QUELQUES ÉLÉMENTS DU LANGAGE PHP

1. Types de base (http://php.net/manual/fr/language.types.intro.php)

En PHP, il existe 8 types de base (4 scalaires, 2 composés, 2 spéciaux). Pour le moment, nous n'utiliserons que les types scalaires.

1. **boolean** (*true* et *false*; insensible à la casse)
2. **integer** (*-2 147 483 648* à *2 147 483 647*; autrement interprété comme un *float*)
3. **float** (*double*)
4. **string** (0 à 2 Go caractères)

2. Le type *string* (http://php.net/manual/fr/language.types.string.php)

- Même si une chaîne de caractères peut être encadrée d'apostrophe ('), nous privilégierons l'utilisation de guillemets (").

  Le principal avantage d'utiliser les guillemets est qu'il sera possible d'incorporer les noms de variables à même notre chaîne de caractères.

  Exemple :

```
$strPrenom = "Nicole";
$strNom = "Laliberté";

echo "Votre nom est $strPrenom $strNom.<br />";
```

  equivaut à...

```
echo "Votre nom est " . $strPrenom . " " . $strNom . ".<br />";
```

- Il est possible d'utiliser des séquences d'échappement comme le « **\n** » pour provoquer un saut de ligne, le « **** » pour afficher une barre oblique inversée (« **** »), le « **\$** » pour le signe *dollar* et naturellement le classique « **\"** » pour afficher le *guillemet*.

- Le **point** permet de concaténer deux chaînes de caractères.

  Exemple :

```
echo "Votre nom est " . $strPrenom . " " . $strNom . ".<br />";
⇨ Votre nom est Nicole Laliberté.
```

  Mise en garde : Une erreur fréquente est d'utiliser le symbole « **+** » qui sert d'opérateur de concaténation en *JavaScript*.

  Exemple :

```
echo "Votre nom est " + $strPrenom + " " + $strNom + ".<br />";
⇨ 0
```

3. Variable en PHP (http://php.net/manual/fr/language.variables.php)

Syntaxe

En PHP, une variable débute par le symbole « **$** » suivi d'une lettre ou d'un souligné (_), puis de lettres, chiffres ou soulignés. (Notez que le nom **$this** est un nom réservé du langage.) Minuscules et majuscules sont traitées différemment. **Puisque le langage est faiblement typé, vous devrez précéder chaque nom de variable de son type.**

Exemple :

```
function dollar($dblNombre) {
   return number_format($dblNombre, 2, ",", " ") . " $";
}

$binTrouve = false;                                                    var_dump($binTrouve);
$booAbsent = true;                                                     var_dump($booAbsent);
$intQteProduit = 1000;                                              var_dump($intQteProduit);
$dblPrixUnitaire = 99.99;                                        var_dump($dblPrixUnitaire);
$float_Total_Ventes = 2147483647;                             var_dump($float_Total_Ventes);
$float_Total_Ventes = 2147483648;                             var_dump($float_Total_Ventes);
$strNom20 = "Brodeur";                                                 var_dump($strNom20);

echo "<p style=\"font-family:consolas; font-size:16px;\">";
echo "\$binTrouve = [$binTrouve]<br />";
echo "\$booAbsent = [$booAbsent]<br />";
echo "\$intQteProduit = $intQteProduit<br />";
echo "\$dblPrixUnitaire = $dblPrixUnitaire $<br />";
echo "Total : $intQteProduit*$dblPrixUnitaire $<br />";
echo "Total : " . $intQteProduit*$dblPrixUnitaire . " $<br />";
echo "Total : " . number_format($intQteProduit*$dblPrixUnitaire, 2, ",", " ") ." $<br />";
echo "\$float_Total_Ventes = " . dollar($float_Total_Ventes) . "<br />";
echo "\$strNom20 = \"$strNom20\"";
echo "</p>";
```

Si nous affichons le code source généré par la partie « **echo** » du code ci-dessus, voici ce qui s'affiche à l'écran du navigateur :

```
<p style="font-family:consolas; font-size:16px;">$binTrouve = []<br />$booAbsent = [1]<br
/>$intQteProduit = 1000<br />$dblPrixUnitaire = 99.99 $<br />Total : 1000*99.99 $<br />Total
: 99990 $<br />Total : 99 990,00 $<br />$float_Total_Ventes = 2 147 483 648,00 $<br />$strNom20
= "Brodeur"</p>
```

L'ajout du caractère d'échappement « **\n** » à la fin de chaque ligne et du caractère d'échappement « **\t** » au début des lignes affichant le contenu des variables permettra d'afficher le code de façon plus visible :

```
<p style="font-family:consolas; font-size:16px;">
    $binTrouve = []<br />
    $booAbsent = [1]<br />
    $intQteProduit = 1000<br />
    $dblPrixUnitaire = 99.99 $<br />
    Total : 1000*99.99 $<br />
    Total : 99990 $<br />
    Total : 99 990,00 $<br />
    $float_Total_Ventes = 2 147 483 648,00 $<br />
    $strNom20 = "Brodeur"
</p>
```

```
boolean false

boolean true

int 1000

float 99.99

int 2147483647

float 2147483648

string 'Brodeur' (length=7)

$binTrouve = []
$booAbsent = [1]
$intQteProduit = 1000
$dblPrixUnitaire = 99.99 $
Total : 1000*99.99 $
Total : 99990 $
Total : 99 990,00 $
$float_Total_Ventes = 2 147 483 648,00 $
$strNom20 = "Brodeur"
```

3. Variable en PHP (suite)

a. Initialisation d'une variable

Il est suggéré d'initialiser chaque variable avant son utilisation même si ce n'est pas obligatoire. Par défaut, la valeur assignée est soit **false** (*boolean*), **0** (*integer* et *float*) ou **vide** (*string*).

b. Noms réservés

Référez-vous à la page **http://php.net/manual/fr/reserved.variables.php** pour la liste des variables prédéfinies du langage.

c. Portée d'une variable
Passage par valeur et par référence

- Toute variable définie dans une fonction est locale à cette dernière.
- Une variable définie dans un script a une portée globale pour ledit script, mais ne peut être accédée via une fonction à moins qu'elle ne soit passée en paramètre à cette dernière ou précédée du mot-clé **global** lors de sa déclaration.
- Par défaut, une variable passée en paramètre l'est toujours par valeur. Pour être passé en référence, le symbole « **&** » doit précéder le nom de la variable dans l'en-tête de cette dernière seulement.

```php
<?php
   $a = 5; $b = 7; $c = 100; $d = 200; $br = "<br />";
   function somme1() {
      global $br;
      $c = $a + $b;
      echo "Dans Somme1 :$br=> a=$a, b=$b, c=$c<br /><br />";
   }
   function somme2($x, $y, $z) {
      global $br;
      $z = $x + $y;
      $d = 1000;
      echo "Dans Somme2 :$br=> a=$x, b=$y, c=$z, d=$d<br /><br />";
   }
   function somme3($x, $y, &$z) {
      global $br;
      global $d;
      $z = $x + $y;
      $d = $z;
      echo "Dans Somme3 :$br=> a=$x, b=$y, c=$z, d=$d<br /><br />";
   }
   echo "<p style=\"font-family:consolas; font-size:16px;\">";
   echo "Avant Somme1 :$br=> a=$a, b=$b, c=$c<br />";
   somme1();
   echo "Avant Somme2 :$br=> a=$a, b=$b, c=$c, d=$d<br />";
   somme2($a, $b, $c);
   echo "Apres Somme2, Avant Somme3 :$br=> a=$a, b=$b, c=$c, d=$d<br />";
   somme3($a, $b, $c);
   echo "Apres Somme3 :$br=> a=$a, b=$b, c=$c, d=$d<br />";
   echo "</p>"
?>
```

```
Avant Somme1 :
=> a=5, b=7, c=100

( ! ) Notice: Undefined variable: b
( ! ) Notice: Undefined variable: a
( ! ) Notice: Undefined variable: a
( ! ) Notice: Undefined variable: b

Dans Somme1 :
=> a=, b=, c=0

Avant Somme2 :
=> a=5, b=7, c=100, d=200
Dans Somme2 :
=> a=5, b=7, c=12, d=1000

Apres Somme2, Avant Somme3 :
=> a=5, b=7, c=100, d=200
Dans Somme3 :
=> a=5, b=7, c=12, d=12

Apres Somme3 :
=> a=5, b=7, c=12, d=12
```

4. Les opérateurs (**http://php.net/manual/fr/language.operators.php**)

Pour l'essentiel de ce que nous utiliserons, il n'y a pas de différence notable entre les opérateurs **arithmétiques**, de **comparaison** et **logiques** *PHPiens* et ceux des langages similaires comme *C#* et *JavaScript*. Il en est de même pour la priorité des opérateurs.

Par contre, il n'en sera pas de même pour les autres types d'opérateurs. Référez-vous à la page Web référencée ci-dessus pour plus de détails.

5. Les structures de contrôles

(**http://php.net/manual/fr/language.control-structures.php**)

Comme vous pourrez le constater, il y a beaucoup de similitudes entre les structures de contrôle *PHPiennes* et ceux des langages déjà étudiés.

```
if (expression)
   instruction;

if (expression) {
   instruction;
   instruction;
   ...
}
```

```
if (expression) {
   instruction(s);
}
else if (expression) {
   instruction(s);
}
else {
   instruction(s);
}
```

```
if (expression) {
   instruction(s);
}
elseif (expression) {
   instruction(s);
}
elseif (expression) {
   instruction(s);
}
else {
   instruction(s);
}
```

```
switch (expression) {
   case valeur1 :
   ...
   case valeurN :
      instruction(s);
      break;
   case valeur2 :
   ...
   case valeurM :
      instruction(s);
      break;
   ...
   default :
      instruction(s);
      [break;]
}
```

```
Utilisation de l'opérateur ternaire
(ou affectation conditionnelle)

Expression ? ValeurSiVRAI : ValeurSiFAUX
```

```
while (expression) {
   instruction(s);
}
```

```
do {
   instruction(s);
}
while (expression);
```

```
for (expr1; expr2; expr3) {
   instruction(s);
}
```

```
où expr1 = Initialisation
   expr2 = Test de la condition
   expr3 = Réinitialisation
```

```
foreach (array_expression as $value) {
   instruction(s);
}
```

6. Les fonctions (http://php.net/manual/fr/language.functions.php)

a. Forme générale d'une fonction utilisateur

```
function nomFonction1($arg1, $arg2, …, $argM) {
   Déclarations;
   instruction(s);
   [return expression]
}
```

```
function nomFonction2($arg3, $arg4, …, $argN) {
   Déclarations;
   instruction(s);
   [return expression]
}
```

En *PHP* (comme en *JavaScript*), une fonction peut être incluse dans une autre fonction.

Note : Tant que la fonction **nomFonction1** ne sera pas appelée, la fonction **nomFonction2** sera inaccessible.

Exemple :

```
<?php
   function principale($arg1, $arg2) {

      function secondaire1($arg) {
         echo "secondaire1 : $arg<br />";
      }

      function secondaire2($arg) {
         echo "secondaire2 : $arg<br />";
      }

      secondaire1($arg1);
      secondaire2($arg2);
   }

   secondaire1(1);
   secondaire2(2);
   principale(3, 4);
   secondaire1(5);
   secondaire2(6);
?>
```

```
secondaire1 : 3
secondaire2 : 4
secondaire1 : 5
secondaire2 : 6
```

b. Passage par valeur et par référence

- Par défaut, une variable passée en paramètre l'est toujours par valeur.
- Pour être passé en référence, le symbole « **&** » doit précéder le nom de la variable dans l'en-tête de la fonction seulement.
- **Assigner un ordre de priorité à une fonction qui utilise des paramètres par référence et qui retourne une valeur.**

Exemple :

```
<?php
   function somme2Nombres($strN1, $strN2, &$numSomme) {
      $binN1 = is_numeric($strN1);
      $binN2 = is_numeric($strN2);
      if ($binN1 && $binN2) {
         $numSomme = $strN1 + $strN2; /* PHP convertit chaque chaîne numérique en nombre */
      }
      return $binN1 && $binN2;
   }
```

```
   function testeFonction($strN1, $strN2) {
      $numSomme;
      if (somme2Nombres($strN1, $strN2, $numSomme)) {
         echo "$strN1 + $strN2 = $numSomme<br />";
      }
      else {
         echo "\"$strN1\" et/ou \"$strN2\" non numérique(s)<br />";
      }
   }

   testeFonction("1", "2");
   testeFonction(3, 4);
   testeFonction("5a", 6);
   testeFonction(7, "a8");
   testeFonction("9a", "a10");
?>
```

```
1 + 2 = 3
3 + 4 = 7
"5a" et/ou "6" non numérique(s)
"7" et/ou "a8" non numérique(s)
"9a" et/ou "a10" non numérique(s)
```

c. Juste pour le plaisir : Fonctions variables (**http://php.net/manual/fr/functions.variable-functions.php**)

```
<?php
   function addition($numN1, $numN2, &$strOperateur) {
      $strOperateur = "+";
      return $numN1 + $numN2;
   }
   function soustraction($numN1, $numN2, &$strOperateur) {
      $strOperateur = "-";
      return $numN1 - $numN2;
   }
   function multiplication($numN1, $numN2, &$strOperateur) {
      $strOperateur = "x";
      return $numN1 * $numN2;
   }
   function divisionReelle($numN1, $numN2, &$strOperateur) {
      $strOperateur = "/";
      return $numN2 != 0 ? $numN1 / $numN2 : "N/A";
   }
   function divisionEntiere($numN1, $numN2, &$strOperateur) {
      $strOperateur = "/";
      return $numN2 != 0 ? floor($numN1 / $numN2) : "N/A";
   }
   function modulo($numN1, $numN2, &$strOperateur) {
      $strOperateur = "MOD";
      return $numN2 != 0 ? $numN1 % $numN2 : "N/A";
   }

   $tOperations = Array("addition", "soustraction", "multiplication",
                        "divisionReelle", "divisionEntiere", "modulo");
   $numN1 = 18;
   $numN2 = 4;
   $numN3 = 0;
   $strOperateur;
   $strFluxZero = "";
   foreach($tOperations as $strOperation) {
      $numResultat = $strOperation($numN1, $numN2, $strOperateur);
      echo "$numN1 $strOperateur $numN2 = $numResultat<br />";
      $numResultat = $strOperation($numN1, $numN3, $strOperateur);
      $strFluxZero .= "$numN1 $strOperateur $numN3 = $numResultat<br />";
   }
   echo "<br />";
   echo $strFluxZero;
?>
```

```
18 + 4 = 22
18 - 4 = 14
18 x 4 = 72
18 / 4 = 4.5
18 / 4 = 4
18 MOD 4 = 2

18 + 0 = 18
18 - 0 = 18
18 x 0 = 0
18 / 0 = N/A
18 / 0 = N/A
18 MOD 0 = N/A
```