l'exporter de la base de donnée. Ecrire un fichier de configuration ou stocker des valeurs. Limiter la taille d'une image,...

Il doit y avoir une bonne centaine de fonction en Php pour faire cela. C'est donc un sujet complexe.

Il est possible de manipuler l'ensemble des fichiers, que cela soit des docx, xlsx, odt, pdf, jpg,...

Nous nous limiterons à regarder ce que l'on peut faire avec des fichiers txt.

## Lire un fichier entier

L'une des opérations de base relative aux fichiers en PHP va tout simplement être de lire le contenu d'un fichier entierement.

Il existe 3 fonctions pour cela :

- `file_get_contents()` qui va placer le contenu du fichier dans une variable string
- `file()` qui le placera dans un tableau indexé par ligne
- `readfile()` qui va afficher directement le fichier

```
$contenu = file_get_contents('./file.txt');  
echo $contenu;
```

```
$contenu = file('./file.txt');  
var_dump($contenu);
```

```
readfile('./file.txt');
```

```
"Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme  
"Un code supprimé est un code débogué." - Anonyme  
"N'importe quel idiot peut écrire du code qu'un ordinateur peut comprendre. Les bons programmeurs écrivent du code que les humains peuvent comprendre." - Martin Fowler  
"Le développement informatique, c'est transformer l'impossible en code et les rêves en réalité numérique." - Sokeo  
"Le code est comme l'humour. Quand vous devez l'expliquer, c'est qu'il n'est pas bon." - Cory House  
"Ajouter des personnes à un projet en retard accroît son retard." - Frederick Brooks  
  
array(6) {  
  [0]>>  
    string(79) "Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme  
  "  
  [1]>>  
    string(53) "Un code supprimé est un code débogué." - Anonyme  
  "  
  [2]>>  
    string(172) "N'importe quel idiot peut écrire du code qu'un ordinateur peut comprendre. Les bons programmeurs écrivent du code que les humains peuvent comprendre." - Martin Fowler  
  "  
  [3]>>  
    string(120) "Le développement informatique, c'est transformer l'impossible en code et les rêves en réalité numérique." - Sokeo  
  "  
  [4]>>  
    string(100) "Le code est comme l'humour. Quand vous devez l'expliquer, c'est qu'il n'est pas bon." - Cory House  
  "  
  [5]>>  
    string(87) "Ajouter des personnes à un projet en retard accroît son retard." - Frederick Brooks  
  "  
}  
  
"Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme  
"Un code supprimé est un code débogué." - Anonyme  
"N'importe quel idiot peut écrire du code qu'un ordinateur peut comprendre. Les bons programmeurs écrivent du code que les humains peuvent comprendre." - Martin Fowler  
"Le développement informatique, c'est transformer l'impossible en code et les rêves en réalité numérique." - Sokeo  
"Le code est comme l'humour. Quand vous devez l'expliquer, c'est qu'il n'est pas bon." - Cory House  
"Ajouter des personnes à un projet en retard accroît son retard." - Frederick Brooks
```



## Ouvrir, lire et fermer un fichier en PHP

Précédemment, nous avons vu comment lire un fichier texte entièrement. Cependant, il arrive que l'on ne désire récupérer qu'une partie d'un fichier.

### Ouvrir un fichier en PHP

Pour lire un fichier partie par partie, nous allons avant tout devoir l'ouvrir. Pour ouvrir un fichier en PHP, nous allons utiliser la fonction `fopen()`, abréviation de « file open ».

On va devoir passer le chemin relatif menant à notre fichier ainsi qu'un « **mode** » en arguments à cette fonction qui va alors retourner en cas de succès une ressource de pointeur de fichier, c'est-à-dire pour le dire très simplement une ressource qui va nous permettre de manipuler notre fichier.

Le mode choisi détermine le type d'accès au fichier et donc les opérations qu'on va pouvoir effectuer sur le fichier. On va pouvoir choisir parmi les modes suivants :

- `r` : Ouvre un fichier en lecture seule. Il est impossible de modifier le fichier.
- `r+` : Ouvre un fichier en lecture et en écriture.
- `a` : Ouvre un fichier en écriture seule en conservant les données existantes. Si le fichier n'existe pas, PHP tente de le créer.
- `a+` : Ouvre un fichier en lecture et en écriture en conservant les données existantes. Si le fichier n'existe pas, PHP tente de le créer.
- `w` : Ouvre un fichier en écriture seule. Si le fichier existe, les informations existantes seront supprimées. S'il n'existe pas, crée un fichier.
- `w+` : Ouvre un fichier en lecture et en écriture. Si le fichier existe, les informations existantes seront supprimées. S'il n'existe pas, crée un fichier.
- `x` : Crée un nouveau fichier accessible en écriture seulement. Retourne `false` et une erreur si le fichier existe déjà.
- `x+` : Crée un nouveau fichier accessible en lecture et en écriture. Retourne `false` et une erreur si le fichier existe déjà.
- `c` : Ouvre un fichier pour écriture seulement. Si le fichier n'existe pas, il sera créé. Si il existe, les informations seront conservées.
- `c+` : Ouvre un fichier pour lecture et écriture. Si le fichier n'existe pas, il sera créé. Si il existe, les informations seront conservées.

Comme vous pouvez le constater, le choix du mode influe fortement sur les opérations que nous allons pouvoir réaliser sur le fichier en question.

Par exemple lorsqu'on ouvre un fichier en lecture seulement toute modification de ce fichier est impossible, ce qui n'est pas le cas si on choisit un mode permettant l'écriture dans ce fichier.

Notez par ailleurs qu'on ajoutera généralement la lettre `b` au paramètre « mode » de `fopen()`. Cela permet une meilleure compatibilité et évite les erreurs pour les systèmes qui différencient les fichiers textes et binaires comme Windows par exemple.

## Lire un fichier partie par partie

PHP met à notre disposition plusieurs fonctions qui vont nous permettre de lire un fichier partie par partie :

Les fonctions `fread()`, `fgets()` et `fgetc()` vont nous permettre de lire un fichier partie par partie.

### Lire un fichier jusqu'à un certain point avec `fread()`

La fonction `fread()` (abréviation de « file read ») va prendre en arguments le fichier renvoyé par `fopen()` ainsi qu'un nombre qui correspond aux nombres d'octets du fichier qui doivent être lus.

Pour lire le fichier entièrement, on peut utiliser la fonction `filesize()` en lui passant le nom du fichier qu'on souhaite lire en deuxième argument de `fread()`. En effet, `filesize()` renvoie la taille d'un fichier. Cela va donc nous permettre de lire entièrement un fichier à condition de commencer la lecture au début.

Exemple :

```
$fichierOuvert = fopen('./file.txt', 'rb');  
  
echo fread($fichierOuvert, filesize('./file.txt'));
```

donne :

```
"Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme  
"Un code supprimé est un code débogué." - Anonyme  
etc...
```

On commence par utiliser `fopen()` pour lire notre fichier et on récupère la ressource renvoyée par la fonction dans une variable.

On passe ensuite le contenu de notre variable à `fread()` et on lui demande de lire le fichier jusqu'au bout en lui passant la taille exacte du fichier obtenue avec `filesize()`. Finalement, on affiche le résultat renvoyé par `fread()` grâce à une structure `echo`.

### Lire un fichier ligne par ligne avec `fgets()`

La fonction `fgets()` va nous permettre de lire un fichier ligne par ligne. On va passer le résultat renvoyé par `fopen()` en argument de `fgets()` et à chaque nouvel appel de la fonction, une nouvelle ligne du fichier va pouvoir être lue.

On peut également préciser de manière facultative un nombre en deuxième argument de `fgets()` qui représente un nombre d'octets. La fonction lira alors soit le nombre d'octets précisé, soit jusqu'à la fin du fichier, soit jusqu'à arriver à une nouvelle ligne (le premier des trois cas qui va se présenter). Si aucun nombre n'est précisé, `fgets()` lira jusqu'à la fin de la ligne.

### Lire un fichier caractère par caractère avec `fgetc()`

Dans certains cas, il va également être intéressant de lire un fichier caractère par caractère notamment pour récupérer un caractère en particulier ou pour arrêter la lecture lorsqu'on arrive à un certain caractère. (via une boucle et l'instruction `break`;)

Pour faire cela, nous allons cette fois-ci utiliser la fonction `fgetc()`. Cette fonction va

**s'utilise exactement comme fgets()**, et chaque nouvel appel à la fonction va nous permettre de lire un nouveau caractère de notre fichier. Notez que les espaces sont bien entendus considérés comme des caractères.

## Trouver la fin d'un fichier de taille inconnue

En pratique, les fichiers que l'on lit ont une taille non connues à l'avance. On risque donc de ne pas pouvoir utiliser les fonctions fgets() et fgetc() de manière optimale.

Il existe plusieurs moyens de déterminer la taille ou la fin d'un fichier. La fonction filesize(), par exemple, va lire la taille d'un fichier. Dans le cas présent, cependant, nous cherchons plutôt à déterminer où se situe la fin d'un fichier (ce qui n'est pas forcément équivalent à la taille d'un fichier à cause de la place du curseur, notion que nous allons voir en détail par la suite).

La fonction PHP feof() (« eof » signifie « end of the file » ou « fin du fichier ») va nous permettre de savoir si la fin d'un fichier a été atteinte ou pas. Dès que la fin d'un fichier est atteinte, cette fonction va renvoyer la valeur true. Avant cela, elle renverra la valeur false.

On va donc pouvoir utiliser cette fonction pour boucler dans un fichier de taille inconnue.

```
$fichierOuvert = fopen('./file.txt', 'rb');
while (!feof($fichierOuvert)) {
    $ligne = fgets($fichierOuvert);
    echo "<br>La ligne: " . $ligne . " contient: " . strlen($ligne) . " caractères";
}
```

La ligne: "Les ordinateurs sont rapides ; les programmeurs les gardent lents." -  
Anonyme contient:79 caractères

## La place du curseur interne ou pointeur de fichier

La position du curseur va impacter le résultat de la plupart des manipulations qu'on va pouvoir effectuer sur les fichiers. Il est donc essentiel de toujours savoir où se situe ce pointeur et également de savoir comment le bouger. Le curseur étant l'endroit dans un fichier à partir duquel une opération va être faite.

Ce curseur indique l'emplacement à partir duquel vous allez écrire votre requête ou supprimer un caractère, etc. Le curseur dans les fichiers va être exactement la même chose à la différence qu'ici on ne peut pas le voir visuellement.

Le mode d'ouverture choisi va être la première chose qui va influencer sur la position du pointeur. En effet, selon le mode choisi, le pointeur de fichier va se situer à une place différente et va pouvoir ou ne va pas pouvoir être déplacé :

r / r+ Début du fichier  
a / a+ Fin du fichier  
w / w+ Début du fichier  
x / x+ Début du fichier  
c / c+ Début du fichier

Certaines fonctions vont modifier la place du curseur à chaque exécution. Cela va par exemple être le cas des fonctions `fgets()` et `fgetc()`.

En effet, la première fois qu'on appelle `fgets()` par exemple, le pointeur est généralement au début de notre fichier et c'est donc la première ligne de notre fichier qui est lue par défaut. Cependant, lors du deuxième appel à cette fonction, c'est bien la deuxième ligne de notre fichier qui va être lue.

Pour savoir où se situe notre pointeur de fichier, on peut utiliser la fonction `ftell()` qui renvoie la position courante du pointeur. Nous allons devoir lui passer la valeur renvoyée par `fopen()` pour qu'elle fonctionne correctement.

## Déplacer le curseur manuellement

Pour commencer la lecture d'un fichier à partir d'un certain point, ou pour écrire dans un fichier à partir d'un endroit précis ou pour toute autre manipulation de ce type, nous allons avoir besoin de contrôler la position du curseur. Pour cela, nous allons pouvoir utiliser la fonction `fseek()`.

Cette fonction va prendre en arguments l'information renvoyée par `fopen()` ainsi qu'un nombre correspondant à la nouvelle position en octets du pointeur.

La nouvelle position du pointeur sera par défaut calculée par rapport au début du fichier). Pour modifier ce comportement et faire en sorte que le nombre passé s'ajoute à la position courante du curseur, on peut ajouter la constante `SEEK_CUR` en troisième argument de `fseek()`.

Notez cependant que si vous utilisez les modes `a` et `a+` pour ouvrir votre fichier, utiliser la fonction `fseek()` ne produira aucun effet et votre curseur se placera toujours en fin de fichier.

## Fermer un fichier

Pour fermer un fichier en PHP, nous utiliserons cette fois la fonction `fclose()`. On va une nouvelle fois passer le résultat renvoyé par `fopen()` en argument de cette fonction.

Notez que la fermeture d'un fichier n'est pas strictement obligatoire. Cependant, cela est considéré comme une bonne pratique : cela évite d'user inutilement les ressources de votre serveur.

## Pendu V0.4

Sur base du fichier fourni qui contient un grand nombre de mots en français, je vous propose de jouer avec les différentes notions précédentes pour prendre un mot au hasard. Il existe plusieurs solutions. Comparer le temps d'exécution des divers moyens (L'objet datetime ou un autre est votre ami :-)

## Créer et écrire dans un fichier en PHP

Le mode d'ouverture va conditionner les opérations qu'on va pouvoir effectuer sur le fichier ouvert. La position du pointeur est également très importante dans l'écriture.

### Créer et écrire dans un fichier en PHP : les méthodes disponibles

Il existe différentes façons de créer un fichier et d'écrire dans un fichier déjà existant ou pas et contenant déjà du texte ou pas en PHP.

Les deux façons les plus simples vont être d'utiliser soit la fonction `file_put_contents()`, soit les fonctions `fopen()` et `fwrite()` ensemble.

Notez déjà que l'utilisation des fonctions `fopen()` et `fwrite()` va nous donner plus de contrôle sur notre écriture, en nous permettant de choisir un mode d'ouverture, d'écrire à un endroit du fichier, etc.

### Écrire dans un fichier avec `file_put_contents()`

La fonction `file_put_contents()` va nous permettre d'écrire simplement des données dans un fichier.

Cette fonction va accepter en argument le chemin vers le fichier dans lequel on doit écrire les données, les données à écrire (qui peuvent être une chaîne de caractères ou un tableau) ainsi qu'un drapeau (nous reviendrons là-dessus plus tard).

Si le fichier spécifié dans le chemin du fichier n'existe pas, alors il sera créé. S'il existe, il sera par défaut écrasé, ce qui signifie que ses données seront supprimées.

Vous pouvez déjà noter qu'appeler `file_put_contents()` correspond à appeler successivement les fonctions `fopen()`, `fwrite()` et `fclose()`.

```
file_put_contents('./file2.txt', 'Hello World');
```

Comme indiqué si le fichier existait déjà il aura été écrasé, pour conserver les données de base de notre fichier et lui ajouter de nouvelles données à la suite des données déjà précédentes, on va pouvoir passer le drapeau `FILE_APPEND` en troisième argument de notre fonction `file_put_contents()`.

Un drapeau est une constante qui va correspondre à un nombre. De nombreuses fonctions utilisent des drapeaux différents. Le drapeau ou la constante `FILE_APPEND` va ici nous permettre d'ajouter des données en fin de fichier (c'est-à-dire à la suite du texte déjà existant) plutôt que d'écraser les données déjà existantes.

```
file_put_contents('./fileN.txt', 'Hello');  
file_put_contents('./fileN.txt', ' World', FILE_APPEND);
```

Donnera Hello World

## Créer un fichier PHP avec fopen()

Pour créer un nouveau fichier en PHP (sans forcément écrire dedans), nous allons à nouveau utiliser la fonction `fopen()`. En effet, rappelez-vous que cette fonction va pouvoir créer un fichier si celui-ci n'existe pas à condition qu'on utilise un mode adapté.

Pour créer un fichier avec `fopen()`, nous allons devoir lui passer en arguments un nom de fichier qui sera le nom du fichier créé ainsi qu'un mode d'ouverture adapté. En mentionnant simplement cela, le fichier sera par défaut créé dans le même dossier que notre page PHP.

```
fopen('./fileN.txt', 'c+b');
```

Ici, nous créons un nouveau fichier avec le mode `c+`. Le fichier est donc accessible en lecture et en écriture et si celui-ci existait et contenait déjà des informations, celles-ci ne seraient pas effacées à la différence du mode `w+`.

On pense bien également à rajouter l'option `b` (pour binaire) afin de maximiser la compatibilité pour les différents systèmes.

## Écrire dans un fichier avec fwrite()

Une fois un fichier ouvert ou créé avec `fopen()`, on va pouvoir écrire dedans en utilisant la fonction `fwrite()`.

Cette fonction va prendre la valeur retournée par `fopen()` ainsi que la chaîne de caractères à écrire dans le fichier en arguments.

Si notre fichier est vide, on va très simplement pouvoir écrire du texte dedans :

```
$file = fopen('./file2.txt', 'c+b');  
fwrite($file, "Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme');
```

Dans le cas où le fichier ouvert contient déjà du texte, cela va être un peu plus complexe. En effet, il va déjà falloir se poser la question d'où se situe notre pointeur.

Si on utilise la fonction `fwrite()` plusieurs fois de suite, le texte va être ajouté à la suite car `fwrite()` va déplacer le pointeur après le texte inséré après chaque utilisation.

```
$file = fopen('./file3.txt', 'c+b');  
fwrite($file, "Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme);  
fwrite($file, "Un code supprimé est un code débogué." - Anonyme);
```

```
"Les ordinateurs sont rapides ; les programmeurs les gardent lents." - Anonyme"Un code  
supprimé est un code débogué." - Anonyme
```

## Tester l'existence d'un fichier

PHP met à notre disposition deux fonctions qui vont nous permettre de vérifier si un fichier existe et si un fichier est un véritable fichier.

La fonction `file_exists()` vérifie si un fichier ou si un dossier existe. On va devoir lui passer le chemin du fichier ou du dossier dont on souhaite vérifier l'existence en argument. Si le fichier ou le dossier existe bien, la fonction renverra le booléen `true`. Dans le cas contraire, elle renverra `false`.

La fonction `is_file()` indique si le fichier est un véritable fichier (et non pas un répertoire par exemple). Nous allons également devoir lui passer le chemin du fichier supposé en argument. Si le fichier existe et que c'est bien un fichier régulier, alors `is_file()` renverra le booléen `true`. Dans le cas contraire, elle renverra `false`.

## Effacer un fichier

La fonction `unlink()` permet d'effacer un fichier. On va lui passer le chemin du fichier à effacer en argument. Cette fonction va retourner `true` si le fichier a bien été effacé ou `false` en cas d'erreur.

## Introduction aux permissions des fichiers et au Chmod

```
$file = fopen('./file3.txt', 'c+b');
```

**Warning:** `fopen(./file3.txt): Failed to open stream: Permission denied in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/exercices/syllabus/fichiers/ecriture.php on line 9`

Vous mettez votre site en production et vous avez cette erreur. Le répertoire existe bien mais le fichier `file3.txt` n'est pas créé dedans comme attendu. A n'y rien comprendre si on oublie le haut niveau de sécurité présent sur les serveurs web et donc « linux ». Votre site n'a juste pas les droits en écriture sur le serveur...

Le sujet des permissions liés aux fichiers (et aux dossiers) est un sujet très vaste et complexe et nous n'allons ici que l'aborder en surface et dans les grandes lignes afin que vous puissiez comprendre la relation avec les fichiers en PHP.

Le système Linux (utilisés par la plupart des hébergeurs) définit différents types d'utilisateurs pouvant interagir avec les fichiers (et les dossiers) :

- Le propriétaire ;
- Les membres du groupe ;
- Les autres utilisateurs.

La façon dont est créé le système Linux permet à plusieurs utilisateurs d'avoir accès au système en même temps et va nous permettre de définir des groupes d'utilisateurs.

Cependant, pour que le système fonctionne toujours bien, il a fallu définir différents niveaux de permission d'accès aux différents fichiers pour les différents utilisateurs.

Pour pouvoir manipuler des fichiers (ou le contenu de dossiers), c'est-à-dire effectuer des opérations dessus, nous allons donc avant tout avoir besoin de permissions. Différentes opérations (lecture du fichier, écriture, etc.) vont pouvoir nécessiter différents niveaux de permission.

*« Mais pourtant moi je peux créer un fichier dans ce repertoire ! »*

Oui, mais pas le serveur web... Vous êtes connecté sur le serveur comme un utilisateur nommé « prenom ». Le serveur web « apache », lui n'utilise pas votre compte mais le siens. Habituellement « www-data » c'est son nom d'utilisateur. Il n'a pas le droit en écriture dans les dossiers de « prenom ».

Les permissions d'accès accordées pour chaque groupe d'utilisateurs à un fichier (ou à un dossier) sont symbolisées par 3 chiffres allant de 0 à 7 ou par 3 lettres. Le premier caractère indique les droits accordés au propriétaire du fichier, le deuxième caractère indique les droits accordés au groupe et le troisième caractère indique les droits accordés aux autres utilisateurs.

Exemple :

Pierre est membre du groupe X75-1-A

Marie également

Juliette est membre du groupe X75-1-B

Voici les permissions liées à chaque caractère :

<b>Droit</b>	<b>alphanumérique</b>	<b>octale</b>
---	aucun droit	0
--x	exécution seulement	1
-w-	écriture seulement	2
-wx	écriture et exécution	3
r--	lecture seulement	4
r-x	lecture et exécution	5
rw-	lecture et écriture	6
rwX	tous les droits(lecture, écriture et exécution)	7

Si un fichier ou un dossier possède les permissions 750 par exemple, cela signifie qu'on accorde tous les droits (lecture, écriture et exécution) sur les fichiers contenus dans ce dossier à l'auteur, des droits de lecture et d'exécution aux membres du groupe et des droits de lecture uniquement aux autres utilisateurs.

Donc :

Pierre(propriétaire) à tous les accès, Marie peut lire et exécuter mais pas écrire et Juliette peut juste lire.

Pour notre problème c'est la même chose « prenom » a l'ensemble des droits mais « www-data » n'a par défaut que la lecture.

La solution qui peut sauter aux yeux et de mettre tous les fichiers en 777. Cela corrige en effet la solution définitivement mais n'est vraiment pas une bonne idée. Imaginez qu'un hacker trouve une



faille dans votre code. Il prendra donc le contrôle du serveur web et aura les droits de www-data. Il aura donc le droit en lecture sur l'ensemble des fichiers. Entre autre celui qui contient le mot de passe de la base de donnée...

Il aura également la possibilité de modifier votre code pour spammer, hameçonner,...

La bonne pratique est de créer un répertoire spécifique et de donner les droits nécessaires à www-data dessus. Par la même occasion d'enlever les droits de lecture aux fichiers de config par exemple. (exécution et pas lecture).

## Pour info : Vérifier et modifier les permissions d'un fichier

### Sur le serveur :

En utilisant une ligne de commandes du genre :

`chmod -R 756 ./files` (R signifie récursif, les sous dossiers seront donc impactés.)

Si www-data a les droits d'écriture il pourra à ce moment modifier les droits du répertoire/fichier également.

### Via PHP

Pour vérifier les permissions d'un fichier via un script PHP, c'est-à-dire dynamiquement, on va pouvoir utiliser la fonction PHP **fileperms()** qui renvoie les permissions pour un fichier.

Cette fonction va renvoyer les permissions d'un fichier sous forme numérique. On pourra ensuite convertir le résultat sous forme octale avec la fonction **decoct()** pour obtenir la représentation des permissions selon nos trois chiffres.

Notez que `fileperms()` peut renvoyer des informations supplémentaires selon le système utilisé. Les trois derniers chiffres renvoyés correspondent aux permissions.

On va également pouvoir utiliser les fonctions **is\_readable()** et **is\_writable()** qui vont respectivement déterminer si le fichier peut être lu et si on peut écrire dedans. Ces fonctions vont renvoyer true si c'est le cas ou false dans le cas contraire ce qui les rend très pratiques d'utilisation au sein d'une condition.

Pour modifier les permissions d'un fichier en PHP, on va devoir utiliser la fonction **chmod()** qui va prendre en arguments le fichier dont on souhaite modifier les permissions ainsi que les nouvelles permissions du fichier en notation octale (on placera un zéro devant nos trois chiffres).

Le réglage des permissions concernant les membres du groupe et les autres utilisateurs va être particulier à chaque cas selon votre site et la sensibilité des informations stockées.

## **Pendu V0.5**

Un bug persiste dans le jeux. En effet, si un mot contient un caractère accentué, cette lettre ne sera pas trouvée. Je vous propose de réécrire le fichier en remplaçant les accent par son équivalent sans accent. C'est évidemment pas la meilleure facon de faire pour arranger le bug mais est un exercice amusant.

Php ne propose pas de fonction pour le faire mais vous trouverez facilement des scripts sur internet ou via l'IA.

## **Pendu V1.0**

On reprend le bon fichier et on corrige le bug intelligemment :-). Pour ceux qui ont fini la version V0.5.

The End (ouf...)

## **Et maintenant ?**

Si on faisait une application pour afficher les horaires pour mélanger l'ensemble des concepts et créer un vrai site applicatif. L'idée est simple et se fera par groupe.

Un site avec un accès sécurisé pour les étudiants et formateurs. Après identification, ils voient dans un agenda les cours qui les concernent et peuvent télécharger un fichier pour encodage dans leur agenda.

Un accès admin qui permet d'uploader les pdf's. Qui seront converti dans un fichier ou une base de donnée. Full automatique.