PHP

BASES

TYPES

- **Types scalaires**
  boolean
  integer
  float (double)
  string

- **Types composés**
  array
  object

- **Types spéciaux**
  resource
  NULL

TYPES

- **Déclaration**
  fonction de la valeur de la variable

- **Test**
  fonction *gettype ($var)*
  fonctions *is_xxx (var)*

- **Conversion (cast)**
  cast (cf C)
  settype ($var, type) *avec type = « boolean », « integer », « float », « string », « array », « object », « NULL »*

FONCTIONS is_xxx

- **type d'une variable (passée en paramètre)**
  retour true ou false
- **Versions**
  is_int ($var) / is_integer ($var) / is_long ($var)
  is_float ($var) / is_double($var) / is_real ($var) *is_numeric ($var)*
  is_bool ($var)
  is_string ($var)
  is_resource ($var)
  is_object ($var)

```
<?php
if (!is_int($quantite))
                ...
}?>
```

- ***is_null ($var)***
  équivalent à $var === NULL

GETTYPE

- **Retourne le type d'une variable (passée en paramètre)**

- **Retour**
  boolean, integer, double, string, array, object, resource, NULL unknown type

```php
<?php
$exemples = array(1, 3.14, NULL, new MaClasse, 'bonjour');

foreach ($exemples as $variable) {
   echo gettype($variable) .  "\n";
}?>
```

```
integer
double
NULL
object
string
```

ECHO

- **Affiche son argument**
- **Construction du langage**
  pas fonction
  appel avec ou sans parenthèse
- **Syntaxe**

```
<?php
echo « Bonjour »;
echo « Bonjour Monsieur » . $nomUtilisateur;
?>
```

- **Version courte**

```
<h1>Bonjour Monsieur <? = $nomUtilisateur ?> </h1>
```

PRINT

- **Affiche son argument**
- **Retour**
  toujours 1
- **Syntaxe**

```
<?php
print  « Bonjour »;
print (« Bonjour Monsieur » . $nomUtilisateur);
?>
```

PRINT_R

- **Affiche une expression dans un format lisible**
  retour à l'écran ou dans une variable

- **Syntaxe**

```
<?php
$tab= array (   'lundi' => 'ravioli',
                'mardi' => 'purée',
                'weekend' => array ('v', 's', 'd'));
print_r ($tab);
?>
```

```
Array
(
        [lundi] => ravioli
        [mardi] => purée
        [weekend] => Array
                (
                        [0] => v
                        [1]  => s
                        [2] => d
                )
)
```

INCLUDE / REQUIRE

- **Inclut et exécute le fichier en paramètre**

- **Chemins de recherche**
  - chemin précisé en paramètre (
  - chemins listés dans la variable include_path
  - dossier courant

- **Si fichier non trouvé**
  include : avertissement (warning)
  require : erreur fatale

```
<?php
include ‘connexion.php’;
require ‘etudiant.class.php’;
?>
```

INCLUDE_ONCE / REQUIRE_ONCE

- **Même comportement que include/require**

- **Différence**
  ne fait rien si le fichier a déjà été inclu

```
<?php
include_once ‘connexion.php’;
include_once ‘connexion.php’;
require_once ‘etudiant.class.php’;
?>
```

TRY … CATCH

```
try {
        ...
}
catch (OutOfRangeException $e) {
        echo 'Index incorrect ! ' . $e->getM
}
catch (Exception $e) {
        echo 'Index incorrect ! ' . $e->get
}
finally {
        echo 'fin !';
}
```

Bloc try
Appels de fonctions pouvant « lancer » des exceptions

Bloc(s) catch
traitement du/des exceptions lancée(s)
rupture de séquence du try
type d'exception (sous-classes d'Exception)

Bloc finally (optionnel)
actions finales (après try et catch)

THROW

- **« Jette » une exception**
  rupture de séquence

- **Syntaxe**

```
If ($valeur == 0)
                throw new Exception ('Division par zéro !');
…
return 1/$valeur
```

- **Exception utilisateur (héritage)**

```
class CalculException extends Exception {… }

If ($valeur == 0)
                throw new CalculException ('Division par zéro !');
…
return 1/$valeur
```

ARRAY

- **Carte ordonnée**
  ensemble ordonné de paires <clé, valeur>
  taille illimitée

- **Clé**
  numérique ou chaîne

- **Tableau indexé**
  clé non mentionnée (0, 1, 2, …)

- **Tableau associatif**
  clé spécifiée

ARRAY

- **Création**
  $tableau = array(valeur$_1$, valeur$_2$, …);
  $tableau = array(valeur$_1$ => clé$_1$, valeur$_2$ => clé$_2$, …);

- **Ajout d'une valeur**
  sans clé : *$tableau [] = valeur;*
  avec clé : *$tableau [clé] = valeur;*

- **Suppression d'une valeur**
  sans clé : *unset ($tableau [index]);*
  avec clé : *unset ($tableau [clé]);*

- **Nombre d'éléments**
  count($tableau);

- **Opérations**
  recherches (clé, valeurs), tri, …

ARRAY (INDEXE)

- **Déclaration**

```
<?php
$semaine = array(« lundi », « mardi », « mercredi », « jeudi »,
                                « vendredi », « samedi », « dimanche »);
?>
```

- **Parcours**

```
<?php
foreach ($semaine as $jour)
                echo « $jour  <br /> »;
?>
```

lundi
mardi
mercredi
jeudi
vendredi
samedi
dimanche

ARRAY (ASSOCIATIF)

- **Déclaration**

```
<?php
$gouvernement = array(« Premier Ministre » => « Mickey »,
                      « Ministre de la Culture » => « Simplet »,
                      « Ministre de l'Intérieur » => « Tintin »,
                      « Ministre des Finances » => « Picsou »,
                      « Ministre de la Justice » => « Zorro »,
                      « Ministre de la Danse » => « Minnie »,
                      « Ministre de l'Ecologie » => « Tarzan »);
?>
```

- **Parcours**

```
<?php
echo « Mon gouvernement : <br/> »;
foreach ($gouvernement as $poste => $ministre)
        echo « $poste : <b>$ministre</b> »;
?>
```

Mon gouvernement :
Premier Ministre : **Mickey**
Ministre de la Culture : **Simplet**
Ministre de l'Intérieur : **Tintin**
Ministre des Finances : **Picsou**
Ministre de la Justice : **Zorro**
Ministre de la Danse : **Minnie**
Ministre de l'Ecologie : **Tarzan**

ARRAY (MATRICE)

- **Déclaration**
  tableau de tableau

```
<?php
$matrice = array(          array ('X', 'O', 'X'),
                           array ('X', 'X', 'O'),
                           array ('X', 'O', 'O'));
?>
```

- **Parcours**

```
<?php
foreach ($matrice  as $ligne)
{
            foreach ($ligne as $val)
                        echo « $val    »;
            echo « <br /> »
}
?>
```

X O X
X X O
X O O

IF… ELSE/ELSEIF

- **Cf manuel du C/C++**

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

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

SWITCH

- **Cf manuel du C/C++**
  mise en cascade de plusieurs cas (plus de 2)

```
<?php
switch ($numeroSS[0]) {
   case '1':
      echo "Homme... ";
      break;
   case '2':
      echo "Femme... ";
      break;
   default:
     echo "Gremlins...";
}
?>
```

WHILE

- **Cf manuel du C/C++**
  boucle pouvant se faire 0 fois (test en début)

```
<?php
$n = 2014;
while  ($n != 1)
{
        if ($n % 2 == 0)
                $n = $n / 2;
        else
                $n = 3 * $n + 1;
        echo $n . " ";
}
?>
```

```
2014 1007 3022 1511 4534 2267 6802 3401 10204 5102 2551
7654 3827 11482 5741 17224 8612 4306 2153 6460 3230
1615 4846 2423 7270 3635 10906 5453 16360 8180 4090
2045 6136 3068 1534 767 2302 1151 3454 1727 5182 2591
7774 3887 11662 5831 17494 8747 26242 13121 39364 19682
9841 29524 14762 7381 22144 11072 5536 2768 1384 692
346 173 520 260 130 65 196 98 49 148 74 37 112
56 28 14 7 22 11 34 17 52 26 13 40 20 10
5 16 8 4 2 1
```

DO… WHILE

- **Cf manuel du C/C++**
  boucle se faisant au moins une fois (test à la fin)

```php
<?php
$n = 2014;
$res ='';
do
{
        $r = $n % 2;
        $n = $n / 2;
        $res = $r . $res;
} while  ($n != 0) ;
echo $res;
?>
```

```
11111011110
```

FOR

- **Cf manuel du C/C++**

```php
<?php
for ($i=0; $i<10; $i++)
        echo $i;
?>
```

```
0 1 2 3 4 5 6 7 8 9
```

- **Boucle sur un tableau**

```php
<?php
$neveux = array('Riri', 'Fifi', 'Loulou');

for($i = 0, $taille = count($neveux); $i < $taille; $i++)
{
        echo    $neveux[$i] . « <br/> »
}
?>
```

```
Rifi
Fifi
Loulou
```

FOREACH

- **Boucle sur les tableaux**

```
<?php
$menuDeLaSemaine = array (    “lundi” => “ravioli”,
                              “mardi” => “spaghetti”,
                              “mercredi” =>  “macaroni”,
                              “jeudi” => “cannelloni”,
                              “vendredi” => “flambi”);

foreach ($menuDeLaSemaine as $repas)
        echo $repas;

foreach ($menuDeLaSemaine as $jour => $repas)
        echo “le $jour, c’est $repas”;
?>
```

```
ravioli
spaghetti
macaroni
cannelloni
flambi

le lundi c’est ravioli
le mardi c’est spaghetti
le mercredi c’est macaroni
le jeudi c’est cannelloni
le vendredi c’est flambi
```

FUNCTION

- **Déclaration**

```php
<?php
function maFonction($arg_1, $arg_2, /* ..., */ $arg_n)
{
   echo "Exemple de fonction.\n";
   return $resultat;
}
?>
```

- **Appel**

  case insensitive

```php
<?php
 echo maFonction('hello', 'world');
?>
```

- **Ni surcharge, ni redéfinition**

PASSAGE D'ARGUMENTS

- **Passage par valeur**
  non modificable
  par défaut

```
<?php
function maFonction ($arg)
{
  …
?>
```

- **Passage par référence**
  modifiable

```
<?php
function maFonction (&$arg)
{
  …
?>
```

ARGUMENTS PAR DEFAUT

- **Arguments non fournis lors de l'appel**
  valeur par défaut (constante) lors de la déclaration

- **Derniers paramètres de la fonctions**

```php
<?php
function servir_cafe ($type = "expresso")
{
   return "Servir un $type.\n";
}
echo servir_cafe();
echo servir_cafe(null);
echo servir_cafe(" cappuccino ");
?>
```

```
Servir un expresso.
Servir un.
Servir un cappuccino.
```

ARGUMENTS VARIABLES

- **Nombre d'arguments de la fonction**
  func_num_args( )

- **Récupération d'un argument**
  func_get_arg($i)

```php
<?php
function somme() {
   $total = 0;
   foreach (func_get_args() as $n) {
      $total += $n;
   }
   return $total;
}
 echo somme(1, 2, 3, 4);
 echo somme(1, 2, 3, 4, 5, 6, 7, 8, 9);
?>
```

```
10
45
```

- **Utilité**
  surcharge de constructeur de classe

ARGUMENTS VARIABLES *(php5.6)*

- **Ellipse**
  … $var *tableau des arguments restants*

```php
<?php
function somme(… $valeurs) {
   $total = 0;
   foreach ($valeurs as $n) {
      $total += $n;
   }
   return $total;
}
echo somme(1, 2, 3, 4);
echo somme(1, 2, 3, 4, 5, 6, 7, 8, 9);
?>
```

```
10
45
```

FONCTION VARIABLE

- **Stockage du nom d'une fonction dans une variable**
  $var = 'nomFonction';

- **Appel**
  $var($arg…
  call_user_func($var, $arg, …

```php
<?php
function somme($a, $b) {
   return $a + $b;
}
function produit ($a, $b) {
   return $a * $b;
}
$fct = 'somme';
echo $fct (5, 7);

$fct = 'produit';
echo call_user_func($fct , 5, 7) ;
?>
```

```
12
35
```

FONCTION CALLBACK

- **Référence vers une fonction**

- **Déclaration**
  $var = function ($arg)
  { …

- **Closure**
  variables du contexte de définition
  $var = function ($arg) use ($varContexte$_1$, $varContexte$_2$, …)
  { …

- **Appel**
  $var (…

FONCTION CALLBACK

```
class Panier {
   const PRIX_BEURRE  = 2.00;
   const PRIX_LAIT    = 1.00;
   const PRIX_OEUF    = 0.50;

   protected $articles = array();

   public function ajouter($article, $quantite)   { $this->articles[$article] = $quantite;   }
   public function getquantite($article)   { return isset($this->articles[$article]) ? $this->articles[$article] :  FALSE;   }

   public function getTotal($tva)   {
      $total = 0.00;
      $fct = function ($quantite, $article) use ($tva, &$total)          {
            $pricePerItem = constant(__CLASS__ . "::PRIX_" . strtoupper($article));
            $total += ($pricePerItem * $quantite) * ($tva + 1.0);
         };

      array_walk($this->articles, $fct);
      return round($total, 2);
   }
}

$mon_panier = new Panier();

// Ajout d'élément au panier
$mon_panier->ajouter('beurre', 1);
$mon_panier->ajouter('lait', 3);
$mon_panier->ajouter('oeuf', 6);

// Affichage du prix avec 5.5% de TVA
print  « Prix total : » . $mon_panier->getTotal(0.055) . "\n";
?>
```

```
Prix total : 6,33
```

CHAINES DE CARACTERES

- **Entourée d'apostrophes**
  '…..'
  seuls échappements possibles : \' \\

- **Entourée de guillements**
  « ….. »
  échappements (\)
  interprétation des variables

- **Syntaxe Heredoc**

- **Syntaxe Nowdoc**

HEREDOC

- **Délimite une chaîne de caractère**
  début : **<<<IDENT**
  fin: **IDENT;** *(en début de ligne, rien après)*

- **Comme avec des guillemets, mais sans**
  Variables interprétées

```
<?php
 $message = <<<EOT
Je, soussigné $nom $prenom, certifie sur l'honneur
-  être né quelque part,
-  avoir fêté mon anniversaire l'année dernière.
Fait pour valoir et servir ce que de droit.
A Bordeaux, le $dateDuJour
EOT;
echo $message;
?>
```

```
Je, soussigné MALDONADO Michel, certifie sur l'honneur
-  être né quelque part,
-  avoir fêté mon anniversaire l'année dernière.
Fait pour valoir et servir ce que de droit.
A Bordeaux, le  26/09/2014
```

NOWDOC *(php5.3)*

- **Délimite une chaîne de caractère**
  début : **<<<'IDENT'** *(identificateur quoté)*
  fin: **IDENT;** *(en début de ligne, rien après)*

- **Comme avec des quotes , mais sans**
  Variables non interprétées

```
<?php
 $message = <<<'EOT'
Je, soussigné $nom $prenom, certifie sur l'honneur
-  être né quelque part,
-  avoir fêté mon anniversaire l'année dernière.
Fait pour valoir et servir ce que de droit.
A Bordeaux, le $dateDuJour
EOT;
echo $message;
?>
```

```
Je, soussigné $nom $prenom, certifie sur l'honneur
-  être né quelque part,
-  avoir fêté mon anniversaire l'année dernière.
Fait pour valoir et servir ce que de droit.
A Bordeaux, le  $dateDuJour
```

VARIABLES SYSTEME

- **$_GET**
  variables postées (formulaire, méthode GET)
- **$_POST**
  variables postées (formulaire, méthode POST)
- **$_FILES**
  fichiers uploadés (formulaires)
- **$_COOKIE**
  Cookies (conservés d'une visite à l'autre)
- **$_SESSION**
  variables sessions (conservées d'une page à l'autre)

CONCLUSION