Module 3 Les bases de PHP

Développement web HENALLUX — IG3 — 2016-2017

PHP « in use »

```
Processeur
PHP
```

```
<html>
 <head></head>
 <body>
   <h1>Bienvenue !</h1>
   Il est actuellement
   <?php
     $heure = date('H:i');
     echo $heure;
    ?>
   et tout va bien.
 </body>
</html>
```

```
<html>
    <head></head>
    <body>
        <h1>Bienvenue !</h1>
        Il est actuellement
        11:18
        et tout va bien.
        </body>
    </html>
```



Bienvenue!

Il est actuellement 11:18 et tout va bien.

Au programme...





- Caractéristiques générales du langage
- Données, instructions, éléments du langage, tableaux...
- Le langage PHP : aspects orientés objets
- Échanges de données entre le client et le serveur

Le langage PHP: généralités

- Règles syntaxiques générales
 - Identificateurs, blancs, indentation...
- Sorties vers le document
 - Comment produire du code HTML/Javascript
- En cas d'erreurs...
 - Aperçu de la gestion des erreurs
- Programmation modulaire
 - Découper un fichier en plusieurs parties

Ensuite : le langage PHP, syntaxe et sémantique

Syntaxe (généralités)

- [idem Javascript] En PHP, tous les blancs (espaces, tabulations, retours à la ligne) sont ignorés, ce qui permet une mise en page claire des programmes (indentations, lignes blanches...).
- [idem Javascript] Deux formats pour les commentaires :
 - // pour tout le reste de la ligne (aussi avec # au lieu de //)
 - /* ... */ pour plusieurs lignes ou une partie de ligne.
- Chaque instruction PHP doit se terminer par un «; ». Le dernier; avant la balise fermante?> peut être omis.

Javascript : on peut parfois remplacer ; par un retour à la ligne.

Syntaxe (généralités)

- En PHP, les identificateurs de variables commencent tous par un \$ suivi par leur nom.
 - Le nom doit commencer par une lettre ou _ et être composé de lettres, de chiffres et/ou de _.
- [idem JS] Les noms de variables PHP sont <u>sensibles à la casse</u>.
 Exemple : mavariable ≠ maVariable ≠ MaVariable ≠ MAVARIABLE
- Par contre, les noms des fonctions, classes et les mots-clefs, eux, ne sont pas sensibles à la casse!

Syntaxe (généralités)

Les identificateurs ne peuvent pas être des mots réservés :

and	as	break	case
class	const	continue	declare
default	die	do	echo
else	empty	eval	exit
for	foreach	function	global
if	include	list	new
or	print	require	return
switch	use	var	while

Sorties (écran ou document)

Sorties simples

```
echo(expr, expr, ...)
print(expr)
```

(parenthèses optionnelles) (parenthèses optionnelles)

Autres options

```
printf(format [, args]* )
sprintf(format [, args]* )
var_export(expr)
print_r(expr)
var_dump(expr)

Pratiques pour les tests...
```

```
(mêmes formats qu'en C)
(idem mais fournit un string)
(affiche au format PHP)
(affichage du contenu)
(affiche le type et la valeur)
```

Erreurs

- Trois niveaux d'erreurs :
 - **ERROR**: interrompt l'exécution (point-virgule manquant, absence d'un fichier nécessaire)
 - **WARNING**: l'exécution se poursuit (argument manquant, essayer de modifier la valeur d'une constante...)
 - NOTICE : l'exécution se poursuit (écrire une variable non définie)
- Forcer une erreur
 - die('message') interrompt l'exécution et affiche le message
 - Souvent utilisé avec une fonction qui renvoie TRUE si tout se passe bien :

```
mysql_connect("host", "user", "pwd") or die('La BD est KO !');
```

PHP, fonctionnement général

- Où placer le code PHP ?
 - Option #1 : directement dans le document-type HTML, à l'endroit où le résultat doit être placé : <?php ...code php... ?>
 - Option #2 : dans un fichier à part qui ne contient que du PHP (module)

```
<?php ... ?> (balise fermante optionnelle dans ce cas-là)
```

- Notes
 - Il existe des versions raccourcies des balises, qu'il vaut mieux éviter.
 - Si un document HTML comporte des scripts PHP, il doit porter l'extension .php (pour signaler au serveur http qu'il doit être traité avant d'être envoyé).

Programmation modulaire

- Pour incorporer des fichiers annexes :
 - include "fichier.php" comme si on copiait/collait le contenu du fichier à cet endroit (simple avertissement si le fichier n'existe pas);
 - require "fichier.php" idem, mais erreur fatale si le fichier n'existe pas;
 - include_once "fichier.php" require_once "fichier.php" n'incorpore le contenu que si le fichier n'a pas encore été incorporé;
- Tout se passe comme si le contenu du fichier était copié à l'endroit de l'include mais en-dehors des balises <?php ?>.
- Si le fichier contient du code, il doit donc être encadré de balises
 <?php ?>!
- Pour les modules, on utilise également l'extension .php (parfois .inc).
- Dans un module, c'est une bonne idée de ne pas mettre de balise fermante pour éviter d'ajouter des espaces en trop à la fin du document.

Programmation modulaire

• Exemple :

```
<html>
    <head></head>
    <body>
        <h1>Bienvenue !</h1>
        Il est actuellement
        <?php
            include("heure.php");
            ?>
            et tout va bien.
        </body>
    </html>
```



```
<?php
    $heure = date('H:i');
    echo $heure;
?>
sur le <em>serveur</em>
```

```
<html>
    <head></head>
    <body>
        <h1>Bienvenue !</h1>
        Il est actuellement
</php
        $heure = date('H:i');
        echo $heure;

}

sur le <em>serveur</em>
        et tout va bien.
        </body>
    </html>
```

PHP: syntaxe et sémantique





 Variables, types de données, valeurs et littéraux, opérateurs

Instructions

Fonctions (modules)

Tableaux

Ensuite : l'orienté objet en PHP

PHP: données manipulées

- Variables
 - Déclaration, type d'une variable, affectation
- Types de données
 - entiers, réels, booléens, ...
- Valeurs, littéraux et opérateurs
- Constantes

Ensuite: instructions en PHP

Variables: déclaration

- Aucune déclaration de variable en PHP!
 Une variable est déclarée quand on lui donne une valeur pour la première fois, ce qu'il faut faire avant de pouvoir l'utiliser.
- Tests courants
 - isset(\$v): vrai si la variable existe et ne vaut pas NULL
 - empty(\$v): vrai si
 - \$v n'existe pas ou
 - \$v vaut FALSE (ou 0 ou "" ou "0" ou NULL ou [])
- Utilisation des variables

Variables: affectation

Affectation par valeur (standard)

```
$a = 'ville';
$b = $a;
```

Affectation par référence (création d'alias)

```
$b = &$a;
$a = 'village'; // modifie aussi $b !
```

Affectation combinée avec une opération (syntaxe standard)

```
+= -= *= /= %= .= (concaténation)
```

Affectation multiple

```
a = b = 0;
```

Variables: typage

- Comme Javascript, PHP n'est pas typé :
 - On ne déclare pas le type des variables / arguments de fonctions.
 - Le type d'une variable peut changer au cours d'un programme.
 - Pas de vérification de type.
 - Nombreuses conversions implicites.
- Les conversions sont effectuées en fonction de ce que demande le contexte.

Types de données

La fonction gettype() peut indiquer :

```
    "boolean": valeurs booléennes
        "integer": entiers
        "double": réels
        "string": chaînes de caractères

    "array": tableaux (un type à part, contrairement à Javascript)
    "object": objets
    "resource": pour les ressources binaires (données génériques)
    "NULL": pour la valeur NULL
```

 Fonctions pour tester le type d'une valeur : is_boolean, is_integer, is_double, is_string, is_array, is_object, is_ressource, is_null, is_numeric (entier ou réel)

Types de données

- On peut **forcer une conversion** explicitement en utilisant une syntaxe semblable à celle de Java et de C :
 - (int)expr, (integer)expr ou intval(expr)
 - si \$var est un réel, on prend sa partie entière ;
 - si \$var est un string, on utilise le plus long préfixe possible Exemple : '12345EDCBA' -> 12345
 - (float)expr, (double)expr, (real)expr ou floatval(expr)
 - (string)expr ou strval(expr)
 - (bool)expr ou (boolean)expr
 - (array)expr
 - (object)expr
- On peut également changer le type d'une variable avec settype(\$x, 'int').

Les valeurs numériques

Les entiers (integer) et les réels (float)

```
42 -78 342.17 1.36E78 037 (octal) 0x3BFF (hexa)
```

- Contrairement à Javascript, PHP différencie les entiers des réels.
- PHP possède également une valeur NAN (not a number) et la fonction test is nan(expr).

```
Ex: var = acos(2); résulte en var = NAN
```

Opérateurs numériques

```
+ - * / % ++ --
```

Note : la division fournit un réel, même pour des opérandes entières.

Plus de nombreuses fonctions mathématiques (abs, exp...)

Les valeurs booléennes

Les booléens (boolean)

true false TRUE FALSE

- Autres valeurs interprétées comme FALSE : 0 (entier), 0.0 (réel), "" (chaîne vide), "0" (chaîne avec 0), [] (tableau vide), {} (objet vide), NULL.
- En Javascript, "0" est converti en true!
- [ATTENTION] Affichage des valeurs booléennes

```
echo TRUE produit 1
echo FALSE produit (rien)
```

Note. FALSE est en fait implémenté comme NULL.

Opérateurs booléens

```
! && and || or xor ? : (opérateur conditionnel)
```

Note : les opérateurs && et | | d'une part et and et or d'autre part sont identiques à la priorité près ; en cas de doute, utilisez des parenthèses !

La valeur null

- Valeur null (ou NULL)
 - Cette valeur représente les variables "sans valeur".
 - PHP n'a pas de valeur "undefined" contrairement à Javascript.
 - La fonction isset(\$var) permet de tester si la variable en question possède une valeur (donne false si \$var est null).
 - Aussi: unset(\$var); pour supprimer une variable

Les chaînes de caractères

- PHP offre plusieurs types de chaînes de caractères (string).
- 1) Chaînes littérales (avec des apostrophes)

```
    Caractères échappés : \' \\ (les autres sont écrits tels quels)
    Ex : 'aujourd\'hui' 'En PHP, on écrit $nomVar.'
```

- 2) Chaînes non-littérales (avec des guillemets)
 - Les variables y sont remplacées par leur valeur !

Note: on peut aussi utiliser \xA5 ou \x12B6 (code hexa) et \123 (code octal).

Les chaînes de caractères

3) Chaînes "here-document" (ou heredoc)

Pour les messages multi-lignes avec une mise en page spécifique

4) Chaînes "now-document" (ou nowdoc)

Comme heredoc mais les variables ne sont pas remplacées

Les chaînes de caractères

- Concaténation de chaînes de caractères : . \$txt1.\$txt2
 En Javascript, il est noté + !
- Accès aux caractères d'une chaîne : \$txt[3] ou \$txt{3}
- Affichage formatté (similaire au C) :printf(format,...,...) ou \$res = sprintf(format,...,...)
- Longueur d'une chaîne de caractères : strlen(\$s)

Opérateurs et conversion

Opérateurs de comparaison

```
== : égalité ; === : identité
!= ou <> === !===
```

Note : les deux derniers testent l'égalité sans conversion implicite.

Conversions implicites: si possible, convertir en nombres.

```
echo '13' + '17 vaches'; affiche 30!

$txt1 = '12';

$txt2 = '+12';

$txt3 = '12pommes';

echo ($txt1 == $txt2); affiche 1 (= vrai)!

echo ($txt1 === $txt2); affiche <rien> (= faux)!

echo ($txt1 == $txt3); affiche 1 (= vrai)!

echo (12 == $txt3); affiche 1 (= vrai)!
```

Constantes

PHP permet de définir des constantes

```
define('MACONSTANTE', 'le contenu de ma constante');
En Javascript, pas de véritable notion de constante :
var MACONSTANTE = "le contenu de ma constante";
```

• Les constantes s'utilisent sans \$.

 Les constantes ne sont pas remplacées par leur valeur dans un string encadré de guillemets.

```
define('PI', 3.14); $e = 2.72;
echo "Pi = PI et e = $e."; produit "Pi = PI et e = 2.72."
echo 'Pi = PI et e = $e.'; produit "Pi = PI et e = $e."
```

Constantes

- Quelques remarques :
 - Une fois définie, une constante est visible partout.
 - Si on utilise MACONSTANTE sans la définir d'abord, PHP suppose qu'elle contient la chaîne "MACONSTANTE".
 - Quelques constantes PHP prédéfinies :
 - __LINE__ : numéro de la ligne courante dans le fichier
 - __FILE__ : fichier courant (__DIR__ pour son répertoire)

	Constantes	Variables
Utilisation	Sans \$	Avec\$
Déclaration	Via define	Lors de la première affectation
Visibilité	Partout	Locale ou globale
Valeur	Doit être scalaire	Quelconque
Modification	Impossible	Possible, ainsi que unset

PHP: syntaxe et sémantique

- Données manipulées
 - Variables, types de données, valeurs et littéraux, opérateurs
- Instructions



- Fonctions (modules)
- Tableaux

Ensuite : l'orienté objet en PHP

PHP: les instructions

Instructions conditionnelles

Instructions itératives (boucles)

Syntaxe alternative

Ensuite : les fonctions en PHP

Instructions conditionnelles

Structure conditionnelle simple

```
if (condition)
    instruction
[else instruction]
```

Version simplifiée pour les if imbriqués

```
if (condition)
    instruction

[elseif (condition)
    instruction ]*

[else instruction]
```

```
if ($prix < 10)
    echo 'Pas cher';
elseif ($prix < 100)
    echo 'Assez cher';
elseif ($prix < 1000)
    echo 'Très cher';
else
    echo 'Hors de prix';</pre>
```

Instructions conditionnelles

Structure conditionnelle avec gardes

```
switch (expr)
{
    case val : instruction
    ...
    [default : instruction]
}
```

Note : ajouter des break pour éviter de passer d'un cas à l'autre.

Instructions itératives

Boucles while et do (aussi : boucle foreach pour les tableaux)

```
while (condition)
    instruction
    while (condition);
```

Boucle for (voir aussi for-in pour les objets)

```
for (init; condition; incrémentation)
    instruction
Note. Utilisation de, pour les composantes complexes du for.
```

Échappements

break continue

Syntaxe alternative

• Écriture alternative : remplace les accolades par des deuxpoints et endif/endswitch/endfor/endwhile/endforeach :

```
if ($cote >= 10):
    echo 'Réussi';
    $ccl = 'Bravo';
else:
    echo 'Raté';
    $ccl = 'Bouh';
endif;

while ($i > 0):
    echo $i, ' ';
    $i--;
endwhile;
```

```
if ($prix < 10):
    echo 'Pas cher';
elseif ($prix < 100):
    echo 'Assez cher';
elseif ($prix < 1000):
    echo 'Très cher';
else:
    echo 'Hors de prix';
endif;</pre>
```

PHP: syntaxe et sémantique

- Données manipulées
 - Variables, types de données, valeurs et littéraux, opérateurs
- Instructions
- Fonctions



Tableaux

Ensuite : l'orienté objet en PHP

PHP: les fonctions

Déclaration de fonctions

- Portée des variables
 - Variables locales et variables globales
- Variables superglobales

Ensuite: les tableaux en PHP

Définition d'une fonction

Déclaration d'une fonction

```
function nom ([$ident[, $ident]*]) {
    instructions
}
```

- Sortie ou valeur de retour d'une fonction return [expr]
- Appel de fonction : nom ([expr [, expr]*])
 Pas de vérification de type mais (contrairement à Javascript,) il faut passer au moins le nombre d'arguments de la définition !
- Les identificateurs de fonction ne sont pas sensibles à la casse.
 (Mais respectez la tout de même Clean code!)

Fonctions anonymes

Fonctions sous la forme d'expressions (fonctions anonymes)
 function ([\$ident[, \$ident]*]) { instructions}

```
Exemple: $double = function ($x) { return $x * 2; };
echo $double(3);
```

Définition de fonctions : portée

 PHP lit toutes les définitions de fonctions situées <u>au niveau</u> global avant de commencer l'exécution : on peut donc utiliser une fonction définie plus bas.

Exemple:

(affiche 9)

```
f(3);
function f($x) {
  echo $x*$x;
}
```

Définition de fonctions : portée

• Lorsque PHP rencontre une définition de fonction, il définit celle-ci au <u>niveau global</u>.

```
Ex: function f () {
    function g() { echo 'fonction g'; }
    echo 'fonction f';
}
g(); // produit une erreur : g pas définie
f();
g(); // fonctionne correctement
```

Définition de fonctions : portée

• Lorsque PHP rencontre une définition de fonction, il définit celle-ci au niveau global.

- En PHP : PAS de redéfinition/surcharge (sinon, erreur fatale !)
- Quelques cas particuliers :
 - Valeurs par défaut
 - Passage par référence/adresse
 - Renvoyer plusieurs valeurs en même temps
 - Nombre d'arguments variable

• Valeurs par défaut pour les arguments (il doit s'agir des derniers arguments dans la liste) :

```
function pourcentage ($cote, $max = 20) {
  return ($cote / $max) * 100;
}
pourcentage(15);
```

Passage par référence (automatique pour les objets) :

```
function swap (&$x, &$y) {
    $tmp = $x;
    $x = $y;
    $y = $tmp;
}

$nb1 = 3;
$nb2 = 7;
swap($nb1,$nb2);
```

• Fonctions qui renvoient plusieurs valeurs : les assembler dans un tableau (voir plus loin).

```
function plusEtFois ($x, $y) {
   $somme = $x + $y;
   $prod = $x * $y;
   return array($somme, $prod);
}
```

On peut décomposer le tableau reçu grâce au mot-clef list.

```
list($total,$produit) = plusEtFois(5,7);
```

- Fonctions à nombre d'arguments variable
 - func_num_args() : nombre d'arguments transmis à la fonction
 - func_get_arg(i) : le i^e argument (en comptant à partir de 0)
 - func_get_args : tous les arguments (tableau)

```
function moyenne () {
    $n = func_num_args();
    $total = 0;
    for ($i = 0; $i < $n; $i++) {
        $total += func_get_arg($i);
    }
    return ($total/$n);
}</pre>
```

Portée des variables

```
$intro = 'Le résultat est ';
$outro = '.<br/>';
$portée globale
```

```
function calculeDouble ($val) {
  $double = $val * 2;
  return $double;
}
portée locale
```

```
function afficheDouble ($v) {
   global $intro, $outro;
   $d = calculeDouble($v);
   echo $intro, $d, $outro;
}
```

\$v, \$d (autre) portée locale

Les variables globales ne sont pas automatiquement accessibles en local!

Portée des variables

```
$prix = 15;

function prixHorsSaison() {
    $prix = 10;
    echo "Le prix passe à $prix.<br/>}

$prix

$prix

$prix

$var locale
```

```
echo "Le prix est $prix. <br/>prixHorsSaison();
echo "Le prix est $prix. <br/>";
```

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 15.
```

Portée des variables

```
$prix = 15;

function prixHorsSaison() {
   global $prix;
   $prix = 10;
   echo "Le prix passe à $prix.<br/>}
```

```
echo "Le prix est $prix. <br/>prixHorsSaison();
echo "Le prix est $prix. <br/>";
```

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 10.
```

Portée des variables

```
$prix = 15;

function prixHorsSaison(&$prix) {
    $prix = 10;
    echo "Le prix passe à $prix.<br/>}

$prix \( \frac{1}{2} \)
$prix \( \frac{1} \)
$prix \( \frac{1}{2} \)
$prix \( \frac{
```

```
echo "Le prix est $prix. <br/>prixHorsSaison($prix);
echo "Le prix est $prix. <br/>";
```

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 10.
```

Portée des variables : variable statique (comme en C)

```
prix = 15;
                                               $nbModif
                                               var locale statique
function modifPrix($modif) {
  global $prix;
                                    La variable n'est initialisée
  static $nbModif = 0;
                                    à 0 que la première fois.
  $prix += $modif;
  $nbModif++;
  echo "Le prix passe à $prix ({$nbModif}e modif).<br/>";
                     Affichage
modifPrix(+5);
                     Le prix passe à 20 (1<sup>e</sup> modif).
modifPrix(-3);
                     Le prix passe à 17 (2<sup>e</sup> modif).
modifPrix(+7);
                     Le prix passe à 24 (3<sup>e</sup> modif).
```

Variables superglobales

- Variables superglobales (= prédéfinies et accessibles partout !)
 - \$GLOBALS : toutes les variables définies dans le contexte global (dont les variables globales déclarées : \$_GLOBALS['mavar'])
 - \$_SERVER : informations provenant du serveur web
 - \$ ENV : informations sur l'environnement et l'OS
 - \$_GET, \$_POST : variables reçues dans la requête http
 - \$ FILES : fichiers attachés à la requête http
 - \$_COOKIE : variables passées sous la forme de cookies http
 - \$_REQUEST : l'ensemble des informations attachées à la requête http (redondant avec \$_GET, \$_POST, \$_COOKIE et \$_FILES).
 - \$_SESSION: variables de session
- Ces variables prennent la forme de tableaux associatifs.
- Sécurité contre les injections :
 \$origine = htmlentities(\$_SERVER["HTTP_REFERRER"]);

PHP: syntaxe et sémantique

- Données manipulées
 - Variables, types de données, valeurs et littéraux, opérateurs
- Instructions
- Fonctions
- Tableaux



Ensuite : l'orienté objet en PHP

PHP: les tableaux

- Tableaux numériques et tableaux associatifs
- Tableaux multidimensionnels
- Utilisation des tableaux
 - Résultat d'une fonction, argument d'une fonction, foreach

Tableaux

• En PHP, tous les tableaux sont des **tableaux associatifs** (associations clefs/valeurs).

```
Exemple: père -> Homer
mère -> Marge
fils -> Bart
fille -> Lisa
```

• Les tableaux à index numérique sont un cas particulier (les clefs sont 0, 1, 2, 3...) pour lequel il existe certaines facilités syntaxiques.

```
Exemple: 0 -> Lisa
1 -> Bart
2 -> Maggie
```

Tableaux

- En PHP, les tableaux associatifs peuvent avoir pour clefs
 - des chaînes de caractères ou
 - des entiers.

Mais les chaînes correspondant à un entier valide sont automatiquement converties en entier : $tab[3] \equiv tab["3"]$

- Si on utilise d'autres types de clefs :
 - clefs réelles : converties en entiers
 - clefs booléennes : converties en entiers (0 ou 1)
 - valeur « null » : convertie en une chaîne vide ""
 - tableaux / objets : erreur Illegal Offset Type !

Tableaux à index numérique

Tableaux hétérogènes à index numérique

```
$tab1 = []; // tableau vide
$tab2 = array(); // tableau vide
$tab3 = [2, 7, "hello", TRUE]; // tableau à 4 cases
$tab4 = array(2, 7, "hello", TRUE); // le même tableau
```

- Accès aux éléments : \$tab[index] ou \$tab{index}
- Écriture dans la case suivante

```
$tab5 = [2, "oui"];
$tab5[] = "ensuite";
$tab5[] = 47;
Aussi:array_push($tab5, "ensuite", 47);
```

Tableaux associatifs

Tableaux associatifs

- Accès aux éléments : \$tab[clef] ou \$tab{clef}
- Note : on peut également construire des tableaux mixtes.

```
$tab = array(2, 4, 7, 10);
$tab['info'] = 'Data';
```

Tableaux multidimensionnels

- Simulés par des tableaux dont les éléments sont des tableaux.
- Tableaux numériques multidimensionnels

```
$matrice = array (
          array (1, 0, 0, 0),
           array (0, 1, 0, 0),
          array (0, 0, 1, 0),
           array (0, 0, 0, 1)
);
```

Accès aux éléments

```
$matrice[2][2]
```

Tableaux multidimensionnels

Tableaux multidimensionnels

```
$perso = array (
    'Simpson' => array (
        'père' => 'Homer',
        'mère' => 'Marge'),
    'Nahasapeemapetilon' => array (
        'père' => 'Apu',
        'mère' => 'Manjula'),
    'van Houten' => array (
        'père' => 'Kirk',
        'mère' => 'Luann')
);
```

Accès aux éléments

```
$perso['van Houten']['père']
```

Utilisation des tableaux

• Décomposition (des premières valeurs) d'un tableau

```
$tab = array(4,7,9,10);
list($a,$b) = $tab; // met 4 dans $a et 7 dans $b
```

Fonction renvoyant plusieurs valeurs

```
function carreCube ($x) {
    $tab[0] = $x * $x;
    $tab[1] = $tab[0] * $x;
    return $tab;
}
$res = carreCube(4);
echo $res[0], ' ', $res[1], '<br/>';
list ($carre, $cube) = carreCube(5);
echo $carre, ' ', $cube, '<br/>';
```

Utilisation des tableaux

Les tableaux fonctionnent par valeur ! (contrairement à JS)

```
$tab = [1, 2, 3, 4];

$tab2 = $tab; // $tab2 est une copie de $tab

$tab2[0] = 7; // $tab est inchangé

function f($t) { $t[0] = 7; }

f($tab); // $tab est inchangé : f travaille sur une copie
```

Travailler explicitement par adresse :

```
$tab2 = &$tab; // $tab et $tab2 désignent le même tableau
function f(&$t) { $t[0] = 7; }
```

Utilisation des tableaux

Parcours d'un tableau à index numérique

```
for ($i = 0; $i < 5; $i++)
echo $i, ' -> ', $tab[$i], '<br/>';
```

Boucle foreach (clef et info)

```
foreach ($tab as $clef => $info)
  echo $clef, ' -> ', $info, '<br/>';
```

Boucle foreach (info seulement)

```
foreach ($tab as $info)
  echo $info, '<br/>';
```