Programmation Web 1

GI1

R.J.

Printemps 2024

المدرسة الوطنية للعلوم التطبيقية بتطوان
ⵜⵜⵉⵍⵍⵉⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵎⵓⵙⵙⴰⵙⵉⵏ ⵜⴰⵏⵉⵎⵓⵔⵜ
Ecole Nationale des Sciences Appliquées de Tétouan

Premiers pas en PHP (suite)

Les fonctions – Fonctions prédéfinies

- Il existe une multitude de fonctions prédéfinies (natives) dans PHP.
- Les différentes catégories et sous-catégories de ces fonctions sont répertoriées et documentées à partir de la page Web : https://www.php.net/manual/fr/funcref.php :

Les fonctions – Fonctions prédéfinies

- Affecte le comportement de PHP
  - APCu — APC User Cache
  - Componere
  - Gestion des erreurs — Gestion des erreurs et des journaux
  - FFI — Foreign Function Interface
  - OPcache
  - Contrôle de l'affichage — Bufferisation de sortie
  - Options PHP et informations PHP
  - phpdbg — Interactive PHP Debugger
  - runkit7
  - uopz — Opérations utilisateurs pour Zend
  - WinCache — Windows Cache pour PHP
  - Xhprof — Profilage hiérarchique
  - Yac
- Manipulation audio
  - OpenAL — Gestion Audio OpenAL
- Services d'identification
  - Radius
- Extensions spécifiques à la ligne de commande
  - Readline — GNU Readline

Les fonctions – Fonctions prédéfinies

- Extensions sur l'archivage et la compression
  - Bzip2
  - LZF
  - Phar
  - Rar — Archives RAR
  - Zip
  - Zlib — Compression Zlib
- Extensions sur la cryptographie
  - Hash — HASH
  - Mcrypt
  - Mhash
  - OpenSSL
  - Table de hachage de mot de passe
  - Rnp
  - Sodium
- Extensions sur les bases de données
  - Interface d'abstraction
  - Extensions spécifiques des fabricants de bases de données

Les fonctions – Fonctions prédéfinies

- Extensions relatives aux dates et aux heures
  - Calendar
  - Date/Heure — Date et Heure
  - HRTime — Gestion du temps à haute résolution
- Extensions relatives aux systèmes de fichiers
  - Direct IO
  - Les dossiers
  - Fileinfo — Informations sur les fichiers
  - Système de fichiers
  - Inotify
  - xattr
  - xdiff
- Support du langage humain et de l'encodage de caractères
  - Enchant — Bibliothèque d'orthographe Enchant
  - Gender — Détermine le sexe d'un prénom
  - Gettext
  - iconv
  - intl — Fonctions d'internationalisation
  - Chaînes de caractères multi-octets
  - Pspell
  - Recode — GNU Recode

Les fonctions – Fonctions prédéfinies

- Génération et traitement des images
  - Exif — Informations Exif
  - GD — Traitement des images et GD
  - Gmagick
  - ImageMagick — Traitement des images (ImageMagick)
- Extensions relatives aux emails
  - IMAP — IMAP, POP3 et NNTP
  - Mail
  - Mailparse
- Extensions sur les mathématiques
  - BC Math — Fonctions BCMath
  - GMP — GNU Multiple Precision
  - Math — Fonctions mathématiques
  - Statistics
  - Trader — Analyse technique pour les marchands

Les fonctions – Fonctions prédéfinies

- Affichage des données non-textuelles
  - FDF — Format de formulaire
  - GnuPG — GNU PG
  - wkhtmltox
  - PS — Création de document PostScript
  - RpmInfo
  - XLSWriter

- Extensions sur le contrôle des processus
  - Eio
  - Ev
  - Expect
  - PCNTL — Contrôle du processus
  - POSIX
  - Exécution de programme — Système d'exécution de programme
  - parallel
  - pthreads
  - Semaphore — Sémaphore
  - Mémoire partagée
  - Sync

Les fonctions – Fonctions prédéfinies

- Autres extensions basiques
  - GeoIP — Localisation géographique des IPs
  - FANN — FANN (Fast Artificial Neural Network)
  - Igbinary
  - JSON — Notation Objet JavaScript
  - Simdjson
  - Lua
  - LuaSandbox
  - Misc. — Fonctions diverses
  - Random — Générateurs de nombres aléatoires et fonctions liées à l'aléatoire.
  - Seaslog
  - SPL — Bibliothèque standard PHP (SPL)
  - Flux
  - Swoole
  - Tidy
  - Tokenizer
  - URLs
  - V8js — Intégration du moteur Javascript V8
  - Yaml — Linéarisation de données YAML
  - Yaf — Yaf (i.e. Yet Another Framework)
  - Yaconf
  - Taint
  - Data Structures
  - var_representation

Les fonctions – Fonctions prédéfinies

- Autres services
  - cURL — Bibliothèque d'URL client
  - Event
  - FTP
  - Gearman
  - LDAP — Lightweight Directory Access Protocol
  - Memcache
  - Memcached
  - mqseries
  - Réseau
  - RRD — RRDtool
  - ScoutAPM
  - SNMP
  - Sockets
  - SSH2 — Fonctions Shell2 sécurisé
  - Stomp — Client Stomp
  - SVM — Support Vector Machine
  - SVN — Subversion
  - TCP — TCP Wrappers
  - Varnish
  - YAZ
  - 0MQ messaging — ZMQ
  - ZooKeeper

Les fonctions – Fonctions prédéfinies

- Extensions spécifiques aux moteurs de recherche
  - Solr — Apache Solr
- Extensions spécifiques aux serveurs
  - Apache
  - Gestionnaire de processus FastCGI
- Extensions sur les Sessions
  - Sessions — Gestion des sessions
- Traitement du texte
  - CommonMark
  - Parle — Parsing and lexing
  - PCRE — Expressions rationnelles (compatible Perl)
  - ssdeep — Hachages flous ssdeep
  - Chaîne de caractères

Les fonctions – Fonctions prédéfinies

- Extensions relatives aux variables et aux types
  - Les tableaux
  - Classes/Objets — Les Classes/Objets
  - ctype — Vérification des types de caractères
  - Filter — Filtrage des données
  - Gestion des fonctions
  - Quickhash
  - Réflexion
  - Gestion des variables
- Services Web
  - OAuth
  - SOAP
  - Yar — Yet Another RPC Framework
  - XML-RPC
- Extensions pour Windows uniquement
  - COM — COM et .Net (Windows)
  - win32service

Les fonctions – Fonctions prédéfinies

- Manipulation XML
  - DOM — Model Objet d'un document
  - libxml
  - SimpleXML
  - WDDX
  - XMLDiff — XML : Différence et fusion
  - Analyseur syntaxique XML
  - XMLReader
  - XMLWriter
  - XSL
- Extensions GUI
  - UI

Les fonctions – Fonctions prédéfinies

- Voici des exemples de fonctions prédéfinies qui permettent de faire des calculs mathématiques :
  - – exp($X) : retourne la valeur de l'exponentielle de base e de $X,
  - – log($X) : retourne la valeur du logarithme népérien de $X,
  - – sqrt($X) : retourne la valeur de la racine carrée de $X,
  - – pow($X_1,$X_2) : retourne la valeur de $X_1 à la puissance $X_2,
  - – abs($X) : retourne la valeur absolue de $X,
  - – min($X_1,. . . ,$X_n) : retourne la plus petite valeur parmi les $X_i,
  - – max($X_1,. . . ,$X_n) : retourne la plus grande valeur parmi les $X_i,
  - – mt_rand($X_1,$X_2) : retourne un nombre entier aléatoire compris entre $X_1 et $X_2 (il peut être égal à un des deux),
  - – round($X) : retourne $X arrondi à l'entier le plus proche,
  - – acos($X) : retourne la valeur de arc cosinus de $X.
  - – atan($X) : retourne la valeur de arc tangente de $X.
  - – . . .

Les fonctions – Fonctions prédéfinies

- Et voici des exemples de fonctions prédéfinies qui permettent de faire des manipulations sur les chaînes de caractères :
  - – strlen($ch) : retourne la longueur de la chaîne de caractères $ch,
  - – str_replace($ch_1,$ch_2,$chaine) : remplace la chaîne de caractères $ch_1 par la chaîne de caractères $ch_2, dans la chaîne de caractères $chaine, tout en retournant le résultat,
  - – substr_count($ch1,$ch2) : retourne le nombre d'occurrences de la chaîne de caractères $ch2 dans la chaîne de caractères $ch1,
  - – str_repeat($ch,$N) : retourne une chaîne de caractères, qui résulte de la répétition $N fois de la chaîne de caractères $ch,
  - – strpos($ch1,$ch2) : retourne la position de la première occurrence de la chaîne de caractères $ch2 dans la chaîne de caractères $ch1,
  - – strrchr($ch,$car) : retourne le segment de la chaîne de caractères $ch qui commence de la dernière occurrence du caractère $car, jusqu'à la fin de la chaîne de caractères $ch,

Les fonctions – Fonctions prédéfinies

– – strtolower($ch) : retourne la chaîne de caractères $ch en minuscules,

– – strtoupper($ch) : retourne la chaîne de caractères $ch en majuscules,

– – ucwords($ch) : retourne la chaîne de caractères $ch après avoir mis en majuscule la première lettre de chacun de ses mots,

– – str_shuffle($ch) : mélange aléatoirement les caractères de la chaîne $ch, tout en retournant le résultat,

– – nl2br() : permet de faire des retours à la ligne en utilisant les \n,
Exemple :
echo nl2br("Bonjour ...\n");echo nl2br("... Bonsoir\n");
//Affiche à l'écran:
//Bonjour ...
//... Bonsoir

Les fonctions – Fonctions prédéfinies

– – uniqid() : retourne un identifiant unique en se basant sur la date et sur l'heure courante en microsecondes ; ça sera 13 caractères hexadécimaux,
uniqid($prefixe) : retourne un identifiant unique (préfixé) en se basant sur la date et sur l'heure courante en microsecondes ; ça sera la chaîne de caractères $prefixe suivie par 13 caractères hexadécimaux,
uniqid($prefixe,true) : retourne un identifiant unique (préfixé) en se basant sur la date et sur l'heure courante en microsecondes ; ça sera la chaîne de caractères $prefixe suivie par 13 caractères hexadécimaux suivis par 10 autres caractères générés via l'algorithme CLCG[a],

– – . . .

- Remarque : Beaucoup de fonctions de manipulation des chaînes de caractères ne marchent pas bien, quand on a des caractères accentués.

[a] CLCG (Combined Linear Congruential Generator) est un algorithme de génération de nombres pseudo-aléatoires.

Les fonctions – Fonctions prédéfinies

- On peut obtenir des informations sur la date et/ou sur l'heure actuelle(s) du serveur en utilisant par exemple :
  - – date('l') : retourne le jour de la semaine (Monday . . . Sunday),
  - – date('D') : retourne le jour de la semaine en 3 lettres (Mon . . . Sun),
  - – date('d') : retourne le jour du mois sur deux chiffres (01 . . . 31),
  - – date('j') : retourne le jour du mois (1 . . . 31),
  - – date('S') : retourne un suffixe ordinal sur deux lettres, par rapport au rang du jour dans le mois (st, nd, rd, th . . . th),
  - – date('z') : retourne le numéro du jour dans l'année (0 . . . 365[a]),
  - – date('W') : retourne le numéro de la semaine dans l'année (01 . . . 52),
  - – date('F') : retourne le nom du mois (January . . . December),
  - – date('M') : retourne le nom du mois en 3 lettres (Jan . . . Dec),
  - – date('m') : retourne le mois sur deux chiffres (01 . . . 12),
  - – date('n') : retourne le mois (1 . . . 12),

[a] Rappel : Les années bissextiles comptent 366 jours.

Les fonctions – Fonctions prédéfinies

- – date('Y') : retourne l'année sur quatre chiffres (par exemple 1970, 2000, 2015 . . .),
- – date('y') : retourne l'année sur deux chiffres (par exemple 70, 00, 15 . . .),
- – date('a') : retourne am ou pm. am, i.e., ante meridiem. pm, i.e., post meridiem,
- – date('g') : retourne les heures au format 12h (1 . . . 12), selon le fuseau horaire (UTC+01:00),
- – date('h') : retourne les heures au format 12h sur deux chiffres (01 . . . 12), selon le fuseau horaire (UTC+01:00),
- – date('G') : retourne les heures au format 24h (0 . . . 23), selon le fuseau horaire (UTC+01:00),
- – date('H') : retourne les heures au format 24h sur deux chiffres (00 . . . 23), selon le fuseau horaire (UTC+01:00),
- – date('i') : retourne les minutes (00 . . . 59),
- – date('s') : retourne les secondes (00 . . . 59),

Les fonctions – Fonctions prédéfinies

– – date('c') : retourne la date et l'heure complètes, au format ISO 8601 (par exemple 2015-04-08T20:49:42+02:00),

– – date('r') : retourne la date et l'heure complètes, selon la RFC 2822 (par exemple Wed, 08 Apr 2015 20:49:42 +0200),

– – . . .

- On peut obtenir des informations à propos d'une date passées ou futures, en faisant par exemple :
  – – date('l', mktime(heure_au_format_24h, minute, seconde, mois_de_1_à_12, jour_de_1_à_31, année)) : retourne le jour de la semaine (Monday . . . Sunday) correspondant à l'heure et à la date passées à mktime(),
  – – date('D', mktime(heure_au_format_24h, minute, seconde, mois_de_1_à_12, jour_de_1_à_31, année)) : retourne le jour de la semaine en 3 lettres (Mon . . . Sun) correspondant à l'heure et à la date passées à mktime(),
  – – . . .
  Exemple : echo date('l', mktime(8, 0, 0, 10, 25, 2019)) . '
';/*Affiche à l'écran Friday.*/

Les fonctions – Nos propres fonctions

- Outre les fonctions prédéfinies, il est possible de créer nos propres fonctions.
- Une fonction se déclare à l'aide du mot-clef function.
  Les instructions de la fonction sont mises entre des accolades.
- Syntaxe générale :

```
function nom_fonction($argument_1 , . . . , $argument_N)
{
        instruction_1;
        :
        instruction_L;
}
```

Les arguments (séparés par des virgules) ne sont pas obligatoires, mais les parenthèses qui suivent le nom de la fonction le sont.

Les fonctions – Nos propres fonctions

- On appelle (utilise) bien évidement la fonction à travers son nom :

  nom_fonction(paramètres)

  L'appel ou non avec des paramètres (séparés par des virgules), dépend de la définition de la fonction.
- Rappel : Lorsqu'on déclare une fonction, son code est enregistré en mémoire mais ne s'exécute pas tant qu'on ne l'a pas appelé.

Les fonctions – Nos propres fonctions

- Une fonction peut retourner une valeur, grâce au mot-clef return :
  - return $nom_variable;
  - return valeur;
  - return true;
  - return false;
- Notez bien qu'une fonction ne peut retourner qu'une seule et unique valeur ; mais on peut contourner ceci en retournant un tableau ou un objet.

Les fonctions – Nos propres fonctions

- Exemple :

```
<?php
function calcul_distance_euclidienne_dim2($p1,$p2,$r1,$r2){
$terme1=pow($r1-$p1 , 2);
$terme2=pow($r2-$p2 , 2);
return sqrt($terme1+$terme2);}
function affichage_distance_euclidienne_dim2($p1,$p2,$r1,$r2){
$resultat=calcul_distance_euclidienne_dim2($p1,$p2,$r1,$r2);
echo "La distance euclidienne entre [$p1,$p2] et [$r1,$r2] est $resultat <br />";}
$Vect11=1;$Vect12=3;$Vect21=4;$Vect22=2;
affichage_distance_euclidienne_dim2($Vect11,$Vect12,$Vect21,$Vect22);
/*Affiche à l'écran : La distance euclidienne entre [1,3] et [4,2] est
3.1622776601684.*/
$Vect11=9;$Vect12=0;$Vect21=7;$Vect22=6;
affichage_distance_euclidienne_dim2($Vect11,$Vect12,$Vect21,$Vect22);
/*Affiche à l'écran : La distance euclidienne entre [9,0] et [7,6] est
6.3245553203368.*/
?>
```

Les fonctions – Nos propres fonctions

Portée des variables :

- La portée d'une variable définie dans une fonction est locale à la fonction.
- Prenons un exemple d'illustration :

```
<?php
function annuler($var2) {
$var1 = 0;/*Par rapport à ce qui suivra après, il y a création ici d'une autre variable $var1 qui a cette fois-ci une portée locale.*/
$var2 = 0;/*Mise à 0 de $var2, mais en local uniquement.*/
$var3 = 0;/*Création d'une variable locale.*/}
$var1 = 93;
$var2 = -74;
annuler($var2);
echo "$var1 <br />";/*Affiche à l'écran 93.*/
echo "$var2 <br />";/*Affiche à l'écran -74.*/
echo "$var3 <br />";/*Donnera la notice: Undefined variable: var3.*/
?>
```

Les fonctions – Nos propres fonctions

Portée des variables :

- La création et l'utilisation de variables "globales" peuvent se faire via le mot-clef global ou le mot-clef $GLOBALS.
- Prenons un exemple d'illustration :

```
<?php
function creation_variables()
{
global $var1; $var1 = 36;/*Création d'une variable globale.*/
$GLOBALS["var2"] = "Bonjour";/*Création d'une variable globale.*/
}
creation_variables();
echo "$var1 <br />";/*Affiche à l'écran 36.*/
echo "$var2 <br />";/*Affiche à l'écran Bonjour.*/
?>
```

Les fonctions – Nos propres fonctions

Portée des variables :

- Notez bien que les variables globales d'un code PHP, i.e., déclarées dans la balise PHP <?php ?>, ne sont directement accessibles dans les fonctions créées dans le code PHP en question.
  Pour pouvoir utiliser une variable globale au sein d'une fonction, il faudra préciser au sein de la fonction que la variable en question est une variable globale.
- Prenons un exemple d'illustration :

```
<?php
function afficher() {
echo "$var1 <br />";/*Donnera la notice: Undefined variable: var1.*/
global $var2; echo "$var2 <br />";/*Affiche à l'écran -14.*/
$var2 = 77;}
$var1 = 35;
$var2 = -14;
afficher();
echo "$var2 <br />";/*Affiche à l'écran 77.*/
?>
```

Les fonctions – Nos propres fonctions

Portée des variables :

- Notez bien qu'il y a une différence lorsqu'on utilise global et lorsqu'on utilise $GLOBALS ; c'est qu'avec $GLOBALS, on peut accéder dans les fonctions à la fois au contenu global et au contenu local.
- Prenons un exemple d'illustration :

```
<?php
function contenus() {
$var = "Bonsoir";
echo 'Le contenu global est : ' . $GLOBALS["var"] . '.<br />';/*Affiche à
l'écran : Le contenu global est : Bonjour.*/
echo 'Le contenu local est : ' . $var . '.<br />';/*Affiche à l'écran : Le
contenu local est : Bonsoir.*/
}
$var = "Bonjour";
contenus();
?>
```

Les fonctions – Nos propres fonctions

Utilisation des variables statiques :

- Une variable statique, i.e., une variable déclarée (à l'intérieur d'une fonction) avec le mot-clef static, est une variable qui a une portée locale (à la fonction) uniquement mais qui ne perd pas sa valeur lors des appelles ultérieurs de la fonction.
  L'initialisation d'une variable statique se fait au début de la fonction mais uniquement lors de son première appel, et à chaque nouvel appel de la fonction elle garde la valeur du dernier appel.
- Prenons un exemple d'illustration :
  <?php
  function compter() {
  static $compt = 0;/*$compt est initialisée uniquement lors du première appel à la fonction.*/
  echo "$compt ";
  $compt++;/*La valeur suite au précédent appel est incrémentée de 1, à chaque nouvel appel.*/}
  for($i=0;$i<5;$i++) compter();/*Affiche à l'écran 0 1 2 3 4.*/
  ?>

Les fonctions – Nos propres fonctions

Utilisation des variables statiques :

- Les variables statiques s'utilisent généralement dans le cadre de fonctions qu'on appelle récursivement.

Les fonctions – Nos propres fonctions

Passage par référence :

- Par défaut, les paramètres sont passés aux fonctions par valeur ; on parle de passage par valeur ou de passage par copie.
  Dans ce cas de figure, modifier la valeur d'un paramètre à l'intérieur d'une fonction ne modifiera pas sa valeur à l'extérieur de la fonction.
- Les fonctions peuvent modifier les valeurs des paramètres à l'extérieur de celles-ci, en les leurs passant par référence ; on parle de passage par référence ou de passage par adresse.
  Pour ce faire, on utilise le symbole du passage par référence &.

Les fonctions – Nos propres fonctions

Passage par référence :

- Prenons un exemple d'illustration :

```
<?php
function multiplier_par_2($var) //Passage par valeur.
{$var *= 2;}
function diviser_par_2(&$var)//Passage par référence.
/*Notez bien qu'on ne met pas le signe de référence lors de l'appel de la
fonction ; on le met uniquement lors de sa déclaration.*/
{$var /= 2;}
$var=1024;
multiplier_par_2($var);
echo "$var <br />";//Affiche à l'ecran 1024.*/
diviser_par_2($var);/*Notez bien qu'on ne met pas le signe de référence
lors de l'appel de la fonction ; on le met uniquement lors de sa
déclaration.*/
echo "$var <br />";//Affiche à l'ecran 512.*/
?>
```

Les fonctions – Nos propres fonctions

Passage par référence :

- Grâce au symbole &, plusieurs noms de variables peuvent pointer vers un même espace mémoire (vers une même valeur).
- Exemple :

```
<?php
$var1 = 23; $var2 = &$var1;/*var2 pointe vers le même emplacement en mémoire que var1.*/
echo "$var1 <br />";/*Affiche à l'écran 23.*/
echo "$var2 <br />";/*Affiche à l'écran 23.*/
$var2 = -76;
echo "$var1 <br />";/*Affiche à l'écran -76.*/
?>
```

Les tableaux – Tableaux numérotés

Création d'un tableau :

- Un tableau numéroté de N éléments est crée via :

  $nom_tableau = array(valeur_1 , valeur_2 , . . . , valeur_N);

  ou via :

  $nom_tableau[0] = valeur_1;
  $nom_tableau[1] = valeur_2;
  $\vdots$
  $nom_tableau[$N - 1$] = valeur_$N$;
- L'indice du premier élément d'un tableau est 0 (non pas 1). Alors par conséquent, l'indice du Nème élément est le numéro $N - 1$.

Les tableaux – Tableaux numérotés

Accès à un tableau :

- Pour accéder à un élément d'un tableau nom_tableau, on utilise nom_tableau[indice_element], où indice_element est l'indice de l'élément voulu.
- Exemples :
  $couleurs=array("rouge","vert","bleu");
  echo "Le second élément du tableau contient : $couleurs[1]";//Lecture du contenu d'un élément.
  $couleurs[1]="noir";//Modification d'un élément.
  $couleurs[3]="blanc";//Ajout d'un nouvel élément à la fin du tableau.

Les tableaux – Tableaux associatifs

- Rappel : On peut indexer les éléments d'un tableau à travers des chaînes de caractères au lieu des habituels nombres, i.e., on peut utiliser des chaînes de caractères en tant qu'indices. On parle alors d'un tableau associatif ; il associe une valeur à une chaîne de caractères (qu'on va appeler une clé).
- Un tableau associatif de *N* éléments se crée via :

```
$nom_tableau = array(
'cle_1' => valeur_1,
'cle_2' => valeur_2,
                :
'cle_N-1' => valeur_N-1,
'cle_N' => valeur_N
);
```

ou via :

```
$nom_tableau['cle_1'] = valeur_1; $nom_tableau['cle_2'] = valeur_2; ...$nom_tableau['cle_N-1'] = valeur_N-1; $nom_tableau['cle_N'] = valeur_N;
```

Les tableaux – Tableaux associatifs

- Remarque : Entre 'cle_I' et valeur_I il y a un caractère d'égalité qui est collé à un caractère strictement supérieur.
- Exemples :
  $CIN = array('Youssef' => 'L234476' ,
  'Layla' => 'AB281190' , 'Saad' => 'Z136587');
  $CNE['Youssef'] = 1125349608; $CNE['Layla'] = 8126734451;
  $CNE['Saad'] = 6310979714;

Les tableaux – La boucle foreach

- Outres while, do . . . while et for, il y a un autre type de boucles qui est adapté aux tableaux : c'est la boucle foreach.
- foreach va parcourir automatiquement tout le tableau, en copiant à chaque fois la valeur d'un des éléments du tableau en une variable donnée.
- Prenons un premier exemple d'illustration, dans le cas d'un tableau numéroté :

```
<?php
$villes = array('Agadir','Dakhla','Errachidia','Fès','Marrakech','Nador',
'Rabat','Tétouan');
foreach($villes as $valeur_element)
{echo "$valeur_element ";
/*Affiche à l'écran Agadir Dakhla . . . Rabat Tétouan.
$valeur_element contiendra tour à tour chacune des valeurs des
éléments de $villes ; tout d'abord Agadir, puis Dakhla, . . . , puis Rabat et
finalement Tétouan.*/ }
?>
```

Les tableaux – La boucle foreach

- Prenons maintenant un exemple avec deux tableaux associatifs :

```
<?php
$telephones = array('Youssef' => '0645678909' , 'Layla' => '0622547890' ,
'Saad' => '0611477532');
foreach($telephones as $valeur_element) {
echo "$valeur_element ";
/*Affiche à l'écran : 0645678909 0622547890 0611477532.
$valeur_element contiendra tour à tour chacune des valeurs des éléments de
$telephones.*/}
$poids['Youssef'] = 73.5; $poids['Layla'] = 55; $poids['Saad'] = 66.7;
foreach($poids as $cle_element => $valeur_element) {
/*$cle_element et $valeur_element vont contenir respectivement la clé et la valeur
de l'élément courant.*/
echo "poids[$cle_element]=$valeur_element <br />";
/*Affiche à l'écran :
poids[Youssef]=73.5
poids[Layla]=55
poids[Saad]=66.7.*/}
?>
```

Les tableaux – Opérations sur les tableaux

- array_key_exists($cle , $tab) : retourne true si la clé $cle existe dans le tableau $tab, false sinon.
  Notez bien que la fonction array_key_exists() est sensible à la casse.
- in_array($valeur , $tab) : retourne true si la valeur $valeur se trouve dans le tableau $tab, false sinon.
  Notez bien que la fonction in_array() est sensible à la casse.
- array_search($valeur , $tab) : retourne l'indice (dans le cas d'un tableau numéroté) ou la clé (dans le cas d'un tableau associatif) de la valeur $valeur dans le tableau $tab, false si $valeur n'existe pas dans $tab.
  Notez bien que la fonction array_search() est sensible à la casse.
- count($tab) : retourne le nombre d'éléments qui existent dans le tableau $tab.
- sort($tab) : modifie le tableau $tab en le triant par ordre croissant.
- array_reverse($tab) : retourne un nouveau tableau qui contient les mêmes éléments que le tableau $tab, mais dans l'ordre inverse.
- array_flip($tab) : retourne un tableau, dont les clés sont les valeurs du tableau $tab et dont les valeurs sont les clés de $tab.

Les tableaux – Opérations sur les tableaux

- – – array_slice($tab , $D) : retourne un tableau extrait du tableau $tab, en commençant de l'élément d'indice $D.
  – – array_slice($tab , $D , $N) : retourne un tableau extrait du tableau $tab, en commençant de l'élément d'indice $D et en prenant $N éléments.
- array_merge($tab_1 , $tab_2 , . . . , $tab_*N*) : retourne un tableau qui résulte de la fusion des tableaux $tab_1 , $tab_2 , . . . , $tab_*N* (on aura les éléments de $tab_1 , suivis par ceux de $tab_2 , . . . , suivis finalement par ceux de $tab_*N*).
- array_combine($tab_1 , $tab_2) : retourne un tableau associatif, dont les clés sont les valeurs du tableau $tab_1, et dont les valeurs sont les valeurs du tableau $tab_2.
- array_fill(0 , $N , $val) : retourne un tableau de $N éléments (d'indices allant de 0 à $N-1) dont les valeurs sont toutes égales à $val.
- array_intersect($tab , $tab_1 , . . . , $tab_*N*) : retourne un tableau contenant toutes les valeurs du tableau $tab qui sont présentes dans tous les tableaux $tab_1 , . . . , $tab_*N*.

Les tableaux – Opérations sur les tableaux

- array_diff($tab , $tab_1 , . . . , $tab_*N*) : retourne un tableau contenant toutes les valeurs du tableau $tab qui ne sont pas présentes dans un des tableaux $tab_1 , . . . , $tab_*N*.
- . . .

Les tableaux – Structures de données

- On peut mettre en place des structures de données en exploitant :
  – – array_pop($tab) : retire le dernier élément du tableau $tab (ça modifie donc le tableau), tout en retournant la valeur retirée. NULL sera retournée, si $tab est vide.
  – – array_push($tab , $valeur_1 , $valeur_2 , . . . , $valeur_N) : ajoute $valeur_1 puis $valeur_2 puis . . . puis $valeur_N (ça ajoute donc *N* nouveaux éléments) à la fin du tableau $tab, tout en retournant le nouveau nombre d'éléments dans $tab.
  – – array_shift($tab) : retire le premier élément du tableau $tab (ça modifie donc le tableau), tout en retournant la valeur retirée. NULL sera retournée, si $tab est vide.
  Notez bien que les éléments restants sont décalés, pour combler la place qui a été vidée.
  – – array_unshift($tab , $valeur_1 , $valeur_2 , . . . , $valeur_N) : ajoute $valeur_N puis . . . puis $valeur_2 puis $valeur_1 (ça ajoute donc *N* nouveaux éléments) au début du tableau $tab, tout en retournant le nouveau nombre d'éléments dans $tab.
  Notez bien que les anciens éléments sont décalés, pour permettre l'ajout des nouveaux éléments.
- Ainsi, l'utilisation d'un tableau comme une pile est tout à fait possible, grâce à array_push (pour empiler) et array_pop (pour désempiler). Et avec array_shift et array_push, on peut retirer et ajouter des éléments dans une file.

Les tableaux – Structures de données

- Exemple :

```
<?php
$structure_donnees = array("C","O","M","P","L","E","T");
$ancien_dernier_element=array_pop($structure_donnees);
$ancien_dernier_element=array_pop($structure_donnees);
echo "$ancien_dernier_element <br />";//Affiche à l'écran E
foreach($structure_donnees as $val) echo "$val "; echo "<br />";
//Affiche à l'écran C O M P L
$nouvelle_longueur=array_push($structure_donnees , "È","T","E","S");
echo "$nouvelle_longueur <br />";//Affiche à l'écran 9
foreach($structure_donnees as $val) echo "$val "; echo "<br />";
//Affiche à l'écran C O M P L È T E S
$ancien_premier_element=array_shift($structure_donnees);
echo "$ancien_premier_element <br />";//Affiche à l'écran C
foreach($structure_donnees as $val) echo "$val "; echo "<br />";
//Affiche à l'écran O M P L È T E S
$nouvelle_longueur=array_unshift($structure_donnees,"I","N","C");
echo "$nouvelle_longueur <br />";//Affiche à l'écran 11
foreach($structure_donnees as $val) echo "$val "; echo "<br />";
//Affiche à l'écran I N C O M P L È T E S
?>
```

Les tableaux – Tableaux à plusieurs dimensions

- Il est possible de créer des tableaux à plusieurs dimensions. Par exemple, Un tableau à deux dimensions est un tableau de tableaux (une matrice).
- Exemple 1:

```
//Création de la matrice
$matrice1 = array(array("element_00","element_01","element_02") ,
array("element_10","element_11","element_12") ,
array("element_20","element_21","element_22"));
//Affichage du contenu de la matrice
foreach($matrice1 as $ligne)
        {
        foreach($ligne as $val)
                {
                echo "$val ";
                }
        echo "<br />";
        }
```

L'exécution du code PHP donnera :
element_00 element_01 element_02
element_10 element_11 element_12
element_20 element_21 element_22

Les tableaux – Tableaux à plusieurs dimensions

- Exemple 2:

```
//Création de la matrice
$matrice2[0][0]=1;$matrice2[0][1]=9;$matrice2[0][2]=5;
$matrice2[1][0]=2;$matrice2[1][1]=8;$matrice2[1][2]=6;
$matrice2[2][0]=7;$matrice2[2][1]=4;$matrice2[2][2]=3;
//Modification du contenu de la matrice
for($i=0;$i<3;$i++)
        {
        for($j=0;$j<3;$j++)
                {$matrice2[$i][$j]=$matrice2[$i][$j]*100;}
        }
//Affichage du contenu de la matrice
foreach($matrice2 as $ligne)
        {
        foreach($ligne as $val)
                {echo "$val ";}
        echo "<br />";
        }
```

Les tableaux – Tableaux à plusieurs dimensions

L'exécution du code PHP donnera :
100 900 500
200 800 600
700 400 300

Programmation orientée objet en PHP

- La création d'une classe suit la syntaxe générale suivante :

```
class Ma_Classe
{
    visibilité_attribut_1 $nom_attribut_1;
    :
    visibilité_attribut_N $nom_attribut_N;
    visibilité_méthode_constructeur function __construct($argument_1, ..., $argument_N)
    {
        $this->nom_attribut_1 = $argument_1;
        :
        $this->nom_attribut_N = $argument_N;
    }
    visibilité_méthode_1 function nom_méthode_1( arguments ) { instructions }
    :
    visibilité_méthode_L function nom_méthode_L( arguments ) { instructions }
}
```

La classe Ma_Classe est définie donc par les attributs nom_attribut_1, ..., nom_attribut_*N* et par les méthodes nom_méthode_1(), ..., nom_méthode_*L*() (en plus de la méthode constructeur __construct() qui permet d'instancier des objets de la classe).

- Attention : Il y a deux caractères underscores avant construct.
- Remarque : Entre $this et nom_attribut_*I* il y a un caractère moins qui est collé à un caractère strictement supérieur.
- La visibilité d'un attribut ou d'une méthode indique à partir d'où est ce qu'on peut y avoir accès.
  Par exemple, si la visibilité d'un attribut ou d'une méthode est *public*, alors on pourra y avoir accès depuis n'importe où, i.e., depuis l'intérieur de la classe (dans les méthodes créées) et depuis l'extérieur.
  La visibilité *private* quand à elle rend l'accès à un attribut ou à une méthode possible seulement depuis l'intérieur de la classe.
- La création d'une instance instance_*I*_Ma_Classe de la classe Ma_Classe se fera via l'instruction :

  $instance_*I*_Ma_Classe = new Ma_Classe(arguments);

- Exemple :

```
<?php
  class Position
  {
    private $latitude;
    private $longitude;
    public function __construct($la, $lo)
    {$this->latitude = $la; $this->longitude = $lo;}
    public function get_latitude()
    {return $this->latitude;}
    public function get_longitude()
    {return $this->longitude;}
    public function set_latitude($la)
    {$this->latitude = $la;}
    public function set_longitude($lo)
    {$this->longitude = $lo;}
  }
  $pos = new Position("35.561210" , "-5.364427");
```

```
    echo 'La position initiale est : ' . $pos->get_latitude() . ' , ' .
    $pos->get_longitude() . '.<br />';/*Affiche à l'écran La position initiale
    est : 35.561210 , -5.364427..*/
    $pos->set_latitude("33.999010"); $pos->set_longitude("-6.843553");
    echo 'La position finale est : ' . $pos->get_latitude() . ' , ' .
    $pos->get_longitude() . '.<br />';/*Affiche à l'écran La position finale
    est : 33.999010 , -6.843553..*/
?>
```

- Notez bien qu'on peut définir une/des constante(s) de classe dans une classe.
- Une constante de classe est en quelque sorte un attribut spécial qui appartient à la classe et non à un quelconque de ses objets, et dont la valeur ne change jamais.
- La déclaration d'une constante de classe à l'intérieur d'une classe se fera via l'instruction :

  const NOM_CONSTANTE_CLASSE = valeur_CONSTANTE_CLASSE;
- L'accès à valeur_CONSTANTE_CLASSE dans une instruction d'une méthode de la classe se fera via l'expression :

  self::NOM_CONSTANTE_CLASSE
- L'accès à valeur_CONSTANTE_CLASSE dans une instruction en dehors du bloc de création de la classe se fera via une expression du genre :

  Nom_Classe::NOM_CONSTANTE_CLASSE

- Exemple :

```
<?php
  class Transformateur
  {
    private $marque;
    private $puissance;
    const TENSION = 220;
    public function __construct($ma, $pu)
    {$this->marque = $ma; $this->puissance = $pu;}
    public function get_marque()
    {return $this->marque;}
    public function get_puissance()
    {return $this->puissance;}
    public function calcul_intensite()
    {return $this->puissance / self::TENSION;}
  }
  $trs = new Transformateur("TEMPSA" , "500");
```

```
    echo 'Ce transformateur de la marque ' . $trs->get_marque() .
    ' a une puissance égale à ' . $trs->get_puissance() . '. Sa tension
    est ' . Transformateur::TENSION . ' ; donc son intensité est ' .
    $trs->calcul_intensite() . '.';/*Affiche à l'écran Ce transformateur de la
    marque TEMPSA a une puissance égale à 500. Sa tension est 220 ;
    donc son intensité est 2.2727272727273..*/
?>
```

- Notez bien qu'on peut également avoir dans une classe un/des attribut(s) et une/des méthode(s) statiques.
- Un attribut statique est un attribut particulier qui appartient à la classe et non à ses objets ; il n'existe qu'en un seul exemplaire. Tous les objets de la classe auront accès à cet attribut et cet attribut aura la même valeur pour tous les objets.
- Remarque : Si un des objets de la classe modifie la valeur d'un attribut statique, alors tous les autres objets de la classe auront cette nouvelle valeur.
- Pour déclarer un attribut statique, on met le mot-clef static avant le nom de l'attribut.
- Une méthode statique est une méthode qui est faite pour agir sur une classe et non sur un objet. Donc généralement, elle servira à manipuler un/des attribut(s) statique(s).
- Pour déclarer une méthode statique, on met le mot-clef static avant le mot-clef function.

- L'accès à un attribut statique à l'intérieur de sa classe se fera via une expression du genre :

  self::$nom_attribut_statique

- L'appel d'une méthode statique en dehors de sa classe se fera via une expression du genre :

  Nom_Classe::nom_méthode_statique(arguments)

- Exemple :

```
<?php
  class Compteur
  {
    private static $nombre_objets = 0;
    public function __construct()
    {self::$nombre_objets++;/*La création d'un objet incrémente
    $nombre_objets.*/}
    public static function get_nombre_objets()
    {return self::$nombre_objets;}
  }
```

```
    echo 'Il y a ' . Compteur::get_nombre_objets() . ' objet(s) de la classe
    Compteur.<br />';/*Affiche à l'écran Il y a 0 objet(s) de la classe
    Compteur..*/
    $obj1 = new Compteur(); $obj2 = new Compteur();
    $obj3 = new Compteur(); $obj4 = new Compteur();
    echo 'Il y a ' . Compteur::get_nombre_objets() . ' objet(s) de la classe
    Compteur.<br />';/*Affiche à l'écran Il y a 4 objet(s) de la classe
    Compteur..*/
?>
```

- Sachez que PHP supporte le mécanisme d'héritage.
- Voici un exemple d'utilisation :

```
<?php
  class Plane
  {
    /*protected signifie que l'attribut ne pourra être accessible que de
    l'intérieur de sa classe ou d'une classe dérivée. Si on utilise private
    à la place de protected, on n'aura pas accès dans la classe Boeing
    (qui suivra) aux attributs seats et range.*/
    protected $seats;
    protected $range;
    public function __construct($se, $ra)
    {$this->seats = $se; $this->range = $ra;}
    public function get_seats()
    {return $this->seats;}
    public function get_range()
    {return $this->range;}
    public function set_seats($se)
    {$this->seats = $se;}
  }
```

```
class Boeing extends Plane
{
  private $model;
  public function __construct($mo, $se, $ra)
  {$this->model = $mo;//model est un attribut qui est propre à Boeing.
  parent::__construct($se, $ra);/*Le constructeur parent est appelé.*/}
  public function get_model()
  {return $this->model;}
}
$av = new Boeing("787-9" , 302 , 14140);
echo 'Le Boeing ' . $av->get_model() . ' peut accueillir ' .
$av->get_seats() . ' passagers et parcourir ' . $av->get_range() .
'km.';/*Affiche à l'écran Le Boeing 787-9 peut accueillir 302 passagers
et parcourir 14140km..*/
?>
```

Projet

- Objectif : Créer une plateforme d'apprentissage en ligne.
- Cahier des charges :
  – – Votre site Web doit permettre à un(e) nouveau/nouvelle professeur(e) de s'inscrire dans la plateforme. Pareil, il doit permettre à un(e) nouvel(le) étudiant(e) de s'inscrire dans la plateforme. L'inscription se fera en indiquant différentes informations identitaires, e.g., le nom et le prénom de la personne, son adresse email (elle sera utilisée comme login), son mot de passe, . . . ,
  – – après s'être authentifié, tout(e) professeur(e) (déjà inscrit(e) dans la plateforme) pourra ajouter un nouveau cours dans la plateforme. Pour ce faire, il/elle devra indiquer toutes les informations descriptives de ce cours, e.g., l'intitulé du cours, une présentation du contenu du cours, des mots clés se rapportant au cours, le public visé, les prérequis à avoir pour pouvoir suivre le cours, ... Le/la professeur(e) devra ensuite ajouter les ressources du cours, i.e., des fichiers .pdf relatifs aux différentes parties du cours,

Projet

– – notez bien qu'un(e) professeur(e) pourra décider de rendre visible tout le contenu d'un cours donné, comme il/elle pourra rendre visible progressivement les parties de ce cours. Donc la plateforme devra permettre d'avoir ces deux cas de figure,
– – la plateforme devra permettre à un(e) professeur(e) de modifier ou de supprimer un cours s'il/si elle le désire,
– – après s'être authentifié, tout(e) étudiant(e) (déjà inscrit(e) dans la plateforme) pourra avoir accès aux ressources d'un cours donné qui est disponible dans la plateforme, en s'inscrivant à ce cours. Un(e) étudiant(e) pourra bien entendu s'inscrire en même temps à différents cours,
– – notez bien que la plateforme devra permettre à un(e) professeur(e) de voir la liste de toutes les personnes qui suivent un de ses cours,
– – la plateforme devra permettre à un(e) étudiant(e) donné(e) de rechercher un cours selon tous les critères de recherche possibles,
– – la plateforme devra lui permettre également de consulter (directement dans la plateforme) tout fichier .pdf visible d'un cours auquel il/elle s'est inscrit(e), comme elle devra lui permettre de le télécharger,

Projet

– – par rapport à un cours donné, la plateforme devra permettre à tout(e) étudiant(e) d'envoyer une question au/à la professeur(e) responsable de ce cours. Le/la professeur(e) utilisera la plateforme pour répondre à ce message. Le/la professeur(e) pourra également faire des annonces en relation avec son cours, i.e., envoyer d'un seul coup des messages à toutes les personnes qui suivent ce cours,
– – votre site Web doit avoir également des pages Web qui permettront à un(e) professeur(e)/étudiant(e) authentifié(e) de modifier son profil, i.e., ses informations identitaires,
– – pour qu'un compte d'un(e) professeur(e)/étudiant(e) soit supprimé de la base de données de la plateforme, il faudra qu'après l'authentification cette personne envoie un message (de demande de suppression) à l'administrateur de la plateforme. Seul ce dernier pourra procéder à la suppression d'un(e) professeur(e)/étudiant(e) après la lecture de son message. Vous devez donc avoir les pages Web nécessaires pour permettre la suppression des comptes,

Projet

– – dotez bien sûr votre site Web de pages Web principales qui joueront le rôle de tableaux de bord. Vous devrez évidemment sécuriser l'accès à ces différentes pages Web. Le tableau de bord de l'administrateur devra lui donner une idée claire sur toutes les personnes inscrites dans la plateforme, ainsi que sur le suivi des cours disponibles. Le tableau de bord d'un(e) professeur(e) devra lui permettre d'accéder au(x) cours qu'il/elle propose, alors que le tableau de bord d'un(e) étudiant(e) devra lui permettre d'accéder au(x) cours qu'il/elle suit.
– – faites en sorte que votre site Web soit fonctionnel, facile et agréable à utiliser.
Vous devez tenir compte de tout ce qui est demandé dans le cahier des charges. Et vous pouvez ajouter d'autres choses si vous jugez qu'elles sont utiles.

Projet

- Consignes :
  – – Le projet doit être réalisé par des groupes de quatre ou de cinq personnes,
  – – une personne choisie de chaque groupe devra m'envoyer par mail à **reda.jourani@yahoo.fr** un fichier compressé qui contient tout ce qui se rapporte à votre réalisation, à savoir : le script de création et de remplissage de la base de données[a](qui va contenir toutes les informations nécessaires au bon fonctionnement de la plateforme), et l'ensembles des fichiers .html .css .js .php .jpg de votre site Web.
  Attention : Il faudra que vous commentiez chaque ligne de code HTML, CSS, JAVASCRIPT et PHP pour expliquer/justifier sa présence.
  Il faudra également envoyer dans le mail une vidéo dans laquelle vous présentez en détails toutes les fonctionnalités de votre site Web,
  – – envoyez obligatoirement le mail depuis **vos adresses emails institutionnels**, et donner lui obligatoirement comme titre **PROJET-GI1-2024**,
  – – vous aurez au plus tard jusqu'au **11 Mai** pour rendre vos réalisations.

[a]Le serveur de base de données utilisé doit être obligatoirement MySQL, et le port utilisé par le serveur doit être obligatoirement 3308.

Exercice 1

- Créez avec PHP une classe Point qui est définie par les attributs x, y et z, et par les méthodes __construct, get_x, get_y, get_z, set_x, set_y, set_z, coordonnees_cylindriques et coordonnees_spheriques.
  Les deux dernières méthodes afficheront les coordonnées cylindriques (ρ,φ,z) et sphériques (r,θ,φ) correspondant aux coordonnées cartésiennes (x,y,z).
- Les coordonnées cylindriques s'obtiendront grâce à :

$$\begin{cases} \rho & = & \sqrt{x^2+y^2} \\ \varphi & = & \begin{cases} 0 & si \quad x=0 \quad et \quad y=0 \\ arcsin(y/\rho) & si \quad x \geq 0 \\ arctan(y/x) & si \quad x>0 \\ -arcsin(y/\rho)+\pi & si \quad x<0 \end{cases} \\ z & = & z \end{cases}$$

Exercice 1

- Les coordonnées sphériques s'obtiendront grâce à :

$$\begin{cases} r & = & \sqrt{x^2+y^2+z^2} \\ \theta & = & \begin{cases} 0 & si \quad x=0 \quad et \quad y=0 \quad et \quad z=0 \\ arccos(z/r) & sinon \end{cases} \\ \varphi & = & \begin{cases} 0 & si \quad x=0 \quad et \quad y=0 \\ arcsin(y/\sqrt{x^2+y^2}) & si \quad x \geq 0 \\ arctan(y/x) & si \quad x>0 \\ -arcsin(y/\sqrt{x^2+y^2})+\pi & si \quad x<0 \end{cases} \end{cases}$$

- Créez des objets de la classe Point pour tester les méthodes de celle-ci.

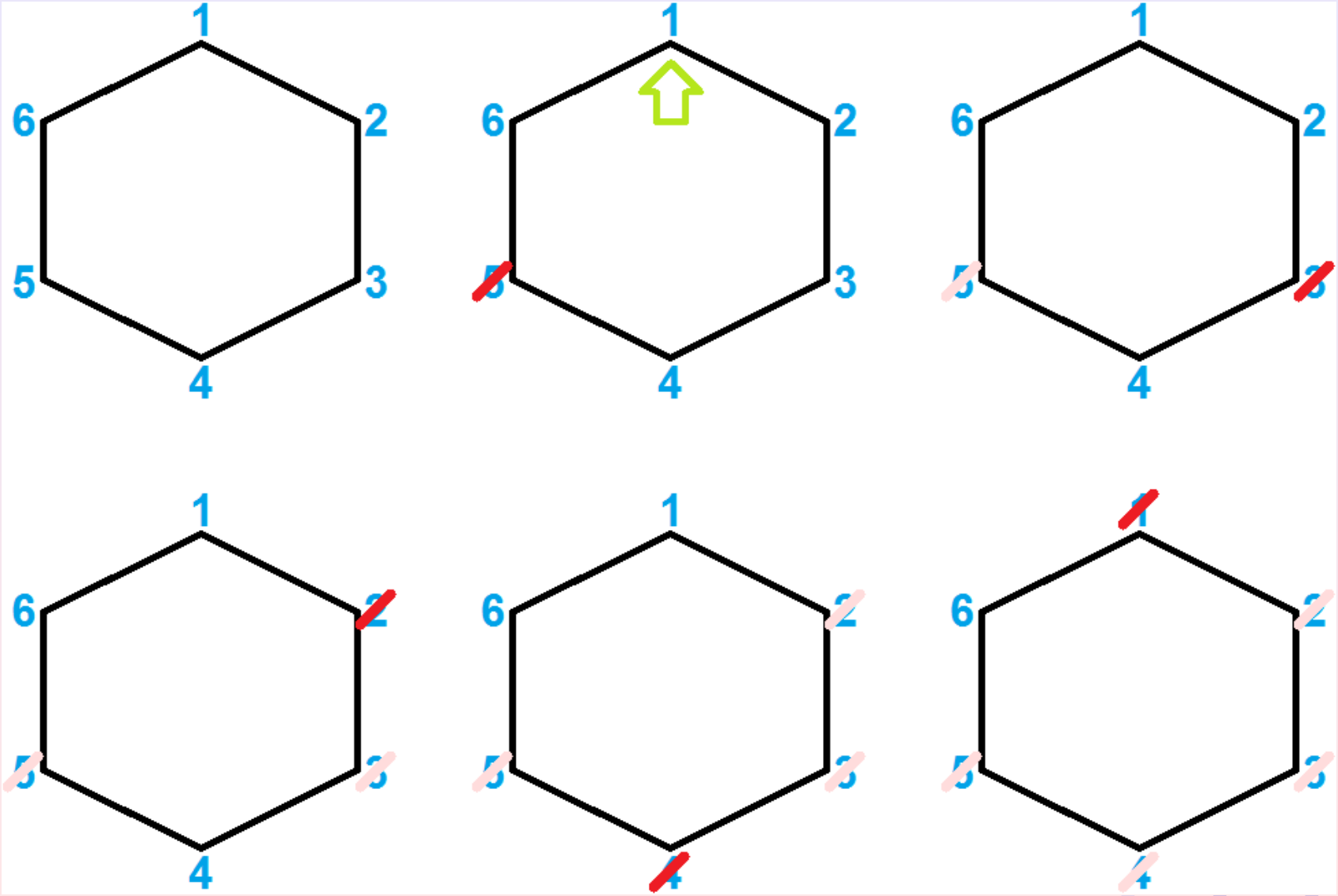
Exercice 2

- Jadis durant une guerre, Flavius Josèphe et ses 40 soldats ont été assiégés par des troupes ennemies. Les 40 soldats ont fini par préférer que le groupe se suicide plutôt qu'ils soient capturés.
  Ils décidèrent donc de former un cercle avec Josèphe, et de choisir au hasard quelqu'un parmi eux pour qu'il tue la troisième personne sur sa gauche. La personne à droite du mort devait tuer à son tour la troisième personne sur sa gauche. Ils allaient continuer ainsi à se tuer mutuellement et successivement jusqu'à ce qu'il ne reste qu'une seule personne. Cette dernière devait finalement se tuer elle-même.
  Josèphe, qui ne souhaitait pas mourir, trouva rapidement la place sûre, i.e., la place de la dernière personne restante et qui est censée se suicider après, et fini donc par se sauver.
  Trouver cette place sûre a été après appelé la résolution du problème de Josèphe, qui a été généralisé en fin de compte à un nombre quelconque de soldats et à un saut quelconque pour tuer.

Exercice 2

- Si on prend par exemple le cas de 6 soldats (Josèphe inclus), qu'on tue à chaque fois le quatrième soldat sur la gauche, et que le premier soldat soit celui qui commence à tuer, l'ordre dans lequel les soldats seront tués sera : 5 3 2 4 1 6, comme le montre la figure suivante :

Exercice 2



Exercice 2

- Faites un programme en PHP dans lequel vous créez et indiquez trois variables : le nombre de soldats, le saut pour tuer, et le premier soldat qui commencera à tuer, et qui affiche en conséquent la séquence de mise à mort.