LES BASES DU LANGAGE PHP

http://php.net/manual/fr/langref.php

http://www.w3schools.com/php/

Open Class Room

SOMMAIRE

Edition : mars 2018

BASES DU LANGAGE PHP

Installation des fichiers de tests

Les exemples du cours sont dans un fichier zip fournis avec l'article du cours : 00-BASES-DU-LANGAGE.zip.

Les exemples sont présentés dans un chapitre en vert.

Chargez ce fichier et mettez-le dans le dossier Partie_2 du répertoire web « www » du serveur WAMP.

Les exercices à faire sont présentés dans un chapitre en jaune.

REMARQUE :

Tous les fichiers d'exemples 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. Ca 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.

Les variables

Définition

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 et une signification.

En PHP, le nom d'une variable commence par un $

Par exemple : $username est une variable de type string dont la signification sera de contenir une information qui est un nom de l'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 ressource (une ressource est une référence à une ressource externe : voir la doc PHP).

Vérifier le type

Quand on utilise 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

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

Avec du texte entre apostrophes ou entre guillemets, on peut concaténer une variable avec l'opérateur « . »

```
Echo 'bonjour' . $username. '. Comment allez-vous ?'
```

Mieux vaut utiliser les apostrophes « ' » dans le PHP et les guillemets « " » dans le HTML.

On peut afficher le contenu d'une variable dans une chaine de caractère

Avec du texte entre guillemets, on peut mettre la variable directement dans le texte entre guillemets.

```
Echo "bonjour $username. Comment allez-vous ?"
```

On évite cet usage. On utilise le précédent, avec la concaténation

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.

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;
?>
```

On peut utiliser les symboles classiques de calcul : +, - , x, /, % (modulo), et toutes les fonction classiques (sin(), cos(), sqrt(), pow() : http://php.net/manual/fr/function.pow.php

A noter que à gauche du signe =, on modifie la valeur de la variable. Ce qu'il y avait dans la variable avant la modification est perdu. A droite du signe =, on utilise la valeur de la variable pour faire le calcul.

A noter aussi qu'on ne met pas de point autour de la variable dans l'echo, au début de l'echo, au début de la variable, à la fin de la variable à la fin de l'echo.

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 « $ ».

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

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

Tests : exemples 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, !==

opérateurs logiques

binaires : AND, &&, OR, || ($a>0 and $a<10)
unaires : ! (!is_int($a))
http://php.net/manual/fr/language.operators.logical.php

switch

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

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

opérateur de comparaison ternaire : ? :

$monResultat = ($maVarialble == 0) ? true : false ;

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

HTML dans le PHP ou séparé du PHP : exemple 4

HTML dans le PHP

```
<?php
  if ($variable <0 ) {
    echo <strong>variable '.$variable.'</strong> : c\'est négatif
! ';
  }
?>
```

On peut mettre des variables dans l'echo.

Il faut utiliser des \'.

HTML séparé du PHP

```
<?php if ($variable <0 ) { ?>
  <strong>variable</strong> est négatif !
<?php } ?>
```

Cette solution sépare le code HTML du code PHP.

Si on a beaucoup de code HTML, sans variable, c'est plus pratique. On évite les \'

Si on doit ajouter des variables, c'est inadapté

HTML dans le PHP avec balise heredoc : <<<

L'opérateur <<<, qu'on peut appeler balise heredoc, permet d'ouvrir une deux balises _BALISE.

Tout le texte contenu dans la balise est une chaîne de caractère.

Les apostrophes et guillemets peuvent être traités tels quels, sans \'

On peut y mettre directement des variables $variable

La balise fermante _BALISE doit être collée à la marge gauche et être seule avec le ;

```
if ($variable <0 ) {
    echo <<<_BALISE
        <strong>variable $variable</strong> : c'est négatif !
<br>
_BALISE;
```

On peut facilement utiliser des variables et on évite les \'

C'est peu utilisé.

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

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

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

```
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 : 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.

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

http://php.net/manual/fr/control-structures.goto.php

Le goto permet d'aller de n'importe où vers n'importe où !

C'est à éviter !

Bibliothèque de Fonctions

Présentation

Il existe des milliers de fonctions qu'on peut utiliser en PHP.
http://php.net/manual/fr/funcref.php

Fonctions de calcul mathématique

http://php.net/manual/fr/book.math.php
sqrt : racine carré, pow : puissance, round : arrondi, rand : aléatoire,
min, max, cos, sin, etc.

Fonction de traitement de chaine de caractères

http://php.net/manual/fr/ref.strings.php
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
printf, fprintf : print formaté (f de fin), print formaté dans un fichier (f de début)
sscanf : scan formaté (f de fin) dans une string (s de début)

Fonction de traitement de date

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

https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/creer-des-images-en-php

Traiter des expressions régulières en PHP

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

https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-expressions-regulieres-partie-1-2

https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-expressions-regulieres-partie-2-2-2

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

Tableau numéroté et tableau associatif

Tableau numéroté – exemples 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.

Instruction echo : afficher un élément du tableau

echo $prenoms[2] ;

Boucle for : afficher tout le tableau

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

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

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

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

Affichage sur une seule ligne

```
<?php
  $lesPrenoms=array('Aurélien','Isabelle','Ahmed');
  print_r($lesPrenoms);
?>
```

Affichage ligne par ligne : avec la balise <pre>

```
<?php
  $lesPrenoms=array('Aurélien','Isabelle','Ahmed');
  echo $lesPrenoms;
  echo '<pre>'; print_r($lesPrenoms); echo '</pre>';
?>
```

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

Affichage sur une seule ligne avec le type en plus :

```
<?php
  $lesPrenoms=array('Aurélien','Isabelle','Ahmed');
  var_dump($lesPrenoms);
  $i=5;
  var_dump($i);
?>
```

Attention : il ne doit pas y avoir de trous dans le tableau

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

  // ça affiche les 4 prénom
  print_r($lesPrenoms);

  // le for n'affichera rien pour $lesPrenoms[3]
  // et  n'affichera pas $lesPrenoms[4]
  for ($indice = 0 ; $indice <count($lesPrenoms) ; $indice++) {
    echo 'lesPrenoms['.$indice.'] = ' .$lesPrenoms[$indice].
'<br/>' ;
  }
?>
```

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.

Dans la boucle foreach on met le tableau et le nom de la variable qu'on récupère dans le tableau.

Les deux éléments sont séparés par un as.

La boucle foreach gère automatiquement le fait de démarrer au premier élément du tableau et d'aller jusqu'au dernier.

```
<?php
```

```
  $lesPrenoms=array('Aurélien','Isabelle','Ahmed');
  foreach ($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 peut ajouter le $key dans la boucle foreach : ça permet d'accéder à la clé directement.

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

chaine vers tableau : implode et explode

explode pour convertir une chaine avec une liste de valeurs en tableau numéroté

implode pour convertir un tableau numéroté en une chaine contenant la liste des valeurs du tableau.

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

$chaine=implode ($sep, $tableau) ;

$chaine=implode («
 », $tableau) ;

$tab = explode ($sep, $chaine) ;

$tab = explode (« , », $chaine) ;

Exercice – 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 :

Affichage des prénoms

- Aurélien
- Isabelle
- Ahmed
- Olivier
- Nour
- Chang

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 ».

Tableau associatif – exemple 9

Présentation

Un tableau associatif est un tableau particulier qui permet de mettre plusieurs valeurs de type différents dans une même variable.

C'est l'équivalent d'une structure (struct) ou d'un objet dans d'autres langages.

A chaque valeur on associe un nom qu'on appelle clé ou attribut. Ce nom 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'
) ;
```

Créer le tableau par les indices

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

Instruction echo : afficher un élément du tableau

echo $utilisateur[‘nomUtilisateur’] ;

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

Affichage sur une seule ligne

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

  print_r($utilisateur);
?>
```

Affichage ligne par ligne : avec la balise <pre>

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

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

Afficher tout le tableau associatif : boucle foreach

Dans la boucle foreach on met le tableau et le nom de la variable qu'on récupère dans le tableau.

Les deux éléments sont séparés par un as.

La boucle foreach gère automatiquement le fait de démarrer au premier élément du tableau et d'aller jusqu'au dernier.

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

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

boucle foreach $key => $value

Chaque élément d'un tableau associatif, comme pour un tableau numéroté, correspond à un couple (key, value). La key correspond à un attribut, la value à sa valeur.

On peut ajouter le $key dans la boucle foreach : ça permet d'accéder à l'attribut.

```
<?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: '.$key. '<br/>' ;
    echo '$value: '.$value. '<br/>' ;
  }
?>
```

Exercice – tableau-users-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 – 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

Tableau numéroté de tableau associatif – exemple 10

Présentation

On utilise souvent des tableaux numérotés qui contiennent des tableaux associatifs.

Par exemple, un tableau d'élèves.

On utilise un foreach et un foreach $key=>$value.

```
$utilisateurs[] = array (
    'nom' => 'Toto',
    'prenom' => 'Aurélien',
    'dateNaissance' => 1995,
    'nomUtilisateur' => 'aurelien1995'
) ;
$utilisateurs[] = array (
    'nom' => 'Tata',
    'prenom' => 'Bertrand',
    'dateNaissance' => 1990,
    'nomUtilisateur' => 'bertrand1990'
) ;

$utilisateurs[] = ['Toto3', 'Olivier', 1992, 'olivier1992'] ;
$utilisateurs[] = ['Titi4', 'Ahmed', 1990, 'ahmed1990'] ;

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

echo '<h2>affichage foreach skey => $value</h2>';
foreach ($utilisateurs as $index => $user){
    echo '<p><b>$index : '.$index. '</b></p>';
    echo '<pre>';
        print_r($user);
    echo '</pre>';

    foreach ($user as $key => $value){
        echo '<p>$key : '.$key. '<br>';
        echo '$value : '.$value. '</p>';
    }
}
```

Fonctions de manipulation de tableau – exemples 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.

Exemple 11.

Toutes les fonctions de tri :

http://php.net/manual/fr/array.sorting.php

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».

arsort() pour trier en sens inverse.

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.

array-search

array_search('Nour', $lesPrenoms) renvoie l'indice de la valeur 'Nour' dans le tableau $lesPrenoms, soit 4 dans notre exemple (cf. plus haut).

shuffle – exemple 11

shuffle($monTableau) pour mélanger un tableau.

Exercice – tableau-users-Etape-1

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.

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

Créer ce tableau avec 5 utilisateurs.

On créera le premier utilisateur avec la syntaxe suivante :

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

Et les autres avec cette syntaxe :

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

3) Afficher les utilisateurs créés avec un print_r

4) Afficher les utilisateurs avec un for

5) Afficher les utilisateurs avec un foreach

6) Afficher le tableau des 5 utilisateurs en HTML dans un tableau HTML.

Ecrire ses propres fonctions – exemple 13 et 14

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

On veut afficher « Olivier a 20 ans » à partir du prénom et de l'année de naissance

```
<?php
  function afficherAge($prenom, $anneeNaissance){
    $today=getdate();
    $age=$today['year']-$anneeNaissance;
    echo($prenom. ' a ' .$age. ' ans');
  }

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

  afficherAge
    ($utilisateur['prenom'], $utilisateur['anneeNaissance']);
?>
```

La fonction a deux paramètres en entrée : le prénom et l'année de naissance.

Elle calcul l'âge en fonction de l'année de naissance et de l'année de la date du jour.

La fonction getdate permet de récupérer l'information.

Ensuite on utilise cette fonction à partir, dans l'exemple, d'un tableau associatif déjà vu.

Fonction qui renvoie un résultat – exemple 14

On va écrire une fonction qui renvoie l'âge à partir de l'année de naissance

```
<?php
  function calculerAge($anneeNaissance){
    $today=getdate();
    $year=$today['year'];
    $age=$year-$anneeNaissance;
    return $age;
  }

  $anneeNaissance=1990;
  $age=calculerAge($anneeNaissance);
  echo('né en' .$anneeNaissance. ' : ' .$age. ' ans.');
?>
```

La fonction a un paramètre en entrée : l'année de naissance.

Elle calcul l'âge en fonction de l'année de naissance et de l'année de la date du jour.

Elle renvoie le résultat avec le return.

La fonction peut alors être utilisée pour donner une valeur à une variable, par exemple.

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

  function augmenter(&$employe, $augmentation){
    $employe['salaire']= $employe['salaire']+$augmentation;
  }
  $emp=array('nom'=>'Toto', 'salaire'=>2000);
  print_r($emp);echo'<br>';
  augmenter($emp, 100);
  print_r($emp);
?>
```

Dans la fonction, on met un « & » devant $employe : c'est ce qui fait qu'il ressortira modifié.

Le test fait passer de 2000 à 2100. Sans le « & », on reste à 2000.

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

➢ ***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
?>
```

- ***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.

Exercice – tableau-users-Etape-2 : codage avec fonctions

On prend le fichier de correction de Exercice-tableau-users-Etape-1. On le duplique et on l'appelle Etape-2.php

1) Dans un nouveau fichier appelé « unUser.php » on va créer une fonction permettant de créer un utilisateur. On l'appelle « newUser ». Ecrivez cette fonction. Mettez à jour la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

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

2) Dans le fichier « unUser.php », ajoutez une fonction qui permette d'afficher un utilisateur avec un print_r. Mettez à jour la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

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

3) Dans le fichier « unUser.php », ajoutez une fonction qui permette d'afficher un utilisateur champ par champ. Mettez à jour la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

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

4) 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 la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

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

5) Ensuite on se dote d'une fonction qui affiche les utilisateurs avec un print_r. Ajouter l'appel à cette fonction dans la copie du fichier de l'étape1.

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

6) Ensuite on se dote d'une fonction qui affiche avec un for. Mettez à jour la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

```
// affichage du tableau de façon basique avec un for
print_rLesUsersFor ($lesUtilisateurs);
```

7) Ensuite on se dote d'une fonction qui affiche avec un foreach. Mettez à jour la copie du fichier de l'étape1 pour qu'elle utilise cette fonction.

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

8) **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 et met à jour la fonction printUnUser(), le tout dans le fichier unUser.php.

9) Créer une fonction d'affichage qui soit stylé : par exemple on affiche dans un **tableau HTML et on gère un peu de CSS** pour le tableau.

Filtrer un tableau : fonction array_filter – exemple 17

La fonction « array_filter » permet de filtrer un tableau :

```
<?php
  $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>';
?>
```

Explication : on passe un tableau et le nom d'une fonction en paramètre à la fonction « array_filter ».

Ici, la fonction s'appelle « filtreLesPairs ».

On définit cette fonction. Elle a un paramètre qui correspond à un élément du tableau passé en paramètre à array_filter. On peut appeler ce paramètre $element.

On va choisir quel «$element on conserve. Si on retourne true, on conserve le $element dans le tableau, sinon, on ne le garde pas.

Ici, on teste si l'entier est divisible par 2.

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

1) 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.

```
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 résultat de l'Etape 2.

2) Avec la même méthode, trier par âge décroissant

3) Avec la même méthode, trier par âge et nom décroissant

4) Ensuite on va généraliser la fonction de tri pour qu'elle puisse trier selon n'importe quel champ.
5) Enfin on écrit une fonction qui tri selon 2 champs, de façon générale.