PROGRAMMATION PHP-8 – MYSQL-8
LES BASES DU LANGAGE PHP
L2S – 7-9 AVRIL 2025
3 JOURNEES

http://php.net/manual/fr/langref.php
http://www.w3schools.com/php/
https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql

Prof : Bertrand Liaudet
Supports de cours en ligne : http://bliaudet.free.fr/rubrique.php3?id_rubrique=276

SOMMAIRE

Edition : avril 2025

BASES DU LANGAGE PHP

0. Présentation du document

- Les exemples sont présentés dans un chapitre en vert avec le mot clé : exemple-
- Les dossiers d'exemples sont fournis dans l'article qui contient ce fichier de cours.

  ⇒ http://bliaudet.free.fr/IMG/zip/PHP-01-Bases_du_langage_exemples.zip

  ⇒ Chargez ce fichier et mettez-le dans le dossier « php » du répertoire web « www » du serveur WEB. Vous pouvez aussi structurer les choses avec des dossiers J1, J2, etc. correspondant aux journées de travail.

- Les exercices à faire sont présentés dans un chapitre en jaune avec les mots-clés : TP- ou exercice-

  ⇒ Les sources pour les exercices, quand il y en a, sont fournis dans le dossier des exemples.

- Les chapitres entièrement en jaune correspondent au cours pour L2S
- Les exercices en jaune correspondent au cours pour L2S.

Remarque

- Certains fichiers d'exemple commencent par ces trois lignes :

```
echo '<h1>CODE PHP</h1>';
highlight_file('fichier.php');
echo '<h1>RESULTATS</h1>';
```

- Ce code affiche deux balises h1 avec CODE PHP puis RESULTATS
- La fonction « highlight_file » permet d'afficher le contenu du fichier proposé. Quand on teste le code, on commence par affiche le code. Ça permet de voir le code en même temps que les résultats.
- Pour généraliser le code, on écrit : highlight_file(basename(__FILE__));
- basename(__FILE__) permet de récupérer le nom du fichier en cours de traitement.

Document « slidé »

- Ce document Word est en partie « slidé » : chaque page tient, en général, sur un écran, comme un slide.

Plan général du cours pour ce document :

- Premiers usages « propres » du PHP.
- Bases du PHP (25 « pages jaunes »).

1. Les variables		
3. Tests	exemple 2 et 3	
4. HTML dans le PHP ou séparé du PHP	exemple 4	
5. Boucles	exemple 5	exercice 1
6. Débranchements		
7. Bibliothèque de Fonctions		
8. Tableau numéroté	exemple 6, 7 et 8	exercice 2
9. Tableau associatif	exemple 9	exercice 3, 4
10. Tableau numéroté de tableaux associatifs	exemple 10	
11. Fonctions de manipulation de tableau	exemple 11 et 12	exercice 5
12. Écrire ses propres fonctions	exemple 13, 14 et 15	
14. Technique de base d'organisation du code : include()		

Exemples : sont dans un dossier joint

- exemple-1 : variable dans un calcul
- Exemple-1.b : utilisation de constantes
- exemple-s 2 et 3 : tests
- exemple-4 : HTML dans le PHP ou séparé du PHP
- exemple-5 : Boucles
- exemples 6, 7 et 8 : Tableau numéroté
- exemple-9 : Tableau associatif
- exemple-10 : Tableau numéroté de tableau associatif
- exemples 11 et 12 : Fonctions de manipulation de tableau
- exemple-13 : Exemple d'affichage sans fonction
- exemple-13-a : Fonction d'affichage, qui ne renvoie rien
- exemple-14 : Fonction qui renvoie un résultat
- exemple-15 : Fonction avec un paramètre en sortie : qui est modifié - &
- exemple-16 : Visibilité des variables – global, GLOBALS, static
- exemple-17 : Filtrer un tableau : fonction array_filter
- exemple-18 : algorithme de tri récursif

Exercices

- Récapitulatif des exercices du document :

  - Exercice-1 : 10 premiers entiers, carrés et racines dans un tableau HTML
    ⇒ Objectif : boucle et affichage HTML dans le PHP
  - Exercice-2 : tableau de prénoms et liste à puces
    ⇒ Objectif : tableau numéroté et affichage HTML dans le PHP
  - Exercice-3 : un-user-Etape-0
    ⇒ Objectif : tableau associatif
  - Exercice-4 : tableau périodique des éléments
    ⇒ Objectif : tableau associatif
  - Exercice-5 : tableau-users-Etape-1
    ⇒ Objectif : tableau numéroté de tableaux associatifs (équivalent tableau d'objets)
  - Exercice-6 : tableau-users-Etape-2 : codage avec fonctions
    ⇒ Objectif : codage avec fonctions et regroupement des fonctions.
    ⇒ Approche objet sans POO.
  - Exercice-7 : tableau-users-Etape 3 : tri des données
    ⇒ Objectif : utilisation de fonctions pour trier.

1. Les variables

Définition d'une variable

- Une variable est un moyen pour stocker en mémoire une information le temps de la génération de la page PHP

Une variable a :

- un nom
- une valeur
- un type
- une signification.

Nom d'une variable : $nomVar

- En PHP, le nom d'une variable commence par un $
  - Par exemple : $username = « Bertrand »
    - ⇒ Nom : username
    - ⇒ valeur : « Bertrand »
    - ⇒ type : String
    - ⇒ signification : le nom d'un utilisateur

Les types

Présentation

- Les variables peuvent enregistrer des informations de différents types :
- Entier, décimal (nombre à virgule), texte et booléen (vrai ou faux) sont les principaux.
- http://php.net/manual/fr/language.types.php

Le type entier : int

- 0, 1, 2, 3, etc et les entiers négatifs : -1, -2, etc.

Le type décimal : float

- Ce sont les nombres à virgules : 1.234
- On écrit la virgule avec un point.
- Tant qu'on n'a pas besoin d'une précision extrême, ils conviennent très bien à tous les calculs.

Le type texte (ou chaine de caractères) : string

- Les chaines de caractères sont écrites entre guillemets ou apostrophe.
- On parle aussi de quotte ou simple quotte ou simple guillemet pour les apostrophes, de double quotte pour les guillemets.

Le type booléen : bool

- Peut prendre les valeurs true ou false.

La valeur NULL

- On peut donner la valeur NULL à toutes les variables, quel que soit leur type.
- Cela veut dire que la variable ne contient rien.

Types tableau, objet, ressource

- Il existe aussi un type pour les tableaux (suite du cours), pour les objets (cours POO) et pour les ressources (une ressource est une référence à une ressource externe : voir la doc PHP).

Vérifier le type

- Quand on crée une variable, on n'a pas besoin de lui préciser son type.
- Selon la valeur qu'on donne à la variable, le type est défini automatiquement.
- Des fonctions permettent de savoir de quel type est quelle variable. Par exemple : is_bool($maVariable) retourne vrai (1) si $maVariable est un booléen, faux (0) sinon.

  ⇒ http://php.net/manual/fr/function.is-bool.php

  ⇒ is_bool, is_float, is_numeric, etc.

La signification

- Une variable sert à quelque chose, par exemple à enregistrer le nom de l'utilisateur.

  ⇒ C'est sa signification.
- On lui donne un nom en rapport avec sa signification.

  ⇒ Ce n'est pas obligé, mais c'est préférable.

Que peut-on faire avec une variable ?

Principes

- Une variable commence par un $
- Une variable est typée par défaut, en fonction de ce qu'on met dedans (int, float, string, bool, array, object, NULL)
- On peut récupérer le type avec la fonction gettype()

```
$maVariable = 42;
echo gettype($maVariable); // Affiche "integer"
```

On peut donner une valeur à une variable

```
$username= "Barack";
```

- On parle d'affectation ou d'assignation

On peut afficher le contenu d'une variable

```
echo $username;
```

- Ici pas de guillemets comme quand on affiche un texte.

On peut concaténer une variable à une chaine de caractère

- Concaténation : l'opérateur « . »

- Avec du texte entre « " » ou entre « ' » : on peut concaténer une variable avec un « . » :

```
$username= "Barack";
echo "bonjour" . $username. ". Comment allez-vous ? "
```

- La concaténation ne marche pas pour les booléen. Il faut utiliser la fonction strval() :

```
$a = True;
echo "variable a vaut " . strval($a);
```

Opérateurs de base

Concaténation : opérateur « . »

- Le « . » permet de concaténer deux textes, une variable et du texte, deux variables.

Opérateurs arithmétique

- On utilise les opérateurs classiques : +, -, /, *, %

Opérateur d'incrémentation

- $i++ équivaut à $i=$i+1
- Attention, c'est une post-incrémentation : si on fait « echo $i++ ; c'est le $i avant l'incrémentation qui est affiché.

Fonctions mathématiques

- On peut utiliser les fonction mathématiques classiques : sin(), cos(), sqrt(), pow() : http://php.net/manual/fr/function.pow.php

exemple-1 : on peut utiliser une variable dans un calcul

```
<?php
    $prix_unitaire=11.6;
    $quantite=5;
    $produit="clé USB 32 GO";
    $prix_total=$quantite*$prix_unitaire;
    echo 'bonjour <br>';
    echo $quantite. ' ' .$produit. ' : ' .$prix_total;
?>
```

2. Constantes

- Une constante est une sorte de variable qui ne peut pas être modifiée.
- Son nom est donné en majuscule par convention.
- Sa valeur est donnée par la fonction « define ».
- On y accède sans utiliser le « $ ».

Exemple-1.b : utilisation de constantes

```php
<?php
define("CONSTANT", "Bonjour le monde.");
echo CONSTANT; // affiche "Bonjour le monde."

define('ANIMALS', array(
    'chien',
    'chat',
    'oiseaux'
));
echo ANIMALS[1]; // affiche "chat"

define('ANIMAL', array(
    'chien' => 'Droopy',
    'chat' => 'Felix',
    'canari' => 'Titi',
));
echo ANIMAL['chat']; // affiche "Felix"

?>
```

⇒ On utilise un tableau et un tableau associatif. On verra dans la suite de ce cours ces deux types de tableau.

3. Tests : exemple-s 2 et 3

if, else, elseif

```
if ($maVariable == 0){
   instructions ;
}
elseif ($maVariable >0 {
   instructions ;
}
else {  /* $maVariable<0 */
   instructions ;
}
```

- **if** : http://php.net/manual/fr/control-structures.if.php
- **else** : http://php.net/manual/fr/control-structures.else.php
- **elseif** : http://php.net/manual/fr/control-structures.elseif.php

opérateurs de comparaison

- ==, !=, <, <=, >, >=
- http://php.net/manual/fr/language.operators.comparison.php#language.operators.comparison
- === : vrai si les valeurs sont identiques et de même type, !==

```
echo 5 == '5' ;   // Affiche: 1 (true)
echo 5 === '5' ;  // Affiche:   (false)
```

opérateurs logiques

- binaires : AND, &&, OR, ||

```
$a = 3; // Définir la variable $a

if ($a > 0 and $a < 10) {
    echo 'a est entre 0 et 10';
} else {
    echo 'a n\'est pas entre 0 et 10';
}
```

- unaires : ! : négation

```
$a = 3; // Définir la variable $a

if (is_int($a)) {
    echo 'a est un entier';
} else {
    echo "a n'est pas un entier";
}
```

⇒ http://php.net/manual/fr/language.operators.logical.php

switch

```
switch ($maVariable )
{
  case(0) :
    instructions ;
    break ;
  case (1) :
    instructions ;
    break ;
  default :
    instructions ;
}
```

⇒ http://php.net/manual/fr/control-structures.switch.php

Opérateur de comparaison ternaire : ? :

```
$maVariable = 5;
$monResultat = ($maVariable == 0) ? 'maVariable vaut 0' :
'maVariable est différente de 0';
echo $monResultat;
```

```
echo (1 == '1' ? 'true' : 'false');  // Affiche "true"
echo (1 === '1' ? 'true' : 'false'); // Affiche "false"
```

⇒ http://php.net/manual/fr/language.operators.comparison.php#language.operators.comparison.ternary

4. HTML dans le PHP ou séparé du PHP : exemple-4

HTML dans le PHP : ce qu'on ne fait jamais plus !

- Cette solution réunit le code HTML et le code PHP.
- On peut mettre les balises HTML dans l'echo et les variables dans l'echo.
- Il faut utiliser des \' ou des guillemets pour gérer les apostrophes dans le texte.

```php
<?php
  if ($variable <0 ) {
    echo '<strong> La valeur : </strong>'.$variable.' est négative ! <br>';
    echo 'C\'est une façon de faire<br>';
    echo "C'en est une autre";
  }
?>
```

⇒ La coloration syntaxique n'est pas pratique : on ne voit pas le HTML.

⇒ C'est verbeux : il faut ajouter des echo

HTML séparé du PHP : ce qu'il faut faire tout le temps !

- Cette solution sépare le code HTML du code PHP.

```php
<?php if ($variable <0 ) { ?>
  <strong>La valeur : </strong> <?php echo $variable ?> est négative !
  C'est une façon de faire<br>
  C'en est une autre
<?php } ?>
```

⇒ C'est plus adapté car la coloration syntaxique est plus claire.

⇒ De plus, ça se rapproche des langages de template comme Blade de Laravel.

- On n'a plus qu'un seul bloc : le bloc « if »
- Et trois lignes qui correspondent à 3 lignes HTML (ça pourrait être une balise <p> à chaque fois).

5. Boucles : exemple-5

While

⇒ http://php.net/manual/fr/control-structures.while.php

Présentation

```
while ($cpt < 10){
  instructions ;
}
```

- tant que maVariable <10, on répète les instructions.
- Il faut bien sûr que maVariable soit modifiée dans les instructions pur qu'on puisse sortir de la boucle.

Boucler 10 fois

- Pour boucler 10 fois on peut écrire

```
$cpt =1
while ($cpt <= 10){
  instructions ;
  $cpt ++ ;
}
```

- Le ++ permet d'incrémenter $cpt.
- $cpt ++ ; est équivalent à $cpt =$cpt +1 ;

for

- La boucle for est l'équivalent de la boucle while pour boucler 10 fois (ou autant de fois qu'on veut)

⇒ http://php.net/manual/fr/control-structures.for.php

```
for ($cpt =1 ; $cpt < 1 ; $cpt++){
          instructions ;
}
```

- Il y a trois parties dans le for :
- L'initialisation de maVariable est dans le for, en premier.
- La condition de sortie est au milieu.
- L'incrémentation de maVariable est en troisième position.

Exercice-1 : 10 premiers entiers, carrés et racines dans un tableau HTML

- Écrivez un script qui affiche les 10 premiers entiers, le carré et leur racine carré dans un tableau.
- Cherchez la fonction racine carré.

  ⇒ Résultat attendu :

Nombre	Carré	Racine
1	1	1
2	4	1.41421356237
3	9	1.73205080757
4	16	2

etc.

- On fera une version avec 2 chiffres après la virgule pour la racine.

6. Débranchements

Présentation

- 2 instructions permettent de quitter le fonctionnement standard des boucles : break qui fait quitter la boucle et continue qui permet de passer au suivant.

break

⇒ http://php.net/manual/fr/control-structures.break.php

```
while ($maVariable < 10){
  if($casParticulierQuiFaitSortir ==true){
            ce qu'il y a à faire
            break; //
          }
          cas général
}
```

continue

⇒ http://php.net/manual/fr/control-structures.continue.php

```
while ($maVariable < 10){
  if($casParticulierQuiFaitPasserAuSuivant ==true){
            ce qu'il y a à faire
            continue; // (on passe au suivant)
          }
          cas général
}
```

goto

- Le goto permet d'aller de n'importe où vers n'importe où !
- C'est à éviter !
- http://php.net/manual/fr/control-structures.goto.php

7. Bibliothèque de Fonctions

Présentation

- Il existe des milliers de fonctions qu'on peut utiliser en PHP.
- Certaines sont simples et courantes (mathématiques, traitement de string, traitement de date).
- D'autres sont complexes : envoyer des mails, générer des pdfs, générer des images, traiter des expressions régulières.
- On va surtout s'intéresser aux usages élémentaires. Pour le reste, on cherche en fonction du besoin.

  ⇒ http://php.net/manual/fr/funcref.php

Fonctions de calcul mathématique

- **sqrt** : racine carré, **pow** : puissance, **round** : arrondi, **rand** : aléatoire,
- **min**, **max**, **cos**, **sin**, etc.

  ⇒ http://php.net/manual/fr/book.math.php

Fonctions de traitement de chaine de caractères

- **length**, **substr**, **strpos**, **str_replace** : longueur, extraction, position, remplacement.
- **trim**, **ltrim** : pour supprimer les espaces au début ou à la fin d'une chaîne.
- **chr**, **ord** : passer d'un caractère à un nombre

  ⇒ http://php.net/manual/fr/ref.strings.php

Fonctions de traitement de date

- getdate() pour récupérer la date et heure du jour :

  ⇒ http://php.net/manual/fr/ref.datetime.php

  ⇒ https://www.php.net/manual/fr/function.getdate.php

Envoi de mail

⇒ http://php.net/manual/fr/function.mail.php

Générer des PDF en PHP

⇒ http://php.net/manual/fr/book.pdf.php

Générer des images en PHP

⇒ http://php.net/manual/fr/book.image.php

⇒ OCR

Traiter des expressions régulières en PHP

Généralités sur les expressions régulières

⇒ https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/memento-des-expressions-regulieres

Expression régulière en PHP

⇒ http://php.net/manual/fr/book.pcre.php

⇒ OCR_1 et OCR_2

8. Tableau numéroté – exemple-s 6, 7 et 8

Présentation

- Tableau « classique » : permet de mettre plusieurs valeurs d'un même type dans une même variable.

Fonction array : créer le tableau

- $prenoms = array ('Aurélien', 'Isabelle', 'Ahmed', 'Olivier' , 'Nour', 'Chang') ;
- // crée $prenoms[0], $prenoms[1] jusqu'à $prenoms[5]

Créer le tableau par les indices

```
$lesPrenoms[0]=' Aurélien' ;
$lesPrenoms[1]=' Isabelle';
$lesPrenoms[2]=' Ahmed' ;
```

⇒ Notez que le premier élément du tableau est à l'indice 0.

⇒ L'indice est aussi appelé clé.

Créer le tableau par les indices automatiques

```
$lesPrenoms [ ]=' Aurélien' ; // crée $prenoms[0]
$lesPrenoms [ ]=' Isabelle';  // crée $prenoms[1]
$lesPrenoms [ ]=' Ahmed' ;    // crée $prenoms[2]
```

⇒ Quand on utilise les crochets vides, le nouvel élément est placé automatiquement à la suite des précédents.

Accès à une valeur du tableau : par la clé

- Pour accéder à un élément du tableau on écrit : $tableau[indice]
- Par exemple : $lesPrenoms[1] ;

afficher tout le tableau

```
print_r($prenoms); // affichage « moche » pour débug
```

afficher un élément du tableau

```
echo $prenoms[2] ;
```

Boucle for : afficher tout le tableau

```php
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';

   for ($i = 0 ; $i <count($lesPrenoms) ; $i++) {
      echo 'lesPrenoms['.$i.'] = ' .$lesPrenoms[$i].  '<br/>' ;
   }
?>
```

⇒ Notez la présence de la fonction count pour avoir le nombre d'éléments du tableau.

⇒ Attention : avec une boucle for, il ne doit pas y avoir de trous dans le tableau

```php
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   // le for n'affichera rien pour $lesPrenoms[3]
   // et  n'affichera pas $lesPrenoms[4]
   for ($i=0 ; $i <count($lesPrenoms) ; $i++) {
      echo 'lesPrenoms['.$i.'] = ' .$lesPrenoms[$i].  '<br/>' ;
   }
?>
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   echo '<h2>affichage foreach $key => $value</h2>';
   foreach ($lesPrenoms as $key => $value){
      echo '$key : '.$key. ' => ' ;
      echo '$value : '.$value. '<br/>' ;
   }
?>
```

⇒ **Notice**: Undefined offset: 3 in **C:\laragon\www\php\test.php** on line **10**
lesPrenoms[3] =

boucle foreach : afficher tout le tableau, quel que soit l'indice

- Pour passer en revue tous les éléments du tableau, quel que soit leur numéro dans le tableau, on utilise le foreach.
- La boucle foreach gère automatiquement le fait de démarrer au premier élément du tableau et d'aller jusqu'au dernier en sautant les trous.
- Dans les paramètres du foreach, on précise le tableau et le nom de chaque élément qu'on va récupérer, séparés par un « as ».

```php
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   foreach ($lesPrenoms as $unPrenom){
      echo $unPrenom. '<br>' ;
   }
?>
```

boucle foreach $key => $value

- Chaque élément d'un tableau correspond à un couple (key, value) :
  - ⇒ La clé est le numéro de l'élément dans le tableau, la value sa valeur.
  - ⇒ On écrit : $key => $value

```php
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Bérénice';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   echo '<h2>affichage foreach $key => $value</h2>';
   foreach ($lesPrenoms as $key => $value){
      echo '$key : '.$key. ' => ' ;
      echo '$value : '.$value. '<br>' ;
   }
?>
```

  - ⇒ $key : 0 => $value : Aurélien
  - ⇒ $key : 1 => $value : Bérénice
  - ⇒ Etc.

Fonction print-r() : affichage basique du tableau sans mise en forme

- Affichage sur une seule ligne et en une seule ligne :

```
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   print_r($lesPrenoms);
?>
```

⇒ Array ([0] => Aurélien [1] => Olivier [2] => Ahmed [5] => Hang)

- Affichage ligne par ligne : avec la balise <pre>

```
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   echo '<pre>'; print_r($lesPrenoms); echo '</pre>';
?>
```

⇒ Array
(
[0] => Aurélien
[1] => Olivier
[2] => Ahmed
[5] => Hang
)

Fonction var_dump() : affichage basique des informations d'une variable

- Affichage sur une seule ligne avec le type en plus :

```
<?php
   $lesPrenoms[0]='Aurélien';
   $lesPrenoms[1]='Olivier';
   $lesPrenoms[2]='Ahmed';
   $lesPrenoms[5]='Hang';

   var_dump($lesPrenoms);
   $i=5;
   var_dump($i);
?>
```

⇒ array(4) { [0]=> string(9) "Aurélien" [1]=> string(7) "Olivier" [2]=> string(5) "Ahmed" [5]=> string(4) "Hang" } int(5)

Chaine vers tableau : implode et explode

Présentation

- explode() : pour convertir une chaine avec une liste de valeurs en tableau numéroté.
  - ⇒ https://www.php.net/manual/en/function.explode.php
  - ⇒ https://www.w3schools.com/php/func_string_explode.asp

- implode() : pour convertir un tableau numéroté en une chaine contenant la liste des valeurs du tableau.
  - ⇒ https://www.php.net/manual/en/function.implode.php
  - ⇒ https://www.w3schools.com/php/func_string_implode.asp

  - On précise aux fonction le séparateur qu'on trouve entre les valeurs dans la chaine.

Exemples

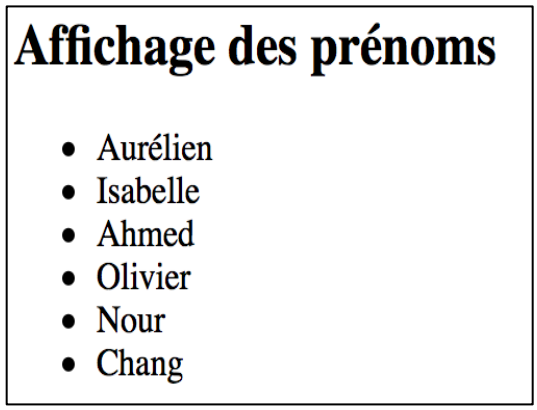
```
$chaine=implode ($sep, $tableau) ;
$chaine=implode (« <br> », $tableau) ;

$tab = explode ($sep, $chaine) ;
$tab = explode (« , », $chaine) ;
```

Exercice-2 : tableau de prénoms et liste à puces

- Écrire un script qui affiche le contenu d'un tableau de prénoms dans une liste à puces. Les prénoms sont écrits en minuscules avec une majuscule en premier.

  Résultat attendu :



- Faites une version qui n'affiche que les prénoms dont la première lettre vient après le H.

  ⇒ Chercher des informations sur la fonction strcmp() dans la documentation PHP.

  ⇒ On utilisera un « continue ».

9. Tableau associatif – exemple-9

Présentation

- Un tableau associatif est un tableau particulier qui permet de mettre plusieurs valeurs de types différents dans une même variable.
- C'est **l'équivalent d'un objet** ou d'une structure dans d'autres langages.
- **A chaque valeur on associe un nom qu'on appelle clé ou attribut**. Cette clé est donc « associé » à la valeur. PHP parle de tableau associatif.

fonction array : créer le tableau :

```
$utilisateur = array (
  'nom' => 'Toto',
  'prenom' => 'Aurélien',
  'dateNaissance' => 1995,
  'nomUtilisateur' => 'aurelien1995'
) ;
```

⇒ 'nom' est une clé qui permet d'accéder à la valeur 'Toto'

Créer le tableau par les indices

```
$utilisateur['nom'] = 'Toto';
$utilisateur['prenom'] = 'Aurélien';
$utilisateur['dateNaissance'] = 1995;
$utilisateur['nomDUtilisateur'] = 'aurelien1995';
```

⇒ 'nom' est une clé qui permet d'accéder à la valeur 'Toto'

Accès à une valeur du tableau : par la clé

- Pour accéder à un élément du tableau on écrit : $tableau['clé']

⇒ Par exemple : $utilisateur['nomUtilisateur'] ;

afficher tout le tableau

```
print_r($utilisateur); // affichage « moche » pour débug
```

afficher un élément du tableau

```
echo $utilisateur['nomDUtilisateur']
```

Afficher tout le tableau associatif : boucle foreach

- Pour passer en revue tous les attributs du tableau associatif, on utilise le foreach.
- Dans les paramètres du foreach, on précise le tableau et le nom de chaque élément qu'on va récupérer, séparés par un « as ».

```php
<?php
   $utilisateur = array (
      'nom' => 'Toto',
      'prenom' => 'Aurélien',
      'dateNaissance' => 1995,
      'nomUtilisateur' => 'aurelien1995'
   );

   foreach ($utilisateur as $value){
      echo $value. '<br/>';
   }
?>
```

boucle foreach $key => $value

- Chaque élément d'un tableau correspond à un couple (key, value) :

  ⇒ La clé est le nom de l'attribut, la value sa valeur.

  ⇒ On écrit : $key => $value

```php
<?php
   $utilisateur = array (
      'nom' => 'Toto',
      'prenom' => 'Aurélien',
      'dateNaissance' => 1995,
      'nomUtilisateur' => 'aurelien1995'
   );

   echo '<h2>affichage foreach as $key => $value</h2>';
   foreach ($utilisateur as $key => $value){
      echo '$key: '.$key. ' => ' ;
      echo '$value: '.$value. '<br/>' ;
   }
?>
```

  ⇒ $key: nom => $value: Toto

  ⇒ $key: prenom => $value: Aurélien

  ⇒ Etc.

Fonction print-r() : affichage basique du tableau associatif sans mise en forme

- Affichage sur une seule ligne

```php
<?php
   $utilisateur = array (
      'nom' => 'Toto',
      'prenom' => 'Aurélien',
      'dateNaissance' => 1995,
      'nomUtilisateur' => 'aurelien1995'
   );

   print_r($utilisateur);
?>
```

⇒ Array ([nom] => Toto [prenom] => Aurélien [dateNaissance] => 1995 [nomUtilisateur] => aurelien1995)

- Affichage ligne par ligne : avec la balise <pre>

```php
<?php
   $utilisateur = array (
      'nom' => 'Toto',
      'prenom' => 'Aurélien',
      'dateNaissance' => 1995,
      'nomUtilisateur' => 'aurelien1995'
   );

   echo '<pre>'; print_r($utilisateur); echo '</pre>';
?>
```

⇒ Array
(
[nom] => Toto
[prenom] => Aurélien
[dateNaissance] => 1995
[nomUtilisateur] => aurelien1995
)

Exercice-3 : un-user-Etape-0

- On veut gérer des utilisateurs avec les caractéristiques suivantes :
  - ⇒ Prénom et NOM (dans un seul champ),
  - ⇒ mail,
  - ⇒ motDePasse,
  - ⇒ age

- Créer un utilisateur (vous !) avec ces informations dans un tableau associatif.
  - Afficher ce tableau associatif avec un print_r avec une ligne par information.
  - Afficher ce tableau associatif dans un tableau HTML.

Exercice-4 : tableau périodique des éléments

En chimie, le tableau périodique des éléments associe un symbole à un nom d'élément chimique :
H pour Hydrogène,
He pour Helium,
etc.
Faites un programme qui affiche au moins les 5 premiers éléments dans un tableau HTML.
Vous pouvez trouver les autres sur internet.

Résultats attendus : vous devez mettre les résultats dans une page HTML.

Symbole	Element
H	Hydrogene
He	Helium
Li	Lithium
Be	Beryllium
B	Bore

10. Tableau numéroté de tableaux associatifs – JSON - exemple-10

Présentation

- On utilise souvent des tableaux numérotés qui contiennent des tableaux associatifs.
- Par exemple, un tableau d'élèves.
- Ca correspond à ce qu'on va récupérer en BD.
- Pour le parcourir, on utilise un foreach et un foreach $key=>$value.

```php
<?php
   // ecriture 1
   $utilisateurs[0] = array ( // c'est $utilisateurs[0]
      'nom' => 'Toto',
      'prenom' => 'Lolo',
      'dateNaissance' => 1995,
      'nomUtilisateur' => 'toto1995'
   ) ;
   // ecriture 2
   $utilisateurs[] = array ( // c'est $utilisateurs[1]
      'nom' => 'Tata',
      'prenom' => 'Lala',
      'dateNaissance' => 1992,
      'nomUtilisateur' => 'tata1992'
   ) ;
   // ecriture 3
   $utilisateurs[2]['nom'] ='Titi';
   $utilisateurs[2]['prenom'] ='Lili';
   $utilisateurs[2]['dateNaissance'] ='1990';
   $utilisateurs[2]['nomUtilisateur'] ='titi1990';

   echo '<h2>affichage print_r</h2>';
   print_r($utilisateurs);

   echo '<h2>affichage foreach</h2>';
   foreach ($utilisateurs as $user){
      echo '<pre>'; print_r($user); echo '</pre>';
   }

    echo '<h2>affichage foreach sindex => $user</h2>';
    foreach ($utilisateurs as $index => $user){
      echo('----------------------------------------<br>');
      echo '<b>$index '.$index. ' : </b><br>';
      foreach ($user as $key => $value){
           echo '$key : '.$key. ' => ';
           echo '$value : '.$value. '<br>';
       }
    }
?>
```

Equivalent JSON

Le tableau numéroté de tableaux associatif PHP est équivalent à du JSON standard :

Tableau de 3 élèves en PHP avec array()

```
$utilisateurs = array (
    array(
        'nom' => 'Toto',
        'prenom' => 'Lolo',
        'dateNaissance' => 1995,
        'nomUtilisateur' => 'toto1995'
    ),
    array(
        'nom' => 'Tata',
        'prenom' => 'Lala',
        'dateNaissance' => 1992,
        'nomUtilisateur' => 'tata1992'
    ),
    array(
        'nom' => 'Titi',
        'prenom' => 'Lili',
        'dateNaissance' => '1990',
        'nomUtilisateur' => 'titi1990'
    )
);
```

Tableau de 3 élèves en PHP sans array()

```
$utilisateurs = [
    [
        'nom' => 'Toto',
        'prenom' => 'Lolo',
        'dateNaissance' => 1995,
        'nomUtilisateur' => 'toto1995'
    ],
    [
        'nom' => 'Tata',
        'prenom' => 'Lala',
        'dateNaissance' => 1992,
        'nomUtilisateur' => 'tata1992'
    ],
    [
        'nom' => 'Titi',
        'prenom' => 'Lili',
        'dateNaissance' => '1990',
        'nomUtilisateur' => 'titi1990'
    ]
];
```

Tableau de 3 élèves en JSON :

```
[
  {
    "nom": "Toto",
    "prenom": "Lolo",
    "dateNaissance": 1995,
    "nomUtilisateur": "toto1995"
  },
  {
    "nom": "Tata",
    "prenom": "Lala",
    "dateNaissance": 1992,
    "nomUtilisateur": "tata1992"
  },
  {
    "nom": "Titi",
    "prenom": "Lili",
    "dateNaissance": "1990",
    "nomUtilisateur": "titi1990"
  }
]
```

11. Fonctions de manipulation de tableau – exemple-s 11 et 12

- Il existe beaucoup de fonction de manipulation de tableau

  ⇒ http://php.net/manual/fr/ref.array.php

- Citons particulièrement :

array

⇒ pour créer un tableau

count - sizeof

⇒ count($monTableau) ou sizeof($monTableau) pour récupérer le nombre d'éléments du tableau.

sort – exemple-11

⇒ sort($monTableau) pour trier un tableau numéroté.

⇒ rsort() pour trier en sens inverse.

⇒ Toutes les fonctions de tri : http://php.net/manual/fr/array.sorting.php

shuffle – exemple-11

⇒ shuffle($monTableau) pour mélanger un tableau.

array-search – exemple-11

⇒ array_search('Isabelle', $lesPrenoms) renvoie l'indice de la valeur 'Isabelle' dans le tableau $lesPrenoms.

ksort

⇒ ksort($monTableau) pour trier un tableau associatif selon la « key ».

⇒ krsort() pour trier en sens inverse.

asort

⇒ asort($monTableau) pour trier un tableau associatif selon la « value». C'est équivalent à sort() pour un tableau numérotée.

⇒ arsort() pour trier en sens inverse. C'est équivalent à rsort() pour un tableau numéroté.

in_array

⇒ in_array('valeur', $monTableau) est vrai si 'valeur' est dans $monTableau.

array_key_exists – exemple-12

⇒ array_key_exists($cle, $monTableauAssociatif) est vrai si $cle est une clé du tableau associatif $monTableauAssociatif.

Exercice-5 : tableau-users-Etape-1

On veut gérer des utilisateurs avec les caractéristiques suivantes :

- Prénom et NOM (dans un seul champ), mail, motDePasse, age

  ⇒ Créer un utilisateur (vous !) avec ces informations dans un tableau associatif.

  ⇒ Afficher ce tableau associatif avec un print_r avec une ligne par information.

On veut créer non plus un seul utilisateur mais un tableau d'utilisateurs.

- Créer ce tableau avec 5 utilisateurs.
- On créera le premier utilisateur, vous, avec la syntaxe suivante :

```
$lesUtilisateurs[0]['nom']='Toto TOTO';
etc.
```

- Et les autres avec cette syntaxe :

```
$lesUtilisateurs[] = array(
    'nom'=>'Toto TOTO',
    etc.
```

⇒ Afficher les utilisateurs créés avec un print_r

⇒ Afficher les utilisateurs avec un for

⇒ Afficher les utilisateurs avec un foreach

⇒ Afficher le tableau des 5 utilisateurs en HTML dans un tableau HTML.

12. Écrire ses propres fonctions – exemples 13, 14, 15

Exemple d'affichage sans fonction– exemple-13

- On veut afficher « Lolo a 25 ans » à partir d'un utilisateur avec son nom, son prénom, son année de naissance et son nom d'utilisateur.

- On structure le code en format MVC
  - D'abord le modèle : on définit les données : l'utilisateur
  - Ensuite le Contrôleur : on fait les calculs : ici l'âge à partir de la date du jour.
  - Enfin la vue : on affiche le résultat souhaité.

```php
<?php
    echo '<h1>CODE PHP</h1>';
    highlight_file(basename(__FILE__));
    echo '<h1>RESULTATS</h1>';

// Modèle
    $utilisateur = array (
        'nom' => 'Toto',
        'prenom' => 'Lolo',
        'anneeNaissance' => 1995,
        'nomUtilisateur' => 'lolo1995'
    );

// Contrôleur
    $today=getdate();
    $age=$today['year']-$utilisateur['anneeNaissance'];

// Vue
    echo '<h2>utilsation de getDate sans fonction</h2>';
    echo $utilisateur['prenom']. ' a ' .$age. ' ans';
?>
```

Fonction d'affichage, qui ne renvoie rien – exemple-13-a

- On va définir une fonction qui fait le calcul et l'affichage.
  - La fonction a deux paramètres en entrée : le prénom et l'année de naissance.
  - Elle calcule l'âge et affiche le résultat souhaité.

- On structure le code en format MVC
  - D'abord le modèle : on définit les données : l'utilisateur
  - Ensuite le Contrôleur : on définit la fonction d'affichage
  - Enfin la vue : on appelle la fonction d'affichage

```php
<?php
    echo '<h1>CODE PHP</h1>';
    highlight_file(basename(__FILE__));
    echo '<h1>RESULTATS</h1>';

// Modèle
    $utilisateur = array (
        'nom' => 'Toto',
        'prenom' => 'Lolo',
        'anneeNaissance' => 1995,
        'nomUtilisateur' => 'lolo1995'
    );

// Contrôleur
    // fonction d'affichage du prénom et de l'âge à partir
    //     du prénom et le l'année de naissance
    // E : prénom, année de naissance
    // S : affichage du résultat
    // Return : rien
    function afficherAge($prenom, $anneeNaissance){
        $today=getdate();
        $age=$today['year']-$anneeNaissance;
        echo $prenom. ' a ' .$age. ' ans' ;
    }

// Vue
    echo '<h2>utilsation de getDate avec fonction</h2>';
    afficherAge($utilisateur['prenom'], $utilisateur['anneeNaissance']);
?>
```

Fonction qui renvoie un résultat – exemple-14

- On reprend l'exemple précédent et on écrit une fonction qui retourne l'âge à partir de l'année de naissance.
  - La fonction a un seul paramètre en entrée : l'année de naissance.
  - Elle calcule l'âge et le retourne.

- On structure le code en format MVC
  - D'abord le modèle : on définit les données : l'utilisateur
  - Ensuite le Contrôleur : on définit la fonction de calcul. Ensuite on fait les calculs : on récupère le prénom et l'âge.
  - Enfin la vue : on affiche le résultat souhaité à l'aide des variables calculées dans le contrôleur.

```php
<?php
    echo '<h1>CODE PHP</h1>';
    highlight_file(basename(__FILE__));
    echo '<h1>RESULTATS</h1>';

// Modèle
    $utilisateur = array (
        'nom' => 'Toto',
        'prenom' => 'Lolo',
        'anneeNaissance' => 1995,
        'nomUtilisateur' => 'lolo1995'
    );

// Contrôleur
    // les fonctions

    // fonction de calcul de l'âge avec l'année de naissance
    // E : année de naissance
    // S : rien
    // Return : age
    function calculerAge($anneeNaissance){
        $today=getdate();
        $year=$today['year'];
        $age=$year-$anneeNaissance;
        return $age;
    }

    // le calcul
    $prenom=$utilisateur['prenom'];
    $age=calculerAge($utilisateur['anneeNaissance']);


// Vue
    echo '<h2>utilsation de getDate avec fonction</h2>';
    echo $prenom. ' a ' .$age. ' ans';
?>
```

Fonction avec un paramètre en sortie : qui est modifié - & - exemple-15

- On va écrire une fonction qui augmente un employé.
- L'employé est un tableau associatif avec son nom et son salaire.
- La fonction reçoit en paramètre l'employé et l'augmentation.

```php
<?php
    echo '<h1>CODE PHP</h1>';
    highlight_file(basename(__FILE__));
    echo '<h1>RESULTATS</h1>';

    // fonction d'augmentation d'un employé
    // E : employé, augmentation
    // S : employé (on modifie son salaire)
    function augmenter(&$employe, $augmentation){
        $employe['salaire']= $employe['salaire']+$augmentation;
    }

    $emp=array('nom'=>'Toto', 'salaire'=>2000);
    echo '<h3>Avant augmentation : </h3>';
    print_r($emp);
    augmenter($emp, 200);
    echo '<h3>Après augmentation : </h3>';
    print_r($emp);
?>
```

⇒ L'employé qui est un paramètre en entrée de la fonction, est modifiée : il faut donc le passer « par adresse » : c'est un vieil usage du langage C !

⇒ Pour passer un paramètre par adresse en PHP, on met un « & » avant le paramètre dans la définition de la fonction.

⇒ Si on ne met pas le « & », la modification n'est pas prise en compte : le salaire reste à 2000.

13. Visibilité des variables – exemple-16 – global, GLOBALS, static

3 niveaux de visibilité pour les variables

http://php.net/manual/fr/language.variables.scope.php

- **Les variables de la page ou variables globales** : elles sont visibles dans la page après leur première apparition, sauf dans les fonctions de la page, sauf si elles sont redéclarées « global » dans les fonctions. Elles ne sont pas visibles dans les autres pages.

  ⇒ A noter que les variables de page sont appelées en général variables globales (globales à la page).

  ⇒ A noter aussi qu'on peut utiliser le tableau associatif $GLOBALS qui contient les couples key-value correspondant aux couples nomDeVariable-valeurDeVariable.

- **Les variables locales** : elles ne sont visibles que dans la fonction où elles sont définies.
- **Les variables « static »** ou encore « locales-globales » : elles ne sont visibles que dans la fonction ou elles sont définies mais elles gardent leur valeur quand on revient dans la fonction.

Circulation de l'information

Circulation entre fonctions et entre page et fonctions

- Pour passer de l'information à une fonction, on la passe en paramètre de la fonction.
- Si une variable veut accéder à une variable de la page, elle doit la déclarer « global » dans la fonction.

```
<?php
   function test1() {
      $varGlobale='test1';
   }

   function test2() {
      global $varGlobale;
      $varGlobale='test2';
   }

   $varGlobale='global';
   test1();
   echo $varGlobale.'<br>'; // Affiche : global
   test2();
   echo $varGlobale.'<br>'; // Affiche : test2
?>
```

Circulation entre pages

- Pour faire circuler de l'information entre pages, on utilise les variables $_GET et $_POST et aussi la variable $_SESSION.
- Cf. chapitre suivant.

14. include() : technique de base d'organisation du code

Include de fichier

- Pour structurer notre code, on va le répartir dans plusieurs fichiers.
- Pour ça, on utilise la fonction « include » qui permet d' « includer » un fichier ce qui veut dire copier son contenu.
- Par exemple :

```
< ?php
  include(unUser.php);//unUser.php contient les fonctions pour unUser
  $user = newUser(…);// newUser() vient de unUser.php
  echo unUserToString($unUser) //unUserToString() vient de unUser.php
```

Variantes technique : include, include_once, require

- Ces 3 fonctions se ressemblent :
  -

⇒ **include** : cherche à inclure. Si ce n'est pas possible, ce n'est pas bloquant.

⇒ **include_once** : doit être utiliser si on risque de faire plusieurs fois le même include : dans ce cas ça plante.

⇒ **require** : include obligatoire ! Si le fichier à includer n'existe pas, ça plante.

Usages de base 1 : pour factoriser le code HTML

- On se sert de l'include pour factoriser le code HTML
-
- Exemple : pour include le header, nav, footer dans toutes nos pages HTML :

```
    <body>
        <header>
            <?php include("header.php");?>
        </header>
        <nav>
            <?php include("header.php");?>
        </nav>
        <main>
            <!-- à définir ->
        </main>
        <footer>
            <?php include("footer.php"); ?>
        </footer>
    </body>
```

Usages de base 2 : pour le MVC

Structure d'un « contrôleur » :

Le contrôleur include les fonctions dont il aura besoin pour traiter les données : c'est le modèle.

Le contrôleur fait les traitements dont il a besoin : ici il charge les produits

Le contrôleur include la vue qui va permettre d'afficher les résultats de ses traitements. Ici les résultats, c'est $lesProduits. La vue, c'est une page HTML qui affichera la variables $lesProduits.

```
<?php
// Modèle : inclusion des fonctions de traitement de données
    include('modele/modelProduits.php');

// Contrôleur : le main
    $lesProduits = modeleGetLesProduits();

// VUE : On affiche la page
    include('vue/vueProduits.php');
```

15. Exercice et dernières fonctionnalités sur les tableaux : filtre et tri par colonne

Exercice-6 : tableau-users-Etape-2 : codage avec fonctions et avec include

- On reprend l'étape 1, mais on va la traiter avec des fichiers séparés et des fonctions.

Étape 1 :

Dans un **nouveau fichier appelé « unUser.php »** on va créer une fonction permettant de créer un utilisateur. On l'appelle « newUser ». Écrivez cette fonction.

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// creation d'un utilisateur
newUser( parametres à déterminer);
```

Dans le fichier « unUser.php », ajoutez une fonction qui permette d'afficher un utilisateur avec un print_r.

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// affichage du tableau avec un print_r
print_rUnUser( parametres à déterminer);
```

Dans le fichier « unUser.php », ajoutez une fonction qui permette de retourner une string avec les infos de tous les champs d'un utilisateur .

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// affichage du tableau avec un print_r
unUserToString( parametres à déterminer);
```

Étape 2 :

Dans un **nouveau fichier appelé « lesUsers.php »,** ajoutez une fonction qui permette de créer un tableau d'utilisateurs. Les utilisateurs sont fournis dans la fonction. On appelle cette fonction initTab.

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// initialisation d'un tableau d'utilisateurs
initLesUsers( parametres à déterminer );
```

Ensuite on se dote d'une fonction qui affiche les utilisateurs avec un print_r.

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// affichage du tableau avec un print_r
print_rLesUsers($lesUtilisateurs);
```

Ensuite on se dote d'une fonction qui affiche avec un foreach.

⇒ Mettez à jour tableau-users-Etape-2.php pour qu'il utilise cette fonction.

```
// affichage du tableau de façon basique avec un foreach
print_rLesUsersForeach($lesUtilisateurs);
```

Étape 3 :

A la place des âges, on entre l'année de naissance. Adapter le programme pour qu'on continue à afficher les âges. On se dote d'une fonction calculerAge($anneeNaissance) et on met à jour la fonction printUnUser(), le tout dans le fichier unUser.php.

Étape 4 :

A la fin du code, ajoutez une page HTML complète qui affiche les utilisateurs dans une **table HTML à bordures.**

Pour finir, mettez ce code dans un fichier séparé qu'on appelera « vue.php » et includez ce fichier.

Exercice-7 : tableau-users-Etape 3 : tri des données

- On veut trier les utilisateurs par nom.
  - ⇒ On utilise la fonction « array_multisort ».
  - ⇒ Son principe est de créer un tableau avec uniquement les noms, puis de faire appel à la fonction « array_multisort » en passant en paramètre le tableau avec les noms, une constante (SORT_ASC pour dire que c'est un tri croissant) et enfin le tableau des utilisateurs.
  - ⇒ https://www.php.net/manual/fr/function.array-multisort

- On écrit une fonction qui fait le tri par nom avec un SORT_DESC
  - ⇒ On passe les Users en paramètre

```
// fonction de tri décroissant par nom des utilisateurs
// E : le tableau $lesUsers
// S : le tableau $lesUsers trié
// Return : rien
function triParNomDesc(&$lesUsers){
   // tri du tableau par nom

   // on commence par fabriquer le tableau des noms
   foreach ($lesUsers as $indice => $unUser){
      $lesNoms[$indice]  = $unUser['nom'];
   }

   // appel à array_multisort : le tableau à trier est en dernier
   array_multisort($lesNoms, SORT_DESC, $lesUsers);
}
```

- Enregistrer cette fonction dans un fichier « tri.php » et tester cette fonction à partir des résultats de l'Etape 2.

- Avec la même méthode, trier par âge décroissant
-
- Avec la même méthode, trier par âge et nom décroissant
-
- Ensuite on va généraliser la fonction de tri pour qu'elle puisse trier selon n'importe quel champ.

exemple-17 - Filtrer un tableau : fonction array_filter

- La fonction « array_filter » permet de filtrer un tableau.
- On lui passe le tableau à filtrer en paramètre ainsi qu'une fonction de filtre qu'on peut écrire comme on veut.
    - ⇒ Cette fonction de filtre a en paramètre un élément du tableau et retourne true si on veut conserver l'élément, false si on ne veut pas le conserver.
    - ⇒ Dans notre exemple, on garde les nombres pairs.

```php
<?php
    echo '<h1>CODE PHP</h1>';
    highlight_file(basename(__FILE__));
    echo '<h1>RESULTATS</h1>';

    $tab=array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    echo '<h2>affichage tableau complet</h2>';
    echo'<pre>';print_r($tab);echo'</pre>';

    function filtreLesPairs($element){
        if($element%2==0) return true;
        else return false;
    }

    $tab_filtre=array_filter($tab, "filtreLesPairs");
    echo '<h2>affichage des pairs</h2>';
    echo'<pre>';print_r($tab_filtre);echo'</pre>';
?>
```

exemple-18 : algorithme de tri récursif

- L'exemple 18 propose une fonction récursive d'affichage indentée un peu complexe !

TP 0 - INSTALLATION

Exercice-0 : Visual Studio Code

- Installation de Visual Studio Code : facile !
- VS Code doit s'utiliser avec des dossiers, pas directement avec des fichiers !
- Il faut charger des packages pour faciliter l'écriture du code. Au minimum :
  - ⇒ French Language Pack For Visual Studio Code
  - ⇒ PHP Intellephense
  - ⇒ Prettier Code Formater
  - ⇒ Prettier SQL VSCode
- Avec ça, les bugs de syntaxe apparaissent en coloration syntaxique.
- On peut formatter le code avec maj-alt-f.

  ⇒ Raccourcis VS Code : http://bliaudet.free.fr/IMG/txt/raccourcis_VS_Code.txt

Exercice-1 : WAMP (MAMP sur Mac)

- Installation de WAMP
- Vérifier la version de PHP : C :> php --version
- Démarrer WAMP
- Vérifier que les serveurs tournent : ctrl-alt-suppression, gestionnaire des tâches : voir 2 apache http server (serveur WEB) et 3 mysqld (serveur de BD).
- Vérifier la version de MySQL : Bouton gauche sur le W, MySQL, console MySQL, ok, entrée, select version() ;

Exercice-2 : Premiers paramétrages de WAMP

- Démarrer WAMP : un W vert apparaît dans la barre des tâches, côté droit (s'il est dans le menu de ^, descendez-le dans la barre des tâches).
- Bouton droit sur le W, paramètres WAMP, affichez le dossier www dans le menu
- Bouton droit sur le W, langue, french
- Bouton droit sur le W, outils, vérifiez que la BD par défaut est MySQL
- Bouton gauche sur le W, ouvrir le répertoire www : créer un dossier home

Exercice-3 : Premiers tests :

- Allez dans le dossier www/home
- Créez un dossier premierTest
- Ouvrez ce dossier avec VS-Code
- Créez un fichier _index.php
- Copiez le code d'exemple des principes du PHP
- Ouvrez un navigateur
- Tapez l'URL : localhost/home
- Cliquez sur le fichier _index.php
- Modifiez le code pour tout mettre en minuscules
- Changez TOTO par Titi et modifiez le code pour tout mettre en majuscules
- Ajoutez une variable prénom et affichez votre prénom et votre nom minuscule avec la première lettre en majuscule avec un Bonjour devant et un point à la fin

Résultats :

Bonjour Bertrand Liaudet.

- prénom: Bertrand
- nom: Liaudet

Exercice-4 : Cyber Sécurité

Exemple 1

- Exemple : http://www.amberieunatation.fr/spip/IMG/jpg/
  - Sur ce site, on constate qu'on peut accéder au contenu du dossier !
  - On peut remonter le dossier : http://www.amberieunatation.fr/spip/IMG
  - Jusqu'à la racine du site : http://www.amberieunatation.fr
  - La racine contient un fichier index.php : c'est la page d'accueil du site qui s'affiche.

- Il n'est pas satisfaisant qu'on puisse accéder à tous les fichiers.
- Pour éviter le problème, on peut par exemple mettre un fichier index.php dans tous les répertoires qui renvoie par exemple sur la page d'accueil (fonction header en PHP).

Exemple 2

- Googlez : index of ".sql"
  - On trouve des liens vers des données de site en tout genre
  - Allez sur ces liens : on accède à un dossier d'un site distant
  - ➔ ces sites ne sont donc pas protégés !

Solutions

- Pour les protéger il y a 2 méthodes :
  - ➔ Interdire l'accès au dossier aux utilisateurs non autorisés
    - C'est la solution propre !
  - ➔ Mettre un fichier index.php dans tous les répertoires qui renvoie par exemple sur la page d'accueil (fonction header en PHP ou balise meta avec http-equiv et url).
    - C'est une solution de secours si on ne contrôle pas le serveur
    - Exemple pour mon site :

```
<meta http-equiv="refresh" content="0;url=
http://bliaudet.free.fr">
```

TP 1 - BASES DU LANGAGE

On réécrit ici les exercices du poly :

Exercice-1 : 10 premiers entiers, carrés et racines dans un tableau HTML

- Ecrire un script qui affiche les 10 premiers entiers, le carré et leur racine carré dans un tableau.
- Chercher la fonction racine carré.
- Résultat attendu :

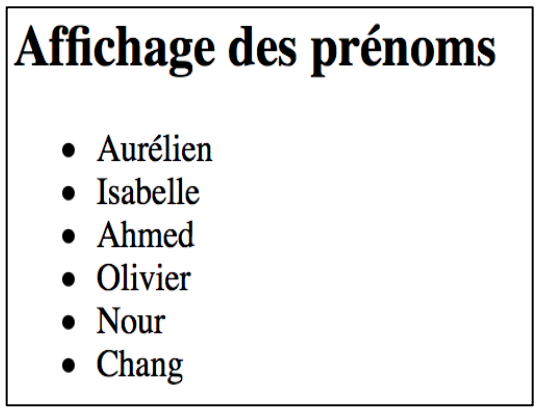
Nombre	Carré	Racine
1	1	1
2	4	1.41421356237
3	9	1.73205080757
4	16	2

etc.

- On fera une version avec 2 chiffres après la virgule pour la racine.

Exercice-2 tableau de prénoms et liste à puces

- Ecrire un script qui affiche le contenu d'un tableau de prénoms dans une liste à puces. Les prénoms sont écrits en minuscules avec une majuscule en premier.
- Résultat attendu :



- Faites une version qui n'affiche que les prénoms dont la première lettre vient après le H.
- Chercher des informations sur la fonction strcmp() dans la documentation PHP.
- On utilisera un « continue ».

Exercice-3 un-user-Etape-0

- On veut gérer des utilisateurs avec les caractéristiques suivantes :
  - ⇒ Prénom et NOM (dans un seul champ),
  - ⇒ mail,
  - ⇒ motDePasse,
  - ⇒ age

- Créer un utilisateur (vous !) avec ces informations dans un tableau associatif.

Afficher ce tableau associatif avec un print_r avec une ligne par information.

- Afficher ce tableau associatif dans un tableau HTML.

Exercice-4 tableau périodique des éléments

- En chimie, le tableau périodique des éléments associe un symbole à un nom d'élément chimique :
  - ⇒ H pour Hydrogène,
  - ⇒ He pour Helium,
  - ⇒ etc.

- Faites un programme qui affiche au moins les 5 premiers éléments dans un tableau HTML.
- Vous pouvez trouver les autres sur internet.

- Résultats attendus : vous devez mettre les résultats dans une page HTML.

Symbole	Element
H	Hydrogene
He	Helium
Li	Lithium
Be	Beryllium
B	Bore

Exercice-5 tableau-users-Etape-1

- On veut gérer des utilisateurs avec les caractéristiques suivantes :
  - Prénom et NOM (dans un seul champ), mail, motDePasse, age
  - Créer un utilisateur (vous !) avec ces informations dans un tableau associatif.
  - Afficher ce tableau associatif avec un print_r avec une ligne par information.

- On veut créer non plus un seul utilisateur mais un tableau d'utilisateurs.
  - Créer ce tableau avec 5 utilisateurs.
  - On créera le premier utilisateur, vous, avec la syntaxe suivante :

```
$lesUtilisateurs[0]['nom']='Toto TOTO';
etc.
```

  - Et les autres avec cette syntaxe :

```
$lesUtilisateurs[] = array(
    'nom'=>'Toto TOTO',
    etc.
```

- Afficher les utilisateurs créés avec un print_r
  - Afficher les utilisateurs avec un for
  - Afficher les utilisateurs avec un foreach
  - Afficher le tableau des 5 utilisateurs en HTML dans un tableau HTML.

- Variante MVC :
  - On fait une version avec uniquement l'affichage dans un tableau HTML et on met cet affichage dans un fichiers HTML séparé avec un include.