Les sous programmes

Table des matières

Objectifs

À la fin de cette leçon, vous serez capable de :

- Définir et utiliser une fonction ou procédure en PHP
- Définir et utiliser des variables globales et locales
- Identifier quelques fonctions prédéfinies du langage

Introduction

Lorsqu'un programme est long, il est irréaliste d'écrire son code d'un seul tenant. En fait, on décompose le programme en plusieurs parties plus petites qu'on assemble pour former le programme finale.

- Lorsque la complexité d'un problème s'accroît, il devient donc nécessaire d'utiliser les sous-programmes pour alléger la tâche. Ces sous-programmes sont les procédures et les fonctions.
- Les sous-programmes (procédures et fonctions) permettent donc de découper un gros programme en morceaux plus petits et donc plus simples à coder et à comprendre, et de plus permet d'éviter de répéter plusieurs morceaux de code identiques.
- Un sous-programme possède un nom, des variables, des instructions, un début et une fin.
- Un sous-programme ne peut pas s'exécuter tout seul, il a besoin qu'on appel pour qu'il puisse s'exécuter.

De façon générale, une fonction ou procédure permet :

- d'éviter de copier un segment de code plusieurs fois, c-à-d, elle permet de n'écrire qu'une seule fois une série d'instructions et de pouvoir réutiliser le code écrit autant de fois que l'on souhaite dans notre script en appelant simplement la fonction.
- d'offrir une meilleure organisation du code source
- de décomposer un grand problème en petit module ou sous problème, chacun effectuant une tâche bien précise. Les modules peuvent être écrits par plusieurs personnes de façon indépendante.

En PHP, la manière de définir une fonction et une procédure est identique, à la différence qu'une fonction retourne une valeur, tandis qu'une procédure est une fonction ne retournant rien. A la suite de ce cours, on parlera tout simplement de fonction en lieu et place de sous-programme.

Liens Utiles :

https://www.pierre-giraud.com/php-mysql/cours-complet/php-presentation-fonctions.php

https://www.vulgarisation-informatique.com/fonctions.php

https://www.commentcamarche.com/contents/792-php-les-fonctions

https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/912352-les-fonctions

Définition et utilisation des fonctions

I

Une fonction est un bloc de code PHP destiné généralement à être réutilisé plusieurs fois. Plutôt que d'écrire X fois le morceau de code, on le met dans une fonction, et c'est cette fonction que l'on appellera dès qu'on l'aura décidé. On pourra utiliser les fonctions par exemple pour effectuer des calculs répétitifs ou pour afficher des informations.

1. Définition d'une fonction

Syntaxe : Déclaration d'une fonction

```
function Nom_De_La_Fonction(argument1, argument2, ...) {
  liste d'instructions
}
```

Remarque

- Le nom de la fonction suit les mêmes règles que les noms de variables (les règles d'identification) :
  - le nom doit commencer par une lettre
  - un nom de fonction peut comporter des lettres et des chiffres, mais ne doit pas comprendre de caractères spéciaux (espace, #, $, *,", ', /, etc.) à l'exception des caractères _ et &.
  - le doit doit pas comprendre de caractères en indice ou en exposant
  - etc.
- le nom de la fonction, comme celui des variables est sensible à la casse (différenciation entre les minuscules et majuscules)
- Les *arguments* (ou paramètres) sont facultatifs, mais s'il n'y a pas d'arguments, les parenthèses doivent rester présentes
- les arguments représentes des variables ou des données que la fonction prend en entrée afin de les utiliser pour effectuer le traitement
- il est possible de définir une valeur par défaut à un argument pendant la déclaration. La syntaxe est : `argument=valeur` .*Ex* : `function somme($x, $y=1){...}`
- Il faut respecter l'ordre des paramètres pendant l'appel de la fonction
- si la fonction doit retourner une valeur, on utilise le mot clé *return* à la fin de la fonction de ma manière suivante : `return valeur` ; *Ex* : `return $x ;`

Exemple : fonction pgcd

Si le code ci-dessous ne s'affiche pas bien, veuillez cliquez sur le lien (bleu) ci-dessous

```php
<?php
function pgcd($a,$b){
  for ($i=1;$i<=$a;$i++){
    if( $a%$i == 0  && $b%$i ==0 ){
        $pg=$i;
    }
  }
  return $pg;
}
?>
```

2. Utilisation des fonctions

- Pour qu'une fonction puisse s'exécuter, on doit d'abord l'appeler par le biais d'une instruction d'appel
- Pour appeler une fonction, on écrit tout simplement le nom de la fonction suivit des parenthèses avec les valeurs des arguments. *Ex* : `$d= pgcd(12,16) ;`
- Dans le cas où une fonction retourne une valeur, on peut assigner cette valeur à une variable pendant l'appel de la fonction comme dans l'exemple ci-dessus
- Si la fonction n'a pas d'argument alors on écrit simplement les parenthèses; dans le cas où la fonction dispose d'argument il faut obligatoirement préciser la valeur des arguments pendant l'appel
- on peut appeler autant de fois qu'on veut une fonction pour qu'elle puisse s'exécuter
- on peut appeler une fonction avant ou après sa définition dans le fichier
- une fonction peut être appelée directement dans le fichier ou bien dans une autre fonction
- Lorsque les instructions d'un fichier sont exécutées, une fonction écrite dans le fichier ne peut s'exécuter que si elle est appelée dans le fichier.

Exemple : Calcul de moyenne

Écrire un programme qui calcule la moyennes des notes stockées dans un tableau et l'affiche, on utilisera une fonction pour le calcul

Si le code ci-dessous ne s'affiche pas bien, veuillez cliquez sur le lien (bleu) ci-dessous

```php
<?php
// Definition de la fonction

function calcul_moyenne($tab){
  $som= array_sum($tab); //on somme les éléments du tableau grâce à la fonction php prédefinie array_sum()
  $nb= sizeof($tab);
  $moy= $som/$nb;
  return $moy;
}

//Utilisation de la fonction

$notes=array(12,13.5,17,9,8,19,16,17);
$moyenne= calcul_moyenne($tab); //Appel de la fonction
echo "La moyenne des notes est : $moyenne";
?>
```

Exercice : Activité d'auto-évaluation No 1

II

Exercice

Les fonctions :

- ☐ permettent de gérer des taches répétitives
- ☐ permettent de coder automatiquement
- ☐ offre un meilleur affiche des informations à l'écran
- ☐ offrent une meilleure organisation du code source
- ☐ permettent de découper une tache en de petites tâches afin de faciliter le traitement
- ☐ sont des mots prédéfinis d'un langage informatique

Exercice

Le nom de la fonction :

- ☐ peut comprendre des indices ou exposant
- ☐ doit commencer par une lettre
- ☐ peut comprendre le caractère $ et undurscore (_)
- ☐ peut comprendre des caractères accentués et des espaces
- ☐ peut comprendre des chiffre
- ☐ doit commencer par un $

Exercice

Une fonction retourne une valeur grâce au mot clé

Exercice

On aimerais écrire une fonction qui prend en entrée un entier n puis calcule somme des carrées des n premier nombres pairs non nuls. Veuillez compléter le code

```
<?php

 somme_caree $n
```

```
$som=          ;
       ($i=1;$i<=2*n;$i++){
        ($i      2      0){
   $som      = $i*      ;
  }
}
       $som;

?>
```

Les variables globales et locales

1. les variables locales

Définition

On appelle variable locale, une variable déclarée à l'intérieur d'une fonction.

- Une variable déclarée au sein d'une fonction ne peut être utilisée qu'à l'intérieur de cette fonction.
- Deux variables locales issues de deux fonctions différentes et ayant le même nom ne sont pas identiques. chacune est indépendante de l'autre

Exemple

soit deux fonctions *somme()* et *calcul_moyenne()* définies comme suit :

```php
<?php

function somme($x,$y){
    $som=$x+$y;
    return $som;
}

function calcul_moyenne($tab){
  $som= array_sum($tab);
  $nb= sizeof($tab);
  $moy= $som/$nb;
  return $moy;
}

?>
```

- Les variables locales à la fonction somme() sont : *$som*
- Les variables locales à la fonction calcul_moyenne() sont : `$som, $nb, $moy`
- la variable `$som` de sommes() est différente de la variable `$som` de calcul_moyenne()
- On peut appeler et utiliser la fonction somme() dans la fonction calcul_moyenne() sans que les variables de ces deux fonctions ne se mélangent. On dit que les deux fonctions ont des scopes différents

2. Les variables globales

Définition

Toute variable déclarée directement dans un fichier et ne se trouvant pas à l'intérieur d'une fonction est appelée variable globale. Autrement dit, une variable globale est une variable déclarée dans un script PHP en dehors de toute fonction.

Remarque

- Une variable globale peut être utilisée partout dans le fichier mais n'est pas vu au sein d'une fonction, c-à-d, on ne peut pas utiliser ou accéder à une variable globale étant à l'intérieur d'une fonction.
- Une variable globale et une variable locale ayant le même nom ne sont pas identiques ; une variable globale n'existe qu'à l'exterieur des fonctions, et une variable locale n'existe qu'à l'interieur d'une fonction.
- Pour accéder à une variable globale au sein d'une fonction on la fait précéder du mot clé *global* ou bien on utilise une variable array nommée *$GLOBALS*

Exemple : Utilisation des variables globales et locales

```
<!DOCTYPE HTML>
<html>
<head>
  <meta http-equiv="content-type" content="text/html" />
  <title>Exemple</title>
</head>
<body>
    <h1><center> Utilisation des variables globales et locales</center></h1>
<?php
function calcul_moyenne($tab){
  $som= array_sum($tab);
  $nb= sizeof($tab);
  $moy= $som/$nb;
  return $moy;
}

function somme($x,$y){
    $som=$x+$y;
    return $som;
}
?>

<h2>Voici les notes de deux etuidants</h2>

<?php
$notes1=array(12,13.5,17,9,8,19,16,17);
$notes2=array(12,13.5,17,9,8,19,16,17);
print_r($notes1);
echo "<br/>";
print_r($notes2);

//Calcul des moyennes:
$moy1= calcul_moyenne($notes1);
```

```
34 $moy2= calcul_moyenne($notes2);
35 ?>
36 <h2>Affichage des moyennes</h2>
37 <?php
38 echo "Les moyennes sont : $moy1 et $moy2";
39 ?>
40 <h2>La moyenne des deux moyennes est:</h2>
41 <?php
42 $moy= somme($moy1,$moy2)/2;
43 echo $moy;
44 ?>
45 </body>
46 </html>
47
```

Remarque

Dans l'exemple ci-dessus :

- `$notes1, $notes2, $moy1,$moy2,$moy` sont des variables globales, on peut les utiliser partout dans le fichier
- les fonctions peuvent être définie partout dans le fichier (au début, au milieu ou la fin du fichier). On peu les utiliser avant ou après leur définition
- La variable local $moy de calcul_moyenne() et la variable globale $moy sont différentes
- L'argument $tab de calcul_moyenne() représente tout simplement un tableau qui doit être fourni en entré à la fonction calcul_moyenne pendant sont appel. Dans notre cas ici on fournit comme tableau $notes1 et $notes2 que la fonction utilise pour calculer la moyenne.
- Un variable globale peut avoir le même nom qu'un argument de fonction sans que cela ne porte confusion L'argument n'est pas identique à la variable globale même si les deux ont le même nom.

Exemple : Utilisation d'une variable globale à l'intérieur d'une fonction

Dans les exemples précédents pour calculer la moyenne des notes, on fait passer la variable $notes en entrée à la fonction calcul_moyenne(). On peut utliser directement la variable $notes au sein de la fonction calcul_moyenne() sans passer par un argument ou paramètre. Voici comment on procède :

```
1 <?php
2 // Definition de la fonction
3
4 function calcul_moyenne(){
5   global $notes;
6   $som= array_sum($notes); //on utilise directement la variable globale $notes
  dans la fonction
7   $nb= sizeof($notes);
8   $moy= $som/$nb;
9   return $moy;
10 }
11
12 //Utilisation de la fonction
13
14 $notes=array(12,13.5,17,9,8,19,16,17);
15 $moyenne= calcul_moyenne();
16 echo "La moyenne des notes est : $moyenne";
17 ?>
```

On peut même déclarer la variable $moy comme globale aussi, et accéder directement au résultat du calcul sans faire de return .

Pour utiliser la variable $moy comme globale, nous passerons par la variable array $GLOBALS

```php
<?php
// Definition de la fonction

function calcul_moyenne(){
  global $notes;
  $som= array_sum($notes); //on utilise directement la variable globale $notes dans la fonction
  $nb= sizeof($notes);
  $GLOBALS['moy']= $som/$nb; //on assigne le resultat du calcul à la variable globale $moy
}

//Utilisation de la fonction

$notes=array(12,13.5,17,9,8,19,16,17);
$moy=0; // on initialise la variable globale $moy par une valeur par defaut;
calcul_moyenne();
echo "La moyenne des notes est : $moy"; // après l'appel de calcul_moyenne(), $moy contient le resultat du calcul
?>
```

Remarque

- Cette manière de déclarer les variables globales comme globale à l'intérieur des fonctions et de les utiliser directement est utile dans certaine situation. Toute fois, elle est à utiliser avec circonspection, car elle rend moins souple l'utilisation des fonctions. Dans la pratique, il est conseillé de transmettre des valeurs aux fonctions en passant par des arguments, et d'accéder au résultat du calcul en faisant un return.
- Dans un projet PHP, il est conseillé de mettre les fonctions dans un fichier spécial, puis d'inclure ce fichier dans les pages où on aimerait les utiliser, plutôt que de les definir directement dans la page contenant les balises HTML où doit être affichées les informations. Dans notre exemple précédent, on devrait mettre les fonctions calcul_moyenne() et somme() dans un autre fichier PHP, puis on inclut ce fichier dans notre page et on les appel pour faire les calculs. Cela offre une bonne visibilité du code, une meilleure organisation, structuration, gestion, et maintenance du code.

3. Les variables super-globales

Les variables super-globales permettent de transmettre des informations d'une page PHP à une autre. En effet, les variables locales n'existent qu'on sein de la fonction où elles sont définies, et les variables globales n'existent qu'au sein du fichier où elles sont créées ; lorsqu'on exécute un autre fichier, ces variables n'existent plus. Toutefois, il est possible grâce au variables super-gobales de stocker des informations dans une variable et de l'utiliser partout peut importe la page PHP qui est exécute.

Les variables superglobales sont:

- écrites en majuscules et commencent toutes, à une exception près, par un tiret du huit ou underscore (_). *Ex* : $_GET et $_POST
- des array car elles contiennent généralement de nombreuses informations ;

- automatiquement créées par PHP à chaque fois qu'une page est chargée. Elles existent donc sur toutes les pages et sont accessibles partout : au milieu de votre code, au début, dans les fonctions, etc.

Pour afficher le contenu d'une superglobale et voir ce qu'elle contient, le plus simple est d'utiliser la fonction *print_r*.

Lien utile : *https://www.pierre-giraud.com/php-mysql/cours-complet/php-variables-superglobales.php*

Quelques variables super-globales sont :

"

- *$_SERVER* : ce sont des valeurs renvoyées par le serveur. Elle donne des informations sur le serveur. Tels que :
    - $_SERVER['REMOTE_ADDR']. Elle nous donne l'adresse IP du client qui a demandé à voir la page, ce qui peut être utile pour l'identifier.
    - $_SERVER['REMOTE_PORT']. Elle nous donne le port utilisé par la machine cliente pour communiquer avec le serveur web.
    - $_SERVER['REMOTE_HOST']. Elle nous donne le nom de l'hôte qui lit le script courant.
    - etc. Pour plus d'informations : *http://php.net/manual/fr/reserved.variables.server.php*
- *$_ENV* : ce sont des variables d'environnement toujours données par le serveur. C'est le plus souvent sous des serveurs Linux que l'on retrouve des informations dans cette superglobale.. *Ex* : $_ENV["USER"] fournit le nom de l'utilisateur.
- *$_SESSION* : on y retrouve les variables de session. Ce sont des variables qui restent stockées sur le serveur le temps de la présence d'un visiteur. Elle permet de conserver des informations sur l'utilisateur et ses actions.
- *$_COOKIE* : contient les valeurs des cookies enregistrés sur l'ordinateur du visiteur. Cela nous permet de stocker des informations sur l'ordinateur du visiteur pendant plusieurs mois, pour se souvenir de son nom par exemple.
- *$_GET* : Elle contient les données envoyées en paramètres dans l'URL.
- *$_POST* : Elle contient les informations qui viennent d'être envoyées par un formulaire.
- *$_FILES* : elle contient la liste des fichiers qui ont été envoyés via le formulaire précédent.

4. Travaux dirigés

Calcul de factoriel et combinaison

Libellé :

Écrire une fonction qui prend comme paramètre deux variables n et p, puis la fonction calcule la combinaison de p dans n

Correction

Pour calculer la combinaison de p dans n nous allons utiliser l'expression mathématique : combinaison de p dans n égale à n!/(p!*(n-p)!). Pour cela nous allons d'abord écrire une fonction qui permet de calculer le factoriel d'un entier.

Si le code ci-dessous ne s'affiche pas bien, veuillez cliquez sur le lien (bleu) ci-dessous

```php
<?php
function factoriel($n){
    $facto=1;
    for($i=1;$i<=$n;$i++){
        $facto *= $i;
    }
    return $facto;
}

function combinaison($n,$p){
    $c= factoriel($n)/(factoriel($p)*factoriel($n-$p));
    return $c;
}

$n=8;
$p=3;
echo "La combinaison de $p dans $n est : ". combinaison($n,$p);
?>
```

Exercice : Activité d'auto-évaluation No 2

IV

Exercice

Une variable locale est une variable :

- qui disparaît après la fin de l'exécution de la page
- une variable qui contient l'heure local du serveur
- déclarée à l'intérieur du fichier où se trouve la fonction
- déclarée au sein d'une fonction
- déclarée dans une seule page php

Exercice

Un argument d'une fonction est :

- une variable qu'on déclare au sein d'une fonction
- une valeur qu'on définie à la fin de la fonction
- une valeur qu'on fournit à l'entrée à une fonction et que la fonction utilise pour effectuer son traitement
- une sorte de symbole identifiant la fonction

Exercice

Les variables super-globales :

- sont des variables que l'on déclare partout dans des fichiers php
- permettent de transmettre des informations d'une page PHP à une autre
- sont des variables qu'on déclare directement dans un fichier PHP
- sont des variable se trouvant à l'intérieur d'une fonction
- sont des variables que le client transmet au serveur

Exercice

Parmi ces variables, lesquelles sont des variables superglobales

- ☐ $_COOKS
- ☐ $_POST
- ☐ $_GLOBAL
- ☐ $_FILES
- ☐ $_POSTER
- ☐ $_USERS
- ☐ $_COOKIE
- ☐ $_HOST
- ☐ $_SERVER
- ☐ $_SESSION

Les fonctions prédéfinies

V

Ce qui fait le force d'un langage de programmation, c'est les fonctions déjà prédéfinies que le langage met à la disposition des développeur afin de faciliter aux programmeur le développement d'application. PHP pour ce fait dispose de nombreuses fonctions qui facilitent amplement le développement de projet. Au fur et à mesure que vous travaillerez sur des projets php et que vous apprendrez de nouvelles notions, vous découvrirez progressivement ces fonctions.

Voici une liste de fonction PHP :

http://www.info-3000.com/phpmysql/listecompletefonction.php

http://php.net/manual/fr/indexes.functions.php

1. Les catégories de fonctions prédéfinies

Les catégories de fonctions prédéfinies par le langage sont nombreuses. On peut citer en autres :

- les fonctions mathématiques que nous avons vu dans la leçon 1 de notre formation. Pour de plus amples informations voir le lien : *http://php.net/manual/fr/ref.math.php*
- Les fonctions de manipulation des chaînes de caractères : *http://php.net/manual/fr/ref.strings.php*
- Les fonctions pour interagir avec une base de données mysql :*http://php.net/manual/fr/ref.mysql.php*
- Les fonctions de manipulation des tableaux : *http://php.net/manual/fr/ref.array.php*
- Les fonctions sur le système de fichiers: *http://php.net/manual/fr/ref.filesystem.php*
- Les fonctions de gestion des images : *http://php.net/manual/fr/ref.image.php*
- Les fonctions d'expressions rationnelles pour les vérifications de format ou recherche dans les chaînes de caractères: *http://php.net/manual/fr/ref.pcre.php*
- etc.

2. Quelques fonctions utiles

Dans cette partie, nous présentons quelques fonctions utiles et usuelles qui pourront beaucoup vous aider pour la gestion des chaînes de caractères. La liste n'est pas exhaustive. Vous en trouverez beaucoup ailleurs qui sont aussi très utiles en fonction de l'usage.

- *explode* : transforme une chaîne de caractères en tableau de chaîne en se basant sur un séparateur. Le séparateur peut être un espace, ou n'importe quel caractère

```
<?php
// Exemple 1
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
```

```
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2
// Exemple 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user, $pass, $uid, $gid, $gecos, $home, $shell) = explode(":",
$data);
echo $user; // foo
echo $pass; // *
?>
```

- *html_entity_decode* : Convertit toutes les entités HTML en caractères normaux. ça permet de lieux formater untexte en html pour que tous les caractères spéciaux puissent bien s'afficher. Une fonction silillaire est *htmlentities()* et *htmlspecialchars()*

```
< ?php
$orig = 'J\'ai "sorti" le <strong>chien</strong> tout à l\'heure';
$a = htmlentities($orig);
$b = html_entity_decode($a);
echo $a; // J'ai &quot;sorti&quot; le &lt;strong&gt;chien&lt;
/strong&gt; tout &amp;agrave; l'heure
echo $b; // J'ai "sorti" le <strong>chien</strong> tout à l'heure
?>
```

- *implode* : Rassemble les éléments d'un tableau en une chaîne

```
<?php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated; // lastname,email,phone
// Chaîne vide lors de l'emploi d'un tableau vide :
var_dump(implode('hello', array())); // string(0) ""
?>
```

var_dump() affiche les informations structurées d'une variable, y compris son type et sa valeur.

- *strtolower* : Renvoie une chaîne en minuscules

```
<?php
$str = "Marie A un Petit Agneau, et l'aime TRès fORt.";
$str = strtolower($str);
echo $str; // marie a un petit agneau, et l'aime très fort.
?>
```

- *strlen* : Calcule la taille (nombre de caractère) d'une chaîne

```
<?php
$str = 'abcdef';
echo strlen($str); // 6
$str = ' ab cd ';
echo strlen($str); // 7
?>
```

- *ucfirst* : Met le premier caractère en majuscule

```
<?php
```

```
$foo = 'bonjour tout le monde!';
$foo = ucfirst($foo); // Bonjour tout le monde!
$bar = 'BONJOUR TOUT LE MONDE!';
$bar = ucfirst($bar); // BONJOUR TOUT LE MONDE!
$bar = ucfirst(strtolower($bar)); // Bonjour tout le monde!
?>
```

- *trim* : Supprime les espaces (ou d'autres caractères) en début et fin de chaîne

```
<?php
$text = "\t\tThese are a few words :) ... ";
$binary = "\x09Example string\x0A";
$hello = "Hello World";
var_dump($text, $binary, $hello);
print "\n";
$trimmed = trim($text);
var_dump($trimmed);
$trimmed = trim($text, " \t.");
var_dump($trimmed);
$trimmed = trim($hello, "Hdle");
var_dump($trimmed);
$trimmed = trim($hello, 'HdWr');
var_dump($trimmed);
?>
```

L'exemple ci-dessus va afficher :

```
string(32) " These are a few words :) ... "
string(16) " Example string"
string(11) "Hello World"
string(28) "These are a few words :) ..."
string(24) "These are a few words :)"
string(5) "o Wor"
string(9) "ello Worl"
```

- *strtoupper* : Renvoie une chaîne en majuscules

```
<?php
$str = "Marie A un Petit Agneau, et l'aime fORt.";
$str = strtoupper($str);
echo $str; // MARIE A UN PETIT AGNEAU, ET L'AIME FORT.
// Note : Très aurait été converti en TRèS
?>
```

- *substr_count* : Compte le nombre d'occurrences de segments (mot)dans une chaîne

```
<?php
$text = 'Ceci est un test';
echo strlen($text); // 16
echo substr_count($text, 'est'); // 2
// la chaîne de caractères est réduite à 'st un test', alors elle
affiche 1
echo substr_count($text, 'est', 6);
// le texte est réduit à 'st u', alors elle affiche 0
```

```
echo substr_count($text, 'est', 6, 4);
// génère une erreur parce que 8+10 > 16
echo substr_count($text, 'est', 8, 10);
// affiche seulement 1, parce que elle ne compte pas les chaînes de
caractères
// qui se recouvrent
$text2 = 'gcdgcdgcd';
echo substr_count($text2, 'gcdgcd');
?>
```

- *str_replace* : Remplace un segment (suite de caractère ou mot) dans une texte

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original : $var<hr />\n";
// Remplace toute la chaîne $var par 'bob'.
echo substr_replace($var, 'bob', 0) . "<br />\n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br />\n";
// Insert 'bob' au début de la chaîne
echo substr_replace($var, 'bob', 0, 0) . "<br />\n";
// Remplace la séquence 'MNRPQR' par 'bob'.
echo substr_replace($var, 'bob', 10, -1) . "<br />\n";
echo substr_replace($var, 'bob', -7, -1) . "<br />\n";
// Efface la séquence 'MNRPQR' de $var.
echo substr_replace($var, '', 10, -1) . "<br />\n";
?>
```

- *substr* : Retourne un segment de chaîne. *Syntaxe* : `substr ($string , int $start , $length)`;retourne le segment de string défini par start et length. length est facultatif

  Exemple #1 Exemple de start négatif

```
<?php
$rest = substr("abcdef", -1); // retourne "f"
$rest = substr("abcdef", -2); // retourne "ef"
$rest = substr("abcdef", -3, 1); // retourne "d"
?>
```

  Exemple #2 Utilisation d'une valeur négative pour length

```
<?php
$rest = substr("abcdef", 0, -1); // retourne "abcde"
$rest = substr("abcdef", 2, -1); // retourne "cde"
$rest = substr("abcdef", 4, -4); // retourne false
$rest = substr("abcdef", -3, -1); // retourne "de"
?>
```

- *utf8_encode* : Encode une chaine au format utf8. Permet de mieux gérer l'affichage et le traitement des caractères accentués.

- *utf8_decode* : Decode une chaine au format ut8

Lien utile :

https://www.youtube.com/watch?v=VAxpQifHAmc

3. Travaux dirigés

Afficher une partie d'un texte sans couper un mot

- *Libellé*:

Ecrire une fonction prenant en paramètre un texte et une valeur entière n puis la fonction affiches les n premiers caractères du texte sans couper de mot.

- *Correction :*

La fonction substr() permet certes de tronquer une chaîne de caractères, mais rien n'est plus désagréable que de voir des mots coupés en d... Comme ça par exemple. Pour éviter cela, cette fonction affiche une partie d'un texte sans couper de mot. Elle prend deux paramètres, un argument représentant le texte, un deuxième argument représentant le nombre de caractères a afficher.

Ci-dessous une approche algorithmique, certaines fonctions sont toutes nouvelles, cela nous permettre de voir a quel point php dispose des fonctions prédéfinies et pouvant nous aider à resourdre beaucoup de problèmes:

```php
<?php
// Coupe un texte à $longueur caractères, sur les espaces, et ajoute des points de suspension...

function tronque($chaine, $longueur = 100)
{

  if (empty ($chaine))
  {
    return "";
  }
  elseif (strlen ($chaine) < $longueur)
  {
    return $chaine;
  }
  elseif (preg_match ("/(.{1,$longueur})\s./ms", $chaine, $match))
  {
    return $match [1] . "...";
  }
  else
  {
    return substr ($chaine, 0, $longueur) . "...";
  }
}
print tronque("Ceci n'est pas un tuyau",12);
// Renvoie Ceci n'est pas...
?>
```

Exercice : Activité d'auto-évaluation No 3

VI

Exercice

La fonction transforme une chaîne de caractères en tableau de chaîne en se basant sur un séparateur

Exercice

La fonction transforme une chaîne en majuscules

Exercice

La fonction transforme une chaîne en minuscule

Exercice

La fonction compte le nombre d'occurrences de segments (mot) dans une chaîne

Exercice

La fonction encode une chaine au format utf8.