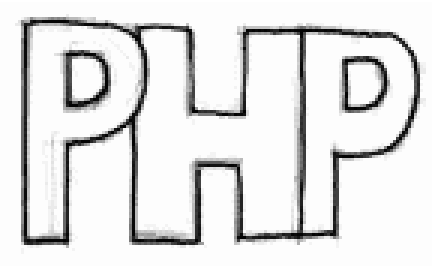
Chapitre

Le langage P.H.P.
Partie 2



III°) Les tableaux

1°) généralités

Un tableau PHP est en fait une association ordonnée (map) à une ou plusieurs dimensions. Ils peuvent se comporter comme :

- des tableaux indicés (des vecteurs)

- des tableaux associatifs (hashes)

Les tableaux sont ordonnés. Vous pouvez modifier l'ordre des valeurs avec de nombreuses fonctions de classement.

2°) Créer un tableau array()

Un tableau peut être créé avec la fonction **array()** ou la fonction **list().** La taille n'est pas définie et n'est pas limitée. Il est possible de donner des valeurs quelconques aux indices des cases, aux contenus des cases.

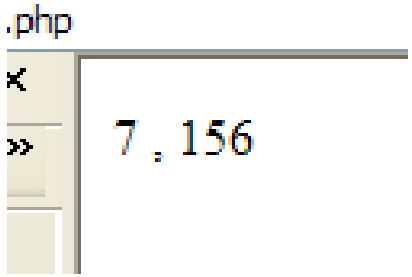
LE LANGAGE P.H.P.

3°) Exemple de tableaux simples

```
<?php   $a = array( 7 , 8 , 0 , 156 , -10 );
        echo ("{$a{0}} , {$a{3}} ");   ?>

     Ou   echo ("$a[0] , $a[3]");
```

```
<?php   $a= array( 0 => 7,
                   1 => 8,
                   2 => 0,
                   3 => 156,
                   4 => -10
                   );
print ("{$a{0}} , {$a{3}}");  ?>
Ou print("a[0] , $a[3]"); ?>
```



NOTE : $tab = array(); // crée un tableau tab vide

LE LANGAGE P.H.P.

4°)Exemple de tableau associatif

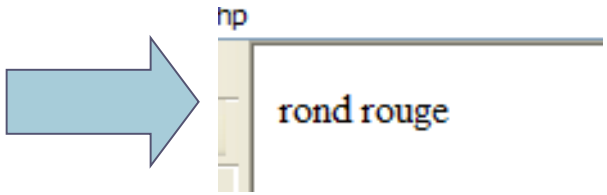
```
<?
$a = array(
        'couleur' => 'rouge' ,
        'gout' => 'sucre' ,
        'forme' => 'rond' ,
        'nom' => 'pomme' ,
    );
```

indice clé valeur

```
print ("{$a{forme}}");
print ("  {$a{couleur}}");
```

Ou

```
print ("$a[forme]");
print ("$a[couleur]");
```

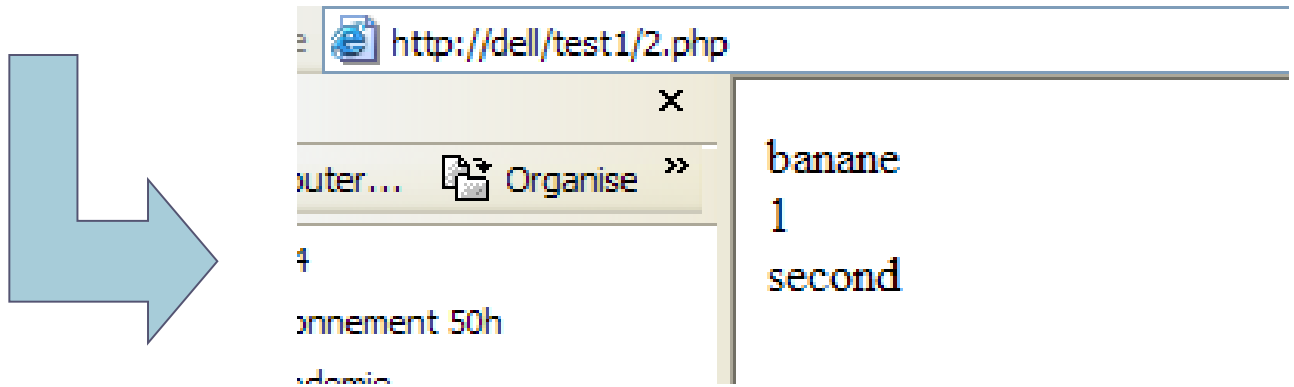



```
<?
$a{'couleur'} = 'rouge';        ou  $a['couleur'] = 'rouge';
$a{'gout'} = 'sucre';           ou  $a['gout']='sucre';
$a{'forme'} = 'rond';           ou  $a['forme']='rond';
$a{'nom'} = 'pomme';            ou  $a['nom']='pomme';
?>
```

LE LANGAGE P.H.P.

5°) Exemples de tableaux à plusieurs dimensions

```
<?php
 $var1 = array (       "fruits"  => array (    "a" => "orange",
                                               "b" => "banane",
                                               "c" => "pomme" )   ,
            "nombres" => array ( 1,2,3,6),
            "trous"   => array ( "premier","second", "troisième")
           );?>
<?php print("{$var1{fruits}{b}}"); ?><br>          ou   <?php print("$var1[fruits][b]"); ?>
<?php print("{$var1{nombres}{0}}"); ?><br>         ou   <?php print("$var1[nombres][0]"); ?>
<?php print("{$var1{trous}{1}}"); ?>               ou   <?php print("$var1[trous][1]"); ?>
```



LE LANGAGE P.H.P.

6°) Les constantes

Une constante est un identifiant (un nom) qui représente une valeur simple. Cette valeur ne peut jamais être modifiée durant l'exécution du script.

Par convention, les constantes sont toujours en **majuscules**.

Un nom de constante valide commence par une lettre ou un souligné (_), suivi d'un nombre quelconque de lettre, chiffres ou soulignés. {a-z, A-Z,0-9,_} et tous les caractères ASCII de 127 à 255.

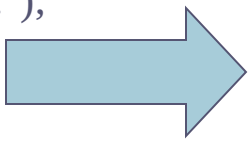
Les constantes sont accessibles de manière globale dans le programme.

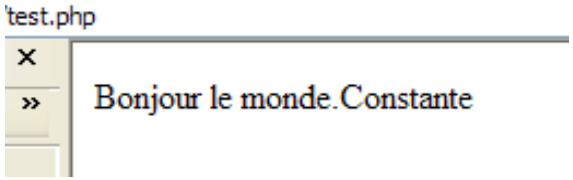
LE LANGAGE P.H.P.

On définit une constante en utilisant la fonction **define()**.

Seuls les types de données scalaires peuvent être placés dans une constante : c'est à dire les types booléen, entier, double et chaîne de caractères (soit **boolean**, **integer**, **double** et **string**).

```
<?php
    define("CONSTANTE", "Bonjour le monde.");
    echo CONSTANTE;
    echo Constante;
 ?>
```





LE LANGAGE P.H.P.

IV°) LES OPERATEURS

1°) opérateurs arithmétiques

Exemple	Nom	Résultat
$a + $b	Addition	Somme de $a et $b.
$a - $b	Soustraction	Différence de $a et $b.
$a * $b	Multiplication	Produit de $a et $b.
$a / $b	Division	Quotient de $a et $b.
$a % $b	Modulo	Reste de $a divisé par $b.

LE LANGAGE P.H.P.

2°) opérateurs de comparaisons

Exemple	Nom	Résultat
$a == $b	Egal	Vrai si $a est égal à $b.
$a === $b	Identique	Vrai si $a est égal à $b et qu'ils sont de même type (PHP 4 seulement).
$a != $b	Différent	Vrai si $a est différent de $b.
$a <> $b	Différent	Vrai si $a est différent de $b.
$a < $b	Plus petit que	Vrai si $a est plus petit strictement que $b.
$a > $b	Plus grand	Vrai si $a est plus grand strictement que $b.
$a <= $b	Inférieur ou égal	Vrai si $a est plus petit ou égal à $b.
$a >= $b	Supérieur ou égal	Vrai si $a est plus grand ou égal à $b.

LE LANGAGE P.H.P.

3°) opérateurs d'incrémentation/décrémentation

Exemple	Nom	Résultat
++$a	Pré-incrémente	Incrémente $a de 1, puis retourne $a.
$a++	Post-incrémente	Retourne $a, puis l'incrémente de 1.
--$a	Pré-décrémente	Décrémente $a de 1, puis retourne $a.
$a--	Post-décrémente	Retourne $a, puis décrémente $a de 1.

LE LANGAGE P.H.P.

4°) opérateurs logiques

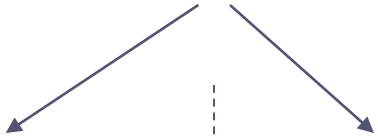
Exemple	Nom	Résultat
$a and $b	ET (And)	Vrai si $a ET $b sont vrais.
$a or $b	OU (Or)	Vrai si $a OU $b est vrai
$a xor $b	XOR (Xor)	Vrai si $a OU $b est vrai, mais pas les deux en même temps.
! $a	NON (Not)	Vrai si $a est faux.
$a && $b	ET (And)	Vrai si $a ET $b sont vrais.
$a || $b	OU (Or)	Vrai si $a OU $b est vrai.

V°) LES INSTRUCTIONS DE CONTROLE

1°) IF, ELSE, ELSEIF

Le principe reste identique au langage C.

```
if  (expression 1)
{// traitement si expression 1 vrai
}
```



```
else
{// traitement si expression 1 fausse
}
```

```
elseif (expression 2)
{// traitement si expression 2 est vrai
}
else
{// traitement si expression 2 est fausse
}
```

LE LANGAGE P.H.P.

Exemples :

```
<?php       if ($a > $b) print "a est plus grand que b";                ?>

<?php
            if ($a > $b)
                        {
                        print "a est plus grand que b";
                        $b = $a;
                        }                                                ?>
  <?php
              if ($a > $b) {
                          print "a est plus grand que b";
                          }
                    else   {
                          print "a est plus petit que b";
                           }                                               ?>
<?php
            if ($a > $b)
                        { print "a est plus grand que b"; }
              elseif ($a == $b)
                        { print "a est égal à b"; }
                        else
                         { print "a est plus petit que b"; }              ?>
```

LE LANGAGE P.H.P.

2°) SWITCH, CASE, BREAK, DEFAULT

Le principe reste identique au langage C. Cette construction est intéressante si on veut comparer une même variable à plusieurs valeurs différentes.

```
switch (variable)
{
        case expression : ….. break;
        case expression 1 : ….break;
        default : …..
}
```

```
<?php
switch ($i)
{
 case 0: print "i égale 0"; break;
 case 1: print "i égale 1"; break;
 case 2: print "i égale 2"; break;
 default: print "i n'est ni égal à 2, ni à 1, ni à 0.";
 }
 ?>
```

LE LANGAGE P.H.P.

3°) WHILE

La boucle while est le moyen le plus simple d'implémenter une boucle en PHP. Cette boucle se comporte de la même manière qu'en C.

```
While  (expression)
{
   …..
}
```

```
<?php
$i = 1;
 while ($i <= 10)
{
 print $i
$i=$i+1;
 }
?>
```

LE LANGAGE P.H.P.

4°) DO…WHILE

Ces boucles ressemblent beaucoup aux boucles while, mais l'expression est testée à la fin de chaque itération plutôt qu'au début. La première itération de la boucle do..while est toujours exécutée

```
do
{
   …..
}
While (expression) ;
```

```
<?php
        $i = 0;
        do {
                print $i;
                $i++;
            }
        while ($i<=10);
?>
```

LE LANGAGE P.H.P.

5°) FOR …

Les boucles for sont les boucles les plus complexes en PHP. Elles fonctionnent comme les boucles for du langage C.

```
for  (exp1;exp2;exp3)
{
  Instructions…..
}
```

La première expression (*expr1*) est évaluée (exécutée), quoi qu'il arrive au début de la boucle.

Au début de chaque itération, l'expression *expr2* est évaluée. Si l'évaluation vaut **TRUE**, la boucle continue et les instructions sont exécutées. Si l'évaluation vaut **FALSE**, l'exécution de la boucle s'arrête.A la fin de chaque itération, l'expression *expr3* est évaluée (exécutée).

```
<?php
for ($i = 1; $i <= 10; $i++)
        {
        print $i;
        }
?>
```

```
<?php    for ($i = 1; $i <= 10; print $i, $i++) ;    ?>
```