

Université Libanaise

Faculté de Technologie  
Génie des Réseaux Informatiques  
et Télécommunications



الجامعة اللبنانية

كلية التكنولوجيا

قسم هندسة شبكات المعلوماتية والاتصالات

# Architecture Client Serveur

## Langage PHP - Les bases



*Préparé par:*

*Dr. Youssef ROUMIEH*

*E-mail : [youssef.roumieh@gmail.com](mailto:youssef.roumieh@gmail.com)*

*Les problèmes rencontrés et les oublis constatés dans ce support sont à signaler à [youssef.roumieh@gmail.com](mailto:youssef.roumieh@gmail.com)*

# Références

1. <http://php.net/manual/fr/>

2. **Bases de données et Internet avec PHP et MySQL**

by Magali Contensin

Copyright © Dunod, Paris, 2004

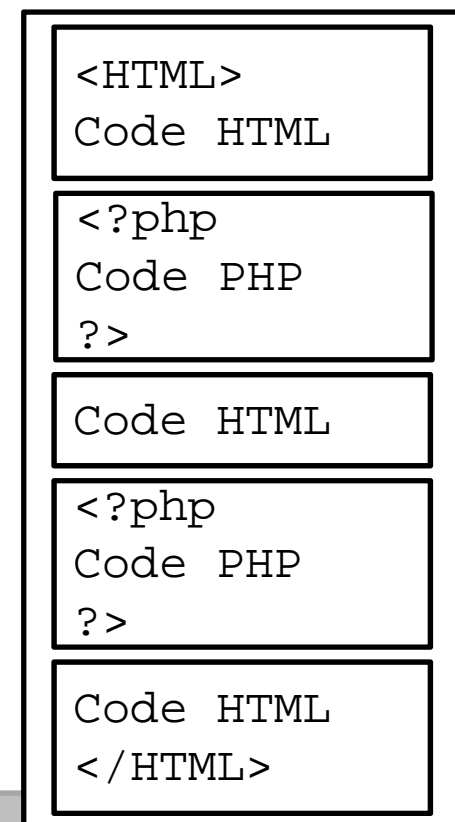
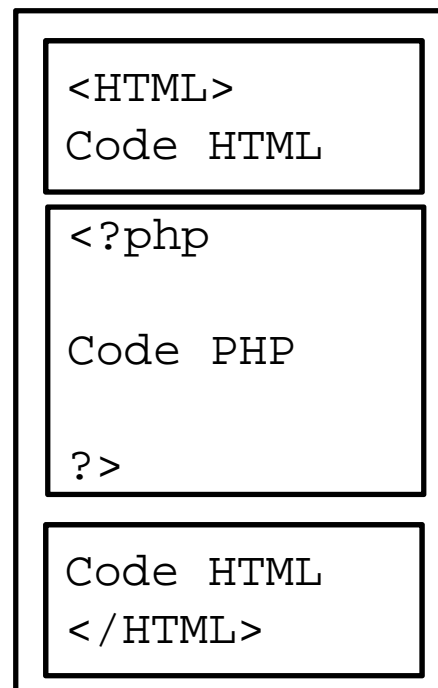
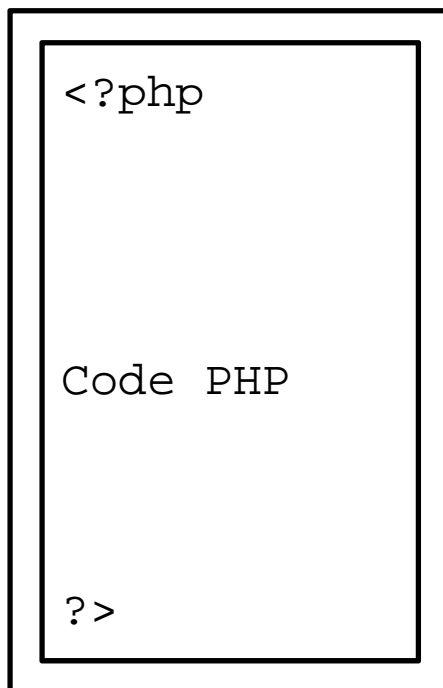
ISBN: 2 10 048340 4

# Caractéristiques

- Le langage PHP (Personnal Home Page)
  - est un langage de scripts exécutés par le serveur qui est dédié à la génération de pages Web dynamiques
  - permet l'extraction et la manipulation de données pour un grand nombre de bases de données
  - s'inspire du langage C du point de vue de la syntaxe, ainsi que de Java pour les objets.
  - est gratuit et Open Source
  - est intégré dans le serveur Web Apache
  - S'intègre facilement au HTML
  - Le client ne reçoit que le résultat du script (code HTML)

# Structure d'un script PHP

- Le code PHP est marqué par la balise spéciale `<?php code ?>`
  - l'interpréteur PHP : il faut traiter le code compris entre la marque de début `<?php` et la marque de fin `?>`.
  - Le fichier peut contenir uniquement du code PHP, mais également être mélangé à du code HTML.



# L'écriture des commentaires

- En HTML, vous pouvez ajouter des commentaires à l'aide de balises spéciales: **<!-- Comment goes here. -->**
  - Les commentaires HTML sont visibles dans la source, mais ne figurent pas dans la page rendue.
- Les commentaires PHP sont différentes en ce qu'ils ne sont pas envoyés au navigateur Web, ce qui signifie qu'ils ne seront pas visibles par l'utilisateur final, même quand on regarde le code source HTML.
- PHP supporte trois types de commentaires.
  - La première utilise le symbole dièse (#) (commentaires pour les lignes simples seulement): **# This is a comment.**
  - La seconde utilise deux barres obliques (Commentaire pour des lignes simples seulement): **// This is also a comment.**
  - Un troisième style permet d'exécuter des commentaires sur plusieurs lignes:  
**/\* This is a longer comment  
that spans two lines. \*/**

# Envoi de données vers le navigateur

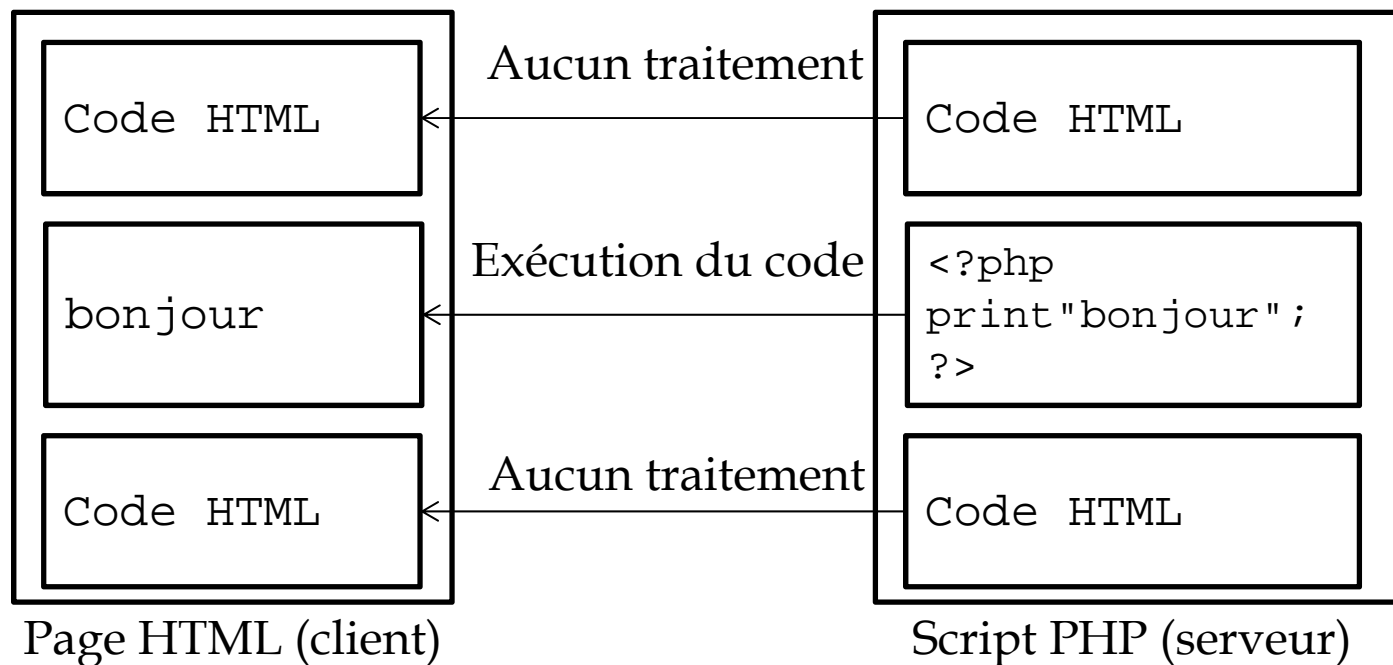
- Pour créer des sites Web dynamiques avec PHP, vous devez savoir comment envoyer des données au navigateur Web.
  - PHP dispose d'un certain nombre de fonctions intégrées à cet effet, la plus courante étant `echo ()` et `print ()`.

```
echo 'Hello, world!';
```

```
echo "What's new?";
```

```
print "Hello, world!";
```

```
print "What's new?";
```



# Besoin d'un Escape

- Impression des guillemets simples et doubles. Un des événements suivants seront provoqué des erreurs:
  - `echo "She said, "How are you?"";`
  - `echo 'I'm just ducky.';`
- Il ya deux solutions à ce problème.
  - Utiliser des guillemets simples lors de l'impression d'un guillemet double et vice versa:

```
echo 'She said, "How are you?";
```

```
echo "I'm just ducky.";
```

- échapper le caractère problématique en le précédant d'une barre oblique inverse (backslash):

```
echo "She said, \"How are you?\"";
```

```
print 'I\'m just ducky.';
```

- **Remarque:** Echo () et print () peuvent toutes deux être utilisées pour imprimer du texte sur plusieurs lignes:

```
echo 'This sentence is
```

```
printed over two lines.';
```

# Variables

- Stocker des données qui peuvent être modifiées au cours de l'exécution du script
- Toute variable commence par le caractère « \$ » suivi d'un identifiant
  - L'identifiant peut contenir une combinaison de chaînes, des nombres, et le trait de soulignement, par exemple, `$my_report1`.
  - Le premier caractère après le symbole dollar doit être une lettre ou un trait de soulignement (il ne peut pas être un nombre).
- PHP est sensible à la casse sauf pour les noms de fonctions.
  - La variable `$heure_courante`  $\neq$  `$Heure_courante`
- En PHP, le typage est implicite et momentané
  - Lorsqu'une valeur est affectée à une variable, le *typage* est réalisé automatiquement.



# Variables et types

- Type Scalaire
  - Le type *integer*: nombres entiers positifs et négatifs en notation décimale, octale et hexadécimale.
  - Les types *double* et *float*: nombres réels tels que 3.14, .4 et 3.5<sup>e</sup>12
  - Le type *boolean*:
    - la constante FALSE ainsi que le chiffre 0 représentent la valeur faux
    - la constante TRUE et tout entier non nul (positif ou négatif) représentent la valeur vrai
  - Le type *string*:
    - Permet de représenter des caractères uniques, des mots et des phrases.
    - Ces chaînes de caractères sont écrites entre guillemets ou encore apostrophes.

# Variables et types

- Type composé:
  - Le type *array*: Permet de manipuler des tableaux, c'est-à-dire des collections de données.
  - Le type **object**: La programmation orientée objet est possible en PHP.
- Type spécial:
  - Le type *resource*: Représente une ressource externe, certaines fonctions de manipulation de répertoires et de fichiers ainsi que des fonctions MySQL utilisent et retournent des valeurs de ce type.
  - Le type *nul*: comporte une unique valeur NULL qui représente l'absence de valeur

# Affichage de la valeur d'une variable

- Afficher la valeur d'une variable:
  - Les variables peuvent être imprimées sans les guillemets: `print $ some_var;`
  - les variables peuvent être imprimés dans les guillemets: `print "Bonjour, $ name";`
  - Vous ne pouvez pas imprimer les variables entre apostrophes: `print 'Bonjour, $name';` // Ne fonctionne pas!
- Pour qu'une **valeur soit affichée**, il faut au préalable avoir affecté une **valeur à la variable**, sinon l'affichage réalisé sera équivalent d'une **chaîne vide**.
  - Lorsque vous afficherez la valeur de la variable `$Heure_courante` au lieu de `$heure_courante` **vous obtiendrez un message d'erreur vous informant que cette variable n'a aucune valeur affectée, vous obtiendrez une chaîne vide.**

# Opérateurs

- Opérateurs arithmétiques:
  - Addition (+), Soustraction (-), Multiplication (\*), Division (/), Modulo (%), Signe opposé(-)
- Affectation : =
- Opérateurs bits à bits et de décalages:
  - et (&), ou (|), ou exclusif (^), complément à un (~), décalage à gauche (<<), décalage à droite (>>)
- Opérateurs relationnels:
  - Egalité (==), égalité et **mêmes types (===)**, Différence (!=) ou (<>), **Différence ou différence de types (!==)**, strictement inférieur (<), inférieur ou égal (<=), strictement supérieur (>), supérieur ou égal (>=)
- Opérateurs logiques:
  - And (&&), Or (| |), Xor, Not (!)
- Opérateur d'incrément (++) , opérateur de décrémentation(--).
- Opérateurs d'affectation élargie:
  - +=, -=, \*=, /=, %=, . =
- L'opérateur de concaténation « . » s'applique uniquement aux chaînes de caractères.

2 === 2	TRUE	2 !== 2	FALSE
2 === "2"	FALSE	2 !== "2"	TRUE
2 === 8	FALSE	2 !== 8	TRUE

# Instructions conditionnelles

- L'instruction if

```
if (expr)
    { instruction block 1 }
```

- L'instruction if ... else

```
if (expr)
    { instruction block 1 }
else
    { instruction bloc 2 }
```

- L'instruction switch

```
switch (expr) {
    case val1 : instructions
    case val2 : instructions
    ...
    case valN : instructions
    default : instructions
}
```

- Opérateur conditionnel ternaire « ?: » :

```
exp1 ? exp2 : exp3
```

# Instructions itératives

- L'instruction while

```
while (expr){  
    instructions block  
}
```

- L'instruction do .. While

```
do{  
    instruction block  
}while (expr);
```

- L'instruction for

```
for(exp1; exp2; exp3){  
    instruction block  
}
```

# Instructions d'interruption

- L'instruction break
  - Provoque la sortie des instructions itératives (for, foreach, while, do ... while), et de l'instruction conditionnelle switch.
- L'instruction continue **Instructions itératives**
  - Utilisée dans instructions itératives for, foreach, while et do ... while
  - Provoque le passage à l'itération suivante si le test de continuation est encore vrai
  - Les instructions de la boucle situées après l'instruction continue sont ignorées.

## Formulaire

Nom d'utilisateur:

Mot de passe:

zone multi-lignes

Message:

Sélectionner un jour:

Dimanche  
Lundi  
Mardi  
Mercredi  
Jeudi  
Vendredi  
Samedi

Sélectionner un jour:

Tarif: ☒ Tarif de jour ☐ Tarif de nuit ☐ Tarif de week-end

☒ Glace vanille ☐ Chantilly ☒ Chocolat chaud ☐ Biscuit

Sélectionner un fichier:  No file chosen

Quantite:

Date:

Heure:

Date et Heure:

Email:

Intervalle:

Couleur:

# Formulaire

- Le code de ce formulaire est posté en Google Classroom.
- **Note :** Disabled et readonly permet de désactiver un élément du formulaire.



# Formulaire

- **La balise form :**

- L'attribut "method": (post/get) précise le mode d'envoi des données
  - method="get" : informations passées dans la barre d'adresse
  - method="post" : envoie les données dans le corps de la requête sans passer par la barre d'adresse, c'est la méthode la plus utilisée
- L'attribut "action" spécifie l'adresse d'expédition des données (l'adresse du fichier ou programme qui va traiter les données).
- L'attribut "enctype" (optionnel) spécifie l'encodage utilisé pour le contenu du formulaire.
  - Ainsi enctype="text/plain" encode le contenu du formulaire en format texte lisible par le destinataire.

- **Variables Prédéfinies:**

- \$\_GET: un tableau associatif de variables transmis au script actuel via les paramètres d'URL (<form method='get'>).
- \$\_POST: un tableau associatif de variables transmis au script actuel via la méthode HTTP POST (<form method='post'>).
- \$\_REQUEST: un tableau associatif contenant par défaut les contenus de \$\_GET, \$\_POST et \$\_COOKIE.

# Exemple

Page ex1.html

```
<form method='post' action='ex1.php'>
  <p>Entrer votre email: <input type='text' name='txtEmail'>
    <input type='submit' name='btnEnvoyer' value='Envoyer'></p>
</form>
```

Page ex1.php

```
<?php
  echo "<p>Votre email est: " . $_POST['txtEmail'] . "</p>";
?>
```

- **Remarque:**

- `isset` — Détermine si une variable est considérée définie, ceci signifie qu'elle est déclarée et est différente de `NULL`

`isset ( mixed $var [, mixed $... ] ) : bool`

- Si plusieurs paramètres sont fournis, alors `isset()` retournera `TRUE` seulement si tous les paramètres sont définis.

# Types

- Les fonctions **is\_int(\$nb)**, **is\_integer**, **is\_long**, retournent TRUE si la variable est de type entier
- Les fonctions **is\_float**, **is\_double**, **is\_real** retournent TRUE lorsque l'argument est un réel
- La fonction **is\_string** retourne TRUE lorsque la variable est une chaîne de caractères
- La fonction **is\_numeric** retourne TRUE lorsque la variable contient une valeur numérique (entier, réel ou chaîne de caractères comportant uniquement des chiffres)
- La fonction **is\_bool** retourne TRUE lorsque la variable est booléenne
- La fonction **is\_null** retournera TRUE si la variable ne contient pas de valeur ou a été affectée avec la constante NULL.
- La fonction **is\_array** retourne TRUE lorsque la variable est un tableau
- La fonction **gettype** retourne une chaîne de caractères contenant le type de la variable passée en argument
- conversion:
  - Il est possible de convertir des données en entier, réel, et chaîne de caractères en utilisant les fonctions intval, doubleval ou floatval et strval.
  - L'opérateur de cast (...) permet également d'effectuer des conversions.

```
<?php
    $nb_stock = 25;
    $a = doubleval($nb_stock);
    echo gettype($a), "<BR>";
    echo (int)($nb_stock/2); //cast
?>
```

# Architecture Client Serveur - TD1: Exercice 1

- La fonction `mt_rand(min, max)` retourne un entier aléatoire compris entre min et max ou false si le paramètre max est inférieur à min.
- Donner les codes des pages PHP qui permettent de réaliser :
  1. une page serveur qui affiche la table de multiplication de 10.
  2. une page qui saisit un nombre et qui le soumet à une autre page qui elle affiche sa table de multiplication.
  3. vérifier dans (2) que le nombre saisi est numérique.
  4. en plus de (3), accepter aussi la virgule.
  5. une page qui saisit un nombre et qui le soumet à une autre page qui elle affiche soit sa table de multiplication, soit son factoriel, soit son carré.
  6. même chose que (5), mais sur une même page.
  7. en plus de (6), s'assurer qu'au premier affichage de la page, on n'affiche pas les résultats.
  8. en plus de (7), s'assurer que le textBox garde sa valeur lors de l'affichage des résultats.

# Functions

- Une fonction est un bloc spécialisé de code PHP qui utilise aucun, un ou plusieurs arguments pour réaliser un traitement et *retourne* éventuellement un résultat.
  - Une fois une fonction est définie, nous pouvons l'appeler dans le programme principal autant de fois que nécessaire.
- Déclaration d'une fonction : Il faut utiliser le mot-clé **function**

```
function function_name (parameter_list) {  
    php code  
    return;  
}
```
- Pour appeler une fonction
  - `$res = function_name (parameter_list);`
  - `function_name (parameter_list);`

- Paramètres peut être passés par valeur ou par adresse
  - Si le paramètre sera passé par adresse, il faut préfixer le paramètre formel par le symbole « & » lors de la déclaration de la fonction.

- Exemple:

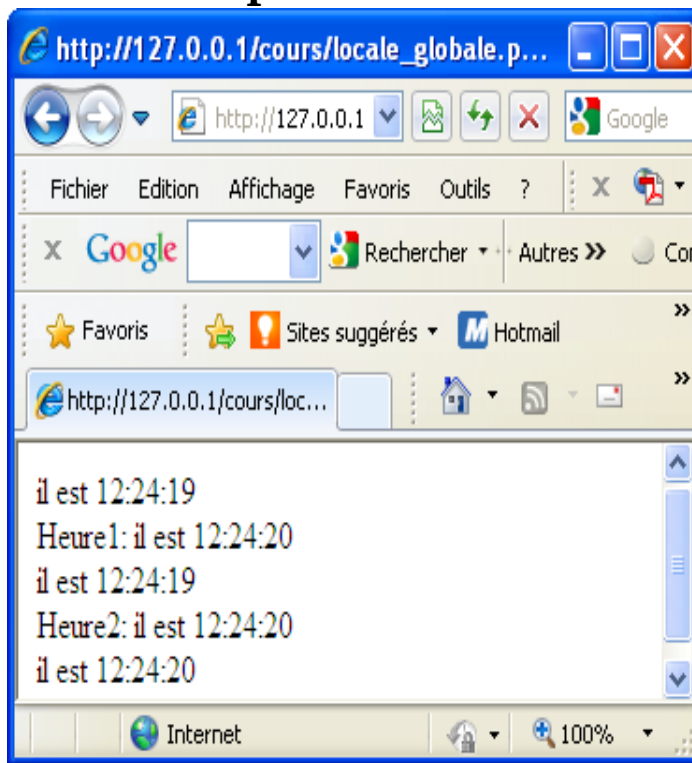
```
function Convertir_en_min1($heure, $min, $res){  
    $res = $heure * 60 + $min;  
}  
function Convertir_en_min2($heure, $min, &$res){  
    $res = $heure * 60 + $min;  
}  
$r1 = 0;  
Convertir_en_min1(21, 30 , $r1);  
echo "<p>r1 = $r1</p>";  
$r2 = 0;  
Convertir_en_min2(21, 30 , $r2);  
echo "<p>r2 = $r2</p>";
```

- Valeurs par défaut : Les valeurs par défaut permettent d'appeler la fonction sans renseigner tous les paramètres
  - Les paramètres comportant une valeur par défaut doivent obligatoirement figurer en fin de liste
- Exemple:

```
function tva($prix_HT, $taux = 19.6){  
    $prix_TTC = ($prix_HT * $taux)/100 + $prix_HT;  
    echo $prix_TTC, « <BR> »;  
}  
...  
//appels  
tva(10); // affiche 11.96 (utilise la TVA pour les  
produits de luxe)  
tva(10, 5.5); // affiche 10.55 (applique la TVA sur les  
produits alimentaires)
```

# Portée des variables globales et locales

- Les variables globales sont visibles dans tout le script, sauf dans les fonctions
- Pour utiliser une variable globale `$x` dans une fonction il faut la précédant par le mot-clé **global**.
- **Exemple:**



```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

...
//fonction Heure1() $h est une var. locale
function Heure1(){
    $h = date("H:i:s");
    echo "Heure1: il est ", $h, "<BR>";
}

//fonction Heure2() $h est une var. globale
function Heure2(){
    global $h;
    $h = date("H:i:s");
    echo "Heure2: il est ", $h, "<BR>";
}

//corps du programme
$h = date("H:i:s");
echo "il est ", $h, "<BR>";
...// du code PHP qui ne produit pas d'affichage
Heure1();
echo "il est ", $h, "<BR>";
...// du code PHP qui ne produit pas d'affichage
for($i=0;$i<1000000;$i++){
    Heure2();
    echo "il est ", $h, "<BR>";
}
```



# Chaînes de caractères (Strings)

- Les chaînes s'écrivent entre guillemets ou apostrophes,
  - `$first_name = 'Tobias';`
  - `$today = "August 2, 2006";`
- Si cette même marque apparaît dans la chaîne, vous devez l'échapper:
  - `$var = "Define \"platitude\", please.";`
- Pour imprimer la valeur d'une chaîne, utilisez `echo ()` ou `print ()`:
  - `echo $first_name;`
- Pour imprimer la valeur de chaîne dans un contexte, utilisez des guillemets doubles:
  - `echo "Hello, $first_name";`
- En PHP, les valeurs enfermés dans des guillemets simples seront traités littéralement, tandis que ceux dans les guillemets sera interprétée.

- La concaténation est effectuée à l'aide de l'opérateur de concaténation, qui correspond à la période (.):

```
$city= 'Seattle' ;
```

```
$state = 'Washington' ;
```

```
$address = $city . $state; //$city.' '.$state;
```

- **Tips**

- Comme le code HTML valide inclut souvent de nombreux attributs entre deux guillemets, il est souvent plus simple d'utiliser des apostrophes lors de l'impression HTML avec PHP:

```
echo '<table width="80%" border="0" cellpadding="2" cellspacing="3" align="center">';
```

# Illustration of the usefulness of \n

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2   "http://www.w3.org/TR/html4/strict.dtd">
3
4 <HTML>
5   <HEAD>
6     <TITLE>Test d'affichage</TITLE>
7     <META http-equiv="Content-Type" content=text/html; charset=ISO-8859-1">
8   </HEAD>
9   <BODY>
10    <DIV>
11      <?php
12        for($i= 0; $i < 5; $i++){
13          echo "test d'affichage<BR>";
14        }
15      ?>
16    </DIV>
17  </BODY>
18 </HTML>
```

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2   "http://www.w3.org/TR/html4/strict.dtd">
3
4 <HTML>
5   <HEAD>
6     <TITLE>Test d'affichage</TITLE>
7     <META http-equiv="Content-Type" content=text/html; charset=ISO-8859-1">
8   </HEAD>
9   <BODY>
10    <DIV>
11      test d'affichage<BR>test d'affichage<BR>test d'affichage<BR>test d'affichage<BR>test d'affichage<BR>
12    </DIV>
13  </BODY>
14 </HTML>
```

# Avec (\n):

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2 "http://www.w3.org/TR/html4/strict.dtd">
3
4 <HTML>
5   <HEAD>
6     <TITLE>Test d'affichage</TITLE>
7     <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8   </HEAD>
9   <BODY>
10    <DIV>
11      <?php
12        for($i= 0; $i < 5; $i++){
13          echo "test d'affichage<BR>\n";
14        }
15      ?>
16    </DIV>
17  </BODY>
18 </HTML>
```

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2 "http://www.w3.org/TR/html4/strict.dtd">
3
4 <HTML>
5   <HEAD>
6     <TITLE>Test d'affichage</TITLE>
7     <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8   </HEAD>
9   <BODY>
10    <DIV>
11      test d'affichage<BR>
12      test d'affichage<BR>
13      test d'affichage<BR>
14      test d'affichage<BR>
15      test d'affichage<BR>
16    </DIV>
17  </BODY>
18 </HTML>
```

# Fonctions prédéfinies

Prototype	Description
<code>string strtolower(string ch)</code>	Retourne ch converti en miniscules
<code>string strtoupper(string ch)</code>	Retourne ch converti en majuscules
<code>string ucfirst(string ch)</code>	Retourne ch avec le 1 <sup>er</sup> caractère en majuscule
<code>string ucwords(string ch)</code>	Retourne ch avec le 1 <sup>er</sup> caractère de chaque mot en majuscule.
<code>string trim(string ch)</code>	Retourne ch sans les espaces de début et fin
<code>int strcmp(string ch1, string ch2)</code>	Retourne 0 si les deux chaînes sont égales, un nombre positif si ch1 est alpha-numériquement plus grand que ch2, et un nombre négatif sinon.
<code>int strcasecmp(string ch1, string ch2)</code>	Même comportement que strcmp mais sans tenir compte de la case

# Fonctions prédéfinies

Prototype	Description
<code>int strpos(string ch, string sc)</code>	Retourne la position de la 1 <sup>re</sup> occurrence de la sous-chaîne sc dans ch, ou FALSE si sc n'est pas dans ch
<code>string substr(string ch, int deb, int n)</code>	Renvoie la sous-chaîne de la taille n qui commence à l'indice deb dans ch (n est optionnel, s'il n'est pas précisé retourne la sous-chaîne de l'indice deb jusqu'à la fin de la chaîne)
<code>int substr_count(string ch, string sc)</code>	Retourne le nombre d'occurrences de sc dans ch
<code>int strlen(string ch)</code>	Retourne le nombre de caractères de ch
<code>String str_repeat(string ch, int n)</code>	Retourne la chaîne ch répétée n fois
<code>string strrev(string ch)</code>	Retourne la chaîne ch inversée

# Fonctions prédéfinies

Prototype	Description
<b><u>mixed</u> count_chars ( string \$string [, int \$mode = 0 ] )</b>	<p>count_chars() compte le nombre d'occurrences de tous les octets présents dans la chaîne string et retourne différentes statistiques.</p> <p>Valeurs de retour : Suivant la valeur de mode, count_chars() retourne les informations suivantes :</p> <ul style="list-style-type: none"><li>▪ 0 : un tableau avec l'octet (ASCII Code) en index, et la fréquence correspondante pour chaque octet.</li><li>▪ 1 : identique à 0, mais seules les fréquences supérieures à zéro sont listées.</li><li>▪ 2 : identique à 0, mais seules les fréquences nulles sont listées.</li><li>▪ 3 : une chaîne contenant tous les octets (caractères) utilisés est retournée.</li><li>▪ 4 : une chaîne contenant tous les octets non utilisés est retournée.</li></ul>
<b>string chr(int ascii)</b>	Retourne le caractère qui correspond au code ascii
<b>int ord(string car)</b>	Retourne le code ascii qui correspond au caractère car

# Fonctions prédéfinies

Prototype	Description
<code>string str_pad(string ch, int nb, string remp, int pos)</code>	Retourne la chaîne ch sur nb caractères, en complétant par des espaces ou par la chaîne de remplissage remp si elle est précisée. Les caractères ajoutés sont par défaut à droite, pos peut prendre les valeurs STR_PAD_LEFT, STR_PAD_RIGHT, STR_PAD_BOTH
<code>array explode(string sep, string ch, int n)</code>	Sépare ch selon le séparateur sep et place chaque sous-chaîne dans une case de tableau, dans la limite de n cases si n est précisé (la dernière case contient alors la fin de la chaîne).



# Tableaux

- Deux types:
  - Tableau avec accès par indice lorsque les clés sont des nombres entiers
  - Tableaux associatifs lorsque les clés sont des chaînes de caractères.
- Tableau avec accès par indice: Initialisation et ajout de cases
  - Appeler **array** qui retourne un tableau contenant une case par argument (array n'est pas une fonction, c'est un élément de langage utilisé pour représenter des tableaux).  
`$tab = array («b», «o», «n»);`
  - Crée la case (ou remplace le contenu de la case)  
`$tab[0]=« b »; $tab[1]=« o »; $tab[2] = « n »;`
  - Il est possible de ne préciser pas le numéro de case (la donnée est ajoutée en fin de tableau)  
`tab[]=« b »; $tab[]=« o »; $tab[] = « n »;`

# Parcours

- Afficher tout le tableau → utiliser une instruction itérative
- Les instructions While, do ... while et for : il faut connaître le nombre d'éléments du tableau pour arrêter le parcours (la fonction count)
- L'instruction foreach(\$tab as \$val) permet d'accéder à chaque case non nulle du tableau \$tab et de stocker sa valeur pour une itération dans \$val, sans que le programmeur précise la taille du tableau.

- Exemple:

Le résultat est le suivant:

b-o-n-

b-o--

```
1  <?php
2      $tab[0] = "b"; $tab[1] = "o"; $tab[3] = "n";
3      foreach ($tab as $i){
4          echo "$i-";
5      }
6      echo "<BR>";
7      $t = count($tab);
8      for($i = 0; $i < $t; $i++){
9          echo $tab[$i], "-";
10     }
11  ?>
```

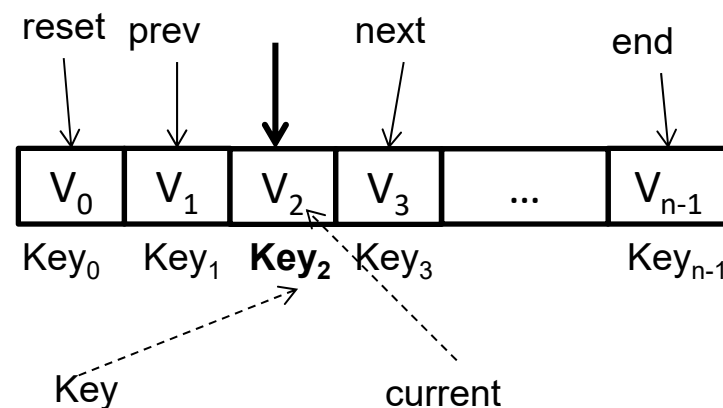
# Tableaux à plusieurs dimensions (Exemple):

```
1 <?php
2 //calculer et stocker
3 for($i = 0; $i <= 10; $i++){
4     for($j = 0; $j <= 10; $j++){
5         $add[$i][$j] = $i + $j;
6     }
7 }
8 //afficher
9 echo "<BR>";
10 foreach ($add as $ligne){
11     foreach ($ligne as $col){
12         printf("%2d", $col);
13     }
14     echo "\n<BR>";
15 }
16 ?>
```

```
1 <BR> 0 1 2 3 4 5 6 7 8 9 10
2 <BR> 1 2 3 4 5 6 7 8 9 10 11
3 <BR> 2 3 4 5 6 7 8 9 10 11 12
4 <BR> 3 4 5 6 7 8 9 10 11 12 13
5 <BR> 4 5 6 7 8 9 10 11 12 13 14
6 <BR> 5 6 7 8 9 10 11 12 13 14 15
7 <BR> 6 7 8 9 10 11 12 13 14 15 16
8 <BR> 7 8 9 10 11 12 13 14 15 16 17
9 <BR> 8 9 10 11 12 13 14 15 16 17 18
10 <BR> 9 10 11 12 13 14 15 16 17 18 19
11 <BR> 10 11 12 13 14 15 16 17 18 19 20
12 <BR>
```

# Tableaux associatifs

- Initialisation
  - En utilisant array:  
`$nomtab = array («cle1»=>valeur1, «cle2»=>valeur2, ...);`
  - En donnant la clé entre crochets: `$nomtab[«cle»] = valeur;`
- Parcours
  - Les instructions for, while : nécessitent l'utilisation de fonctions pour manipuler les pointeurs (reset, next, prev et end)
  - Ces fonctions modifient la position du pointeur courant du tableau passé en paramètre et retournent la valeur de la nouvelle case pointée.



- **L'instruction foreach: beaucoup plus simple d'utilisation**

```

1 <?php
2 //initialisation 1
3 $tarif["pizza napolitaine"] = 6;
4 $tarif["pizza royale"] = 8;
5 $tarif["pizza aux fruits de mer"] = 8.50;
6 $tarif["moules frites"] = 7.30;
7 $tarif["poulet frites"] = 6;
8 //initialisation 2
9 $stock = array("poulets frites"=>50, "pizzas"=>22,
10 "moules frites"=>8);
11 //parcourir le tableau avec for
12 echo "<BR>parcours avec for : ";
13 for(reset($tarif); $cle = key($tarif); next($tarif)){
14     $val = pos($tarif);
15     echo "<BR>$cle = $val euros";
16 }
17 //parcourir le tableau avec while
18 echo "<BR>premier parcours avec while : ";
19 reset($stock);
20 while(list($cle, $valeur) = each($stock)){
21     echo "<BR>nombre de $cle = $valeur portions";
22 }
23 echo "<BR>deuxième parcours avec while : ";
24 while(list($cle, $valeur) = each($stock)){
25     echo "<BR>nombre de $cle = $valeur portions";
26 }
27 //parcourir le tableau avec foreach
28 echo "<BR>parcours avec foreach : ";
29 foreach($stock as $cle => $val){
30     echo "<BR>$cle = $val portions";
31 }
32 ?>

```

parcours avec for :

- pizza napolitaine = 6 euros
- pizza royale = 8 euros
- pizza aux fruits de mer = 8.5 euros
- moules frites = 7.3 euros
- poulet frites = 6 euros

premier parcours avec while :

- nombre de poulets frites = 50 portions
- nombre de pizzas = 22 portions
- nombre de moules frites = 8 portions

deuxième parcours avec while :

parcours avec foreach :

- poulets frites = 50 portions
- pizzas = 22 portions
- moules frites = 8 portions

Il n'y a pas aucun affichage produit par le second while car le pointeur n'a pas été placé en début de tableau.

# Opérateur d'ajout

- L'opérateur d'ajout « + » ajoute au tableau de gauche le tableau de droite sans dupliquer les cases dont les clés sont identiques.

```
1 <?php
2
3     $tarif = array("pizza"=>6,"pizza royale"=>8,
4         "poulet"=>6);
5     $tarif2 = array("couscous"=>7,"pizza"=>6.50,
6         "lasagnes"=>7.50);
7     $res = $tarif + $tarif2;
8     //parcourir le tableau avec foreach
9     foreach($res as $cle => $val){
10         echo "$cle = $val euros<BR>";
11     }
12 ?>
```

pizza = 6 euros  
pizza royale = 8 euros  
poulet = 6 euros  
couscous = 7 euros  
lasagnes = 7.5 euros

# Fonctions prédéfinies

Prototype	Description
<code>void sort (array t)</code>	Trie t par ordre croissant
<code>void rsort (array t)</code>	Trie t par ordre décroissant
<code>void asort (array t)</code>	Trie le tableau associatif t par ordre croissant de valeurs
<code>void arsort (array t)</code>	Trie le tableau associatif t par ordre décroissant de valeurs
<code>void ksort (array t)</code>	Trie le tableau associatif t par ordre croissant de clés
<code>void krsort (array t)</code>	Trie le tableau associatif t par ordre décroissant de clés
<code>array array_keys(array t)</code>	Retourne les clés de t
<code>array array_values(array t)</code>	Retourne les valeurs de t
<code>boolean array_key_exists (\$cle, array t)</code>	Retourne TRUE si la clé existe

# Fonctions prédéfinies

Prototype	Description
<code>boolean in_array( type_element rech, array t)</code>	Retourne TRUE si rech (dont le type n'est pas prédéfini) est dans t
<code>type_cle array_search(type_element rech, array t)</code>	Retourner la clé (un entier si un tableau a accès par indice) de l'élément rech dans t, ou FALSE si rech n'est pas dans t
<code>type_element min(array t)</code>	Retourne la valeur minimale de t. Le type retourné est celui de l'élément minimal
<code>type_element max(array t)</code>	Retourne la valeur maximale de t. Le type retourné est celui de l'élément maximal
<code>int count(array)</code>	Retourne le nombre de cases
<code>array explode(string sep, string ch)</code>	Transforme une chaîne ch en tableau en fonction du séparateur sep



# Fonctions prédéfinies

Prototype	Description
<code>string implode(string sep, array t)</code>	Opération inverse
<code>array range(int min, int max, int step)</code>	Crée un tableau dont les valeurs vont de min à max. si step est précisé (PHP5), il donne l'incrément (1 par défaut)

- Exemple

```
1  <?php
2      $sch = "poulet/pizza royale/pizza napolitaine";
3      $tab = explode('/', $sch);
4      aff_ind($tab, 'explode');
5      $sch = implode('-', $tab);
6      echo "implode: $sch";
7  ?>
```

`explode` : poulet, pizza royale, pizza napolitaine,  
`implode`: poulet-pizza royale-pizza napolitaine

# Autre fonctions

- Les fonctions `require`, `include`, `require_once` et `include_once` permettent d'inclure dans du code PHP le code d'un fichier donné en argument.
- Lorsque le fichier n'existe pas, s'il a été inclus avec la fonction `include` ou `include_once` un message d'avertissement est produit.
- Si c'est la fonction `require` ou `require_once` qui a été utilisée, l'exécution du script est interrompue.
- Dans le cas où le code a déjà été inclus dans le script, `require_once` et `include_once` ne réaliseront pas une nouvelle inclusion.

# Architecture Client Serveur - TD1: Exercice 2 (Devoir)

- Donner le code php qui permet de réaliser le jeu de scrable :

## SCRABLE

Entrer un mot :

Ecrire le script resultat qui calcule le nombre de points correspondant au mot saisi

Rappel des regles:

- Z = 6
- K = 10
- X, Y = 5
- Tout autre lettre vaut 1.

# Architecture Client Serveur - TD2: Exercice 1

- Écrire le code PHP qui permet de lire une variable \$v à partir d'une chaîne de requête (Query String) et qui affiche la forme suivante:

Si \$v=1	Si \$v=2	Si \$v=3	Si \$v=4	...
A	A B B	A B B C C C	A B B C C C D D D D	

# Architecture Client Serveur - TD2: Exercice 2

- Considérons le tableau \$t qui stocke les températures enregistrées dans différentes capitales:

```
$t = array('london' => 3, 'moscow' => 1, 'rome' => 7,  
'brussels' => 6, 'PaRis' => 5, 'bEirut' => 21, 'CAIRO'  
=>27, 'Tokyo' => 15, 'NewYork' =>4)
```

- Écrire un script PHP qui permet de:
  - Afficher toutes les capitales et leurs températures, de la façon suivante:  
La température à **London** est 3°C  
La température à **Moscow** est 1°C  
La température à **Rome** est 7°C  
....
  - Calculer et afficher la température moyenne, les trois températures les plus basses et les trois températures les plus hautes dans différentes capitales, comme suit:  
Température moyenne est: 9.6  
Liste des trois températures les plus basses: Moscow-1, 3-London, NewYork-4  
Liste des trois températures les plus hautes: Cairo-27, Beirut-21, Tokyo-15
- **NB:** la sortie attendue doit être telle que présentée ci-dessus (le format des lettres également).

# Architecture Client Serveur - TD2: Exercice 3

- Etant donné le tableau associatif suivant :

```
$drapeaux = array ( "japon"=>"japon.png",  
"france"=>"france.png", "liban"=>"liban.png",  
"chine"=>"chine.png", "bresil"=>"bresil.png", "usa"=>"usa.png",  
"allemagne"=>"allemagne.png", "espagne"=>"espagne.png" );
```

- Donner le code php qui permet de parcourir le tableau associatif '\$drapeaux' et d'afficher les noms des pays comme suit :

**Drapeaux**

Aficher les drapeaux : ☐ japon ☐ france ☐ liban ☐ chine ☐ bresil ☐ usa ☐ allemagne ☐ espagne

- L'utilisateur choisit un (ou plusieurs) pays et clique ensuite sur le bouton 'Afficher'. Il obtient alors le résultat suivant :

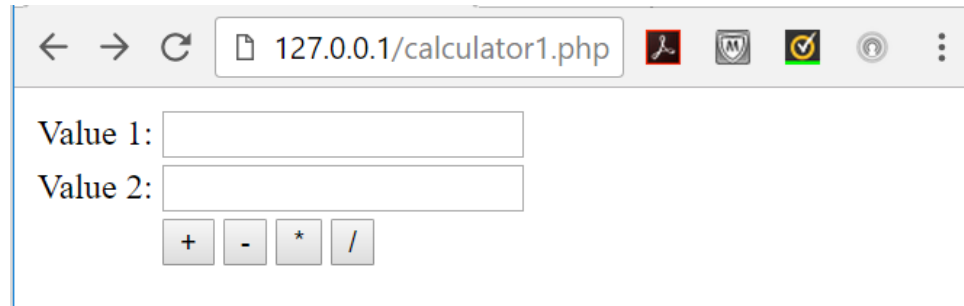
**Drapeaux**

Aficher les drapeaux : ☐ japon ☒ france ☒ liban ☐ chine ☐ bresil ☐ usa ☐ allemagne ☐ espagne



# Exercise

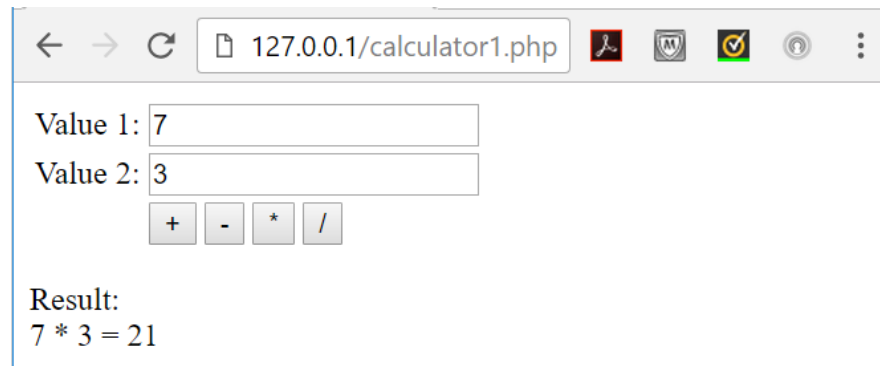
- Write a php script to make a calculator as follows:



Value 1:

Value 2:

- The click on an operation allows to display the result of the expression as follows:



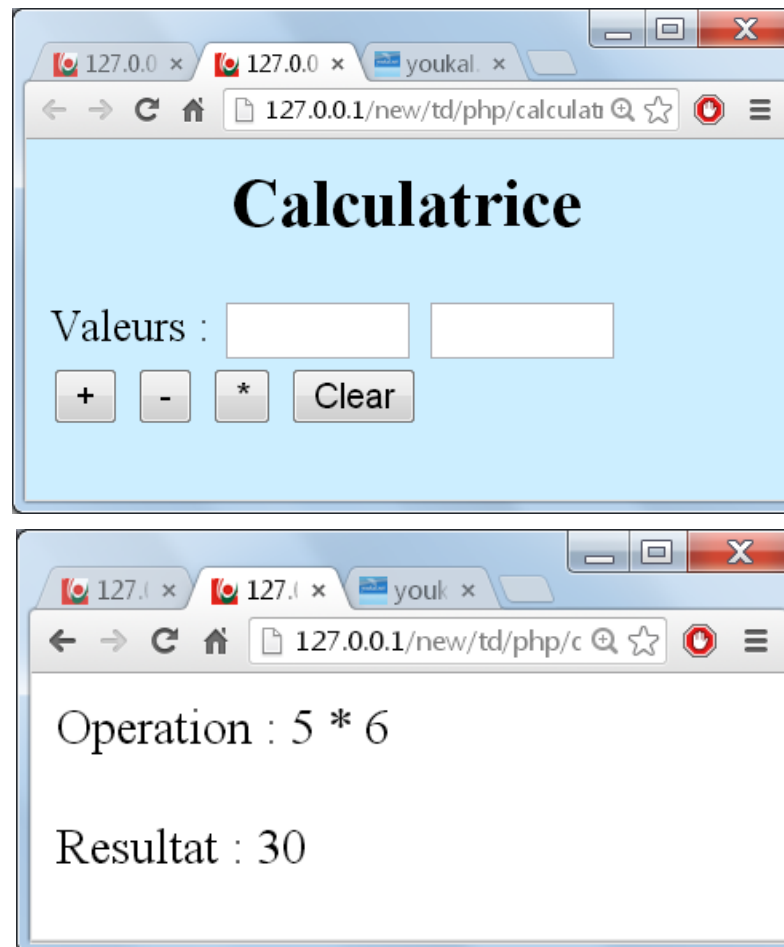
Value 1:

Value 2:

Result:  
7 \* 3 = 21

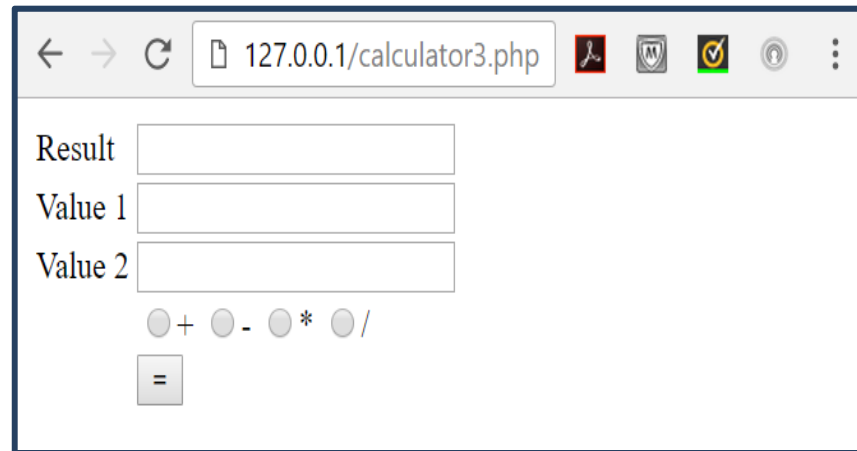
- Note: write the exercise using two methods:
  - Each operation has a different name.
  - All operations has the same name.

- Same exercise, the result is displayed on another page.



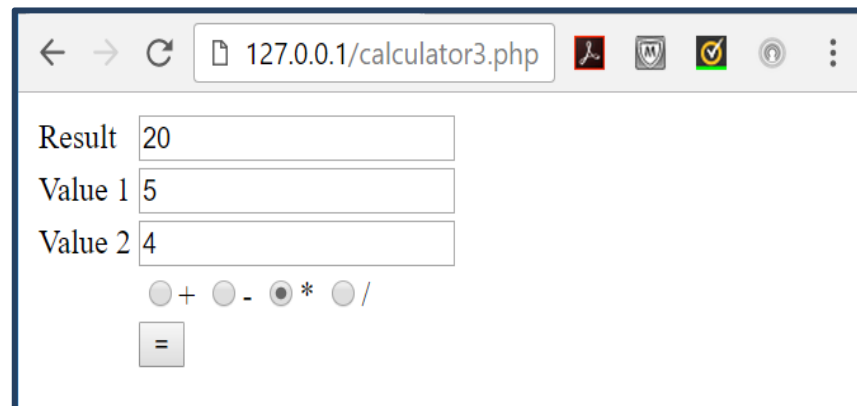


- Same exercise, the operations are represented as radios button instead of buttons.



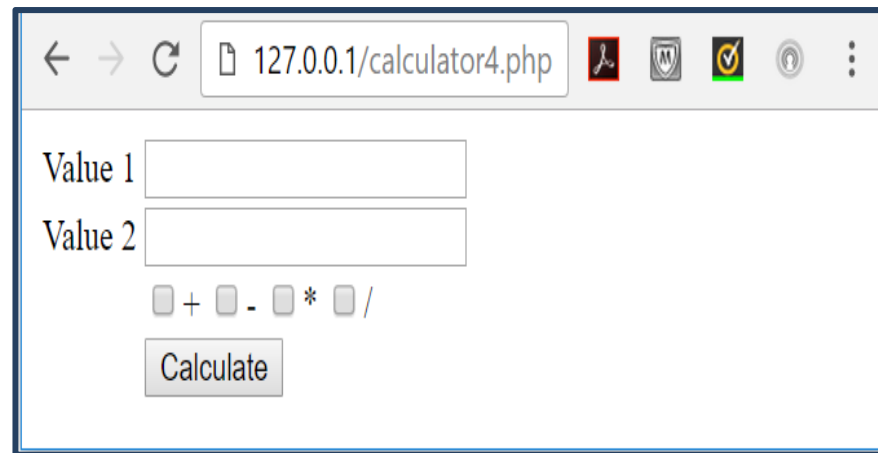
A screenshot of a web browser window displaying a calculator application. The address bar shows the URL `127.0.0.1/calculator3.php`. The form contains three input fields labeled "Result", "Value 1", and "Value 2", all of which are currently empty. Below the input fields, there are four radio buttons for operations: "+", "-", "\*", and "/". The "\*" radio button is selected. Below the radio buttons is a button labeled "=".

- The form keeps its values when the user submit it.



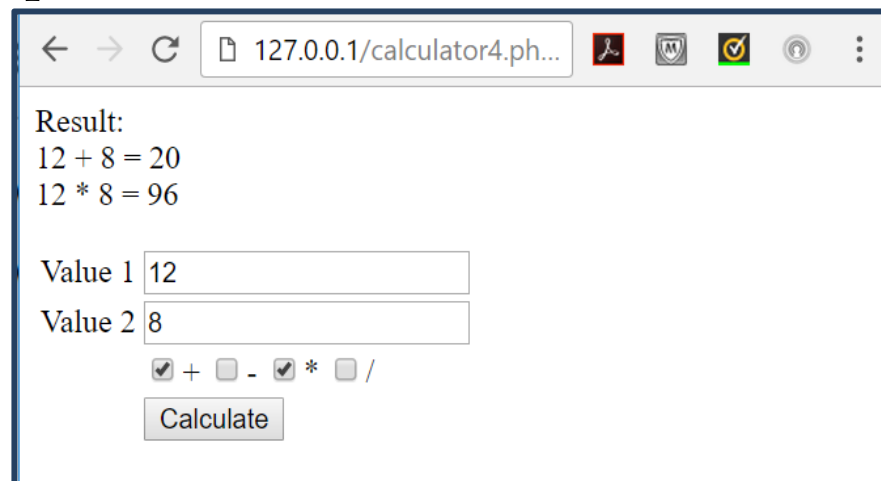
A screenshot of the same web browser window showing the calculator application after a submission. The "Value 1" field now contains the number "5", the "Value 2" field contains the number "4", and the "Result" field contains the number "20". The "\*" radio button remains selected, and the "=" button is still present.

- Same exercise, the operations are represented as checkboxes.



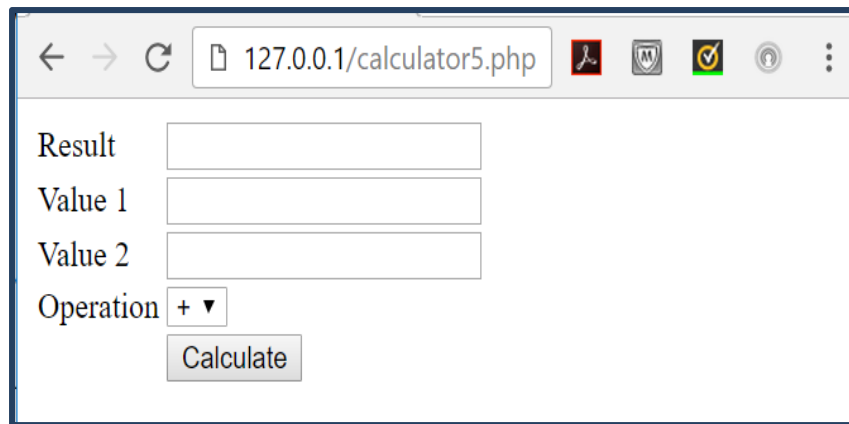
A screenshot of a web browser window displaying a calculator form. The address bar shows the URL `127.0.0.1/calculator4.php`. The form contains two input fields labeled "Value 1" and "Value 2", both of which are empty. Below the input fields, there are four checkboxes for mathematical operations: addition (+), subtraction (-), multiplication (\*), and division (/). All checkboxes are currently unchecked. A "Calculate" button is positioned below the operation checkboxes.

- The form keeps its values when the user submit it.



A screenshot of the same web browser window after the form has been submitted. The address bar still shows `127.0.0.1/calculator4.php`. The form now displays the results of the calculation. At the top, it says "Result:" followed by two lines of text: `12 + 8 = 20` and `12 * 8 = 96`. Below this, the input fields "Value 1" and "Value 2" now contain the numbers "12" and "8" respectively. The checkboxes for addition (+) and multiplication (\*) are now checked, while subtraction (-) and division (/) remain unchecked. The "Calculate" button is still present at the bottom.

- Same exercise, the operations are represented into a list.



127.0.0.1/calculator5.php

Result

Value 1

Value 2

Operation + ▼

- The form keeps its values when the user submit it.



127.0.0.1:8080/dws...

Result

Value 1

Value 2

Operation / ▼

- Note: check whether the values are numeric.