Langage PHP

Initiation à la programmation

Partie 1

Table des matières

I. Introduction :

A. Un peu de théorie :

1. *Du PHP dans du HTML :*

Les pages Web sont généralement composées de quatres langages informatiques *(les deux premiers ne sont pas des langages de programmation)* :

- Le langage `HTML` : tous les élèments affichés par le navigateur viennent du code HTML : mise en page *(ligne, paragraphe, tableau. . .)*, insertion d'images. . . Les trois autres langages se serviront du langage HTML pour afficher leur résultat.
- Le langage `CSS` améliore le rendu visuel des pages internet ; avec ce langage, il est plus facile de contrôler *(apparence et position)* des objets affichés sur la page. Il permet également d'homogénéiser la mise en page des différentes pages d'un site.
- Le langage `Javascript` : principalement, il permet de créer une interaction entre les actions du client *(mouvement de la souris, action sur une touche du clavier)* et les objets de la page : un menu déroule ou une partie de la page change de couleur lors du passage de la souris.

- Le langage PHP permet de contrôler la création des pages d'un site. A chaque demande d'un client, une page d'un site peut être entièrement reconstruite et sa création peut varier suivant des divers paramètres choisies par le programmeur : heure de la journée, identification d'un client...

  Le langage PHP peut également manipuler des bases de données *(généralement MySQL)* pour y accéder et/ou stocker facilement des informations.

Pour mieux comprendre l'utilité des deux langages Javascript et PHP, il est utile de faire une distinction importante :

- Le langage Javascript est un langage "*côté client*" : le code Javascript est tranmis au client en même temps que la page et il s'exécutera directement par le navigateur du client : il pourra alors inter-agir avec les actions du client : clic d'un bouton ou mouvement de la souris, action d'une touche du clavier, re-dimensionnement de la fenêtre...
- Le langage PHP est un langage "*côté serveur*" : avant d'envoyer une page au client, le serveur exécute le code PHP et n'envoye que le résultat du code, c'est à dire du HTML, du CSS et du Javascript : le PHP ne sert qu'à créer la page qui sera envoyée au client. Le code PHP est inaccessible au client.

Pour que le code PHP soit exécuté par le serveur d'hébergement, il est nécessaire que :

- le fichier ait pour extension "*.php*" ; ce fichier peut contenir tout type d'informations *(texte, HTML, Javascript...)*
- la partie du fichier contenant le code PHP doit être "*baliser*".

Il existe deux manières d'insérer du code PHP dans un fichier .php :

```
1 Conformément aux normes du langage HTML :
2 <script language="php">
3    On place ici le code PHP
4 </script>
5
6 L'introduction de code PHP la plus utilisée :
7 <?php
8   On place ici le code PHP
9 ?>
```

Dans ce tutorial, on insérera le code PHP à l'intérieur de la balise <?php ... ?>.

2. *Les instructions :*

Le PHP est un langage de programmation orienté objet, c'est à dire que le PHP est capable de créer des "*objets*" de programmation qui auront leur propre "*vie*" lors de l'exécution du programme. Ceci est une notion avancée de programmation.

Nous utiliserons, dans cette formation, la forme la plus classique de programmation : la programmation linéaire. Le programme est composé de différentes instructions s'exécutant les unes après les autres. La fin d'exécution d'un instruction entraîne l'exécution de la suivante et ainsi de suite jusqu'à arriver à la fin du code.

Voici la première règle de syntaxe du langage PHP : toute instruction doit se terminer par un point-virgule ";" ;

```php
<?php
$i=5;
echo "Hello";$j=7;
?>
```

Le retour à la ligne ne signifie pas la fin d'une instruction : seul le point-virgule ";" indique la fin d'une instruction.
L'exemple ci-dessus montre qu'une ligne peut contenir plusieurs instructions, mais ceci est plutôt à éviter afin de rendre le code plus lisible.

3. *Les noms de variables :*

Comme dans tous les langages de programmation, le langage PHP utilise des variables pour stocker temporairement des données et y travailler dessus.
En PHP, une variable naît *(elle est déclarée et affectée d'une valeur)* à l'intérieur du code et elle est détruite à la fin d'exécution du script.

Le nom d'une variable doit être toujours précédée par le caractère dollar "$" : par exemple, `$compte`.

Voici la seconde règle de syntaxe du langage PHP : le nom d'une variable doit commencer par une lettre ou un tiret-souligné "_" puis il est composé de lettres, de soulignés et/ou de chiffres.

Le nom d'une variable peut contenir des chiffres, mais il ne peut commencer par un chiffre :

Noms de variables autorisés	Noms de variables interdites
`$x` ; `$_tom` ; `$t2t`	`$1` ; `$1tom` ; `$.t`

Attention, les noms des variables et des fonctions sous PHP sont sensibles à la casse ; cela signifie que les deux variables `$x` et `$X` sont distinctes pour le PHP :

4. *Les types de variables, les affectations :*

Le langage PHP est un langage de programmation faiblement typé : au cours de l'exécution du code, une variable peut contenir successivement des entiers, des nombres décimaux, des chaînes de caractèrs...
Ce n'est pas le cas d'autres langages comme, par exemple, le langage C dont les variables sont

définies très précisemment avant d'être utilisée.
Lors de sa définition et avec le langage C, une chaîne de caractères doit être définie avec le nombre maximal de caractères qu'elle contiendra : ce genre de limitation n'existe pas en PHP et rend son apprentissage plus aisé.

On affecte une valeur à une variable à l'aide de l'opérateur "=" :

```php
<?php
$x="thomas";
$x=3;
?>
```

Le code ci-dessus montre la variable $x qui prend successivement deux valeurs de nature différente : une chaîne de caractères puis un nombre.

5. *La concaténation de chaînes de caractères :*

La concaténation est l'action de mettre bout à bout des chaînes de caractères. L'opérateur de concaténation en PHP est le point "." :

```php
<?php
$x="Bonjour";
$y="Thomas";
echo $x;
echo $y;
?>
```

```php
<?php
$x="Bonjour";
$y="Thomas";
echo $x.$y;
?>
```

Ces deux codes afficheront la même chose dans un navigateur : "BonjourThomas". Le code de droite concatène les valeurs des variables $x et $y avant de l'envoyer au navigateur.

Aucun espace n'a été inséré entre les deux mots car aucune des variables n'en contenaient. Voici une possibilité pour espacer ces deux mots :

```
echo $x." ".$y;
```

6. *Les conversions de types :*

Le langage PHP étant faiblement typé, lors d'opérations sur les variables, il s'occupe lui-même de convertir les valeurs afin d'effectuer l'opération :

- **Addition de nombres :**
  Dès que l'opérateur d'addition "+" est utilisé, PHP essaye de transformer les deux valeurs fournies en nombre voici un tableau illustrant différents cas :

	$x	$y	echo $x+$y;
1.	5	2	7
2.	5	"tom"	5
3.	5	"2tom"	7
4.	5	"t2om"	5
5.	5	"5e2tom"	505

Voici quelques explications :

2. La chaîne `"tom"` ne commençant pas par un nombre, elle est remplacé par `PHP` par la valeur 0.
3. La chaîne `"2tom"` est remplacé par la valeur `2`.
4. La valeur de `$y` ne commençant pas par un nombre, la valeur de `$y` est considérée comme étant 0.
5. Lors de l'addition, seule la partie `5e2` de la variable `$y` sera utilisée : cette partie ayant une valeur de 5×10^2.

- **Concaténation de chaines de caractères :**

  L'opérateur de concaténation "." est beaucoup plus facile à comprendre : toute donnée sera transformée en chaîne de caractères.

	$x	$y	echo $x.$y;
1.	5	3	"53"
2.	"a"	3	"a3"
3.	array(5,3)	3	"Array3"

Mais voici une remarque :

3. La variable `$x` est un tableau *(étudié dans un prochain chapitre)*, lors de sa conversion en chaîne de caractère, `PHP` le transformera en `"Array"`.

7. *Passage d'informations via l'URL :*

A chaque demande d'une page, le navigateur du client envoye des informations "*cachées*" au serveur : navigateur utilisé, fuseau horaire défini sur la machine du client...

Ces données peuvent être utilisées par le code `PHP` pour afficher ou non des informations : imaginez le fond d'une page verte le matin et bleu l'après-midi.

Mais ces données restent relativement impersonnelle. Pour créer un plus grand "*dialogue*" entre le client et le serveur, une des méthodes et d'adjoindre des informations via l'adresse *(l'URL)* de la page demandée.

Considérons l'adresse suivante :

`http://chingatome.net/index.php?nom=thomas&age=23`

Lorsque le serveur d'hébergement du site "*chingatome.net*" recevra la demande de cette page : il ira chercher le fichier `index.php` et l'exécutera en créant les deux variables `$_GET["nom"]` et `$_GET["age"]` contenant respectivement les valeurs "*thomas*" et "*23*".

Analysons syntaxiquement cette adresse :

- le nom du fichier et les variables à transmettre sont séparées par le caractère "`?`" ;
- les variables entre elles sont séparés par l'esperluète "`&`" ;
- l'affectation des variables s'effectue avec l'opérateur "`=`".

B. Quelques exercices dirigés :

Exercice 1

1. a. Lancer `WampServer` à partir du bouton démarrer et dans la section tous les programmes. Vérifier que l'icone de `WampServer` passe au vert dans la zone de notification.

   b. A la racine du serveur Web `C:\wamp\www`, créer un nouveau fichier `PHP` à l'aide du menu contextuel "*Nouveau*". Le renommer en `index.php`.

   c. En double-cliquant sur ce fichier, l'éditeur de texte "*NotePad++*" doit se lancer. Compléter le fichier pour obtenir le code suivant :

```
1 <b>Salut</b>
2 <?php
3  echo ". Thomas";
4 ?>
5 <i>Au revoir</i>
```

   d. Enregistrer le fichier puis en effectuant la combinaison de touches "*Ctrl+Alt+Shift+X*", Firefox affichera cette page.

WampServer est un logiciel du domaine libre contenant tous les logiciels-tiers nécessaire à la création d'un serveur Web :

- un serveur Web : le logiciel Apache s'occupe de récupérer les requêtes des clients et de leur renvoyer les pages demandées ;
- un interprétateur de code `PHP` : ce logiciel nommé simplement PHP permet de transformer le code `PHP` en instructions pour l'ordinateur ;
- une base de donnée : on dispose de MySQL qui est la base la plus populaire dans le

domaine des logiciels libres de droits.

Notepad++ est un simple éditeur de texte : il permet de saisir son texte, son code. Il met en évidence *(changement de couleurs, surligne...)* des parties du code pour en faciliter la saisie. La combinaison de touches faite dans ce logiciel a été modifié afin de lancer l'exécution de fichiers à la racine de notre site Web.

2. Intéressons nous d'abord à l'ouverture du navigateur :

   a. Ecrivez ci-dessous l'URL complète de localisation de cette page :

.....

   b. Que signifie `http` ?

.....

.....

   c. Que signifie `127.0.0.1` ?

.....

.....

   d. Quel est le fichier qui est retourné au client ?

.....

Pour tout système d'exploitation, l'adresse `IP 127.0.0.1` ou `localhost` représente l'ordinateur lui-même : cette adresse permet à un ordinateur de s'auto-désigner.

Ainsi, l'adresse `http://127.0.0.1` permet de désigner le serveur Web présent sur la machine utilisée.

3. Intéressons nous à l'affichage du navigateur :

   a. Quel est la fonction de l'élément HTML b défini par les balises `<b>` et `</b>` ?

.....

   b. Quel est la fonction de l'élément HTML i défini par les balises `<i>` et `</i>` ?

.....

   c. Pour qu'elle raison aucun saut de lignes et de paragraphes n'apparaissent à l'affichage de la page ?

.....

   d. Justifiez pourquoi le contenu des lignes 2 à 4 ne s'affiche pas intégralement ?

.....

.....

e. Quel est le rôle de la fonction `echo` en PHP ?

. . . . . .

. . . . . .

Pour le HTML :

- L'élément b affiche son contenu, défini entre les deux balises `<b>` et `</b>`, à l'aide d'une fonte grasse.
- L'élément i affiche son contenu, défini entre ses deux balises `<i>...</i>`, à l'aide d'une fonte en italique.
- Pour indiquer un saut de ligne dans le code HTML, il faut saisir la balise `<br>` et pour un saut de paragraphes, c'est `<p>`.
  Les sauts de lignes saisis au clavier dans le code ne sont pas interprétés dans le langage HTML.

Les ligne 2 et 4 désignent respectivement l'entrée et la sortie du code PHP.

La ligne 3 permet l'affichage de la chaîne `". Thomas"` : la commande `echo` enverra à l'affichage la chaîne de caractère passée en argument.

Exercice 2

1. A la racine de votre site *(le dossier* `C:\wamp\www`*)*, créez un nouveau fichier intitulé `chaine.php` et contenant le code suivant :

```
1 Bonjour,
2 <script language="php">
3   $x="Salut";
4   $y="thomas";
5   echo $x."<br>".$y;
6 </script>
```

2. Dans NotePad++, enregistrez le fichier et effectuez la combinaison de touches "*Ctrl+Alt+Shi X*" pour afficher la page Web.

3. Répondez aux questions suivantes :

   a. Quelles lignes dans ce script définissent l'entrée et la sortie du script PHP ?

. . . . . .

   b. Combien de variables sont utilisés par ce script ? Quels sont leurs noms ?

. . . . . .

   c. Quelle est la fonction du point "." contenu dans l'argument de la commande `echo` ?

. . . . . .

. . . . . .

d. Quelle est la chaîne de caractères affichée par la commande `echo` ?

e. Quelle est la fonction de l'élément `br` dans le langage HTML ?

Bien que très peu utilisées, il est possible de déclarer un script `PHP` à l'aide de l'élément HTML `script` avec les deux lignes suivantes :

```
<script language="php">
  ...Emplacement du script PHP
</script>
```

En `PHP`, les noms de variables doivent commencer par le caractère dollar `$` et se poursuivent par des caractères alphabétiques et des tirets-soulignés *(les chiffres ne sont autorisés qu'au milieu et fin du nom d'une variable).*

Le point "." est l'opérateur de concaténation des chaînes : il permet de mettre "*bout à bout*" deux chaînes de caractères.

La commande `echo` renvoye la chaîne de caractère suivante au navigateur du client :

`Salut<br>Thomas`

L'élément `br` indique dans le code HTML le saut de ligne ; ainsi les deux mots "*Salut*" et "*Thomas*" seront affichés dans le navigateur sur deux lignes différentes.

Exercice 3

1. A la racine de votre site, créez un fichier intitulé `passage.php` et y saisir le code suivant :

```
<?php
$i=$_GET["age"];
echo "Bienvenue ".$_GET["nom"]."<p>";
echo "L'année prochaine vous aurez ".($i+1)." ans";
?>
```

2. a. Affichez cette page dans le navigateur Firefox.

b. Des messages d'erreurs s'affichent à l'écran. Pouvez-vous donner la nature de ces messages d'erreurs ?

. . . . . .

. . . . . .

L'erreur "*Undefined index...*" indique que la valeur demandée dans un tableau n'existe

pas. Les variables de type "*tableau*" seront étudiées dans le troisième paragraphe.

La variable `$_GET["nom"]` fait référence à une valeur passée via l'URL sous la forme `?nom=thomas`. Celle-ci n'étant pas présente dans l'URL du navigateur, PHP ne peut pas récupérer sa valeur et continuer son exécution : d'où l'erreur et l'arrêt de l'exécution du code.

3. a. Complétez l'adresse du navigateur en :
   `http://127.0.0.1/passage.php?nom=thomas&age=50`

   b. Lors de l'exécution de cette page, quelles sont les valeurs des variables `$_GET["nom"]` et `$_GET["age"]`.

   c. Pourquoi l'affichage du navigateur comporte deux paragraphes ?

Lors du nouvel appel de cette page, les variables `$_GET["nom"]` et `$_GET["age"]` sont définies dans l'URL : l'exécution du code s'effectue sans erreur.

L'apparition du saut de paragraphe ne vient pas de la double présence de la commande `echo` mais de la présence de l'élémet HTML p représentant un saut de paragraphe.

C. Quelques travaux pratiques :

Exercice 4

Créez un fichier PHP nommé "*arith.php*" qui modifiera son affichage en fonction de la variable `$_GET["i"]` récupérée dans l'URL à cette page. Voici deux exemples vous permettant d'imaginer la structure de cette page :

`http://127.0.0.1/arith.php?i=3`

```
1 Le nombre saisi est : 3
2 Son carré vaut : 9
3 Son successeur est : 4
4 Son prédécesseur est : 2
```

`http://127.0.0.1/arith.php?i=10`

```
1 Le nombre saisi est : 10
2 Son carré vaut : 100
3 Son successeur est : 101
4 Son prédécesseur est : 99
```

Exercice 5

Créez un script qui prenant les valeurs du nom, du prénom d'un personne renvoye une page de la manière suivante :

```
1 Bonjour Nom Prénom
2
3 Nous sommes le Thuesday 18 May à 18:42:34
```

Voici quelques commandes utiles pour la réalisation de cette page :

- La chaîne de caractères contenant le jour de la semaine, le jour du mois et le mois est obtenu par la commande `date("l j F")` ;
- La chaîne de caractères contenant l'heure, les minutes, les secondes est obtenu par la commande `date("H:i:s")`.

Vous trouverez davantage d'information sur la commande `date` en utilisant les mots-clés suivant dans un moteur de recherche :

```
php manual date
```

Exercice 6

A la racine de votre espace d'hébergement, placez deux images nommées `0.jpg` et `1.jpg`. Créez un script qui affiche une image ou l'autre suivant que les minutes, définissant l'heure du serveur, soient un nombre pair ou impair lors de l'affichage de la page.

Voici quelques indications pour la réalisation de ce script :

- L'élément `img` permet l'affichage d'une image : pour afficher l'image "`1.jpg`", on saisit le code HTML suivant `<img src="1.jpg">`
- Les minutes du serveur s'obtiennent à l'aide de l'instruction `date("i")`.
- L'opérateur "`%`" permet d'obtenir le reste de la division euclidienne de deux entier : en `PHP`, l'instruction `5%3` retourne la valeur `2`.

II. Les structures de contrôle :

A. Un peu de théorie :

1. *Introduction :*

Les structures de contrôle permettent, dans un langage de programmation, de contrôler l'exécution d'un programme :

- elles donnent la possibilité d'exécuter ou non une partie du programme : on parle alors de **structures conditionnelles**.

- elles donnent la possibilité de répéter l'exécution d'une partir du programme : on parle alors de **structures répétitives** ou de **boucles**. On distingue deux types de boucles :
  - la boucle itérative : `for(...){...}`. Elle permet de répéter plusieurs intrustructions, placées entre accolade, un certain nombre de fois connu à l'avance. Une variable permet alors de compter et de controler le nombre de répétition de la boucle
  - les boucles conditionnelles : `while(...){...}` et `do{...}(...)`.
    Elles permettent, tant qu'une condition fixée à l'avance est vérifiée, de répéter plusieurs instructions.

On vient de voir que l'usage des structures de controles nécessitent l'usage de condition. Par exemple, pour voir si la variable `$a` est strictement supérieur à 2, on utilisera la condition `$a>2`. Si une condition est remplie, elle retourne la valeur `true` et si elle n'est pas vérifié, la condition retournera la valeur `fase`.

2. *Booléen et opérateurs de comparaison :*

Les booléens sont des objets informatiques ne prenant que deux valeurs : `true` ou `false` : ils symbolisent les résultats de conditions nécessairent aux structures de contrôles

Voici les opérateurs de comparaisons sur les variables :

`Comparaison`	Renvoi `true` si :
`$a==$b`	`$a` et `$b` contiennent la même valeur
`$a!=$b`	`$a` et `$b` ont de valeurs différentes
`$a<$b`	`$a` a une valeur strictement inférieure à `$b`
`$a<=$b`	`$a` a une valeur inférieure ou égale à `$b`
`$a>$b`	`$a` a une valeur strictement supérieure à `$b`
`$a>=$b`	`$a` a une valeur supérieure ou égale à `$b`
`$a===$b`	`$a` et `$b` ont même valeur et de même type
`$a!==$b`	`$a` et `$b` n'ont pas la même valeur ou ne sont pas du même type

Les deux dernières lignes du tableau méritent une explication. Considérons les deux variables `$a` et `$b` comme suit : `$a="3";` et $b=3; :

- Le type de ces deux variables est distinct mais leur valeur, préalablement converties par `PHP` sont égales : la condition `$a==$b` retournera la valeur `true` ;
- Le test "*triple-égal*" teste la valeur et le type des valeurs des deux variables : `$a===$b` retournera la valeur `false`.

Il faut donc faire attention aux conversions implicites réalisés par PHP. Ainsi, les cinq objets ont tous la même valeur pour PHP mais sont de natures distinctes :

`null` ; `0` ; `""` ; `"0"` ; `array()`

Il existe deux opérateurs sur les booléens afin de rassembler plusieurs conditions : l'opérateur *"ET" (&&)* et l'opérateur *"OU" (||)*. Voici les tableaux de valeurs de ces deux opérateurs binaires :

3. *Structures conditionnelles :*

La structure `if...then...else...` permet d'exécuter une partie du code en fonction de la valeur d'une condition *(*`true` *ou* `false`*)* :

```
if(condition){
   Instruction vraie 1
   Instruction vraie 2
   ...
}
else{
   Instruction fausse 1
   Instruction fausse 2
   ...
}
```

```
if(condition){
   Instruction vraie 1
   Instruction vraie 2
   ...
}
```

Si les clauses vraies ou fausses ne contiennent qu'une unique instruction, il est possible d'omettre les accolades { et } :

```
if(condition)
   Instruction vraie
else
   Instruction fausse 1
Suites des Instructions
```

```
if(condition)
   Instruction vraie
Suites des Instructions
```

Dans le cas où les accolades ne sont pas utilisées, seule la première instruction fait partie de la structure conditionnelle : ci-dessus, le code de droite présente une telle situation.

Il est à noter une une forme abrégée de cette structure conditionnelle utile pour des structures conditionnelles courtes afin d'alléger le code. En voici un exemple montrant l'affectation d'une

valeur à la variable `$x` :

```php
$x=($a>2)?"supérieur à 2":"inférieur à 2";
```

La structure conditionnelle `switch` permet d'exécuter différents parties du code en fonction de la valeur d'une variable :

```php
switch($x){
  case 0:
    Instructions si $x==0 ...
    break;
  case "a":
    Instructions si $x=="a"...
    break;
}
```

```php
switch($x){
  case 0:
    Instructions si $x==0 ...
    break;
  case "a":
    Instructions si $x=="a" ...
    break;
  default:
    Instructions sinon ...
}
```

Le code de droite présente l'instruction `default` : si aucune égalité n'a été vérifiée au cours de l'exécution de `switch` alors `PHP` exécutera les instructions suivant `default`.

4. *Boucles conditionnelles :*

```php
do{
  Instructions
}
while(Condition)
```

```php
while(Condition){
  Instructions
}
```

Ces deux structures répétitives exécuteront les instructions placées dans leur corps *(représenté par les accolades)* tant que la condition sera vérifiée. Dès que la condition n'est plus vérifiée, l'exécution du `PHP` se poursuit après la boucle.

Ces deux boucles présentent seulement une différence :

- La boucle `do{...}(...)` exécute d'abord le corps de ses instruction et ne les répétera que si la condition est vérifiée : on est sur que ses instructions s'exécutent au moins une fois ;
- La boucle `while(...){...}` exécute d'abord la condition et en fonction répétera alors le corps de ses instructions. Les instructions ne seront pas nécessairement exécutées.

Pour que l'exécution d'une boucle s'arrête, il faut que la condition deviennent fausse. Les instructions du corps de la boucle modifient les valeurs entrant en jeu dans la condition afin que celle-ci deviennent fausse à un moment donnée : sinon, le programme ne pourra jamais sortir de l'exécution de la boucle créant un "*bug*".

Un autre moyen de sortir définitivement de l'exécution d'une boucle est d'utiliser l'instruction

`break`.

Il est également possible de passer d'achever prématurement l'exécution du corps d'une boucle et de passer au prochain passage de la boucle à l'aide de l'instruction `continue`.

5. *Boucles itératives :*

Cette boucle permet d'éxécuter un nombre connu par avance de fois un jeu d'instruction. En voici un exemple :

```
for($i=0;$i<30;$i++){
  Instructions à exécuter 30 fois
}
```

Les commandes `continue` et `break` sont également disponibles pour cette boucle.

B. Quelques exercices dirigés :

Exercice 7

1. A la racine de votre site, créez un fichier `login.php` et saisissez le code suivant :

```
<?php
echo "Bienvenue sur Woezon.net<p>";
if($_GET["nom"]=="thomas"){
  echo "Vous êtes l'administrateur du domaine";
}
else{
  echo "Vous êtes un invité du domaine";
}
?>
```

2. Affichez la page `http://127.0.0.1/login.php` dans votre navigateur :

   a. Une erreur doit s'afficher dans votre navigateur. Quelle est la nature de cette erreur ?

.....

.....

   b. Quelle URL doit-on saisir dans le navigateur pour que la page affiche "*Vous êtes l'administrateur...*" ?

.....

Le message d'erreurs a déjà été vu dans le paragraphe précédent : le script `PHP` cherche à utiliser la variable `$_GET["nom"]`.
Or, cette variable, pour qu'elle existe, doit être définie via l'URL de la page ce qui n'est pas le cas dans la question précédente.

C'est en saissant l'adresse `http://127.0.0.1/login.php?nom=thomas` que la variable `$_GET["nom"]` sera affectée de la valeur adéquate pour afficher le message "*Vous êtes l'administrateur du domaine*".

3. a. Saisissez à nouveau l'adresse `http://127.0.0.1/login.php` pour afficher le message d'erreurs.

b. Remplacez la conditionnelle `($_GET["nom"]=="thomas")` par :

```
((isset($_GET["nom"])) && ($_GET["nom"]=="thomas"))
```

c. Rafraichir la page du navigateur. Qu'observez-vous ?

.....

.....

La commande `isset(...)` vérifie si une variable a été définie préalablement. Ainsi, si on accède au fichier `login.php` via l'adresse :

`http://127.0.0.1/login.php`

alors la variable `$_GET["nom"]` n'existera pas et l'instruction `isset($_GET["nom"])` sera fausse entrainant l'affichage du message "*Vous êtes un invité*".

Le message d'erreur n'apparait grâce à une particularité de l'opérateur logique `&&` : dès qu'une succession de `&&` rencontre une première fois la valeur `false`, l'exécution s'arrête et retourne `true`. Ainsi, avec l'adresse `http://127.0.0.1/login.php`, l'exécution de la conditionnelle :

```
((isset($_GET["nom"])) && ($_GET["nom"]=="thomas"))
```

n'exécutera pas la partie `$_GET["nom"]=="thomas"` car la commande `isset(...)` a déjà renvoyer la valeur `false`.

Exercice 8

1. A la racine du site, créez un nouveau fichier nommé `langue.php` et saisissez le code suivant :

```php
<?php
switch($_GET["lang"]){
  case "fr":
    $x="Bonjour";
    break;
  case "en":
    $x="Hello";
    break;
```

```
9    case "es":
10     $x="Ola";
11     break;
12   default:
13     $x="??";
14 }
15 echo $x;
16 ?>
```

2. Quel URL doit-on saisir dans le navigateur pour que la page affiche le mot “*Hello*” ?

. . . . . .

. . . . . .

Le test s’effectue avec la structure conditionnelle `switch(...){...}` qui utilise pour test la variable `lang`.

Pour afficher le mot “*Hello*”, il faut que la condition `$lang=="en"` retourne la valeur `true`. Ainsi, il faut que l’appel à cette page se fasse via l’URL :

`http://127.0.0.1/langue.php?lang=en`

3. a. Placez à la racine de votre hébergement un fichier `fr.txt` contenant quelques phrases en français.

b. Remplacez la ligne 4 du fichier `langue.php` par le code suivant :

```
include("fr.txt"); $x="";
```

c. Lancez l’URL suivante dans un navigateur :

`http://127.0.0.1/langue.php?lang=fr`

Quelle est la fonction de la commande `include(...)` ?

. . . . . .

. . . . . .

d. Justifiez la présence de l’instruction `$x=";` à la ligne 4 :

. . . . . .

La commande `switch(...){...}` permet de simplifier l’écriture d’une structure conditionnelle dans le cas de plusieurs tests à effectuer.

La commande `include(...)` insère le contenu d’un fichier lors de son exécution. Dans le cas où le fichier admet pour extension `.php` et si fichier contient du code `PHP` alors celui-ci sera exécuté.

A la suite de la structure `switch(...){...}`, l’instruction `echo $x;` sera exécutée : si

la variable `$x` n'existe pas alors un message d'erreur sera affiché. Voilà pourquoi dans ce code, chaque partie de `switch` doivent comporter une initialisation de la variable `$x`.

Exercice 9

1. a. A la racine de votre, créez un nouveau fichier intitulé `salutation.php` contenant le code :

```php
<?php
for($i=0;$i<100;$i++){
  if($i%4==0){
    echo "<br>";
  }
  else{
    echo " ";
  }
  echo "Bonjour";
}
?>
```

   b. Afficher cette page dans votre navigateur à l'aide de l'URL :

   `http://127.0.0.1/salutation.php`

2. a. Quelles sont les valeurs prises par la variable `$i` au cours de l'exécution de ce script ? Combien de fois le mot "*Bonjour*" apparait-il dans la page ?

.....

   b. Quelle modification doit-on apporter au code-source de cette page pour afficher 6 fois le mot "*Bonjour*" par ligne ?

.....

.....

La variable `$i` est initialisée avec la valeur `0` ; à chaque passage de la boucle, la variable `$i` voit sa valeur augmentée de `1` *(à l'aide de l'opérateur* `$i++`*)* : on parle d'incrémentation de la variable `$i`.

La condition d'arrêt est définie par `$i<100`. Ainsi, la boucle ne s'exécutera plus lorsque i aura la valeur 100 : la dernière fois que la boucle sera évalué, la variable i aura la valeur 99.

Le mot "*Bonjour*", étant hors de la conditionnelle, il sera affiché à chaque exécution de la boucle : il apparaîtra 100 fois dans la page Web.

L'opérateur `%` donne le reste de la division euclidienne de `$i` par `4`. Ainsi, une fois sur

4, le script PHP renverra à la page Web la balise HTML `<br>` : celle-ci insérant un saut de ligne dans la page Web.

3. a. Dans la ligne 3, modifier le code `($i%4==0)` par `($i%4)` et chnger l'espace de la ligne 7 par le caractère `x`. Exécutez le script en relançant la page dans votre navigateur.

b. Justifiez pourquoi la lettre `x` apparaît toutes les quatres lignes ?

. . . . . .

. . . . . .

. . . . . .

Le langage PHP est un langage informatique peu typé, c'est à dire que PHP peut associer facilement des valeurs de types différentes. Pour se faire, PHP exécute implicitement des changements de variables.

Lors du test d'une condition, les valeurs `0`, chaine vide `""` et `null` sont associées à la valeur booléennes `false`. Toutes les autres valeurs seront associées à `true`.

Ainsi, c'est seulement lorsque la variable `$i` sera divisible par 4 que la clause "*vrai*" de la structure `if` sera évaluée.

Exercice 10

1. a. A la racine de votre site, créez un nouveau fichier intitulé `pgcd.php` et saisissez le code ci-dessous :

```php
<?php
$a=$_GET["a"];
$b=$_GET["b"];
while($a%$b!=0){
  $r=$a%$b;
  $a=$b;
  $b=$r;
}
echo "PGCD(".$_GET["a"].";".$_GET["b"].")=".$b;
?>
```

b. Quelle URL doit-on saisir dans le navigateur afin que cette page affiche le PGCD de 5464 et 368. Quelle est la valeur de ce PGCD ?

. . . . . .

2. Modifier le programme afin que soit affichée l'écriture simplifiée de la fraction $\frac{a}{b}$.

C. Quelques travaux pratiques :

Exercice 11

La table ASCII est une table de correspondance entre les caractères utilisées couramment *(en Europe et aux Etats-Unis)* et les nombres allant de 1 à 255.

Evaluez la chaîne de caractère suivante pour des valeurs de `$i` allant de 1 à 255 :

```
echo "Le nombre ".$i." représente le caractère ".chr($i)."<br>";
```

Exercice 12

Utilisez la structure conditionnelle `switch(){}` pour afficher le jour de la semaine *(lundi, mardi, ...)* en français ; on se servira de la commande suivante :

```
date("w");
```

qui renvoie `0` pour dimanche, `1` pour lundi...

Exercice 13

Ecrire un programme qui donne le PPCM de deux nombres.

III. A propos des variables :

Au cours de ce paragraphe, des fonctions seront données afin de manipuler les variables. La liste des fonctions est longue en `PHP` et la façon de les utiliser *(nombre d'arguments, type de la valeur de retour...)* augmente le champ des possibles.

Seule la consulation d'un site officiel tel que `http://php.net/manual/fr` vous donnera toutes les indications que vous pouvez chercher à propos d'une fonction.
A vous d'aller fouiller cet incroyable site.

A. Un peu de théorie :

1. Les chaînes de caractères :

En `PHP`, une chaine de caractères est définie :

- soit par deux guillemets simples ' : `'Une chaîne de caractères'` ;
- soit par deux guillement double " : `"Une chaîne de caractères"`

Dans les deux cas, le même caractère doit être utilisé pour définir l'entrée dans une chaîne de caractères et pour en définir la fin. Par exemple, l'expression ci-dessous ne définit qu'une seule chaîne de caractères :

```
'Voici une "seule" chaine de caractères'
```

Une seule propriété différencie les chaînes de caractères définies par ' ou par " : à l'intérieur d'une chaîne définie par les guillemets doubles, les variables sont évalués. En voici un exemple :

```
1 $x="Thomas";
2 echo "Bonjour $x"; Son execution affiche "Bonjour Thomas"
3 echo 'Bonjour $x'; Son execution affiche "Bonjour $x"
```

On remarque que la variable `$x` n'est pas évaluée à l'intérieur de la chaîne définie par les simples guillemets.

L'instruction suivante provoque une erreur : `echo 'Aujourd'hui';`
La commande `echo` va afficher la première chaîne `'Aujourd'` et va attendre que `PHP` lui traduise le reste de l'expression : c'est à dire `hui'` qui provoque une erreur.

La remarque ci-dessus entraîne une question : comment afficher un guillemet simple dans une chaîne de caractères définie par des guillemets simples : on fait précéder le guillemet à afficher du caractère \ : la barre contre-oblique.
Ce caractère enlève au guillemet sa fonction de délimiteur au sein du `PHP` de chaîne de caractères et permet au `PHP` de le considérer comme simple caractère : on l'appelle le caractère d'échappement.

Voici quelques utilisations du caractère d'échappement \ :

Caractère à afficher	Chaîne de caractères débutant par	
	des guillemets doubles	des guillements simples
"	\"	"
'	'	\'
saut de ligne	\n	\n
retour chariot	\r	\r
tabulation	\t	\t

Voici quelques fonctions utiles ; la liste est non-exhaustive et beaucoup d'options facultatives de ces fonctions ne sont pas présentées ici ; reportez-vous à la documentation de `PHP` en ligne pour plus d'informations :

Pour repérer un caractère ou une partie d'une chaîne, `PHP` donne la position de chaque caractère d'une chaîne de la manière suivante :

- le premier caractère est repéré par la position `0` ;
- le second caractère est repéré par la position `1` ; ...

Ci-dessous sont présentées les fonctions les plus utilisées en PHP avec les chaînes de caractères. L'abbréviation indiquée en italique devant chaque fonction représente le type de la valeur retournée.

- *integer* `strlen($x)` :
  Cette commande retourne la taille de la chaîne de caractère contenue dans la variable `$x`.
- *string* `str_replace($ancien,$nouveau,$chaine)` :
  Cette fonction cherche, dans la chaîne représentée par `$chaine`, toutes les occurences d'apparition de la sous-chaîne contenue dans la variable `$ancien` et les remplacent par la chaine contenu dans la variable `$nouveau`. La nouvelle chaîne est renvoyée par cette fonction : attention, cette fonction ne modifie pas la valeur de la variable `$chaine`.
- *string* `strtolower($x)` :
  Cette fonction prend la valeur de `$x` est change toutes les majuscules en minuscules. La nouvelle chaîne, ainsi modifiée, est retournée par cette fonction : attention, elle ne modifie pas la valeur de la variable `$x`.
  Quant à la fonction `strtoupper($x)`, elle transforme toutes les minuscules d'une chaîne de caractères en majuscules.
- *string* `substr($chaine,$x,$y)` :
  Cette fonction retourne une sous-chaîne de la valeur de la variable `$chaine` définie par :
  - cette sous-chaîne commencera par le caractère de position `$x` ;
  - cette sous-chaîne aura une longueur de `$y` caractères.

2. *Les tableaux :*

En informatique, un tableau est un objet pouvant contenir lui-même plusieurs objets.

En PHP, un même tableau peut contenir des objets de type différents. Chaque objet contenu dans un tableau doit être repéré par un identifiant : cet identifiant s'appelle la clé de l'objet. Ainsi, un tableau est une collection de couples (clé;objet).

Il existe deux types de tableaux sous PHP, deux types de construction de tableaux. Dans les deux cas, la commande `array(...)` permet de construire un nouveau tableau :

- **Les tableaux indexés :**
  On parle de "*tableaux indexés*" lorsque les clefs de tous les objets sont des entiers naturels.

On définit une variable `$x` comme étant un tableau de la manière suivante :

```
$x=array(0 => "douala", 1 => 2010, 2 =>"mai");
```

On remarque l'opérateur `=>` associant à chaque clé *(ou index)* l'objet associé.

Pour accéder à la valeur "`2010`" du tableau `$x`, on utilise les crochets de la manière suivante `$x[1]`.

`PHP` permet de simplifier la saisie de tableaux indexés. Voici une manière plus rapide de coder le tableau précédent :

```
$x=array("douala",2010,"mai");
```

L'indexation des élèments commencera à 0 et s'incrémentera pour chaque objet *(c'est à dire l'indexation ira de un en un).*

Pour rajouter un élément directement à l'index `15` ou pour modifier l'objet indéxé par le nombre 15, on utilise l'instruction suivante :

```
$x[15]="nouveau";
```

Pour effacer, un index et sa valeur associée, on utilise la commande `unset()` :

```
unset($x[15]);
```

Pour rajouter un élément à la fin d'un tableau indéxé, on utilise l'instruction :

```
$x[]="nouveau";
```

- **Les tableaux associatifs :**
  Dans un tableau associatif, les objets contenus dans le tableau sont identifiés par des chaînes de caractères : on dira que les clés du tableau sont des chaînes de caractères. Voici un exemple de définition de tableau :

  ```
  $x=array("a" => "abricot", "b" => "banane");
  ```

  Dans un tableau associatif, pour rajouter une valeur, il est nécessaire de préciser la clé :

  ```
  $x["c"]="cerise";
  ```

Voici quelques fonctions associées aux variables de type tableau :

- *integer* `count($x)` : cette fonction renvoie le nombre d'éléments contenus dans le tableau `$x`.
- *boolean* `sort($x)` : cette fonction trie les valeurs du tableau ; la variable `$x` est modifiée par cette fonction.
- *boolean* `in_array($x,$tab)` : cette fonction renvoie la valeur `true` si la valeur de `$x` est contenue dans le tableau contenu dans `$tab`.

B. Quelques exercices dirigés :

Exercice 14

1. A la racine de votre site, créez un fichier nommé `casse.php` et saisissez le code suivant :

```
1  <?php
2  if(isset($_GET["ch"]))
3    $x=$_GET["ch"];
4  else
5    $x="Animaux";
6
7  //echo $x."<br>";
8  $x=str_replace("a","bb",$x);
9  //echo $x."<br>";
10 $x=strtolower($x);
11 echo $x."<br>";
12 if(substr($x,1,3)=="hom"){
13   echo "oui";
14 }
15 else{
16   echo "non";
17 }
18 ?>
```

Il est possible d'insérer dans du code des commentaires, c'est à dire du texte qui ne sera pas évalué lors de l'exécution du code :

- Une ligne de commentaire doit être précédée du double caractère `//` ;
- Pour qu'une partie s'étalant sur plusieurs lignes soit considérée comme un commentaire, il faut que cette partie débute par `/*` et se termine par `*/`.

On remarque deux lignes commentées sont présentes dans le code précédent : aux lignes 7 et 9. Ces deux lignes peuvent s'exécuter en supprimant les deux caractères `//` du début de ligne.
Ces deux lignes sont très utiles pour le "*débugage du code*" : en effet, en les décommentant, l'exécution de ces lignes affichera les valeurs intermédiaires de la variable $x.

2. a. Affichez la page dans votre navigateur. Combien de lignes s'affichent ?

. . . . . .

b. Quel est l'utilité de la structure conditionnelle débutant à la ligne 2 ?

. . . . . .

. . . . . .

. . . . . .

c. Quel est l'utilisation de la fonction `str_replace(...)` ? Donnez le rôle joué par chacun de ses trois arguments.

. . . . . .

. . . . . .

. . . . . .

d. Pourquoi le premier "`A`" du mot "`Animaux`" n'a pas été changé en "`bb`" ?

. . . . . .

. . . . . .

e. Pourquoi la dernière ligne d'affichage du navigateur est "*non*" ?

. . . . . .

. . . . . .

Si l'URL, appelant le fichier `casse.php`, définit une valeur pour `ch` alors `PHP` créera automatiquement la *(c'est à dire l'existence de la variable* `$_GET["ch"]`*)*.
Ainsi, la structure conditionnelle débutant à la ligne 2 teste si la valeur de `ch` a été définie dans l'URL :

- si c'est le cas, la variable `$x` récupèrera cette valeur ;
- si ce n'est pas le cas, la variable `$x` recevra la valeur par défaut `"Animaux"`.

La fonction `str_replace()` va remplacer toutes les caractères "`a`" en "`bb`". `PHP` est sensible à la casse et il considère donc les caractères "`a`" et "`A`" comme étant distinct. Seul la lettre "*a*" minuscule sera affectée par la commande `str_replace(...)` de la ligne 8.
On finit la transformation de la variable `$x` à l'aide de la commande `strtolower(...)` qui change tous les caractères majuscules en leurs minuscules correspondantes.
La dernière conditionnelle vérifie si la sous-chaîne de caractères débutant au second caractère et ayant trois caractères est égale à "`hom`". Lors de l'exécution du code dans cette question, `$x` hérite de sa valeur par défaut `"Animaux"`, la valeur de `substr($x,1,3)` est alors `"nim"`.

3. Quels sont les deux URL à saisir dans le navigateur pour la variable `$_GET["ch"]` reçoivent chacune de ces valeurs :

`"chomâge"` ; `"Jolie Fleur"`

. . . . . .

. . . . . .

Les deux URL à saisir pour passer les deux valeurs à la variable `$_GET["ch"]` sont :

- `http://127.0.0.1/casse.php?ch=chomâge`.
  On remarquera que lors de l'affichage de l'URL, la page Web affichera "*oui*" en dernière ligne car l'instruction `substr($x,1,3)` retournera la valeur `"hom"`.
- `http://127.0.0.1/casse.php?ch=Jolie+Fleur`.
  Le changement de l'espace par le caractère "+" est préférable.

Exercice 15

1. A la racine de votre site, créez un nouveau fichier `chaine.php` et saisissez le code suivant :

```
1  <?php
2  $x="bonjour";
3  echo "$x thomas ; ".$x."<br>";
4  echo '$x thomas ; '.$x.'<p>';
5
6  echo 'entre \"guillemet\"<p>';
7  echo "entre \"guillemet\"<br>";
8
9  echo "th".$x[1]."mas";
10
11 echo "entre "guillemet"<br>";
12 ?>
```

2. a. Affichez la page en saisissant dans votre navigateur l'URL :

   `http://127.0.0.1/chaine.php`

   b. Quelle est la ligne mis en cause dans le message d'erreur s'affichant ?

.....

   c. Effacer la ligne 11 puis rafraichir la page afin d'observer la disparition du message d'erreur.

La ligne 11 provoque une erreur lors de l'exécution de l'instruction `echo "entre "guillement"<br>"` car PHP reconnait en premier la chaîne de caractère `"entre "`, puis continue en essayant d'évaluer l'instruction `guillement"<br>"`.

Il est nécessaire d'une technique particulière, que nous allons voir dans la question suivante, pour insérer un guillemet double dans une chaîne de caractères définies par un guillemet double.

3. a. En observant l'exécution des lignes 3 et 4, déterminer la différence principale entre une chaine de caractères définie par des guillemets doubles et celle définie par des

guillemets simples :

. . . . . .

. . . . . .

. . . . . .

b. Quel est l'effet du caractère \ précédant un guillemet double à la ligne `6` ?

. . . . . .

. . . . . .

c. Quel est l'effet du caractère \ précédant un guillemet double à la ligne `7` ?

. . . . . .

. . . . . .

d. Expliquer pourquoi la dernière ligne du script affiche `"thomas"`.

. . . . . .

. . . . . .

L'exécution des lignes 3 et 4 montre la seule différence entre une chaîne de caractères définie par des guillemets simples ou par des doubles guillemets.
Pour une chaîne de caractères définie par `"..."`, toute variable contenue dans la chaîne de caractère sera remplacée par sa valeur.
Les chaînes définies par les guillemets simples ne subissent aucun traitement de la part de `PHP` avant leur affichage.

Pour faire perdre la fonction de délimiteur de chaîne de caractère au double guillemet, il faut le faire précéder du caractère d'échappement "\" à la ligne 7.
Le caractère \ apparaît à la ligne comme indiquè à la ligne `6`. Le caractère d'échappement est inutile à la ligne `7` puisque la chaîne est définie par le simple guillemet : c'est pour la barre contre-oblique s'affiche à l'exécution de la ligne `7` car le double guillemet n'a aucune fonction au sein d'une chaîne de caractères définies par des simples guillemets.

En `PHP`, une chaîne de caractèrse peut être vue comme un tableau de caractère. Ainsi, la chaine de caractère `"bonjour"` peut être codée par :

```
array(0=>"b",1=>"o",2=>"n",3=>"j",4=>"o",5=>"u",6=>"r")
```

Ainsi, le premier caractère a pour position 0, le second caractère a pour position 1... Ainsi, la variable `$x[1]` contient le caractère "`o`".

Exercice 16

1. A la racine de votre site, créez un nouveau fichier intitulé `tableau.php` et saisissez-y le code suivant :

```php
<?php
$x=array(5,"cuivre",0);
echo count($x)."<br>";
$x[]="or";
echo "<pre>";print_r($x);echo "</pre><p>";

$y=array("thomas"=>"administrateur","sonia"=>"utilisateur",
         "georges"=>"invité");
echo $y["thomas"]."<br>";
$y["yves"]="invité";
echo count($y);
echo "<pre>";print_r($y);echo "</pre><p>";
echo $y[1];
?>
```

2. a. Dans votre navigateur, saisissez l'URL `http://127.0.0.1/tableau.php` pour afficher la page Web.

   b. Une erreur se produit lors de l'exécution de ce code, quel est le numéro de la ligne incréminée ?

. . . . . .

   c. Quel est la nature de l'erreur ? Justifier.

. . . . . .

. . . . . .

   d. Effacer la ligne `13` du script.

La variable `$y` est définie par le code comme un tableau associatif. C'est à dire que ses objets sont associés à des clés définies par des chaînes de caractères.
Ainsi, l'instruction `$y[1]` appelle l'élèment de position `1` qui n'existe pas dans un tableau associatif.

3. a. Combien d'éléments contient la variable `$x` lors de sa déclaration ?

. . . . . .

   b. A sa déclaration, quel est l'élément `$x[2]` ?

. . . . . .

   c. Après la ligne `4`, quel sera l'index repérant l'objet `"x"` ?

. . . . . .

d. A sa déclaration, combien d'élément comporte la variable `$y` ?

. . . . . .

e. Après l'exécution de la ligne `10`, citez les quatres index de la variable `$y` :

. . . . . .

. . . . . .

f. Justifier l'erreur obtenue à la ligne `11`.

La varìable `$x` est un tableau indexé par des valeurs numériques ; elle est initialisée avec 3 valeurs. Voici une manière équivalente de définir la variable `$x` via l'instruction :

```
array(0=>5,1=>"cuivre",2=>0);
```

Les objets du tableau `$x` sont donc repérés par des index allant de 0 à 3.
Lors de l'ajout, à la ligne 4, d'un nouvel objet au tableau `$x`, celui-ci sera associé à l'index 3 *(premier index de libre)*. Il aurait été équivalent d'évaluer l'instruction `$x[3]="or";`

La variable `$y` est un tableau associatif : les index sont des chaînes de caractères dont les valeurs sont, ici, également des chaînes de caractères. Pour ajouter une nouvelle valeur, il suffit d'utiliser la syntaxe avec les crochets en indiquant le nouvel index :

```
$y["yves"]="invité";
```

Pour comprendre la construction des tableaux en `PHP`, nous avons utilisé la commande `print_r(...)` qui affiche les objets et les clés associés d'un tableau.
L'élément HTML pre permet d'afficher correctement la sortie de la commande `print_r(...)` : les espaces et les sauts à ligne seront alors entièrement affichés par le navigateur.

Exercice 17

1. A la racine de votre site, créez un fichier nommé `compte.php` et saisissez le code suivant :

```
1  <?php
2  if(isset($_GET["phrase"])){
3    $x=$_GET["phrase"];
4    $x=explode(" ",$x);
5    echo "Cette phrase contient ".count($x)." mots<p>";
6  }
7  ?>
8
9  <form action="" method=get>
10 Saisissez votre phrase : <input type=text name="phrase">
```

```
11 <br>
12 <input type=submit value="Compter le nombre de mots">
13 </form>
```

2. a. Dans votre navigateur, saisissez l'URL `http://127.0.0.1/compte.php` afin d'afficher cette page Web.

b. Pourquoi la phrase `"Cette phrase contient..."` ne s'affiche pas dans le navigateur ?

.....

.....

Pour que cette phrase puisse s'afficher, il faut que le test dans la structure conditionnelle retourne la valeur `true`. Or, cette condition ne sera rempli que si `PHP` génère la variable `$_GET["phrase"]`.
Celle-ci n'existera que si l'URL définie la valeur `ch`.

3. a. Dans le formule présent dans la page Web, saisissez deux mots séparés par un espace, puis soumettez le formulaire.

b. Comment le script `PHP` fait-il pour compter le nombre de mots ?

La soumission du formulaire fait apparaitre dans l'URL la valeur `ch`. Automatiquement, `PHP` va créer la variable `$_GET["ch"]`. Sa valeur sera affectée à la variable `$x`, puis la commande `explode(" ",$x)` va couper la chaîne contenu dans `$x` à chacun de ses espaces et va créer un tableau contenant ces différents bouts de chaînes.

4. a. A nouveau, saisissez deux mots séparés par une phrase mais en rajoutant un espace à la fin du second mot, puis soumettez le formulaire.

b. Pourquoi le nombre de mots affichés par le script n'est pas correct ?

.....

.....

c. Remplacez la ligne 3 par l'instruction `$x=trim($_GET["phrase"]);`

d. Rafraichissez la page.

Le script affichera que la chaine de caractères contient trois mots, car la commande `explode(...)` ayant trouvé deux espaces, la chaîne sera alors découpée en trois parties : même si la troisième partie est une chaîne vide, elle est considérée comme un élément du novueau tableau `$x`.
La commande `trim(...)` va permettre d'éliminer les espaces et les sauts de lignes

en début ou à la fin de chaîne. Cette commande permet de "*néttoyer*" une chaîne de caractères avant son utilisation.

5. a. Testez le script avec une phrase comprenant entre les mots plusieurs espaces. Que pensez-vous du résultat du script sur ce genre de phrases ?

. . . . . .

. . . . . .

b. Dire quand la condition suivante a pour valeur `true` et `false` :

```
$x==str_replace("  "," ",$x)
```

c. Ecrire un boucle `while` permettant d'effacer les espaces multiples qu'une phrase peut contenir entre deux mots.

La présence de double-espace à l'intérieur d'une chaîne de caractères peut provoquer d'autres erreurs lors de l'interprétation de notre code. Il faut donc continuer à supprimer les doubles espaces dans notre code.

La condition $x == str_replace("","",x)$ teste la présence ou non de double-espace dans la variable `$x`.

Voici la boucle à écrire pour élimer tout double-espace dans une chaîne de caractère :

```
while($x==str_replace("  "," ",$x)){
  $x=str_replace("  "," ",$x);
}
```

Exercice 18

1. A la racine de votre site, créez un nouveau fichier `bouton.php` et saissisez-y le code suivant :

```
<script language="javascript">
  function dessus(x){x.style.backgroundColor="55FF99";}
  function hors(x){x.style.backgroundColor="red";}
</script>

<?php
  $liste=array("Accueil","Sport","Actualités","Contact");
  echo "<table  style='background-color:yellow;width:100%;↩
     table-layout:fixed;text-align:center'><tr>".
    "<td width=20> ";
  for($i=0;$i<count($liste);$i++){
    echo "<td onmouseover=dessus(this) onmouseout=hors(this) ↩
```

```
            ".
        "style='background-color:red'>".
        $liste[$i]."<td width=20> ";
    }
?>
```

2. a. Dans votre navigateur, saisissez l'URL `http://127.0.0.1/bouton.php` afin d'afficher cette page Web.

b. Combien de boutons apparaissent dans la page Web ?

. . . . . .

c. Modifiez ce script pour qu'il affiche un nouveau bouton de navigation intitulé "`CAN`".

Le script récupère les boutons à créer à l'aide de la variable `$liste`.
La boucle `for(...){...}` va répéter la même action pour créer à chaque valeur du tableau `$liste` le bouton associé.
Une simple modification de la variable `$liste` permet de modifier le nombre de boutons créés.

3. a. A la racine de votre site, créez un nouveau fichier intitulé `index.php` et saissisez-y le code suivant :

```
<?php
include("bouton.php");
?>
```

b. Dans votre navigateur, saisissez l'URL `http://127.0.0.1/index.php`.

c. Quelle est l'utilité de la commande `include(...)` ?

En plaçant dans un fichier à part, les élèments communs à toutes les pages d'un même site, la commande `include(...)` permet alors de centraliser ces éléments.
Cela présente un gain énorme de temps en terme de création et de maintenance d'un site.

C. Quelques travaux pratiques :

Exercice 19

Voici un formulaire demandant au client de s'identifier avec un nom d'utilisateur et un mot de passe

```
<form method=get>
```

```
<table>
<tr><td align=right>Loggin :<td><input type=text name=loggin>
<tr><td align=right>Mot de passe :
    <td><input type=text name=motPasse>
<tr><td colspan=2 align=center>
    <input type=submit value="Connexion">
</form>
```

Utilisez un tableau associatif afin d'identifier les personnes possédant l'identifiant et le mot de passe adéquat ; les utilisateurs s'identifiant correctement seront redirigés vers une page d'informations, les autres se verront renvoyés vers une page signalant l'erreur d'identification.

Exercice 20

Créez un script qui à travers un formulaire demande, au client, des nombres séparés par des virgules. A l'aide de la commande `sort(...)`, triez les nombres saisis et affichez-les à l'écran.