ISOGRAD

## SUJET : INCRÉMENTER UNE VARIABLE D'UNE FONCTION SANS FAIRE APPEL AUX VARIABLES GLOBALES

SCORE : 0

Sans faire appel à une variable globale, incrémentez la variable **$mynum** au sein de la fonction **iterateNumber()** à chaque fois que cette fonction est appelée.

Après 2 minutes écoulées votre score sera progressivement réduit.

RÉPONSE DU CANDIDAT :

```php
<?php

function iterateNumber() {
    //----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------



    //----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------
    return $mynum;
}

$testnum = 0;
for ( $x=0; $x<10; $x++ ) {
    $testnum = iterateNumber();
}

?>
```

ISOGRAD

## SUJET : CRÉER UNE INTERFACE PUIS UNE CLASSE LA RESPECTANT

SCORE : 100.0

Créez une interface nommée Command avec une unique méthode : **execute()** qui demande un tableau (array) en paramètre (50 points). Créez ensuite une classe CommandImpl qui met en place cette interface (50 points).

Après 2 minutes écoulées votre score sera progressivement réduit.

RÉSULTAT AU TEST

```
Evaluation correcte : Vous avez bien défini l'interface pour la méthode execute.
Evaluation correcte : Vous avez bien défini qu'un tableau doit être passé en paramètre à la méthode execute.
Evaluation correcte : Vous avez bien défini la classe CommandImpl.

**********************************************
Message d'évaluation
**********************************************
Score à la question : 100.0 (temps de calcul 0.000)
**********************************************
```

RÉPONSE DU CANDIDAT :

```php
<?php

// Créez une interface nommée Command avec
// une unique méthode : execute() qui
// demande un tableau en paramètre.

// Créez ensuite une classe CommandImpl
// qui met en place cette l'interface
//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

interface Command
{
    public function execute(array $data);
}

class CommandImpl implements Command
{
    public function execute(array $data){

    }
}


//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------


?>
```

ISOGRAD

## SUJET : EXTRAIRE TOUTE LES INSTANCES D'UNE SOUS-CHAINE À L'INTÉRIEUR D'UNE CHAINE DE CARACTÈRES.

SCORE : 0

La fonction **matchStrings()** dont le code n'est pas terminé accepte en paramètre une chaîne de caractères multi-ligne. Complétez le code de la fonction pour qu'elle extraie toutes les sous-chaînes de la forme suivante : ***sh* suivi par *un ou plusieurs caractères alphanumériques* suivi par *sh***. Il n'est pas nécessaire de vérifier la casse. La fonction doit retourner le résultat dans un tableau contenant uniquement les sous-chaînes correspondantes.

RÉPONSE DU CANDIDAT :

```php
<?php

function matchStrings( $str ) {
    $ret = array();
    //----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

    $ret = array_filter(explode(PHP_EOL,$str), function($element){
        return preg_match('/.*(sh[a-z0-9]+sh)/', $element) === 1;
    });

    //----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------
    return $ret;
}

$test = <<<MULTI
    xx xx sh1sh xx xx
    xx sh2sh xx xx xx
    xx sh**sh xx xx x
    xx xx sh3xxxsh xx
    xx sh4shsh5sh xxx
MULTI;

$matches = matchStrings( $test );

//La variable $matches devrait contenir un tableau
//avec les valeurs suivantes :"sh1sh","sh2sh","sh3xxxsh","sh4sh","sh5sh"
//la chaine sh**sh ne doit pas être retournée car ** ne sont pas des caractères
//alphanumériques.

print_r( $matches );


?>
```

ISOGRAD

## SUJET : INTERCEPTER TOUTES LES MÉTHODES D'UN OBJET DYNAMIQUEMENT

SCORE : 0

La classe ProductDecorator est destinée à contenir une instance de la classe Product (ou un de ses descendants). Améliorez la classe ProductDecorator pour qu'elle intercepte n'importe quel appel à une de ses méthodes.

Si l'objet de classe Product qu'elle contient a une propriété du même nom que la méthode appelée, alors la valeur de la propriété de l'objet de classe Product devra devenir la valeur du paramètre passé à la méthode (70 points). S'il n'y a pas de propriété portant le même nom que la méthode ou si cette propriété n'est pas modifiable, générez une exception PHP.

Deux exemples de produits de la classe Product figurent dans le code ci-dessous, mais l'évaluation de votre code utilisera d'autres descendants de Product , vous ne pouvez donc pas supposer que vous connaissez toutes les propriétés de la classe.

RÉPONSE DU CANDIDAT :

```
<?php

class Product {
    public $price;
    function __construct( $price ) {
        $this->price = $price;
    }
}

// Attention, il peut y avoir des classes qui descendent de la classe produit
// vous ne pouvez pas supposer que l'objet contenu dans le ProductDecorator n'a que deux propriétés

class SizedProduct extends Product {
    public $size;
    function __construct( $price, $size ) {
        $this->size = $size;
        parent::__construct( $price );
    }
}

class ProductDecorator {
    private $product;
    function __construct( Product $product ) {
        $this->product=$product;
    }

//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------
}

// Voici un exemple de l'appel au ProductDecorator
$sizedproduct = new SizedProduct( 5, 3 );
$decorator = new ProductDecorator( $sizedproduct );
$decorator->size( 6 );
$decorator->price( 4 );

// les propriétés $size et $ price doivent valoir 4 et 6.
print "Nouvelle valeur de la propriété size : {$sizedproduct->size}<br />\n";
print "Nouvelle valeur de la propriété price : {$sizedproduct->price}<br />\n";



?>
```

ISOGRAD

## SUJET : HÉRITAGE DU CONTEXTE DES VARIABLES (EXPRESSION USES)

SCORE : 0

**getCallbackFunction()** est une fonction qui génère une fonction anonyme destinée à être utilisée avec **preg_replace_callback**().

Modifiez la fonction anonyme attribuée à la variable $anonfunction. Cette nouvelle fonction devra utiliser la variable $replaceString qui a été passé à la fonction **getCallbackFunction()** ainsi que la variable locale $invokeCount pour construire une chaîne de la forme *"$replaceString $invokeCount"*. La fonction devra renvoyer cette chaîne.

Il existe une méthode en PHP pour qu'une fonction anonyme puisse accéder à une variable qui a pourtant été créée avec un contexte local. La variable $invokeCount devra être incrémentée à chaque fois que la fonction anonyme est appelée. Donc si la fonction **getCallbackFunction()** est appellée avec la chaine *number* en paramètre, chaque occurence trouvée par **preg_replace_callback()** sera successivement remplacée par *number 1, number 2, number 3*.

Après 3 minutes écoulées votre score sera progressivement réduit.

RÉPONSE DU CANDIDAT :

```php
<?php


function getCallbackFunction( $replaceString ) {
    $invokecount=0;
    // remplacez la valeur de la variable $anonfunction
    // par une autre fonction anonyme
    $anonfunction = function () {};

    // cette fonction anonnyme doit se terminer par:
    //
    //    return "$replaceString $invokecount";
    //
    // où $replaceString contient la valeur du paramètre passé à getCallbackFunction( )
    // et $invokecount est incrémenté de 1 à chaque fois que la fonction anonyme est appelée.

//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

    $anonfunction = function()use($replaceString, $invokecount){
        $invokecount++;
        return '"'.$replaceString.' '.$invokecount.'"';
    };

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

    return $anonfunction;
}

$text = <<<MULTI
[item] firstly
[item] secondly
[item] thirdly
MULTI;

$text = preg_replace_callback( "/\[item\]/", getCallbackFunction( "Qu" ), $text );

// La variable $text devrait maintenant contenir:
// Qu 1 firstly
// Qu 2 secondly
// Qu 3 thirdly

?>
```

ISOGRAD

## SUJET : ARRAY_WALK() ET FONCTION PASSÉE EN PARAMÈTRE

SCORE : 100.0

Il existe une fonction en PHP qui permet d'appliquer une fonction (passée en paramètre) à tous les éléments d'un tableau.

Utilisez cette fonction sur le tableau $myarr pour transformer son contenu.

Pour cela créez une fonction anonyme dont l'interface correspond à celle exigée par la fonction PHP. Attribuez cette fonction à la variable $addpipes pour qu'elle puisse être utilisée par ailleurs (ceci est important car l'évaluation de votre code utilisera la variable $addpipes pour faire référence à votre fonction). La fonction anonyme devra ajouter le caractère '|' avant et après l'élément qui lui est passé (*one* deviendra donc |*one*|).

Le contenu du tableau $myarr sera vérifié pour voir si la transformation a bien été effectuée (*50 points*). La fonction anonyme sera utilisée sur un autre tableau pour vérifier son fonctionnement (*50 points*).

Après 3 minutes écoulées votre score sera progressivement réduit.

RÉSULTAT AU TEST

```
Array
(
[0] => |one|
[1] => |two|
[2] => |three|
)

Evaluation correcte : Le tableau a été correctement transformé.


**********************************************
Message d'évaluation
**********************************************
Score à la question : 100.0 (temps de calcul 0.000)
**********************************************
```

RÉPONSE DU CANDIDAT :

```php
<?php

$myarr = array( "one", "two", "three" );
// assignez une fonction anonyme à la variable $addpipes
// passez $addpipes en paramètre à la fonction PHP qui permet de transformer un tableau pour transformer $myarr
$addpipes = null;

//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

$addpipes = function(&$value){
    $value = "|".$value."|";
};

array_walk($myarr, $addpipes);

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------



?>
```

ISOGRAD

## SUJET : OUVRIR UN FICHIER, LIRE SON CONTENU LIGNE À LIGNE ET LE MODIFIER

SCORE : 0

Ouvrez le fichier dont le chemin d'accès est contenu dans la variable $path, stockez chaque ligne du fichier dans les valeurs du tableau $lines. Pendant que vous lisez le fichier, (ou ensuite) transformez chaque ligne pour que le premier mot (les mots sont séparés pas des espaces) soit mis en majuscules et le reste de la ligne inchangé.

Après 4 minutes écoulées votre score sera progressivement réduit.

RÉPONSE DU CANDIDAT :

```php
<?php


$path="path/to/file.txt";

$lines = array();

//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

$handle = fopen($path, 'w+');
 while(! feof($handle))  {
    $content = fgets($handle);
    $allWords = explode(' ', $content);
    $allWords[0] = strtoupper($allWords[0]);
    fwrite($handle,implode($allWords));
}
fclose($handle);

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------
foreach ($lines as $theline)
{    echo $theline."<br>";
}


?>
```

ISOGRAD

## SUJET : OPÉRATIONS SUR LES DATES (CALCUL ET FORMATTAGE)

SCORE : 0

Rédigez le code de la fonction **getPrettyDates** pour qu'il retourne un tableau (array) contenant 7 éléments. Chaque élément doit contenir une date au format suivant:

08:36:07 PM on Monday 08 July, 2012.

La première date du tableau doit être déterminée à partir du paramètre $timestamp. Il contiendra le nombre de secondes depuis la date Unix Epoch (1er Janvier 1970 00:00:00 GMT). Si $timestamp est NULL, utilisez la date actuelle.

Les 6 éléments suivants doivent être la date exactement un an avant la date de l'élément précédent.

Après 5 minutes écoulées votre score sera progressivement réduit.

RÉPONSE DU CANDIDAT :

```
<?php
date_default_timezone_set("Europe/London");


function getPrettyDates( $timestamp=null ) {
    $dates = array();

//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------

    if($timestamp === null){
    	$dates[] = (new DateTime())->format('H:i:sA on l d M, Y');
    }else{
    	$dates[] = (new DateTime($timestamp))->format('H:i:sA on l d M, Y');
    }

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION----------
    return $dates;
}

// now
print_r( getPrettyDates() );

//
print_r( getPrettyDates( mktime( 01, 0, 0, 12, 25, 2012 ) ) );



?>
```

ISOGRAD

## SUJET : ECRIRE UN SOLVEUR DE SUDOKU AVEC 2 MÉTHODE SIMPLES

SCORE : 0

Créez la classe SudokuImpl qui étend l'interface Sudoku.

L'interface Sudoku contient un tableau $values qui représente une grille de sudoku, quand une valeur est inconnue, il contient la valeur *NULL*, sinon il contient le chiffre de la grille. Les lignes et les colonnes du tableau sont indexées de *0* à *8*.

Ecrivez le code des méthodes suivantes :
- **solve_a_row()** : analyse une ligne du tableau $values et complète avec le chiffre manquant s'il ne manque qu'un seul chiffre.
- **solve_a_column()** : analyse une colonne du tableau $values et complète avec le chiffre manquant s'il ne manque qu'un seul chiffre.

Plus d'informations sur ce qu'est un Sudoku sont disponibles ici : http://fr.wikipedia.org/wiki/Sudoku

RÉPONSE DU CANDIDAT :

```php
<?php

interface SudokuInterface
{   function __construct(array $matrix);
    function sample();
    function Display();
    function setValue($row,$colum,$value);
    function getValue($row,$colum);
    function solve_a_row($row);
    function solve_a_column($column);
}

abstract class Sudoku implements SudokuInterface
{   protected $values;

    function __construct(array $matrix)
    {   $this->values = $matrix;

    }
    // Code for other methods has been hidden
    //...
    }

class SudokuImpl extends Sudoku
{
//----------NE MODIFIEZ PAS LE CODE AU DESSUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION---------

//----------NE MODIFIEZ PAS LE CODE EN DESSOUS DE CETTE LIGNE, IL SERA REINITIALISE LORS DE l'EXECUTION---------
}
$my_sudoku = new SudokuImpl(array());
$my_sudoku->sample();
writeStatement("Utilisation de solve_a_column sur le Sudoku suivant :");
$my_sudoku->Display();
for($column=0;$column<=8;$column++)
{   $my_sudoku->solve_a_column($column);
}
writeStatement("Sudoku obtenu :");
$my_sudoku->Display();
$my_sudoku->sample();
writeStatement("Utilisation de solve_a_row sur le Sudoku suivant :");
$my_sudoku->Display();
for($row=0;$row<=8;$row++)
{   $my_sudoku->solve_a_row($row);
}
writeStatement("Sudoku obtenu :");
$my_sudoku->Display();
writeStatement("Utilisation de solve_a_row()  sur le Sudoku suivant");
$my_sudoku->Display();
for($row=0;$row<=8;$row++)
{   $my_sudoku->solve_a_row($row);
}
writeStatement("Sudoku Résolu :");
$my_sudoku->Display();


?>
```