III) Les chaînes de caractères

PHP - Initiation

Les chaînes de caractères

III.1) Guillemets et primes

Une déclaration de chaine peut se faire de 2 façons :

- Par encadrement par des ' (simples quotes) : aucun contenu ne sera interprété à l'exception des '
- Par encadrement par des " (doubles quotes): les variables et les " seront interprétés

Exemple

```
$texte = 'à tous';

echo 'Bonjour $texte';          // affiche : Bonjour $texte
echo "Bonjour $texte";          // affiche : Bonjour à tous
```

PHP - Initiation

Les chaînes de caractères

Remarque

L'insertion des caractères spéciaux pourra se faire par échappement avec le symbole **** (anti-slash)

Exemple

```
echo 'Bonjour \$texte';              // affiche : Bonjour \$texte
echo "Bonjour \$texte";              // affiche : Bonjour $texte
```

On peut donc obtenir le même résultat en utilisant des simples quotes ou des doubles contenant des caractères d'échappement.

Exemple

```
echo "Bonjour \$texte";              // affiche : Bonjour $texte
echo 'Bonjour $texte';               // affiche : Bonjour $texte
```

PHP - Initiation

Les chaînes de caractères

III.2) Fonctions courantes

Pour travailler correctement en **UTF-8**, système dont la longueur des caractères peut être de plusieurs octets, il faudra souvent utiliser des fonctions un peu plus avancées et récentes que les classiques.

Ces fonctions contiennent le préfixe **mb_** (pour «Multi-Byte string»). Il est applicable à de nombreuses fonctions propres aux traitements des chaînes de caractères.

Les fonctions **mb_*nomfonction*()** utilisent un argument optionnel supplémentaire destiné à préciser l'encodage (dans le cas où l'on ne souhaite pas utiliser celui défini dans notre environnement).

PHP - Initiation

Les chaînes de caractères

mb_strlen(*chaîne*, *encodage*)

Retourne la taille de la chaîne **chaine**.
Cette taille s'exprime en nombre d'octets.

Exemple

```
echo mb_strlen('Bonjour à tous');                 // affiche : 15
echo mb_strlen('Bonjour à tous', 'utf8');         // affiche : 14
```

Dans cet exemple, le caractère **à** est encodé sur 2 octets, tous les autres sur 1 seul, ce qui explique que l'on obtienne 15.
En précisant **utf8**, on obtient non plus le nombre d'octets mais le nombre de caractères, car les caractères multi-octets comptent pour 1.

Attention : Les espaces sont des caractères !

PHP - Initiation

Les chaînes de caractères

str_replace(*occur*, *rempla*, *chaîne*)

Cette fonction va remplacer toutes les occurences **occur** rencontrées dans la chaine **chaine** par **rempla**.

Exemple

```
$chaine = 'Ma voiture est rouge';
echo str_replace('rouge', 'verte', $chaine);   // va afficher : Ma voiture est verte
```

strtr(*chaîne*, *char1*, *char2*)

Ici nous remplaçons tous les caractères de **char1** par ceux de **char2** rencontrés dans **chaine**.

Exemple

```
$chaine = 'Ma voiture est rouge';
echo strtr($chaine, 'aeiouy', 'aaaaaa');   // va afficher : Ma vaatara ast raaga
```

PHP - Initiation

Les chaînes de caractères

mb_substr(*chaîne*, *debut*, *nbr*, *encodage*)

Cette fonction extrait **nbr** caractères en partant de la position **debut** dans la chaine **chaine**. **Attention** : le premier caractère est en indice 0.

Exemple

```
$chaine = 'Ma voiture est rouge';
echo mb_substr($chaine, 11, 3, 'utf8');   // va afficher : est
```

mb_strpos(*chaîne*, *occur*, *debut*, *encodage*)

Recherche la première occurrence de **occur** rencontrée dans **chaine** en commençant la recherche en position **debut**.

Exemple

```
$chaine = 'Ma voiture est rouge';
echo mb_strpos($chaine, 'tur', 0, 'utf8');       // va afficher : 6
```

PHP - Initiation

Les chaînes de caractères

mb_strtolower(*chaîne*, *encodage*)

Renvoie **chaine** en ayant mis tous les caractères en minuscule.

Exemple

```
$chaine = 'J\'aimeRai vOir cEtTe cHaÎne eN MinUScule';
echo mb_strtolower($chaine, 'utf8');
// va afficher : j'aimerai voir cette chaîne en minuscule
```

mb_strtoupper(*chaîne*, *encodage*)

Bien évidemmente il existe la fonction inverse permettant de tout passer en majuscule..

Exemple

```
$chaine = 'j\'aimerai voir cette chaîne en majuscule';
echo mb_strtoupper($chaine, 'utf8');
// va afficher : J'AIMERAI VOIR CETTE CHAÎNE EN MAJUSCULE
```

PHP - Initiation

Les chaînes de caractères

trim(*chaîne*)

Très utilisée car elle permet de supprimer tous les espace, tabulations et fin de ligne en début et fin de **chaine**. En effet lorsqu'on récupère des informations en provenance d'un formulaire, il arrive fréquemment que l'utilisateur laisse ce genre de caractère indésirable sans forcément le faire exprès.

Exemple

```
$chaine = '   Bonjour   \n';
$test =  trim($chaine);

echo mb_strlen($chaine, 'utf8');        // va afficher : 14

echo '<hr>';

echo mb_strlen($test, 'utf8');          // va afficher : 7
```

PHP - Initiation

Les chaînes de caractères

III.3) Les expressions régulières

Les expressions régulières (communément appelées *regexp*) sont des **modèles** créés à l'aide de caractères ASCII **permettant de manipuler des chaînes de caractères**, c'est-à-dire permettant de trouver les portions de la chaîne correspondant au modèle.
Ce système est emprunté au système *POSIX* (les spécifications des systèmes UNIX). De nombreux scripts sous *UNIX* les utilisent. (notamment *Perl*).

Pour faire simple, il s'agit d'un **système fort ingénieux et très puissant** permettant de retrouver un mot, une phrase ou plus généralement un *motif* dans un texte.
Les expressions régulières permettent de rechercher des occurrences (c'est-à-dire une suite de caractères correspondant à ce que l'on recherche) grâce à une série de caractères spéciaux.

PHP - Initiation

Les chaînes de caractères

III.3.1) Les symboles ^ et $

Ils indiquent respectivement le début et la fin d'une chaîne et permettent donc de la délimiter.

- ***^debut*** correspond à toute chaîne qui commence par **debut**
- ***fin$*** correspond à toute chaîne qui se termine par **fin**
- ***^chaine$*** correspond à une chaîne qui commence et se termine par **chaine**
- ***abc*** correspond à une chaîne contenant le motif **abc**

PHP - Initiation

Les chaînes de caractères

III.3.2) Les symboles *, + et ?

Ils indiquent respectivement "zéro ou plusieurs", "un ou plusieurs" et "un ou aucun". Ils permettent de donner une notion de nombre.

- ***abc+*** : chaîne qui contient **ab** suivie de un ou plusieurs **c** (abc, abcc...)

- ***abc**** : chaîne qui contient **ab** suivie de zéro ou plusieurs **c** (ab, abcc...)

- ***abc?*** : chaîne qui contient **ab** suivie de zéro ou un **c** (ab ou abc)

- ***^abc+*** : chaîne commençant par **ab** suivie de un ou plusieurs **c** (abc, abccc...)

PHP - Initiation

Les chaînes de caractères

III.3.3) Les accolades {x, y}

Permettent de donner des limites de nombre plus complexes.

- ***abc*{2}** : chaîne qui contient **ab** suivie de 2 **c**

(abcc)

- ***abc*{2,}** : chaîne qui contient **ab** suivie de 2 **c** ou plus

(abcc, abccccccc...)

- ***abc*{2,4}** : chaîne qui contient **ab** suivie de 2, 3 ou 4 **c**

(abcc, abccc ou abcccc)

Remarques :

Le premier nombre de la limite est obligatoire, pas le dernier qui, s'il est omis, signifit qu'il n'y a pas de limite maximale. Les symboles vus précédemment (*****, **+**, et **?**) sont équivalents à **{0,}**, **{1,}**, et **{0,1}**.

PHP - Initiation

Les chaînes de caractères

III.3.4) Les parenthèses ()

Permettent de grouper une séquence de caractères.

- ***a*(*bc*)*** : chaîne qui contient **a** suivie de 0 ou plus **bc**

(a, abcbc...)

III.3.5) La barre verticale |

Se comporte comme l'opérateur **OU**.

- ***un*|*le chien*** : chaîne qui contient **un chien** ou **le chien**

- **(*a*|*b*)*** : chaîne qui contient une suite de **a** ou de **b**

(a, aaabaabba...)

PHP - Initiation

Les chaînes de caractères

III.3.6) Le point .

Remplace n'importe quel caractère, une fois.

- **^.{3}$** : chaîne qui contient exactement 3 caractères quelconques (a6J, A22...)

III.3.7) Les crochets []

Permettent de définir des ensembles de caractères.

- **[*aei*]** : chaîne qui contient un **a**, un **e** ou un **i**

Le signe **-** placé entre 2 caractères permet de définit un intervalle.

- **[*a-z*]** : chaîne qui contient un caractère compris entre **a** et **z**

PHP - Initiation

Les chaînes de caractères

Le caractère **^** après le premier crochet indique la négation de l'ensemble.

- **^[^*a-zA-Z*]** : chaîne qui ne commence pas par une lettre

En dehors des **[]** pour rechercher un caractère faisant partie des caractères spéciaux, il suffit de le faire précéder d'un **** (antislash). Dans les crochets, chaque caractère représente ce qu'il est. Pour représenter un **]** il faut le mettre en premier (ou après un **^** si c'est une exclusion).

- **[\+?{}.]** : chaîne qui contient un des 6 caractères ****, **+**, **?**, **{**, **}** ou **.**
- **[]-]** : chaîne qui contient **]** ou **-**

PHP - Initiation

Les chaînes de caractères

III.3.8) Les classes de caractères

Il peut également être utile de vérifier si une chaîne contient des caractères d'un certain type (numérique, alphanumérique...) sans avoir à les énumérer. Pour cela les expressions régulières définissent des classes de caractères, dont la syntaxe est : **[:classe:]**
Ces classes s'utilisent comme des ensembles de caractères à placer dans les **[]**

- **^[[:alnum:]]+$** : chaîne composée d'un ou plusieurs caractères alphanumériques

- **[[:punct:][:space:]]** : chaîne comportant un caractère de ponctuation ou un caractère d'espacement

- **[[:digit:]]?** : chaîne contenant 0 ou 1 chiffre

PHP - Initiation

Les chaînes de caractères

Voici un petit tableau récapitulant des classes de caractères possibles.

Classe	Description
[:alnum:]	caractères alphanumériques
[:alpha:]	caractères alphabétiques
[:blank:]	caractères blanc (espace, tabulation)
[:ctrl:]	caractères de contrôle (les premiers du code ASCII)
[:digit:]	chiffre (équivalent à 0-9)
[:graph:]	caractère d'imprimerie (qui fait une marque sur l'écran en quelque sorte)
[:print:]	caractère imprimable (qui passe à l'imprimante ... tout sauf les caractères de contrôle)
[:punct:]	caractère de ponctuation
[:space:]	caractère d'espacement

PHP - Initiation

Les chaînes de caractères

III.3.9) Les éléments PCRE

PCRE est l'acronyme de "Perl Compatible Regular Expressions".
Ces éléments sont donc compatibles avec les fonctions PCRE et étant donné que nous privilégierons leur utilisation, il s'avèrent important.
Nous allons gagner en lisibilité et en simplicité.

Classe	Description
\b	indique une limite de mot dans une chaîne de caractères
\B	indique ce qui n'est pas une limite de mot
\d	équivalent à [0-9] (digit)
\D	équivalent à [^0-9] (non digit)
\s	indique un espace blanc \t, \r, \n, \f. (space)
\S	indique ce qui n'est pas un espace blanc \t, \r, \n, \f.
\w	indique un mot qui correspond à la classe [0-9a-zA-Z_]
\W	indique ce qui n'est pas un mot, soit [^0-9a-zA-Z_]

PHP - Initiation

Les chaînes de caractères

III.3.10) Fonctions utilisant les REGEXP

Les fonctions PHP modernes utilisant les expressions régulières sont issues du langage PERL. Voyons en quelques unes.

preg_match('/*regexp*/', *chaîne*, *result*)

Cette fonction permet d'évaluer le texte passé en argument (***chaine***) grâce au motif, c'est à dire l'expression régulière (***regexp***) que l'on insère entre les slashs (on peut également utiliser des dièses). La fonction renvoie **true** s'il y a correspondance entre la chaîne et le motif, **false** dans le cas inverse.

Il est possible de stocker toutes les occurences dans un tableau passé en dernier argument (***result***).

PHP - Initiation

Les chaînes de caractères

Exemples

```
$chaine = 'Bonjour à tous';

$test = preg_match('/^[A-Z]onjour/', $chaine, $result);

echo $test;                    // va afficher : 1
var_dump($result);             // va afficher : array(1) { [0]=> string(7) "Bonjour" }
```

L'expression régulière de ce premier exemple exige que le premier caractère de la chaîne soit absolument une lettre majuscule suivie de '**onjour**'. Modifions la chaîne afin de constater l'efficacité :

```
$chaine = 'bonjour à tous';

$test = preg_match('/^[A-Z]onjour/', $chaine, $result);

echo $test;                    // va afficher : 0
var_dump($result);             // va afficher : array(0) { }
```

PHP - Initiation

Les chaînes de caractères

preg_replace('/*regexp*/', *rempla*, *chaîne*)

Cette fonction permet d'évaluer le texte passé en argument (***chaîne***) grâce au motif (***regexp***). La fonction remplace toutes les occurences trouvées par ***rempla***.

Exemples

```
$chaine = 'Bonjour à tous';
$rempla = 'Au revoir';
$result = preg_replace('/[A-Z]onjour/', $rempla, $chaine);
echo $result;                          // va afficher : Au revoir à tous
```

```
$chaine = 'J\'ai 3 livres de chevet';
$rempla = 'plusieurs';
$result = preg_replace('/[2-9]+/', $rempla, $chaine);
echo $result;                          // va afficher : J'ai plusieurs livres de chevet
```

Attention, le dernier exemple est imparfait : Il ne prends pas en compte les 1 et donc 11, 19 ou 41 ne correspondront pas !

PHP - Initiation

Les chaînes de caractères

preg_split('/*regexp*/', *chaîne*)

Cette fonction permet d'éclater la chaîne passée en argument (***chaîne***) grâce au motif (***regexp***). Un tableau est généré en retour contenant les sous-chaînes qui auront été séparées par le motif.

Exemple

```
$chaine = 'PHP mySQL, Javascript, jQuery';

$tableau = preg_split('/[\s,]+/', $chaine);
print_r($tableau);

 // va afficher :

Array ( [0] => PHP [1] => mySQL [2] => Javascript [3] => jQuery )
```

print_r(*tableau*) permet de visualiser le contenu d'un tableau avec moins d'information que **var_dump()**.