

Introduction à PHP

Code: php-intro

Originaux

url: <http://tecfa.unige.ch/guides/tie/html/php-intro/php-intro.html>

url: <http://tecfa.unige.ch/guides/tie/pdf/files/php-intro.pdf>

Auteurs et version

- Daniel K. Schneider - Vivian Synteta - Olivier Clavel
- Version: 0.9 (modifié le 30/11/00)

Prérequis:

- Avoir une notion minimale de ce qu'est un langage de programmation
- Connaître le langage HTML (simple HTML et formulaires pour plus tard)

Module technique précédent: [html-intro \(HTML simple\)](#)

Module technique précédent: [html-forms \(formulaires\)](#)

Objectifs:

- Se familiariser avec le langage PHP
 1. Les variables
 2. Les structures de contrôle (tests et boucles)

1. Table des matières détaillée

1. Table des matières détaillée	3
2. Généralités	4
2.1 Quelques "features" de PHP	5
2.2 Intégration de HTML et de code PHP	6
2.3 Sensibilisation à Php: Inclusion de fichiers	7
3. Introduction à la programmation avec PHP	9
3.1 Eléments de programmation	9
3.2 Ressources PHP on-line et conventions pour la Syntaxe	10
3.3 Syntaxe de PHP	11
3.4 Variables et assignation	11
3.5 Simples expressions et opérateurs	16
3.6 Sélection (Conditions et tests)	20
3.7 Fonctions PHP	23
3.8 Boucles "for" et génération HTML	25
4. Conseils pratiques pour PHP	29
4.1 Debugging	29
4.2 PHP en "Stand-alone"	30
4.3 Win95:	32

2. Généralités

- "PHP" veut dire aujourd'hui "Hypertext Preprocessor"

url: <http://tecfa.unige.ch/guides/php/>

Histoire:

- Conçu comme "Personal Home Page Generator" (Php2/FI) au début du WWW par Rasmus Lerdorf
- PHP 3 depuis fin 1997, PHP 4 depuis 1999

Définition officielle pour PHP 3.0

- PHP Version 3.0 is an *HTML-embedded scripting language*. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

Principe de base:

- Analogie avec JavaScript: on mélange du code PHP avec HTML
- mais c'est le serveur qui lit la page et qui "calcule" le contenu
- A Tecfa, tout fichier *.php est automatiquement passé à PHP pour exécution AVANT d'être servi au client.

Buts:

- Création de pages WWW dynamiquement construits
- "Middleware" le serveur et d'autres programmes

2.1 Quelques “features” de PHP

Disponibilité

- freeware (open-source)
- cross-plateform (Unix et Win32)

Installation

- peut tourner comme programme CGI
- comme module pour certains serveurs (par ex. Apache ou IIS)
la version compilée dans le serveur est plus rapide et plus puissante
- comme interpréteur de script stand-alone

Atouts principaux

- très bon support pour les bases de données (Oracle, Sybase, Microsoft, MySQL, Postgres, ODBC, etc.)
- bonne intégration avec le système
(fonctions OS et communication avec d’autres programmes)
- langage de programmation complet
- permet de mixer HTML et code PHP, relativement facile à apprendre
- support de fonctions Web (cookies, authentication, sessions, redirection)
- support pour un grand nombre d’autres librairies (LDAP, PDF, XML, GIF,...)

Alternatives

- ASP (Microsoft)
- JSP (Java)

2.2 Intégration de HTML et de code PHP

- Un marqueur spécial permet de délimiter les parties de code à interpréter dans un document avant de le servir.

Il existe 3 variantes (équivalentes pour HTML):

La plus répandue: `<? ?>`

```
<? echo("this is the simplest, an SGML processing instruction\n"); ?>
```

XML compatible: `<?php ?>`

```
<?php echo("if you want to serve XML documents, do like this\n"); ?>
```

Pour survivre avec FrontPage: `<script>`

```
<script language="php">
```

```
echo("some editors (like FrontPage) don't like processing instructions");
```

```
</script>
```

2.3 Sensibilisation à Php: Inclusion de fichiers

- PHP permet de composer une page HTML à partir de plusieurs fichiers. On peut ainsi définir une barre de menu centrale et l'inclure automatiquement dans tous les fichiers.
 - .Cet exemple présente une première application de PHP très simple.
 - enfin avec Apache, pas besoin de PHP, SSI (server side includes) ferait aussi l'affaire ...

Include

permet d'inclure le contenu d'un fichier au moment où l'instruction est évaluée

Syntaxe: `include ("nom du fichier");`

Exemple: `include("style.text");`

Require

permet d'inclure le contenu d'un fichier au moment où le fichier php est chargé

Syntaxe: `require ("nom de fichier");`

Exemple: `require("mes_fonctions.lib");`

Exemple 2-1: Inclusion de fichiers

url: <http://tecfa.unige.ch/guides/php/examples/includes/>

```
<HTML>
  <HEAD>
    <TITLE>Simple Include Demo (21-Apr-1998)</TITLE>
  <? include("style.text"); ?>
  </HEAD>
  <BODY>
    <H1>Simple Include Demo</H1>
```

In this file we include a `style sheet` and
a `footer`.

```
<P>
```

```
  Look at <A HREF="includel.phps">the formatted source</A>  
  or the <A HREF="includel.source">unformatted one</A>
```

if you want to know how this is done.

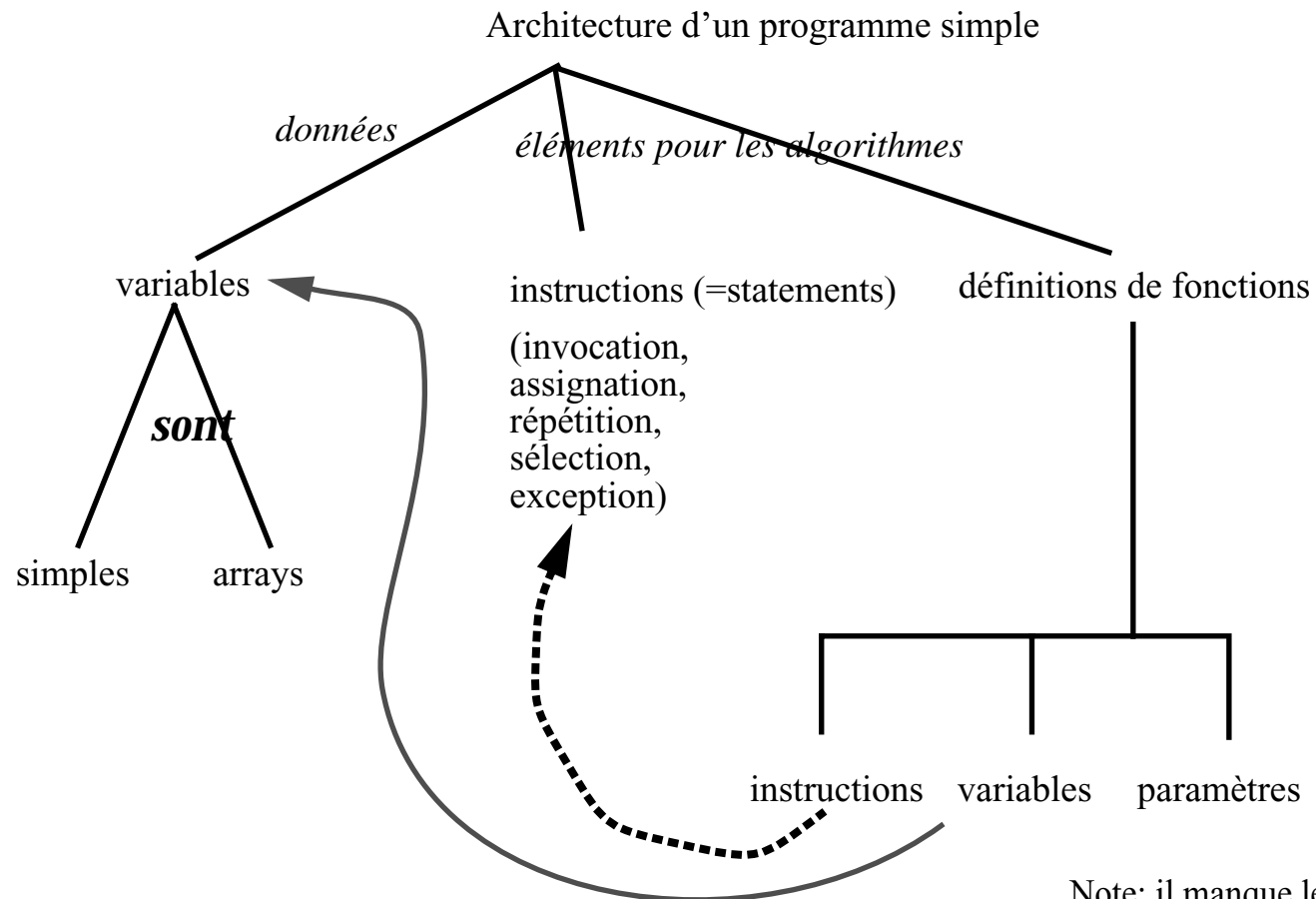
```
<H1>Yet another styled title</H1>
<UL>
  <LI> bullet item </LI>
  <LI> bullet item </LI>
</UL>
```

```
<?
/* A footer */
include("footer.text");
?>
  </BODY>
</HTML>
```


3. Introduction à la programmation avec PHP

3.1 Éléments de programmation

Figure 3-1: Programme = algorithme + structures de données



3.2 Ressources PHP on-line et conventions pour la Syntaxe

Conventions utilisées dans ce document !

fonte	exemple	signification
<i>fixe oblique</i>	<i>contenu</i>	vous devez remplacer
<fixe>	<statement>	pareil
fixe bold	then	Mots clefs, à mettre tel quel

- Attention: Ces conventions ne sont pas toujours respectées !

Ressources:

url: <http://tecfa.unige.ch/guides/php/>

(y compris manuels, exemples etc.)

url: Home page PHP: <http://www.php.net/>

url: Répertoire exemples à TECFA: <http://tecfa.unige.ch/guides/php/examples/>



3.3 Syntaxe de PHP

- La syntaxe de PHP ressemble à celle de famille "C" (C, C++, Java, Perl, etc.)
- Chaque instruction se termine par ";"
- Un commentaire est précédé soit par "//" ou par "#", soit entouré de "/* ...*/"

3.4 Variables et assignation

- Une variable est un "conteneur" qui contient de l'information.
- Tout identificateur précédé par un \$ est une variable
- Il n'est pas obligatoire de déclarer les variables (mais fortement conseillé)
- Pour assigner un contenu à une variable on fait une assignation.

A. Variables simples et assignation

Syntaxe: assignation

```
$variable = contenu ;
```

Illustrations:

```
$a = 10;
```

```
$nom = "Patrick Ott";
```

```
$somme = 123.456;
```

- voir aussi exemple 3-2 "Simple variables, arrays et un peu de génération HTML"
[14]

Exemple 3-1: Imprimer des variables

- Fichiers:

Application	<u>url: /guides/php/examples/simple/simple-echo.php</u>
Source (pour voir)	<u>url: /guides/php/examples/simple/simple-echo.phps</u>
Source (pour copier)	<u>url: /guides/php/examples/simple/simple-echo.text</u>

```
<BODY>
  <H1>Simple Echo of variables with PHP</H1>

  <?php

  <?
  $a = 10;
  $nom = "Patrick Ott";
  $somme = 123.456;

  echo "Le nommé $nom a $somme francs dans la poche, mais il voudrait $a fois plus.";
  ?>

  <p><hr>
</BODY>
```

- echo est une "instruction" qui permet d'imprimer un string
- Notez que les \$xxx sont substitués par leur contenu !

B. Création et utilisation d'arrays simples

- Un "array" (vecteur) est une sorte de liste
- Utiles pour stocker de l'information de même type que l'on veut manipuler ensemble.

Méthode de création 1:

```
$nombres[ ] =1;  
$nombres[ ] =2;  
$nombres[ ] =3;  
$nombres[ ] =4;
```

Méthode de création 2:

```
$nombres = array (1, 2, 3, 4);  
$noms = array ("Pat", "Dave", "Surf", "K");
```

Utilisation:

Syntaxe: Utilisation d'arrays simples

```
$vecteur[index]
```

- L'index commence à 0 ! (zero)

```
echo "Le deuxième élément de noms est: $noms[1];
```

Exemple 3-2: Simple variables, arrays et un peu de génération HTML

url: </guides/php/exemples/simple/simple-arrays.php>

```
<?php

// Some simple HTML
echo "<h1>Simple arrays</h1>";

$utilisateur = "cher étudiant";
$no_utilisateur = 3;

$nombre = array (1, 2, 3, 4);
$noms = array ("Pat", "Dave", "Surf", "K");
$noms[] = "Zorro";

// Note html <br> tag
echo "Salut $utilisateur. Vous êtes le numéro $no_utilisateur.<br>";

// echo with concatenation, use it to print complex things
echo "La quatrième personne s'appelle " . $noms[3] . " ";

// simple echo
echo "et la cinquième personne s'appelle $noms[4].<p>";
$n = sizeof($nombre);

// note that we have to use \ in order to print a $ !
echo "We have $n numbers in array \$nombre.";
?>
```

C. Arrays associatifs et multi-dimensionnels

(pas obligatoire au début !)

```
$fruits = array(
    "fruits" => array("a"=>"orange", "b"=>"banana", "c"=>"apple"),
    "numbers" => array(1, 2, 3, 4, 5, 6)
    "holes" => array("first", 5 => "second", "third")
);
```

D. Récapitulation variables

- Il n'est pas nécessaire de déclarer une variable au préalable, mais c'est conseillé
- Voici les 5 types (avec exemple):

```
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 1.234; $a = 1.2e3; # floating point number
$str = "This is a string"; # chaine de caractères
$a[0] = "abc"; # élément 0 d'un array
$a[1] = "def"; # élément 1 d'un array
$b["foo"] = 13; # élément "foo" d'un array
```

3.5 Simple expressions et opérateurs

A. Opérateurs arithmétiques

- Comme les maths "normales":

exemple	nom	Retourne le resultat:
$\$a + \b	Addition	Somme de $\$a$ et $\$b$
$\$a - \b	Soustraction	Reste de la différence de $\$b$ et $\$a$
$\$a * \b	Multiplication	
$\$a / \b	Division	
$\$a \% \b	Modulo	Reste de la division entière de $\$a$ par $\$b$

- Note: Il existe des fonctions PHP pour effectuer d'autres calculs, par exemple `max()` et `min()` voir le manuel.

Exemple 3-3: Simple Arithmétique

Application	<u>url: /guides/php/examples/simple/simple-calcul.php</u>
Source	<u>url: /guides/php/examples/simple/simple-calcul.phps</u>
Pour copier	<u>url: /guides/php/examples/simple/simple-calcul.text</u>

```
$leisure_satisfaction = 5;  
$work_satisfaction = 7;  
$family_satisfaction = 8;
```

```
$index = ($leisure_satisfaction + $work_satisfaction + $family_satisfaction) / 3 ;
```

```
echo "<p align=center> Satisfaction Index = $index <b>";
```

Assignment + addition en une seule instruction:

```
// sets $a to 8, as if we had said: $a = $a + 5;  
$a += 5;
```

B. Opérateurs sur les chaînes

Addition de chaînes de caractères (concatenation)

Utiliser le ".", exemple:

```
$a = "Hello ";
$b = $a . "World!"; // now $b = "Hello World!"
```

- **Note:** Il existe de fonctions PHP pour manipuler des strings

Assignment + concatenation en une seule fois

```
$b = "Hello ";
// sets $b to "Hello There!", just like $b = $b . "There!";
$b .= "There!";
```

C. Opérateurs logiques

example	name	result
<code>\$a and \$b</code>	"et"	Résultat vrai si \$a et \$b sont vrais
<code>\$a && \$b</code>	"et"	"
<code>\$a or \$b</code>	"ou"	Résultat vrai si \$a ou \$b ou les deux sont vrais
<code>\$a \$b</code>	"ou"	"
<code>\$a xor \$b</code>	Or exclusif	Résultat vrai si \$a ou \$b sont vrais, mais pas les deux
<code>! \$a</code>	"ne pas"	Résultat vrai si \$a n'est pas vrai (est-ce que \$a est faux?)

D. Opérateurs de comparaison

exemple	name	result
<code>\$a == \$b</code>	égal	True if \$a is equal to \$b.
<code>\$a != \$b</code>	différent	True if \$a is not equal to \$b.
<code>\$a < \$b</code>	inférieur	True if \$a is strictly less than \$b.
<code>\$a > \$b</code>	supérieur	True if \$a is strictly greater than \$b.
<code>\$a <= \$b</code>	inférieur ou égal	True if \$a is less than or equal to \$b.
<code>\$a >= \$b</code>	supérieur ou égal	True if \$a is greater than or equal to \$b.

- Utilisez des parenthèses en cas de doute !

Exemple 3-4: Comparaisons simples

url: </guides/php/exemples/simple/simple-compare.php>

url: </guides/php/exemples/simple/simple-compare.phps>

- **Note:** "TRUE" et "FALSE" sont représentés par 1 et 0 dans PHP.

```
$a = "Migros";
$b = "Coop";
$result = $a==$b;
$result2 = $a > $b;
$result3 = $result==TRUE;
echo "Result One = $result. ";
echo "Result TWO = $result2. ";
echo "Result THREE = $result3.";
```

3.6 Sélection (Conditions et tests)

Principe (plusieurs situations typiques):

- Si une condition est vraie alors faire ceci.
- Si une condition est vraie alors faire ceci, sinon faire cela.
- Si une condition est vraie alors faire ceci, sinon si une autre condition est vraie faire autre chose, sinon

"IF" (plusieurs variantes)

Syntaxe: `if (expr) statements`

Syntaxe: `if (expr) statements else statements`

Syntaxe: `if (expr) statements elseif (expr) statements else ...`

Syntaxe: `if (expr) statements elseif (expr) statements [elseif (expr) ...]`

- *expr* = Expression qui doit retourner une valeur TRUE ou FALSE
- *statements* = simple instruction ou bloc d'instructions
 - simple: `$a = 10;`
 - bloc: `{ $a =12; echo "salut"; }`
- déroulement de l'exécution:
 - Lorsque l'expression = TRUE on exécute le(s) statements
 - Lorsque l'expression = FALSE on passe à la clause suivante

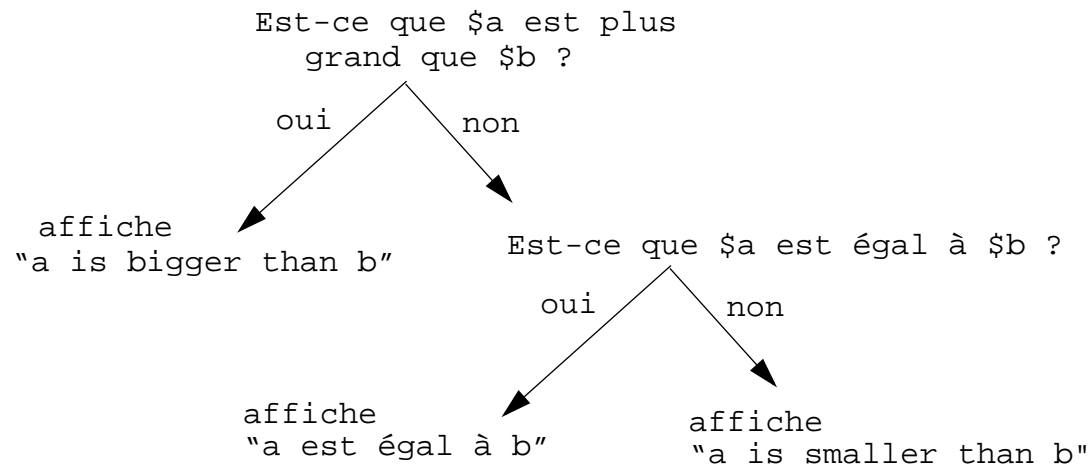
Exemple 3-5: Simple "if" (comparaison)

url: </guides/php/examples/simple/simple-if.php>

url: </guides/php/examples/simple/simple-if.php> (source)

- Cet exemple compare deux nombres \$a et \$b, et affiche un message en fonction du test.
- L'arbre de décision ci-dessous illustre l'ordre des tests qui sont effectués à cet effet.

Figure 3-2: Simple arbre de décision



```
<?php

$a = 10; $b = 11;
print "a was $a, b was $b. ";
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a est égal à b";
} else {
    print "=> a is smaller than b.";
}

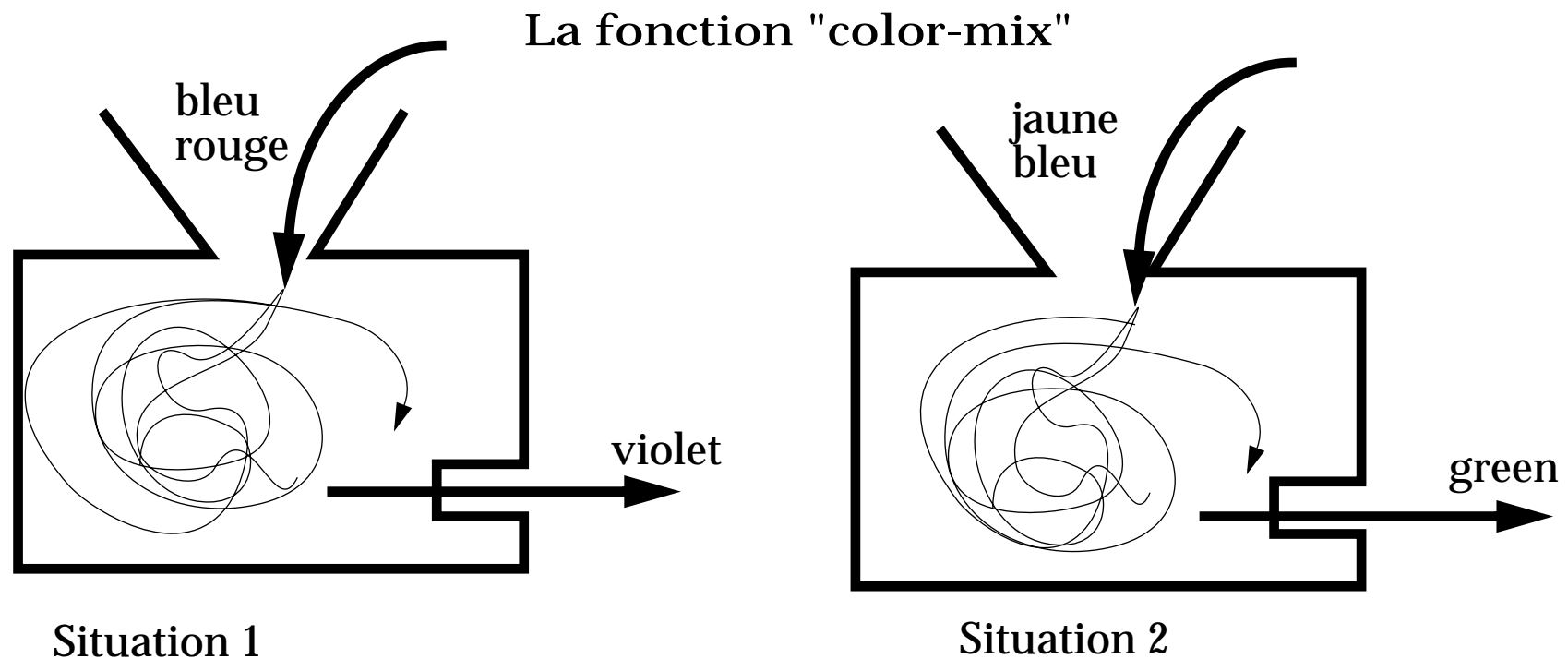
?>
```

Voir aussi les instructions suivantes:

- **switch**
- **foreach**
- **do ... while**
- **break et continue**

3.7 Fonctions PHP

- Comme tous les langages de programmation PHP permet de définir des procédures/fonctions.
- Une fonction dans php est un bout de programme nommé (qu'on peut donc "appeler" et qui fait quelque chose (éventuellement avec des arguments)
"Traite moi ces informations et retourne-moi le résultat"
- On les place au début du fichier car elles doivent être définies avant d'être utilisées



Exemple 3-6: Génération HTML simple avec des fonctions

url: </guides/php/examples/simple/function-demo.php>

```
<?php

// html formats a data element
function pretty_print ($output) {
separator ();
echo "<p align='center'> <strong>ELEMENT:</strong> $output </p>";
}
// outputs a separator
function separator () {
echo "<hr size=4 width=70%>";
}
// data we have
$e11 = "Un arbre jaune";
$e12 = "Ein gelber Hund";
$e13 = "A yellow sky";
// dump the data
pretty_print($e11);
pretty_print($e12);
pretty_print($e13);

separator ();
echo "<hr>";
?>
```


3.8 Boucles "for" et génération HTML

A. Introduction à la boucle "for"

Syntaxe: "boucle FOR":

```
FOR (expr1; expr2; expr3) statement
```

- expr1 est évaluée au début du loop
- expr2 est évaluée au début de chaque boucle, si le résultat = TRUE la boucle continue, sinon on sort de la boucle
- expr3 est évaluée à la fin de chaque boucle,
- statement est exécuté à l'intérieur de chaque boucle.

Exemple 3-7: Love generation

url: voir: </guides/php/examples/html-generate/love.php>

url: voir: </guides/php/examples/html-generate/love.php>s

```
for ($i=1; $i<=10; $i++) {  
    print "I love you so ! ";  
}
```

Résultat:

love you so ! I love you so ! I love you so ! I love you so ! I love you so ! I love you so !

```
for ($i=2; $i<=10; $i++) {  
    echo "Non, je t'aime $i fois plus que moi ! ";  
}
```

Résultat:

Je t'aime plus que moi. Non, je t'aime 2 fois plus que moi ! Non, je t'aime 3 fois plus que moi ! Non, je t'aime 4 fois plus que moi ! Non, je t'aime 5 fois plus que moi ! Non, je t'aime 6

Autres éléments PHP:

- `$i` est utilisée comme variable d'itération. Au début `$i = 1` ou `2`.
- `echo` imprime un ou plusieurs string(s) (et substitue les variables)
- `print` imprime un string (et substitue les variables) ... pareil que `echo` mais c'est une fonction.

B. Fonctions PHP et arrays (génération d'une table HTML)

- `array()` permet de définir un vecteur
- `fonction () { ... }` définit une fonction
- `$<variable>[<entier>]` accède à un élément d'un vecteur

Exemple 3-8: Génération de tables html

url: voir: </guides/php/examples/html-generate/love.php>

url: voir: </guides/php/examples/html-generate/love.phps>

url: voir: </guides/php/examples/html-generate/love.text>

```
$love_list = array ("a lot", "a bit", "somewhat", "à mourir", "forever", "until  
notice", "more than I love my dog");
```

```
<table border align="center">  
<?  
// define a function to generate a table  
function build_table($list) {  
    for ($i=0; $i < sizeof($list); $i++) {  
        $love_text = $list[$i];  
        echo "<tr> <td> ... I love you</td> <td>$love_text</td>";  
    }  
}  
// call the function, generate the table  
build_table($love_list);  
>  
</table>
```

Notez:

- qu'on insère du PHP à l'intérieur d'un tag <table>
- qu'on appelle la fonction `build_table` avec comme argument un vecteur (donc la fonction pourrait être utilisée ailleurs, avec un `include` par exemple)

4. Conseils pratiques pour PHP

4.1 Debugging

- Regardez le code HTML qui est généré (Faites "View Source")
- Pour obtenir un maximum d'information sur la configuration de Php ainsi que sur les variables transmises au programme, insérer qq part dans le fichier:

```
phpinfo();
```

toute l'information ne vous sera pas forcément utile

- Si vous avez un doute sur l'information contenue dans une variable, imprimez!

```
echo "DEBUG: \$var = $var";
```

```
echo "TEST: var = $var";
```

- Insérer au début du fichier (voir PHP options & information dans le manuel) vous permet d'obtenir plus d'avertissements que d'habitude

```
error_reporting(63);
```

4.2 PHP en "Stand-alone"

- Il est possible d'utiliser PHP en "stand-alone"
 - Exemples d'utilisation:
 - Génération de pages HTML statiques (php -> html)
 - Filtres, outils de conversion
 - Pour s'initier à la programmation avec PHP
 - Pour s'initier à la programmation d'outils de "massage"
 - Ce qui ne marche pas:
 - Traitement de formulaires (il faut un serveur pour cela!)
 - Il faut prendre une version "cgi" et non pas "Apache Module"
- url:* Exemples: </guides/php/examples/command-line-php/>



A. Usage:

- soit sous forme de script:
 - sous Unix (ou Windows seulement avec un shell comme bash installé!)
 - la première ligne du script doit indiquer le nom du binaire

```
#!/local/bin/php -q
```
 - remplacer /local/bin/ par l'endroit où se trouve votre script sur VOTRE machine)
 - il faut rendre exécutable le fichier (chmod u+x sous Unix)
- soit avec les formes suivantes (appel dans une fenêtre DOS !!)
 - Si php se trouve dans le PATH ou si vous êtes dans le répertoire PHP:

```
php -q <nom_du_fichier.php>
```
 - Si php ne se trouve pas dans le PATH:

```
\<chemin>\php.exe -q <nom_du_fichier.php>
```
 - L'option "-q" sert à supprimer les header lines HTTP

4.3 Win95:

- Si vous voulez faire des choses plus sophistiqués localement:
 - (command line input par exemple), il faut aussi installer GNU Win32
url: Voir: <http://www.cygnum.com/misc/gnu-win32/>
- utile pour d'autres trucs, mais déconseillé aux débutants !

A. Les conseils de Olivier:

Pour installer Php sur son PC (voir les pages ressources PHP pour le trouver)

1. Décompresser le fichier Php-**-*win32.zip dans le repertoire "C:\Php"
2. Renommer le fichier php3.ini-dist en php3.ini, et le déplacer dans le repertoire c:\windows
3. Cliquer avec le bouton droit sur "php_iis_reg.ing" et faire Installer
4. Ajouter la ligne SET PHPRC=c:\windows\php.ini a la fin de votre Autoexec.bat
5. Redemarrer la machine.
6. Avant de demarrer Php.exe, il peut etre utile de modifier le fichier php.ini en fonction de vos besoins (lire la documentation).

