(suite de la page précédente)

```
Votre âge est stocké dans la variable <b>$a</b>
<br/> dont le type est <i><?php echo gettype($a); ?></i>
<br/> On peut la transformer en <i>integer</i> en faisant :
   <?php settype($a,"integer"); ?>
<br/>
    Type de $a :<?php echo gettype($a); ?>
</body>

</html>
```

3.7.3 Exécution

Formulaire

3.8 Les variables en PHP

3.8.1 Déclaration simple :

```
<?php
$variable = "une variable en PHP";
// Une autre variable :
$Variable = 1000;
```

3.8.2 Existence de variables, la fonction isset() :

```
<?php
$a = "une variable en PHP";
if(isset($a)) echo "la variable a existe";
unset($a);
echo "la variable a a été supprimée ...";
```

3.8.3 Test de variables, la fonction empty() :

```
<?php
$a = "une variable en PHP";
if (!empty($a)) echo " La variable existe et elle n\'est␣
 ↪pas vide !";
```

Avertissement : La fonction empty() répond vrai si la variable n'existe pas et ceci sans faire aucun warning ! En outre, avant PHP 5.5, on ne peut pas l'utiliser sur autre chose que des variables (impossible d'appeller une fonction dans l'argument qu'on lui passe)

3.8.4 Test de variables en PHP 7 avec l'opérateur *coalescent*

L'opérateur *Null coalescent* **??** permet de simplifier certains tests d'existence de variables et d'alternatives, comme par exemple :

```
<?php

// $a non initialisée
$b = 143;

echo $a ?? 3; // affiche 3
echo PHP_EOL;
echo $a ?? $b ?? 7; // affiche 143
echo PHP_EOL;
```

Ce qui permet de limiter le recours à *isset* dans de nombreuses situations comme :

```
<?php
// Récupère la valeur de $_GET['email'] et  retourne 'nobody
↪'
// si elle n'existe pas.
$mail = $_GET['email'] ?? 'nobody@null';
// Equivalent à:
$mail = isset($_GET['email']) ? $_GET['email'] :
↪'nobody@null';

// Coalescing ?? peut être chainé :
// On renvoie la première valeur définie parmi
// $_GET['email'], $_POST['email'], et 'nobody@null.com'.
$mail = $_GET['email'] ?? $_POST['email'] ?? 'nobody@null';
echo "$mail\n";
```

3.8.5 Portée des variables :

— Par défaut, toutes les variables sont **locales**
— Leur portée se réduit à la fonction ou au bloc de leur déclaration
— Pour déclarer une variable globale, on peut utiliser le tableau $_GLOBALS[ ]

```
<?php $_GLOBALS['MaVar']="Bonjour"; ?>
```

3.8.6 Constantes :

```
<?php
define("USER","TOTO");
echo USER; // Notez l'absence de $ ici
```

3.9 Les chaînes en PHP

3.9.1 Les bases :

Guillemets ou Cotes :

```
<?php
$var="Hello PHP";
$machaine="le contenu de \$var est $var<br>";
echo $machaine;
//ou avec des '  ':
$mystring='le contenu de $var est '.$var;
echo $mystring;
```

dont le résultat sera toujours :

```
le contenu de $var est Hello PHP
```

La concaténation :

A l'aide de **.**

La longueur d'une chaine :

```
<?php int lg=strlen($chaine); ?>
```

Accéder au caractère i d'une chaine :

```
<?php echo $chaine[i]; ?>

La chaine est traitée comme un tableau indéxé par un *entier*
La plupart des tableaux de PHP sont indéxés par des chaines...
```

Mettre en majuscules/minuscules :

— avec *strtoupper()* pour obtenir des majuscules
— avec *strtolower()* pour mettre en minuscules
— avec *ucfirst()* pour mettre en majuscule la première lettre d'une chaine
— avec *ucwords()* pour mettre en majuscule la première lettre de chaque mot dans une chaine

3.9.2 Recherche de sous-chaines ou de motifs dans une chaine :

— avec *strstr()*
— avec *stristr()*
— avec *ereg()* ou *eregi()*

Par exemple :

```
<?php
$AGENT=$_SERVER['HTTP_USER_AGENT'];
echo $AGENT;
echo("\n<P>");
if (stristr($AGENT,"MSIE"))
   echo "Vous semblez utiliser Internet Explorer !</b>";
   elseif (ereg("Firefox",$AGENT))
          echo "Vous semblez utiliser Firefox !</b>";
   elseif (eregi("chrome",$AGENT))
          echo "Vous semblez utiliser Chrome !</b>";
```

Indication : Les variantes de ces fonctions comportant un i indiquent une insensibilité à la casse c'est à dire que les majuscules et minuscules sont considérées comme identiques.

Exemple : Test un peu plus complet du UserAgent :

```
<?php
function getBrowser($userAgent){
  if (preg_match("/MSIE(.{5})/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/Firefox(.*)/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/chrome(.{4})/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/safari/i",$userAgent,$num)){
    preg_match("/Version(.{4})/",$userAgent,$num);
    return "Safari ".$num[0];
}
  else return "Navigateur Inconnu";
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
if (!empty($_SERVER['HTTP_USER_AGENT'])){
echo "Votre navigateur semble etre:\n";
echo getBrowser($_SERVER['HTTP_USER_AGENT']);
}

// Test avec des UserAgent connus:
$FF="Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:32.0) Gecko/
 ↪20100101 Firefox/32.0";
$msie="Mozilla/5.0 (compatible; MSIE 10.6; Windows NT 6.1; Trident/
 ↪5.0; InfoPath.2; SLCC1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.
 ↪30729; .NET CLR 2.0.50727) 3gpp-gba UNTRUSTED/1.0";
$chrome="Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.
 ↪36 (KHTML, like Gecko) Chrome/37.0.2049.0 Safari/537.36";
$safari="Mozilla/5.0 (iPad; CPU OS 6_0 like Mac OS X) AppleWebKit/
 ↪536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5355d Safari/
 ↪8536.25";

echo "<br/> Test Firefox:<br/>\n";
echo getBrowser($FF)."<br/>\n";
echo "<br/> Test MSIE: \n";
echo getBrowser($msie)."<br/>\n";
echo "<br/> Test Chrome: \n";
echo getBrowser($chrome)."<br/>\n";
echo "<br/> Test Safari: \n";
echo getBrowser($safari);
```

Résultat :

```
<br/> Test Firefox:<br/>
Firefox/32.0<br/>
<br/> Test MSIE:
MSIE 10.6<br/>
<br/> Test Chrome:
Chrome/37.<br/>
<br/> Test Safari:
Safari Version/6.0
```

3.10 Le typage en PHP

3.10.1 Les fonctions *gettype( )* et *settype( )*

gettype( ) renvoie l'un des résultats suivants :

— integer

— double
— string
— array
— object
— class
— « unknown type »

settype( ) change le type d'un élément

```
<?php
$a=3.5;
settype($a,"integer");
echo "le contenu de la variable a est ".$a;
```

dont le résultat sera :

```
le contenu de la variable a est 3
```

3.10.2 Fonctions de test

— *is_int()*
— *is_long()*
— *is_double()*
— *is_array()*
— *is_object()*
— *is_string()*

Attention : N'oubliez pas comme en JavaScript la différence entre l'opérateur == et ===

Le premier vérifie l'égalité des contenus en ne tenant pas compte d'une éventuelle différence de typage (int ou string par exemple) tandis que le second vérifie une égalité stricte.

En d'autres termes : 5 == « 5 » est VRAI tandis que 5 === « 5 » est FAUX

3.11 Quelques particularités de PHP

3.11.1 Valeurs des variables :

```
    <?php
$toto = "Bonjour<br/>\n";
$var = "toto";
echo $$var;
```

dont le résultat sera toujours :

3.11.2 Résultat brut

```
Bonjour<br/>
```

3.11.3 La fonction *eval()*

— Permet l'évaluation d'expressions arithmetiques directement en PHP.
— Existe aussi en JavaScript.
— Délicat à manipuler, problématique en termes de sécurité.

3.12 Les tableaux en PHP

3.12.1 Tableaux associatifs - parcours avec boucle foreach :

```
<?php
  $jours=array("Lu"=>"Lundi","Ma"=>"Mardi",
        "Me"=>"Mercredi","Je"=>"Jeudi","Ve"=>"Vendredi",
        "Sa"=>"Samedi","Di"=>"Dimanche" );

  foreach($jours as $key=>$val) echo $key." ".$val."<br>\n";
```

Ce qui donne :

```
Lu Lundi
Ma Mardi
Me Mercredi
Je Jeudi
Ve Vendredi
Sa Samedi
Di Dimanche
```

3.12.2 Affichage avec print_r() :

```
<?php
     print_r($jours);
```

3.12.3 Résultat brut html :

```
Array
(
    [Lu] => Lundi
    [Ma] => Mardi
    [Me] => Mercredi
    [Je] => Jeudi
    [Ve] => Vendredi
    [Sa] => Samedi
    [Di] => Dimanche
)
```

3.12.4 Essayez vous-même

tabs

3.12.5 Utilisation de la fonction array_walk :

```
<?php array_walk($jours,'aff_tab'); ?>
```

En ayant défini au préalable :

```
<?php
function aff_tab($val, $key){
echo "$key-$val<br/>\n";
}
```

On obtient le même résultat qu'avec la boucle foreach

3.12.6 Tri simple d'un tableau :

```
<?php
  sort($jours);
  array_walk($jours,'aff_tab');
```

3.12.7 On obtient :

```
0-Dimanche
1-Jeudi
2-Lundi
3-Mardi
```

(suite sur la page suivante)

(suite de la page précédente)

```
4-Mercredi
5-Samedi
6-Vendredi
```

C'est à dire que :

— Le tableau est trié selon l'ordre de ses valeurs

— les clefs sont effacées et réaffectées avec des entiers.

Si on veut préserver également les clefs du tableau associatif, il faut utiliser la méthode suivante :

3.12.8 Tri selon l'ordre naturel avec natsort

```
<?php
    $jours=array("Lu"=>"Lundi","Ma"=>"Mardi",
    "Me"=>"Mercredi","Je"=>"Jeudi","Ve"=>"Vendredi",
    "Sa"=>"Samedi","Di"=>"Dimanche" );
    var_dump($jours);
    natsort($jours);
    var_dump($jours);
```

3.12.9 Résultat brut html

```
/Users/roza/work/iut/prog/php/source/exemples/tritab.php:5:
array(7) {
  'Lu' =>
  string(5) "Lundi"
  'Ma' =>
  string(5) "Mardi"
  'Me' =>
  string(8) "Mercredi"
  'Je' =>
  string(5) "Jeudi"
  'Ve' =>
  string(8) "Vendredi"
  'Sa' =>
  string(6) "Samedi"
  'Di' =>
  string(8) "Dimanche"
}
/Users/roza/work/iut/prog/php/source/exemples/tritab.php:7:
array(7) {
  'Di' =>
  string(8) "Dimanche"
  'Je' =>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  string(5) "Jeudi"
  'Lu' =>
  string(5) "Lundi"
  'Ma' =>
  string(5) "Mardi"
  'Me' =>
  string(8) "Mercredi"
  'Sa' =>
  string(6) "Samedi"
  'Ve' =>
  string(8) "Vendredi"
}
```

3.12.10 Exécution

tritabnat

On peut aussi utiliser la fonction natcasesort() si on ne veut pas se préoccuper de la casse des chaines présentes dans le tableau, soit à peu près l'ordre du dictionnaire ...

3.13 Les tableaux prédéfinis de PHP : Superglobales

3.13.1 Les Superglobales de PHP

Ce sont des tableaux concernant pour l'essentiel le protocole HTTP ou la gestion de Cookies ou des Sessions.

- $_GET[ ], $_POST[ ] ou $_REQUEST[ ] qui englobe les 2
- $_SERVER[ ] : Variables décrivant le client ou la page courante
- $_GLOBALS[ ] variables globales
- $_COOKIE[ ] pour les cookies
- $_SESSION[ ] pour les sessions

3.13.2 Exemple récupération de $_SERVER[ ] grâce à la fonction getenv() :

```
<?php
function infos(){
  $env = array('remote_addr','http_accept_language','http_
↪host',
  'http_user_agent','script_filename','server_addr',
  'server_name','server_signature','server_software',
  'request_method','query_string','request_uri','script_name
↪');
```

(suite sur la page suivante)

(suite de la page précédente)

```
  // Construction d'un tableau associatif
  // Avec les valeurs lues dans l'environnement
  $retour =array();
  foreach ($env as $clef) $retour[$clef] = getenv($clef);
  return $retour;
}

echo("Voici les infos disponibles:<BR>");
$tab = infos();
foreach ($tab as $clef=>$val) echo $clef." :".$val."<br>\n";
```

3.13.3 Résultat

```
Voici les infos disponibles:
remote_addr :::1
http_accept_language :fr-fr
http_host :localhost
http_user_agent :Mozilla/5.0 (Macintosh; U; Intel Mac OS X␣
 ↪10_6_4; fr-fr)
AppleWebKit/533.18.1 (KHTML, like Gecko) Version/5.0.2␣
 ↪Safari/533.18.5
script_filename :/Users/roza/Sites/php/exemples/infospy.php
server_addr :::1
server_name :localhost
server_signature :
server_software :Apache/2.2.14 (Unix) mod_ssl/2.2.14
OpenSSL/0.9.8l DAV/2 PHP/5.3.2
request_method :GET
query_string :
request_uri :/~roza/php/exemples/infospy.php
script_name :/~roza/php/exemples/infospy.php
`User-Agent <http://localhost/~roza/php/exemples/infospy.
 ↪php>`_
```

3.13.4 Exécution

infospy