

Synthèse de Cours PHP

Edouard Ngo SARR

Enseignant à UCAO- SAINT MICHEL

edouard.sarr@ucao.edu.sn

1. Langage côté client : Le Javascript

Un Langage côté client est traité par la machine qui accueille le logiciel de navigation. Ses résultats peuvent varier en fonction de plate-forme utilisée. Un programme en JavaScript pourra fonctionner sous Netscape et poser problème sous Internet explorer. Les résultats peuvent être différents suivant la machine (PC, Mac). Ils Nécessitent de tests importants et ne permettent pas de masquer les sources du programme

2. Langage côté serveur : Les CGI

Le travail d'interprétation du programme est réalisé par le serveur. Ils Sont des composants exécutables (fichiers .exe ou .dll) qui produisent sur le serveur des contenus html à envoyer aux clients. Les CGI sont compilés. Ils sont rapides mais fortement liés à la plate-forme sur laquelle ils tournent. Ils sont indépendants de la machine et du logiciel de navigation utilisés pour la consultation. **Ils** sont aussi compatibles avec tous les navigateurs et toutes leurs versions. Les sources de ses programmes sont masquées. **Exemple** : PERL, ASP, JSP, PHP et JAVA

3. Pages côté serveur et côté client :

Script côté client pour des calculs et des traitements simples

Scripts côté serveur pour des calculs, des traitements et des mises à jour plus conséquents

4. Le PHP (Hypertext PreProcessor)

PHP est un langage de scripts permettant la création d'applications Web indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client. La syntaxe du langage provient de celles du langage C, du Perl et de Java.

L'intégration nécessite l'utilisation de balises :

`<?php ligne de code PHP ?>`

5. Prérequis

Pour utiliser PHP nous aurons besoin d'installer un serveur PHP dans notre ordinateur. Ainsi il nous suffira juste d'installer EasyPHP ou WAMP sous Windows pour avoir une bonne version de PHP

6. Intégration directe de PHP dans HTML

```
<html>
<head>
  <title> Mon script PHP </title>
</head>
<body>
  <?php
    Echo "BIENVENUE UCAO ceci est mon premier code PHP";
  ?>
</body>
</html>
```

7. Inclure un fichier PHP dans un fichier HTML : include()

Le principe consiste à créer un fichier à part avec que du code PHP et de l'appeler dans notre fichier HTML.

Voici notre fichier PHP.

```
1  <?php
2      Echo "BIENVENUE UCAO ceci est mon premier code PHP à inclure";
3  ?>
```

Voici le fichier appelant

```
<html>
<head>
    <title> Mon script PHP </title>
</head>
<body>
    <?php
        include "bienvenue.php" ;
    ?>
</body>
</html>
```

8. Affichage avec Echo et print

- La fonction echo : echo Expression;
 echo "Chaine de caracteres";
 echo (1+2)*87;
- La fonction print : print(expression);
 print("Chaine de caracteres");
 print ((1+2)*87);

9. Typologie

- Toute instruction se termine par un point-virgule ;
- Pas de sensible à la casse Sauf par rapport aux fonctions

10. Les commentaires

/* Voici un commentaire! */

// un commentaire sur une ligne

11. Les constantes :

Define("nom_constante", valeur_constante)

```
<html>
<body>
    <?php
        define ("x",10);
        define ("y",20);
        /* addition de x
           et y */
        echo 'le resultat est : ';
        echo x+y;
        // fin de addition
    ?>
</body>
</html>
```

12. La concaténation

En PHP pour faire la concatenation on utilise le point .

Exemple :

```
echo ('vous etes : '.$prenom.' '.$nom);
echo ('votre salaire est : '.$salaire);
```

13. Les variables :

Commencent par le caractère \$

N'ont pas besoin d'être déclarées

Exemple :

```
<html>
<body>
    <?php
        $prenom = 'edouard';
        $nom= 'sarr';
        $salaire= 100000;
        echo ('vous etes : '.$prenom.' '.$nom);
        echo ('votre salaire est : '.$salaire);
        $new_salaire= $salaire + (($salaire *10)/100);
        echo ('votre nouveau salaire est: '.$new_salaire);
    ?>
</body>
</html>
```

Exemple avec une constante :

```
<html>
<body>
    <?php
        $prenom = 'edouard';
        $nom= 'sarr';
        $salaire= 100000;
        Define ('taux',0.1);
        echo ('vous etes : '.$prenom.' '.$nom);
        echo ('votre salaire est : '.$salaire);
        $new_salaire= $salaire + ($salaire * taux);
        echo ('votre nouveau salaire est: '.$new_salaire);
    ?>
</body>
</html>
```

Exemple : Faire un programme pour calculer la TVA de 1200 000. Utiliser INCLUDE et DEFINE. Le taux est de 0.18.

Le fichier tva.php

```
<?php
    Define ('Taux', 0.18);
    $MHT= 1200000;
    $MTVA= $MHT*Taux;
    $NetPayer= $MHT + $MTVA;
    Echo ('la Tva sur : '.$MHT . 'est : '.$MTVA );
    Echo ('le net a payer est : '.$NetPayer)
?>
```

Le fichier base.php

```
<html>
<body>
    <?php
        Include 'tva.php';
    ?>
</body>
</html>
```

14. Aller a la ligne

```
echo $nom.'</br>';
echo $prenom.'</br>';
```

Exemple:

```
<?php
    echo "bienvenue".'</br>';
    echo "Edouard sarr"
```

15. Fonctions de vérifications de variables par rapport aux types de base :

- Doubleval() : conversion
- empty() : est null
- gettype() : pour recuperer le type de la variable
- intval() : conversion
- is_array(): est un tableau
- is_bool(): est boolean
- is_double(): est double
- is_float(): est float
- is_int(): est entier
- is_integer: est entier
- is_long(): est entiere long
- is_object(): est un objet
- is_real(): est un reel
- is_numeric(): est numerique
- is_string(): est est chaine
-

16. Affectation par valeur et par référence

- Affectation par valeur :

\$b=\$a
- Affectation par (référence) variable :

\$c = &\$a

Exemple :

```

<?php
    $var1= 100;
    $var2=200;

    echo "var1 =" . $var1 . "</br>";
    echo "var2 =" . $var2 . "</br>";
    echo "*****" . "</br>";
    $res1= $var1;
    $res2= &$var2;
    $res1=110;
    $res2= 500;
    echo "*****" . "</br>";
    echo "res 1 =" . $res1 . "</br>";
    echo "res 2 =" . $res2 . "</br>";
    echo "var1 =" . $var1 . "</br>";
    echo "var2 =" . $var2 . "</br>";
?>

```

Resultat :

```

var1 =100
var2 =200
*****
*****
res 1 =110
res 2 =500
var1 =100
var2 =500

```

17. Visibilité des variables

- Variable locale :

Visible uniquement à l'intérieur d'un contexte d'utilisation

- Variable globale :

Visible dans tout le script

Utilisation de l'instruction global() dans des contextes locales

Exemple:

```

<html>
  <head>
    <title> Mon script PHP </title>
  </head>
  <body>
    <?php
      global $var ;
      $var = 100;
      $varlocale = "bienvenue ucao";
      echo $var;
      echo $varlocale;
    ?>
  </body>
</html>

```

18. Les variables dynamiques

Permettent d'affecter un nom différent à une autre variable

Exemple :

```
$nom_variable = 'PRENOM';
```

```
$$nom_variable = valeur; // équivaut à $PRENOM = valeur;
```

Les variables tableaux sont également capables de supporter les noms dynamiques

Exemple :

```
$nom_variable = "TAB";
```

```
${$nom_variable}[0] = valeur;
```

// équivaut à \$TAB[0] = valeur;

Les accolades servent aussi à éviter toute confusion lors du rendu d'une variable dynamique

Exemple :

```
<?php
    $nvar= 'prenom';
    $$nvar= 'Fatou';
    echo $prenom;
?>
```

19. Variables prédéfinies

Les variables d'environnement dépendant du client

Variable	Description
\$_SERVER["HTTP_HOST"]	Nom d'hôte de la machine du client (associée à l'adresse IP)
\$_SERVER["HTTP_REFERER"]	URL de la page qui a appelé le script PHP
\$_SERVER["HTTP_ACCEPT_LANGUAGE"]	Langue utilisée par le serveur (par défaut en-us)
\$_SERVER["HTTP_ACCEPT"]	Types MIME reconnus par le serveur (séparés par des virgules)
\$_SERVER["CONTENT_TYPE"]	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
\$_SERVER["REMOTE_ADDR"]	L'adresse IP du client appelant le script CGI
\$_SERVER["PHP_SELF"]	Nom du script PHP
\$_SERVER["SERVER_NAME"]	Le nom du serveur
\$_SERVER["HTTP_HOST"]	Nom de domaine du serveur

<code>\$_SERVER["SERVER_ADDR"]</code>	Adresse IP du serveur
<code>\$_SERVER["SERVER_PROTOCOL"]</code>	Nom et version du protocole utilisé pour envoyer la requête au script PHP
<code>\$_SERVER["DATE_GMT"]</code>	Date actuelle au format GMT
<code>\$_SERVER["DATE_LOCAL"]</code>	Date actuelle au format local
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

Exemple :

```

<html>
  <head>
    <title> Mon script PHP </title>
  </head>
  <body>
    <?php
      echo $_SERVER["HTTP_HOST"].'<br>';
      echo $_SERVER["HTTP_REFERER"].'<br>';
      echo $_SERVER["HTTP_ACCEPT_LANGUAGE"].'<br>';
      echo $_SERVER["HTTP_ACCEPT"].'<br>';
      echo $_SERVER["REMOTE_ADDR"].'<br>';
      echo $_SERVER["PHP_SELF"].'<br>';
    ?>
  </body>
</html>

```

Resultat :



```

localhost
http://localhost/coursPHP/
fr-FR;fr;q=0.8,en-US;q=0.6,en;q=0.4
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
127.0.0.1
/coursPHP/page1.php

```

20. Les variables d'environnement

Pour affichage des variables d'environnement on utilise la fonction `phpinfo()`

`echo phpinfo(constante);`

La constante prendra l'une des valeurs lister ci apres :

`INFO_CONFIGURATION` affiche les informations de configuration.

`INFO_CREDITS` affiche les informations sur les auteurs du module PHP

`INFO_ENVIRONMENT` affiche les variables d'environnement.

INFO_GENERAL	affiche les informations sur la version de PHP.
INFO_LICENSE	affiche la licence GNU Public
INFO_MODULES	affiche les informations sur les modules associés à PHP
INFO_VARIABLES	affiche les variables PHP prédéfinies.

Exemple :

```

<html>
<head>
<title> Mon script PHP </title>
</head>
<body>
<?php
    echo phpinfo(INFO_VARIABLES);
?>
</body>
</html>

```

21. Principe sur les types de données

Pas besoin d'affecter un type à une variable avant de l'utiliser. La même variable peut changer de type en cours de script. Les variables issues de l'envoi des données d'un formulaire sont du type string.

22. Les différents types de données

Les entiers : le type Integer

Les flottants : le type Double

Les tableaux : le type array

Les chaînes de caractères : le type string

Les objets

23. Le transtypage

La fonction settype() permet de convertir le type auquel appartient une variable.

La fonction Gettype() retourne le type de la variable.

Exemple :

```

<html>
<head>
<title> Mon script PHP </title>
</head>
<body>
<?php
    $nbre=10;
    Settype($nbre, "double");
    Echo " la variable $nbre est de type " , gettype($nbre);
?>
</body>
</html>

```


24. Transtypage explicite : le cast

On utilise le type entre parenthese :

(int)
(integer)
(real)
(double)
(float)
(string)
(array)
(object)

Exemple:

```
<html>
<head>
<title> Mon script PHP </title>
</head>
<body>
<?php
    $var=" 100 FRF ";
    Echo " pour commencer, le type de la variable est ".'<br>';
    echo gettype($var).'<br>';
    $var =(double)$var;
    Echo "Après le cast, le type de la variable est $var ".'<br>';
    echo gettype($var).'<br>';
    Echo "<br> et a la valeur $var "; ?>
</body>
</html>
```

Resultat:

```
pour commencer, le type de la variable est
string
Après le cast, le type de la variable est 100
double

et a la valeur 100
```

25. Détermination du type de données

On utilise :

- Gettype() pour recuperer le type de la variable
- Is_long() pour tester si une variable est un LONG
- Is_double() pour tester si une variable est un DOUBLE
- Is_string() pour tester si une variable est un STRING
- Is_array() pour tester si une variable est un TABLEAU
- Is_object() pour tester si une variable est un OBJET

- `Is_bool()` pour tester si une variable est un BOOLEEN

26. les chaines de caracteres:

Peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux.

Une chaîne de caractères doit être toujours entourée par des guillemets simples (') ou doubles (")

Exemple :

" Ceci est une chaîne de caractères valide."

'Ceci est une chaîne de caractères valide.'

"Ceci est une chaîne de caractères invalide."

Des caractères spéciaux à insérer directement dans le texte, permettent de créer directement certains effets comme des césures de lignes

Car	Code ASCII	Code hex	Description
<code>\car</code>			échappe un caractère spécifique.
" "	32	0x20	un espace simple.
<code>\t</code>	9	0x09	tabulation horizontale
<code>\n</code>	13	0x0D	nouvelle ligne
<code>\r</code>	10	0x0A	retour à chariot
<code>\0</code>	0	0x00	caractère NUL
<code>\v</code>	11	0x0B	tabulation verticale

27. Les operateurs :

Ils peuvent être :

les opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	<code>\$x+3</code>	10
-	opérateur de soustraction	Soustrait deux valeurs	<code>\$x-3</code>	4
*	opérateur de multiplication	Multiplie deux valeurs	<code>\$x*3</code>	21
/	plus: opérateur de division	Divise deux valeurs	<code>\$x/3</code>	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	<code>\$x=3</code>	Met la valeur 3 dans la variable \$x

les opérateurs d'assignation

Opérateur	Effet
<code>+=</code>	addition deux valeurs et stocke le résultat dans la variable (à gauche)
<code>-=</code>	soustrait deux valeurs et stocke le résultat dans la variable
<code>*=</code>	multiplie deux valeurs et stocke le résultat dans la variable
<code>/=</code>	divise deux valeurs et stocke le résultat dans la variable
<code>%=</code>	donne le reste de la division deux valeurs et stocke le résultat dans la variable
<code> =</code>	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
<code>^=</code>	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
<code>&=</code>	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
<code>.=</code>	Concatène deux chaînes et stocke le résultat dans la variable

les opérateurs d'incrémentation

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentatation	Augmente d'une unité la variable	\$x++	8
--	Décrémentatation	Diminue d'une unité la variable	\$x--	6

les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat
==	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	\$x==3	Retourne 1 si \$X est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	\$x<3	Retourne 1 si \$X est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	\$x<=3	Retourne 1 si \$X est inférieur à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	\$x>3	Retourne 1 si \$X est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	\$x>=3	Retourne 1 si \$X est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	\$x!=3	Retourne 1 si \$X est différent de 3, sinon 0

les opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1) (condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1) && (condition2))
XOR	OU exclusif	Opposé du OU logique	((condition1) XOR (condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	!(condition)

les opérateurs bit-à-bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100)	8 (1000)
	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9 12 (1001 1100)	13 (1101)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)
~	Complément (NON)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~1001)	6 (0110)

les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
<<	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	$6 \ll 1$ (110) $6 \ll 1$	12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que le bit non-nul de poids plus fort est recopié à gauche	$6 \gg 1$ (0110) $6 \gg 1$	3 (0011)

Autres opérateurs

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour". "Au revoir"	"BonjourAu revoir"
\$	Référencement de variable	Permet de définir une variable	\$MaVariable = 2;	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	\$MonObjet->Propriete	

28. Les priorités des opérateurs :

Priorité des opérateurs												
()	[]											
--	++	!	~	-								
*	/	%										
+	-											
<	<=	>=	>									
==	!=											
&												
^												
&&												
?	:											
=	+=	-=	*=	/=	%=	<<=	>>=	>>>=	&=	^=	=	
AND												
XOR												

29. L'instruction if

```

if (condition réalisée) {
    liste d'instructions
}
elseif (autre condition ) {
    autre série d'instructions
}
else (dernière condition réalisée) {
    série d'instructions
}

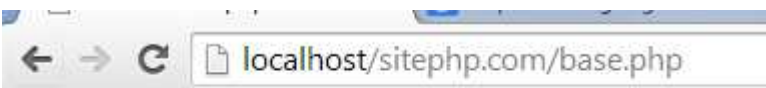
```

Exemple : Ecrire un programme qui compare la salaire d'un employé par 100000. Il affiche la supériorité ou l'infériorité entre les deux valeurs.

```

<?php
$prenom="edouard";
$salaire=230000;
echo 'le salaire de '.$prenom. ' est de ' . $salaire.'

```



le salaire de edouard est de 230000
le salaire de edouard est supérieur À 100000

30. l'instruction switch

```

switch (Variable) {
    case Valeur1: Liste d'instructions break;
    case Valeur1: Liste d'instructions break;
    case Valeurs...: Liste d'instructions break;
    default: Liste d'instructions break;
}

```

Exemple : refaire le même exercice mais cette fois si nous le comparons avec 100000, 200000 et 300000.

```

<?php
$prenom="edouard";
$salaire=230000;
switch ($salaire) {
    case 100000: echo 'le salaire de '.$prenom. ' est 100000'. '<br>';
    case 200000: echo 'le salaire de '.$prenom. ' est 100000'. '<br>';
    case 300000: echo 'le salaire de '.$prenom. ' est 100000'. '<br>';
    default: echo 'le salaire de '.$prenom. ' est de ' . $salaire. '<br>';
}
?>

```

le salaire de edouard est de 230000

31. La boucle for

```

for ($i=1; $i<6; $i++) {
    echo "$i<br>";
}

```

Exemple: Ecrire un programme qui calcule le factoriel de 12.

```

<?php
$valeur=12;
$fact=1;
echo 'la valeur est '.$valeur. '<br>';
for ($i=1; $i<=$valeur; $i++)
{
    $fact=$fact * $i;
}
echo 'le factoriel de '.$valeur. ' est : '.$fact;
?>

```

32. La boucle while

```

While(condition) {
    bloc d'instructions ;
}

```

Exercice: Refaire le même exercice sur les factoriel avec While.

```

<?php
$valeur=12;
$fact=1;
echo 'la valeur est '.$valeur. '<br>';
$i=1;
while ($i<=$valeur)
{
    $fact=$fact * $i;
    $i++;
}
echo 'le factoriel de '.$valeur. ' est : '.$fact;
?>

```

33. La boucle do...while

```
Do {

    bloc d'instructions ;

} while(condition) ;
```

Exercice: Refaire le meme exercice sur les factoriel avec Do While.

```
<?php
$valeur=12;
$fact=1;
echo 'la valeur est '.$valeur.'

```

34. La boucle foreach (PHP4)

```
Foreach ($tableau as $valeur) {
    insts utilisant $valeur ;
}
```

Beaucoup plus utilisé avec les tableaux.

Partie 3 :

LES FONCTIONS

DECLARATION ET APPEL D'UNE FONCTION

```
Function nom_fonction($arg1, $arg2, ...$argn)
{
    //instrcutions
    return $resultat ;
}
```

Exemple :

```
<?php
Function Foncl ($x, $y)
{
    $somme= $x+$y;
    return $somme ;
}
?>
```

Appel d'une fonction

Nous savons qu'une fonction retourne toujours une valeur donc faudra lors de l'appel de fonction declarer une variable pour la recevoir. Exemple :

```
<?php
Function MaFontion($x, $y)
{
    $somme= $x+$y;
    return $somme ;
}
$rep= MaFontion(10,12);
echo $rep;
?>
```

NB : une fonction peut en appeler une autre

Exemple : Calcule de la TVA


```

<?php
Function TVA ($M, $Taux)
{
    $MTVA= $M*$Taux;
    return $MTVA ;
}

Function NetPayer ($MonTva, $Mon)
{
    $net= $MonTva+$Mon;
    Return $net;
}

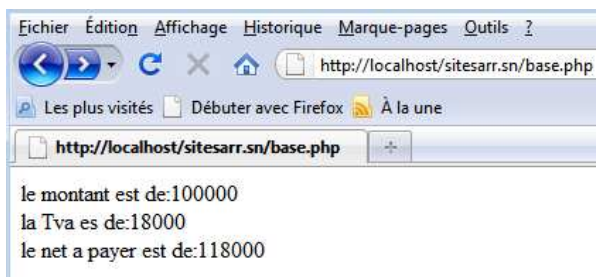
Function CalculeTVA ()
{
    $Montant= 100000;
    Echo 'le montant est de:'. $Montant. '</br>';
    $Taux=0.18;
    $MTVA=TVA ($Montant, $Taux);
    Echo 'la Tva es de:'. $MTVA. '</br>';
    $Net=NetPayer ($Montant, $MTVA);
    Echo 'le net a payer est de:'. $Net;
    return 0;
}

CalculeTVA ();

?>

```

Resultat :



Passage de paramètre par référence

Pour passer une variable par référence, il faut que son nom soit précédé du symbole &

Exemple :

```
$res= &$a ;
```

Exemple simple :

Fonction1.php

```

<?php
$x=12;
$y=100;
echo 'la valeur de x est '.$x.'

```

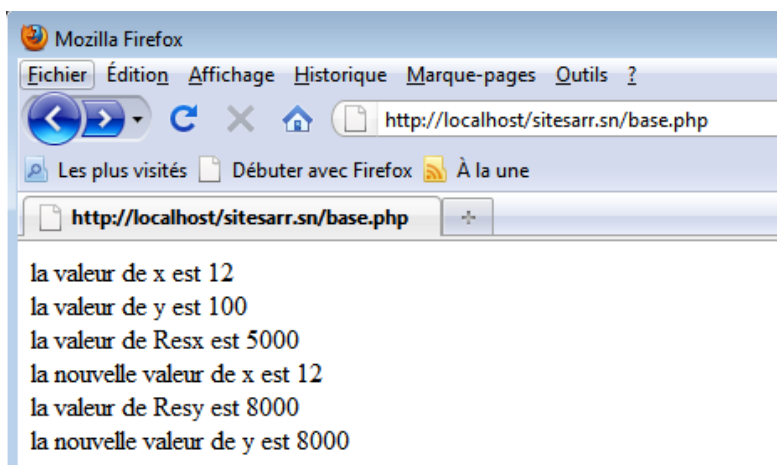
Base.php

```

<html>
<Body>
    <?php Include 'Fonction1.php' ?>
</body>
</html>

```

Resultat :



L'appel récursif

PHP admet les appels récursifs de fonctions. Une fonction peut s'appeler elle-même.

Exemple :

Calcule du factoriel d'un nombre :

```

<?php
    $x=3;
    $f=3;
    $res=Factoriel($f,$x);
    echo $res;

Function Factoriel($fact,$x)
{
    $fact=$fact * ($x-1);
    $x=$x-1;
    If ($x ==1)
    {
        $fact=$fact;
    }
    else
    {
        $fact=Factoriel($fact,$x);
    }
    return $fact;
}
?>

```

Appel dynamique de fonctions

Exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script. L'appel dynamique d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

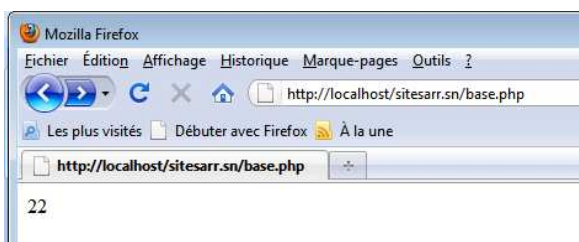
Exemple :

```

<?php
Function Foncl($x, $y)
{
    $somme= $x+$y;
    return $somme ;
}
$nom_Fonction = "Foncl";
$rep= $nom_Fonction(10,12);
echo $rep;
?>

```

Résultat :



Les fonctions mathématiques.

abs(\$x) : valeur absolue
ceil(\$x) : arrondi supérieur
floor(\$x) : arrondi inférieur
pow(\$x,\$y) : x exposant y
round(\$x,\$i) : arrondi de x à la ième décimale
max(\$a, \$b, \$c ...) : retourne l'argument de valeur maximum
pi() : retourne la valeur de Pi
 Et aussi : **cos, sin, tan, exp, log, min, pi, sqrt...**

Quelques constantes :

M_PI : valeur de pi (3.14159265358979323846)

M_E : valeur de e (2.7182818284590452354)

Exemple : Créer un programme avec une fonction :

qui calcule la valeur absolue d'une -234

Donner la plus grande et la plus petite valeur entre les trois valeurs 21, 54 et 9

Arrondir la valeur 1234,3455266 à 3 positions decimales

Calculer la valeur 10 puissance 2 puis arrondire par default le resultat

Donner la racine carré de 25

Calcule la surface d'un cercle de rayon 3.4

```

<?php
    $valeur= -234;
    $valabs= abs($valeur);
    Echo 'la valeur absolue de: '.$valeur.' est de: '.$valabs.'

```

Résultat :

```

http://localhost/sitesarr.sn/base.php
la valeur absolue de: -234 est de: 234
le plus grand est : 54
le plus grand est : 21
avant 1234.3455266 apres 1234.346
puissance de 10 et 2 =100
la racine carre de 25 =5
la surface est de36.316811075498
  
```

Formatage d'un nombre :

number_format (\$nbr,\$dec,\$a,\$b)) : retourne une chaîne de caractères représentant le nombre **\$nbr** avec **\$dec** décimales après formatage.

La chaîne **\$a** représente le symbole faisant office de virgule et **\$b** le séparateur de milliers.

Par défaut, le formatage est anglophone : **\$a** = ‘.’ et **\$b** = ‘,’.

Très utile pour représenter les nombres élevés au format francophone.

Exemples :

```

number_format (1000000.3333);           // affiche 1,000,000
number_format (1000000.3333,2);         // affiche 1,000,000.33
number_format (1000000.3333,2,"",".");  // affiche 1.000.000,33
  
```

Exemple :

Afficher le salaire de toto qui est de \$20,000.500 en FCFA avec un format francais et 1 chiffre apres la virgule.

```

<?php
$salaire= 20000.500;
echo 'le salaire de toto est de : $'.$salaire.'<br>';
$salaireFCFA= $salaire*500;
$frsalaire= number_format($salaireFCFA,1,"",".");
echo 'le salaire de toto est de :'.$frsalaire.' FCFA';
?>
  
```

Résultat :

```

http://localhost/sitesarr.sn/base.php
le salaire de toto est de : $20000.5
le salaire de toto est de :10.000.250,0 FCFA
  
```

Les variables booléennes prennent pour valeurs **TRUE** (vrai) et **FALSE** (faux). Une valeur entière nulle est automatiquement considérée comme **FALSE**. Tout comme une chaîne de caractères vide ‘’’. Ou encore comme les chaînes ‘0’ et ‘0’ castées en l’entier 0 lui même casté en FALSE.

Exemple :

```
if(0) echo 1; // faux
if('') echo 2; // faux
if('0') echo 3; // faux
if('00') echo 4;
if('0') echo 5; // faux
if('00') echo 6;
if(' ') echo 7;
```

Cet exemple affiche **467**. Donc l'espace ou la chaîne **'00'** ne sont pas considérés castés en **FALSE**.

LISTE COMPLETE DES FONCTIONS PHP, LISTEES PAR RUBRIQUE

Ces fonctions permettent la manipulation de chaînes de caractères. Certaines sections plus spécialisées sont disponibles dès les sections sur les expressions régulières et dans la section URL.

AddCSlashes — Ajoute des slashes dans une chaîne, comme en langage C.

AddSlashes — Ajoute un slash devant tous les caractères spéciaux.

bin2hex — Converti une valeur binaire en hexadécimal

Chop — Enlève les espaces de fin de chaîne.

Chr — Retourne un caractère.

chunk_split — Scinde une chaîne en plus petits morceaux.

convert_cyr_string — Converti la chaîne d'un alphabet cyrillique vers un autre.

count_chars — Retourne des informations sur les caractères utilisés dans une chaîne.

crypt — Encrypted une chaîne avec un DES.

echo — Affiche une ou plusieurs chaînes.

explode — Scinde une chaîne en morceau, grâce à un délimiteur.

get_html_translation_table — Retourne la table de traduction utilisée par htmlspecialchars() et htmlentities().

get_meta_tags — Extrait toutes les balises meta d'un fichier, et les retourne sous forme d'un tableau.

htmlentities — Converti tous les caractères spéciaux en équivalent HTML.

htmlspecialchars — Converti tous les caractères spéciaux en équivalent HTML.

implode — Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure.

join — Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure.

ltrim — Enlève les espaces de début de chaîne.

md5 — Calcule un md5 avec la chaîne.

Metaphone — Calcule la clé metaphone d'une chaîne.

nl2br — Converti les nouvelles lignes en HTML (<BR?).

Ord — Retourne la valeur ASCII du caractère.

parse_str — Analyse une chaîne, et en déduit des variables et leur valeur.

print — Affiche une chaîne.

printf — Affiche une chaîne formatée.

quoted_printable_decode — Décode une chaîne
 QuoteMeta — Ajoute un backslash devant tous les caractères méta
 rawurldecode — Décode une chaîne URL.
 rawurlencode — Encode une chaîne en URL, selon la RFC1738.
 setlocale — Change les informations locales.
 similar_text — Calcule la similarité de deux chaînes.
 soundex — Calcule la valeur soundex d'une chaîne.
 sprintf — Retourne une chaîne formatée.
 strcasecmp — Comparaison binaire de chaînes, insensible à la casse.
 strchr — Recherche la première occurrence d'un caractère.
 strcmp — Comparaison binaire de chaînes.
 strcspn — Recherche la longueur du premier segment de chaîne qui ne corresponde pas au masque donné.
 strip_tags — Enlève les balises HTML et PHP.
 StripCSlashes — Déquote une chaîne quotée avec addslashes
 StripSlashes — Enlève les slash ajoutés par la fonction addslashes
 strstr — strstr(), insensible à la casse.
 strlen — Retourne la longueur de la chaîne.
 strpos — Recherche la dernière occurrence d'un caractère dans une chaîne.
 strrchr — Recherche la dernière occurrence d'un caractère dans une chaîne
 str_repeat — Répète une chaîne.
 strrev — Inverse l'ordre des caractères d'une chaîne.
 strrpos — Recherche la dernière occurrence d'un caractère dans une chaîne.
 strspn — Retourne la longueur du premier segment qui vérifie le masque.
 strstr — Trouve la première occurrence d'une chaîne.
 strtok — Morcelle une chaîne
 strtolower — Met tous les caractères en minuscule.
 strtoupper — Met tous les caractères en majuscule.
 str_replace — Remplace toutes les occurrences d'une chaîne par une autre.
 strtr — Remplace toutes les occurrences d'un caractère par un autre.
 substr — Retourne une partie de la chaîne.
 substr_replace — Remplace dans une sous partie de chaîne
 trim — Enlève les espaces de fin et de fin de chaîne.
 ucfirst — Force le premier caractère d'une chaîne en majuscule.
 ucwords — Force le premier caractère de chaque mot d'une chaîne en majuscule

addCSlashes

chaîne_result = addCSlashes(chaîne, caractères);

Ajoute des slashes dans une chaîne devant toutes les occurrences du caractère spécifié

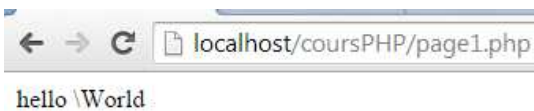
Exemple :

```

<html>
<head>
<title> Mon cours PHP edouard sarr</title>
</head>
<body>
<?php
    $str = addslashes("Hello World!", "W");
    echo($str);
?>
</body>
</html>

```

Resultat :



localhost/coursPHP/page1.php

hello \World

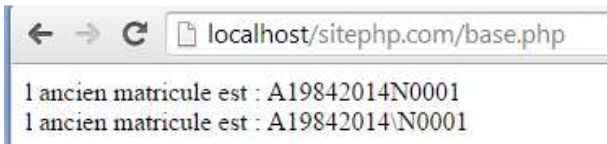
Exemple :

```

<?php
$matricule= 'A19842014N0001';
echo 'l ancien matricule est : '.$matricule.'

```

Résultat :



localhost/sitephp.com/base.php

l ancien matricule est : A19842014N0001
l ancien matricule est : A19842014N0001

addSlashes

chaîne_result = addSlashes (chaîne);

Ajoute un slash devant toutes les occurrences des caractères spéciaux.

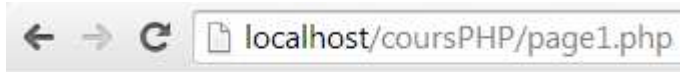
Exemple:

```

<html>
<head>
<title> Mon script PHP </title>
</head>
<body>
<?php
    $str = addslashes('je suis etudiant à UCAO-St michel "merci" ?');
    echo($str);
?>
</body>
</html>

```

Resultat:



je suis etudiant Ã UCAO-St michel \"merci\" ?

chop

```
chaîne_result = Chop(chaîne);
```

Supprime les espaces blancs en fin de chaîne.

chr

```
caractère = chr(nombre);
```

Elle retourne un caractère en mode ASCII

crypt

```
chaîne_result = crypt(chaîne [, chaîne_code])
```

Code une chaîne avec une base de codage.

echo

```
echo expression_chaîne;
```

Affiche à l'écran une ou plusieurs chaînes de caractères.

explode

```
$tableau = explode(délimiteur, chaîne);
```

Scinde une chaîne en fragments à l'aide d'un délimiteur et retourne un tableau.

Autres fonctions de manipulation de chaîne:

Exemple : Soit une chaîne de caractères. Couper le en deux parties égales, le mettre la deuxième en majuscule, la première en minuscule, les concaténer mais inversement.

```

<?php
$chaine= " je suis edouard ngor SARR professeur d'informatique à Ucao St michel de Dakar";
echo 'CHAINE : '.$chaine.'

```

http://localhost/sitesarr.sn/base.php
 CHAINE : je suis edouard ngor SARR professeur d'informatique Ã Ucao St michel de Dakar
 la chaine fait : 80 caracteres
 PARTIE 1 : je suis edouard ngor SARR professeur d'
 PARTIE 2 : informatique Ã Ucao St michel de Dakar
 PARTIE 1 MINISCULE : je suis edouard ngor sarr professeur d'
 PARTIE 2 MAJUSCULE : INFORMATIQUE Ã UCAO ST MICHEL DE DAKAR
 NOUVELLE CHAINE : INFORMATIQUE Ã UCAO ST MICHEL DE DAKAR je suis edouard ngor sarr professeur d'

Variables locales et variables globales

Une variables en PHP est soit : global, static ou local

Toute variable déclarée en dehors d'une fonction est globale

Pour Utiliser une variable globale dans une fonction, il faut utiliser l'instruction **global** suivie du nom de la variable

Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : utiliser l'instruction static.

Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

Exemple : Créer un générateur automatique de mot de passe :

Le mot de passe doit commencer par les deux premiers caractères en majuscule du nom.

Suivit des deux derniers chiffres de son année de naissance

Suivit d'un tirer du 6

Suivit du numéro incrémental du genre 00001.

Suivit des trois premiers caractères de son prénom.

TP à rendre :

Faire un logiciel de cryptage et de décryptage. Tous caractères doivent être transformé et la chaîne initiale inversée.

Partie 4 :
LES TABLEAUX

Principe

Une variable tableau est de type **array**. Un tableau accepte des éléments de tout type. Les éléments d'un tableau peuvent être de types différents et sont séparés d'une virgule.

La Création à l'aide de la fonction `array()`. Uniquement pour des tableaux à une dimension

Les éléments d'un tableau peuvent pointer vers d'autres tableaux

Les éléments d'un tableau peuvent appartenir à des types distincts

L'index d'un tableau en PHP commence de 0

Pas de limites supérieures pour les tableaux

Un tableau peut être initialisé avec la syntaxe **array**.

Exemple :

```
$tab_colors = array('red', 'yellow', 'blue', 'white');
```

```
$tab = array('foobar', 2002, 20.5, $name);
```

Mais il peut aussi être initialisé au fur et à mesure.

Exemples :

```
$prenoms[ ] = "amadou";          $villes[0] = "Paris";
```

```
$prenoms[ ] = "edouard";         $villes[1] = "Londres";
```

```
$prenoms[ ] = "Anissa";          $villes[2] = "Dakar";
```

L'appel d'un élément du tableau se fait à partir de son indice (dont l'origine est zéro comme en C).

Exemple : `echo $tab[10];`

La fonction `count()` pour avoir le nombre d'éléments d'un tableau

Les tableaux indicés et les tableaux associatifs :

Tableau indicé

Il donne la possibilité d'accéder aux éléments par l'intermédiaire de numéros

`$tableau[indice] = valeur;`

Exemple d'affectation de valeurs:

```
$jour[3] = "Mercredi";
```

```
$note[0] = 20;
$tableau = array(valeur0, valeur1,..., valeurN);
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);
```

```
$variable = $tableau[indice];
```

Exemple : récupération de valeurs :

```
$JJ = $jour[6]; // affecte "Samedi" à $JJ
echo $note[1] + $note[5];
```

Tableau associatif (ou table de hachage)

Dans ces genres de tableau les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index.

```
$tableau["indice"] = valeur;
```

Exemple de remplissage :

```
$jour["Dimanche"] = 7
$jour["Mercredi"] = "Le jour des enfants"
$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3, "Mercredi" => 4, "Jeudi" => 5, "Vendredi"
=> 6, "Samedi" => 7);
```

Exemple de récupération de valeurs

```
$variable = $tableau["indice"];
$JJ = $jour["Vendredi"]; //affecte 6 à $JJ
echo $jour["Lundi"]; //retourne la valeur 2
```

Tableaux multidimensionnels

En PHP il n'y pas d'outils pour créer directement des tableaux multidimensionnels Mais l'imbrication des tableaux est possible pour palier à ce manque.

Syntaxe :

```
$tab1 = array(Val0, Val1,..., ValN);
$tab2 = array(Val0, Val1,..., ValN);
```

// Création d'un tableau à deux dimensions

\$tableau = array(\$stab1, \$stab2);

Exemple :

\$mois = array("Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août", "Septembre", "Octobre", "Novembre", "Décembre");

\$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");

&element_date = array(&mois, &jour);

Janvier O,O	Fevrier	Mars O ,2	Avril	Mai	Juin	Juillet	Aout				Decembre
Lundi											
Mardi											
Mercredi											
Jeudi											
Vendredi											
Samedi											
Dimanche											

\$variable = \$tableau[Ligne][colonne];

\$MM = \$element_date[0][0]; //affecte "Janvier" à \$MM

Insertion et Lecture des éléments d'un tableau

Avec une boucle for

```
for ($i=0; $i<count($stab) ; $i++)
{
    Instruction;
}
```

Exemple: Remplir dans chaque case d'un tableau par le double de son indice. Afficher les elements du tableau.

```

<?php
    //remplissage
    for ($i=0; $i<7 ; $i++)
    {
        $Tab[$i]=2*$i;
    }
    //affichage
    for ($i=0; $i<7; $i++)
    {
        echo $Tab[$i]. '<br>';
    }
?>

```

http://localhost/sitesarr.sn/base.php

0
2
4
6
8
10
12

Autre exemple:

```

for ($i=0; $i<count($tab) ; $i++)
{
    if ($tab[$i]== "a")
    {
        echo $tab[$i], "<br />";
    }
}

```

Avec une boucle while

```

$i=0;
while ($tab[$i]){
    if ($tab[$i][0] == "a" )
        {echo $tab[$i], "<br /> ";
    }
}

```

Avec La boucle foreach

```

foreach($tab as $elem)
{

```

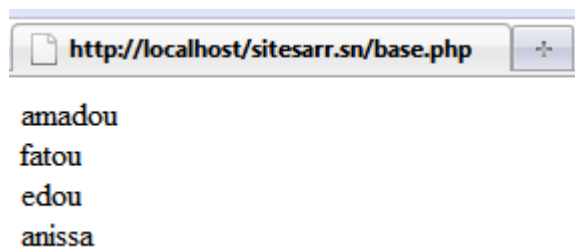


```
echo $elem."\n";
}
```

La variable **\$elem** prend pour valeurs successives tous les éléments du tableau **\$tab**.

Exemple :

```
<?php
//remplissage
$tab=array('amadou','fatou','edou','anissa');
//affichage
foreach($tab as $elem)
{
    echo $elem."</br>";
}
?>
```



http://localhost/sitesarr.sn/base.php

amadou
fatou
edou
anissa

Exemple : Chercher si anissa se trouve dans le tableau.

```
<?php
//remplissage
$tab=array('amadou','fatou','edou','anissa');
//affichage
foreach($tab as $elem)
{
    If ($elem=='anissa')
    {
        echo $elem."</br>";
    }
}
?>
```

Quelques fonctions pour les tableaux:

count(\$tab), sizeof : retournent le nombre d'éléments du tableau
in_array(\$var,\$tab) : dit si la valeur de **\$var** existe dans le tableau **\$tab**
list(\$var1,\$var2...) : transforme une liste de variables en tableau
range(\$i,\$j) : retourne un tableau contenant un intervalle de valeurs
shuffle(\$tab) : mélange les éléments d'un tableau
sort(\$tab) : trie alphanumérique les éléments du tableau
rsort(\$tab) : trie alphanumérique inverse les éléments du tableau
implode(\$str,\$tab), join : retournent une chaîne de caractères contenant les éléments du tableau **\$tab** joints par la chaîne de jointure **\$str**
explode(\$delim,\$str) : retourne un tableau dont les éléments résultent du hachage de la chaîne **\$str** par le délimiteur **\$delim**
array_merge(\$tab1,\$tab2,\$tab3...) : concatène les tableaux passés en arguments
array_rand(\$tab) : retourne un élément du tableau au hasard

Il est possible d'effectuer des opérations complexes sur les tableaux en créant par exemple sa propre fonction de comparaison des éléments et en la passant en paramètre à une fonction de tri de PHP.

Fonctions de tri

Tri selon les valeurs

La fonction **sort()** effectue un tri sur les valeurs des éléments d'un tableau selon un critère alphanumérique :selon les codes ASCII :

« a » est après « Z » et « 10 » est avant « 9 »)

Le tableau initial est modifié et non récupérables dans son ordre original

Pour les tableaux associatifs les clés seront perdues et remplacées par un indice créé après le tri et commençant à 0

La fonction **rsort()** effectue la même action mais en ordre inverse des codes ASCII.

La fonction **asort()** trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs

La fonction **arsort()** la même action mais en ordre inverse des codes ASCII

la fonction **natscasesort()** effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

Tri sur les clés

La fonction **ksort()** trie les clés du tableau selon le critère des codes ASCII, et préserve les associations clé /valeur

La fonction **krsort()** effectue la même action mais en ordre inverse des codes ASCII

Quelques fonctions alternatives pour le parcours de tableaux (normaux ou associatifs) :

reset(\$tab) : place le pointeur sur le premier élément

current(\$tab) : retourne la valeur de l'élément courant
next(\$tab) : place le pointeur sur l'élément suivant
prev(\$tab) : place le pointeur sur l'élément précédant
each(\$tab) : retourne la paire clé/valeur courante et avance le pointeur

Exemple :

```
$colors = array('red', 'green', 'blue');
$nbr = count($colors);
reset($colors);
for($i=1; $i<=$nbr; $i++) {
    echo current($colors)."<br />";
    next($colors);
}
```

SYNTAXES :

\$tableau = array_count_values(\$variable);
 retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.

\$tableau = array_diff(\$var_1, \$var_2, ..., \$var_N);
 retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.

\$tableau = array_intersect(\$var_1, \$var_2, ..., \$var_N);
 retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.

\$tableau = array_flip(\$variable);
 intervertit les paires clé/valeur dans un tableau.

\$tableau = array_keys(\$variable [, valeur]);
 retourne toutes les clés d'un tableau ou les emplacements d'une valeur dans un tableau.

\$tableau = array_map(\$var_1 [, \$var_2, ..., \$var_N], 'fonction');
 applique une fonction à un ou plusieurs tableaux.

\$tableau = array_merge(\$var_1, \$var_2, ..., \$var_N);
 enchaîne des tableaux entrés en argument afin d'en retourner un unique.

\$tableau = array_merge_recursive(\$var_1, \$var_2, ..., \$var_N);

enchaine des tableaux en conservant l'ordre des éléments dans le tableau résultant. Dans le cas de clés communes, les valeurs sont placées dans un tableau.

true | false = array_multisort(\$var, critère1, critère2 [, ..., \$var_N, critère1, critère2])

trie un ou plusieurs tableaux selon un ordre croissant ou décroissant (SORT_ASC ou SORT_DESC) et selon une comparaison alphabétique, numérique ou de chaîne de caractères (SORT_REGULAR, SORT_NUMERIC ou SORT_STRING).

\$tableau = array_pad(\$variable, taille, valeur);

recopie tout un tableau en ajustant sa taille à l'argument correspondant et en bourrant d'une valeur spécifiée les éléments vides.

Une classe est composée de deux parties:

Les attributs: il s'agit des données représentant l'état de l'objet

Les méthodes : il s'agit des opérations applicables aux objets

Php gère la programmation orientée objet à l'aide de classes.

Exemple:

```
<?php
class client {
    var $nom;
    var $ville;
    var $naiss ;
    function age() {
        $jour = getdate();
        $an=$jour["year"];
        $age = $an - $this->naiss;
        echo "Il a $age ans cette année <br />" ;
    }
}

//création d'un objet
$client1 = new client() ;

//affectation des propriétés de l'objet
$client1 -> nom = "Sarr" ;
$client1-> naiss = "1986" ;
$client1->ville = "Mbour" ;

//utilisation des propriétés
echo "le nom du client1 est ", $client1->nom, "<br />" ;
echo "la ville du client1 est ", $client1-> ville, "<br />" ;
echo "le client1 est né en ", $client1->naiss, "<br />" ;

//appel de la méthode age()
$client1->age() ;
```

?>

Autre exemple :

```
class Voiture {           // déclaration de la classe
    var $couleur;         // déclaration d'un attribut
    var $belle = TRUE;    // initialisation d'un attribut

    function voiture() {  // constructeur
        $this->couleur = "noire";
    } // le mot clé $this faisant référence à l'objet est obligatoire

    function Set_Couleur($couleur) {
        $this->couleur = $couleur;
    }
}

$mavoiture = new Voiture(); // création d'une instance
$mavoiture->Set_Couleur("blanche"); // appel d'une méthode
$scoul = $mavoiture->couleur; // appel d'un attribut
```

Manipulation des classes et des objets :

Php n'inclue pas dans sa version 4 de niveaux de visibilité des éléments de la classe, il n'y a donc pas de concept d'encapsulation

Instanciation de la classe

```
$Nom_de_l_objet = new Nom_de_la_classe;
```

Accéder aux propriétés d'un objet

```
$Nom_de_l_objet->Nom_de_la_donnee_membre = Valeur;
```

Accéder aux méthodes d'un objet

```
$Nom_de_l_objet->Nom_de_la_fonction_membre(parametre1,parametre2,...);
```

La variable \$this

```
$this->age = $Age;
```

Tout objet instancié est une variable et peut à se titre être passé en argument à une fonction ou bien être un retour de fonction ou encore être sauvegardée en donnée de session.

Il n'existe pas de destructeur : comme en Java, les objets qui cessent d'être utilisés sont automatiquement détruits.

Il n'y a pas de notion de visibilité : tous les attributs et méthodes sont publiques et une classe hérite forcément de tous les attributs et méthodes de son père.

Une classe fille hérite de tous les attributs et méthodes de la classe parente dont elle est une extension (d'où la syntaxe **extends**). Il est possible de surcharger les méthodes, d'en définir de nouvelles...

L'héritage

On utilise l'Instruction extends :

```
class nouvelle_classe extends super_classe
```

La nouvelle classe hérite des attributs et des méthodes appartenant à la super-classe tout en définissant ses propres fonctions et variables.

Le langage PHP ne supporte pas l'héritage multiple

Exemple :

```
class Voituredeluxe extends Voiture { // déclaration de la sous classe
```

```
    var $couleur;
```

```
    function voituredeluxe() {      // constructeur
        $this->Voiture();
    }
```

```
    function Set_Couleur($couleur) {
        $this->couleur = $couleur;
    }
```

```
    function Get_Couleur() {
        return $this->couleur;
    }
```

```
}
```

La nouvelle classe **Voituredeluxe** hérite de tous les attributs et méthodes de la classe parente **Voiture** dont elle est une extension (**extends**).

Il est possible de surcharger les méthodes, d'en déclarer de nouvelles, etc.

Le constructeur

Une fonction qui est appelée automatiquement par la classe lors de son instantiation avec l'opérateur new
Doit posséder un nom identique à celle de la classe

Avec PHP 3, une fonction définie dans une classe héritée devient un constructeur si son nom est similaire à celle de la nouvelle classe

Avec PHP 4, une fonction constructeur ne peut être définie que dans sa propre classe

Lorsqu'une classe héritant d'une autre est instanciée et si aucun constructeur n'est défini dans cette classe, alors la fonction constructeur sollicitée sera celle de la super-classe

L'opérateur ::

faire référence à une fonction définie dans une super-classe à partir d'une classe héritant de cette dernière .

```
class nouvelle_classe extends super_classe
{
    function fonction()
    {
        echo "Blocs d'instructions de la fonction fonction() . " dans la nouvelle-classe.";
        super_classe::fonction();
    }
}
```

L'opérateur parent

faire référence à des variables ou des fonctions présentes dans la super-classe à partir d'une autre classe héritant de cette dernière

```
class nouvelle_classe extends super_classe
{
    function fonction()
    {
        echo "Blocs d'instructions de la fonction fonction()" . " dans la nouvelle-classe.";
        // se réfère à la fonction fonction() de la super_classe
        parent::fonction();
    }
}
```

Quelques fonctions :

get_declared_classes() : retourne un tableau listant toutes les classes définies

class_exists(\$str) : vérifie qu'une classe dont le nom est passé en argument a été définie

get_class(\$obj), get_parent_class : retournent le nom de la classe de l'objet **\$obj**

get_class_methods(\$str) : retourne les noms des méthodes de la classe **\$str** dans un tableau

get_class_vars(\$str) : retourne les valeurs par défaut des attributs de la classe **\$str** dans un tableau associatif

get_object_vars(\$obj) : retourne un tableau associatif des attributs de l'objet **\$obj** les clés sont les noms des attributs et les valeurs, celles des attributs si elles existent

is_subclass_of(\$obj,\$str) : détermine si l'objet **\$obj** est une instantiation d'une sous-classe de **\$str**, retourne VRAI ou FAUX

method_exists(\$obj,\$str) : vérifie que la méthode **\$str** existe pour une classe dont **\$obj** est une instance, retourne VRAI ou FAUX

Sauvegarde des objets

La sauvegarde et la relecture des objets s'effectuent respectivement par **serialize** et **unserialize**

serialize permet de transformer un objet en une chaîne de caractères pouvant être facilement transmise à une autre page lors d'une session

unserialize permet de reconstituer l'objet à partir de la chaîne de caractères précitée

Exemple : Créer une classe pour gerer les compte en banque. Un compte est caractérisé

Partie 6 :

LA GESTION DES FICHIERS AVEC PHP

Principe

PHP prend en charge l'accès au système de fichiers du système d'exploitation du serveur

Les opérations sur les fichiers concernent la création, l'ouverture, la suppression, la copie, la lecture et l'écriture de fichiers

Les possibilités d'accès au système de fichiers du serveur sont réglementées par les différents droits d'accès accordés au propriétaire, à son groupe et aux autres utilisateurs

La communication entre le script PHP et le fichier est repérée par une variable, indiquant l'état du fichier et qui est passée en paramètre aux fonctions spécialisées pour le manipuler

Ouverture de fichiers

La fonction `fopen()` permet d'ouvrir un fichier, que ce soit pour le lire, le créer ou y écrire

`fopen(chaine nom du fichier, chaine mode)`

mode : indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible:

r (comme read) indique une ouverture en lecture seulement

w (comme write) indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)

a (comme append) indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

lorsque le mode est suivie du caractère **+** celui-ci peut être lu et écrit

le fait de faire suivre le mode par la lettre **b** entre crochets indique que le fichier est traité de façon binaire.

Mode	Description
r	ouverture en lecture seulement
w	ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
a	ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
r+	ouverture en lecture et écriture
w+	ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)
a+	ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

Exemple :

```
$fp = fopen("fichier.txt","r"); //lecture
$fp = fopen("fichier.txt","w"); //écriture depuis début du fichier
```

De plus, la fonction `fopen` permet d'ouvrir des fichiers présents sur le web grâce à leur URL.

Exemple : un script permettant de récupérer le contenu d'une page d'un site web:

```
<?
$fp = fopen("http://www.mondomaine.fr","r"); //lecture du fichier
while (!feof($fp)) {
    //on parcourt toutes les lignes
    $page .= fgets($fp, 4096);
    // lecture du contenu de la ligne
}
?>
```

Il est généralement utile de tester si l'ouverture de fichier s'est bien déroulée ainsi que d'éventuellement stopper le script PHP si cela n'est pas le cas .

```
<?
if (!$fp = fopen("fichier.txt","r")) {
    echo "Echec de l'ouverture du fichier";
    exit;
}
else {
    // votre code;
}
?>
```

Un fichier ouvert avec la fonction **fopen()** doit être fermé, à la fin de son utilisation, par la fonction **fclose()** en lui passant en paramètre l'entier retourné par la fonction `fopen()`

Lecture et écriture de fichiers

```
<?
if (!$fp = fopen("fichier.txt","r"))
{
    echo "Echec de l'ouverture du fichier";
}
else {
    $Fichier="";
```

```

while(!feof($fp)) {
    // On récupère une ligne
    $Ligne = fgets($fp,255);
    // On affiche la ligne
    echo $Ligne;
    // On stocke l'ensemble des lignes dans une variable
    $Fichier .= $Ligne. "<BR>";
}
fclose($fp); // On ferme le fichier
}
?>

```

Exemple :

Fic1.txt

```

je suis etudiant à ucao st michel
je fais informatique
Avec MONSIEUR Edouard Ngor SARR

```

Fic.php

Pour stocker des informations dans le fichier, il faut dans un premier temps ouvrir le fichier en écriture en le créant s'il n'existe pas

Deux choix : le mode 'w' et le mode 'a'.

<?

```

$nom="Jean"; $email="jean@dupont.fr";
$fp = fopen("fichier.txt","a"); // ouverture du fichier en écriture
fputs($fp, "\n"); // on va a la ligne
fputs($fp, $nom."|".$email); // on écrit le nom et email dans le fichier
fclose($fp);
?>

```

Les tests de fichiers

is_dir() permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire.

La fonction **is_dir()** renvoie 1 s'il s'agit d'un répertoire, 0 dans le cas contraire

booléen `is_dir(chaine Nom_du_fichier);`

is_executable() permet de savoir si le fichier dont le nom est passé en paramètre est exécutable.

La fonction `is_executable()` renvoie 1 si le fichier est exécutable, 0 dans le cas contraire

booléen `is_executable(chaine Nom_du_fichier);`

is_link() permet de savoir si le fichier dont le nom est passé en paramètre correspond à un lien symbolique. La fonction `is_link()` renvoie 1 s'il s'agit d'un lien symbolique, 0 dans le cas contraire

booléen `is_link(chaine Nom_du_fichier);`

is_file() permet de savoir si le fichier dont le nom est passé en paramètre ne correspond ni à un répertoire, ni à un lien symbolique.

La fonction `is_file()` renvoie 1 s'il s'agit d'un fichier, 0 dans le cas contraire

booléen `is_file(chaine Nom_du_fichier);`

D'autres façons de lire et écrire

La fonction **file()** permet de retourner dans un tableau l'intégralité d'un fichier en mettant chacune de ces lignes dans un élément du tableau

Tableau `file(chaine nomdufichier);`

Exemple : parcourir l'ensemble du tableau afin d'afficher le fichier

<?

`$Fichier = "fichier.txt";`

`if (is_file($Fichier)) {`

`if ($TabFich = file($Fichier)) {`

`for($i = 0; $i < count($TabFich); $i++)`

`echo $TabFich[$i];}`

`else {echo "Le fichier ne peut être lu...
";}}`

`else {echo "Désolé le fichier n'est pas valide
 " ;}`

`?>`

La fonction **fpasssthru()** permet d'envoyer le contenu d'un fichier dans la fenêtre du navigateur.

booléen `fpasssthru(entier etat);`

Elle permet d'envoyer le contenu du fichier à partir de la position courante dans le fichier.

Elle n'ouvre pas automatiquement un fichier. Il faut donc l'utiliser avec `fopen()`.

Il est possible par exemple de lire quelques lignes avec `fgets()`, puis d'envoyer le reste au navigateur.

Exemple : script permettant de parcourir tous les fichiers HTML contenus dans un site à la recherche de MetaTags

<?php

`function ScanRep($Directory){`

`echo "Parcours: $Directory
\n";`

```

if (is_dir($Directory) && is_readable($Directory)) {
if($MyDirectory = opendir($Directory)) {
    while($Entry = readdir($MyDirectory)) {
        if (is_dir($Directory."/".$Entry)) {
            if (($Entry != ".") && ($Entry != "..")) {
                echo "<b>Repertoire</b>: $Directory/$Entry<br>\n";
                ScanRep($Directory."/".$Entry);
            }
        }
        else {echo "<b>Fichier</b>: $Directory/$Entry<br>\n";
            if (eregi("(\\.html)|\\.htm)", $Entry)) {
                $meta=get_meta_tags($Directory."/".$Entry);
                foreach($meta as $cle => $valeur) echo $cle." ".$valeur."<br>";
            }
        }
    }
    closedir($MyDirectory);}
}
ScanRep(".");
?>

```

Le téléchargement de fichier

Le langage PHP4 dispose de plusieurs outils facilitant le téléchargement vers le serveur et la gestion des fichiers provenant d'un client

Un simple formulaire comportant un champ de type file suffit au téléchargement d'un fichier qui subséquemment, devra être traité par un script PHP adapté

```

<form method="POST" action="traitement.php"
    enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="Taille_Octets">
    <input type="file" name="fichier" size="30"><br>
    <input type="submit" name="telechargement" value="telecharger">
</form>

```

Le téléchargement de fichier en PHP4

Un champ caché doit être présent dans le formulaire afin de spécifier une taille maximum (MAX_FILE_SIZE) pour le fichier à télécharger. Cette taille est par défaut égale à deux mégaoctets.

En PHP 4, le tableau associatif global \$ _FILES contient plusieurs informations sur le fichier téléchargé.

\$ _FILES['fichier']['name'] : fournit le nom d'origine du fichier.

\$ _FILES['fichier']['type'] : fournit le type MIME du fichier.

\$ _FILES['fichier']['size'] : fournit la taille en octets du fichier.

\$ _FILES['fichier']['tmp_name'] : fournit le nom temporaire
du fichier.

Le téléchargement de fichier en PHP3

PHP 3 fait appel au variable globale ou/et au tableau associatif globale \$HTTP_POST_VARS à condition que respectivement les options de configuration register_globals et track_vars soient activées dans le fichier php.ini.

\$fichier : renvoie le nom temporaire du fichier.

\$fichier_name : renvoie le nom d'origine du fichier.

\$fichier_size : renvoie la taille en octets du fichier.

\$fichier_type : renvoie le type MIME du fichier.

\$HTTP_POST_VARS['fichier'] : fournit le nom temporaire du fichier.

\$HTTP_POST_VARS['fichier_name'] : fournit le nom d'origine du fichier.

\$HTTP_POST_VARS['fichier_type'] : fournit le type MIME du fichier.

\$HTTP_POST_VARS['fichier_size'] : fournit la taille en octets du fichier.

Par défaut, le fichier envoyé par le client est stocké directement dans le répertoire indiqué par l'option de configuration upload_tmp_dir dans le fichier php.ini.

upload_tmp_dir = c:\PHP\uploadtemp

Plusieurs fonctions spécialisées permettent la validation d'un fichier téléchargé pour son utilisation ultérieure.

La fonction is_uploaded_file indique si le fichier a bien été téléchargé par la méthode HTTP POST.

```
$booleen=is_uploaded_file($_FILES['fichier']['tmp_name']);
```

La fonction move_uploaded_file vérifie si le fichier a été téléchargé par la méthode HTTP POST, puis si c'est le cas le déplace vers l'emplacement spécifié.

Il est possible de télécharger plusieurs fichiers en même temps, en utilisant des crochets à la suite du nom du champ afin d'indiquer que les informations relatives aux fichiers seront stockées dans un tableau.

```
<form action="traitement.php" method="POST" enctype="multipart/form-data">
  <input type="file" name="fichier[]" ><br>
  ...
  <input type="file" name="fichierN[]" > <br>
  <input type="submit" value="Envoyer" name="soumission">
</form>
```

```
for($i = 0; $i < sizeof($_FILES['fichier']['name']); $i++){
echo "Nom du fichier : ». $_FILES['fichier']['name'][$i];}
```

Les fichiers téléchargés sont automatiquement effacés du répertoire temporaire au terme du script.

il est nécessaire de déplacer les fichiers vers un autre endroit ou de les renommer si ceux-ci doivent être conservés.

```
<!-- Fichier : formulaire.html -->
```

```
<html><body>
```

```
  <form method="POST" action="traitement.php" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
```

```

<input type="file" name="fichier" size="30"><br>
<input type="submit" name="telechargement" value="telecharger">
</form> </body></html>
<?php /* Fichier : traitement.php */
$repertoire = "f:\\PHP\\uploadtemp";
if (is_uploaded_file($_FILES['fichier']['tmp_name'])) {
    $fichier_temp = $_FILES ['fichier']['tmp_name'];
    echo "<h3>Le fichier a été téléchargé avec succès " . "à l'emplacement suivant : <br>" . $fichier_temp .
    ""</h3>";
    $nom_fichier = $_FILES ['fichier']['name'];
    echo "<h3>Le nom d'origine du fichier est " . $nom_fichier . ""</h3>";
    echo "<h3>Le type du fichier est " . $_FILES['fichier']['type'] . ""</h3>";
    echo "<h3>La taille du fichier est de " . $_FILES['fichier']['size'] . " octets.</h3>";
    copy($_FILES['fichier']['tmp_name'], $repertoire . $nom_fichier);}
else
{ echo '<h3 style="color:#FF0000">ATTENTION, ce fichier peut être à l'origine'
. ' d'une attaque : ' . $_HTTP_POST_FILES['fichier']['name'] . " !</h3>";}
?>

```

La manipulation de fichiers distants

Il peut être utile de manipuler des fichiers à distance, c'est-à-dire par le biais des protocoles de transferts HTTP ou FTP.

PHP autorise l'ouverture d'un fichier par l'intermédiaire d'une adresse URL dans la fonction fopen

```
$id_fichier = fopen("http://www.site.com/index.html", "r");
```

A partir de ce moment, toutes les informations contenues dans le fichier sont accessibles en lecture seule dans une application PHP

```
$taille = filesize("fichier.html");
```

```
echo str_replace("<", "&lt;", fread($id_fichier, $taille));
```

L'écriture sur un serveur distant est possible, à condition de passer en argument une adresse FTP à la fonction fopen() et que ce fichier soit nouveau.

```
$id_fichier = fopen("ftp://ftp.site.com/new_page.html", "w");
```

L'accès en écriture directement sur un site, nécessite souvent, la saisie d'un nom d'utilisateur et d'un mot de passe dans l'adresse afin d'éviter toutes intrusions inopportunes

```
ftp://nom_utilisateur:mot_passe@ftp.site.com/nouvelle_page.html
```

La modification d'un fichier distant n'est pas réalisable par ce moyen

```

<?php
function recherche_contenu($adresse){

```

```

$Id_fichier = fopen($adresse, "r");
if ($Id_fichier)
{
    $regex = "<!-- Début contenu -->.*<!-- Fin contenu -->";
    $contenu = fread($Id_fichier, filesize($adresse));
    if (eregi($regex, $contenu, $donnee)){
        echo "<h3><u>Données contenu :</u></h3> "
            . str_replace("<", "&lt;", $donnee[0]);
    }
    else echo "<p>Impossible de le contenu.\n";
}
else echo "<p>Impossible d'ouvrir le fichier distant.\n";
fclose($Id_fichier);
}
recherche_contenu("http://www.site.com/page.html");
?>

```

Les fonctions de système de fichiers

\$chaine = basename(chemin_fichier);

retourne le nom du fichier à partir de l'adresse du fichier spécifiée.

true | false = chgrp(nom_fichier, groupe_propriétaire);

modifie le groupe propriétaire du fichier.

true | false = chmod(nom_fichier, \$mode);

modifie le mode exprimé en nombre octal, du fichier.

true | false = chown(nom_fichier, propriétaire);

modifie le groupe propriétaire du fichier.

clearstatcache();

efface la mémoire cache remplie par les fonctions lsat et stat.

true | false = copy(fichier, nouveau_fichier);

copie un fichier vers une nouvelle destination.

delete(fichier);

efface le fichier.

\$chaine = dirname(chemin);

retourne le nom du dossier parent.

\$nombre = disk_free_space(dossier);

retourne l'espace disponible sur le disque sur lequel est le dossier.

\$nombre = diskfreespace(dossier);

identique à disk_free_space.

\$nombre = disk_total_space(dossier);

retourne la taille totale d'un dossier.

true | false = fclose(ID_fichier);

ferme un fichier indiqué par un identificateur retourné par fopen ou fsockopen.

true | false = feof(ID_fichier);

teste la fin du fichier.

Les fonctions de dossiers

nombre = chroot(\$chaine);

définit la chaîne de caractères comme la nouvelle racine

nombre = chdir(\$chaine);

définit le dossier en cours comme celui précisé par la chaîne de caractères.

\$objet = dir(\$chaine);

crée un objet à partir du dossier spécifié.

closedir(\$identificateur_dossier);

ferme le dossier à partir d'un identificateur retourné par opendir.

\$chaine = getcwd();

retourne le nom du dossier en cours.

\$identificateur_dossier = opendir(\$chaine);

ouvre un dossier et retourne un identificateur de l'objet.

\$chaine = readdir(\$identificateur_dossier);

retourne le fichier suivant dans le dossier spécifié par l'entremise de son identificateur.

Les dates et les heures

Les fonctions de date et d'heure

true | false = checkdate(mois, jour, année);

vérifie la validité d'une date.

\$chaine = date(format [, nombre]);

retourne une chaîne de caractères date/heure selon le format spécifié et représentant la date courante par défaut.

\$tableau = getdate([nombre]);

retourne les éléments de date et d'heure dans un tableau associatif.

\$tableau = gettimeofday();

retourne l'heure courante dans un tableau associatif.

\$chaine = gmdate(format [, nombre]);

retourne une chaîne de caractères date/heure GMT/CUT selon le format spécifié et représentant la date courante par défaut.

\$nombre = gmmktime(heure, minute, seconde, mois, jour, année [, 1/0]);

retourne l'instant UNIX d'une date GMT spécifiée et avec éventuellement une heure d'hiver

\$chaine = gmstrftime(format [, nombre]);

formate une date/heure GMT/CUT en fonction des paramètres locaux définis par setlocale.

\$tableau = localtime([nombre][, tab_associatif]);

retourne l'heure locale dans un tableau indicé par défaut ou associatif (1)

\$chaine = microtime();

retourne l'instant UNIX courant en secondes et microsecondes (1 janvier 1970 à 0H00)

\$nombre = mktime(heure, minute, seconde, mois, jour, année [, 1/0]);

retourne l'instant UNIX d'une date spécifiée et avec éventuellement une heure d'hiver (1)

\$chaine = strftime(format [, instant]);

formate une date/heure locale avec les options locales

\$nombre = time();

retourne l'instant UNIX courant

PARTIE II :

L'interactivité en PHP

