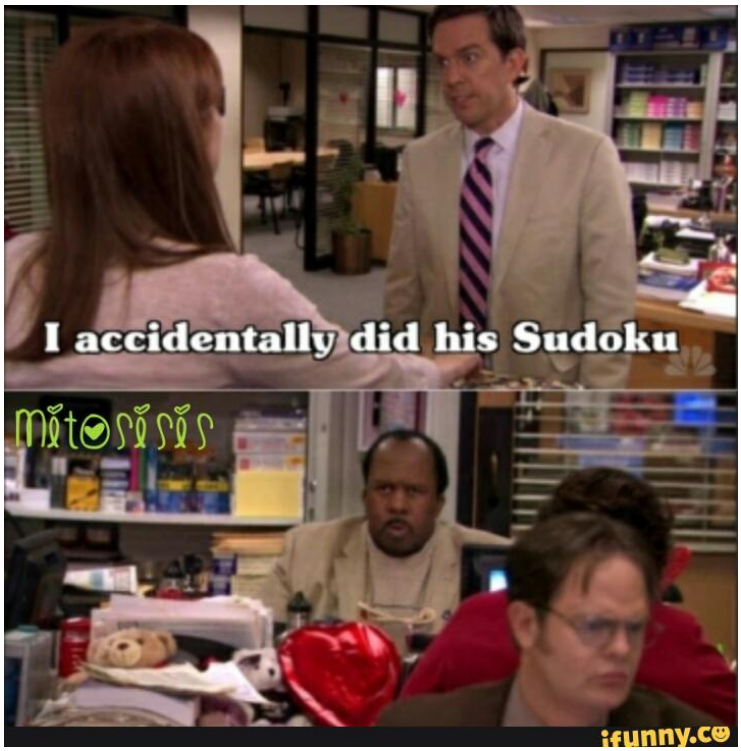
Colle W2

My Sudoku

Vous aimez jouer au Sudoku pendant les cours ? Voilà pour vous l'occasion de vous rattraper et de travailler pour la bonne cause (la vôtre) !



Description du document

Titre	My Sudoku
Date	15/02/2016
Auteur	Odin Duclos
Responsable	Sophie Viger
E-mail	pedagowac@epitech.eu
Sujet	Faire un sudoku pendant une colle
Mots-clés	Sudoku
Version	1.0

Tableau des révisions

Date	Auteur	Sections	Commentaire
15/02/2016	Odin Duclos	Toutes	Création du document.

Sommaire

Introduction

Bonjour à tous ! L'objectif de la présente colle est de réaliser dans le temps imparti un script qui résout des sudoku. Rendez votre grand-mère obsolète avec « my sudoku » !

Procédure

Au travers de trois étapes toutes plus palpitantes les unes que les autres, vous apprendrez (en vous amusant) à créer un script de résolution de sudokus (de débutant à expert !).

Je vous rappelle les règles du sudoku :

> *Les règles du sudoku sont très simples. Un sudoku classique contient neuf lignes et neuf colonnes, donc 81 cases au total.*
>
> *Le but du jeu est de remplir ces cases avec des chiffres allant de 1 à 9 en veillant toujours à ce qu'un même chiffre ne figure qu'une seule fois par colonne, une seule fois par ligne, et une seule fois par carré de neuf cases.*

Obligation : Votre script devra être écrit en PHP.

Introduction

Dans cette partie 1, vous devrez résoudre une grille de 3 cases sur 3. Sachant que votre grille ne devra contenir :

- ⇨ Qu'un chiffre identique par ligne
- ⇨ Qu'un chiffre identique par colonne
- ⇨ Qu'un chiffre identique par grille de 3*3

Votre script devra contenir les fonctions suivantes:

```
/**
 * [fill the grid of cells with values]
 * [this function calls fill_cell for each cells of the grid]
 * [call the draw function at the end of the loops]
 * @param  {$grid [array]}
 */
function fill_grid($grid) ;

/**
 * [fill a cell with a value]
 * [try each values until check_horizontal, check_vertical, and check_square return true]
 * @param  {$grid [array]}
```

```
 * @param  {$y [integer]}
 * @param  {$x [integer]}
*/
function fill_cell($grid, $y, $x);

/**
 * [check if a value is already present in a line]
 * @param  {$grid [array]}
 * @param  {$y [integer]}
 * @param  {$value [integer]}
 * @return {$is_present [boolean]}
 */
function check_horizontal($grid, $y, $value);

/**
 * [check if a value is already present in a column]
 * @param  {$grid [array]}
 * @param  {$x [integer]}
 * @param  {$value [integer]}
 * @return {$is_present [boolean]}
 */
function check_vertical($grid, $x, $value);

/**
 * [check if a value is already present in his square of 3*3]
 * @param  {$grid [array]}
 * @param  {$y [integer]}
 * @param  {$x [integer]}
 * @param  {$value [integer]}
 * @return {$is_present [boolean]}
 */
function check_square($grid, $y, $x, $value);

/**
 * [draw the grid]
 */
function draw($grid);
```

Exemple de grilles :

8		1
	7	

5		
2		
3		

1	3	
2		

Exemple de résultats :

8	2	1
3	4	5
6	7	9

5	1	4
2	6	7
3	8	9

1	3	4
2	5	6
7	8	9

Tableau de départ :

```
$grid = array(
        array(0, 0, 4),
        array(0, 0, 0),
        array(1, 0, 9)
);
```

Un premier essai

Dans cette deuxième partie, vous devrez réutiliser les fonctions de l'étape précédente afin de remplir un sudoku complet de 9 cases sur 9.

L'utilisation du modulo vous sera nécessaire afin de déterminer à quel carré de 3*3 appartient une cellule.

Exemple de grille :

		4			6			7
			1			2		5
1		9	5		7	4		8
4							2	
		2		6			9	3
9			2		3			6
	8	1					5	
7		5	3		2			4
6			9			1		

Exemple de résultat :

2	3	4	8	9	6	⊗	1	7
8	6	7	1	3	4	2	⊗	5
1	⊗	9	5	2	7	4	3	8
4	1	3	7	5	8	⊗	2	⊗
5	7	2	4	6	1	8	9	3
9	⊗	8	2	⊗	3	5	4	6
3	8	1	6	4	⊗	7	5	2
7	9	5	3	1	2	6	8	4
6	2	⊗	9	7	5	1	⊗	⊗

Tableau de départ :

```
$grid = array(
        array(0, 0, 4, 0, 0, 6, 0, 0, 7),
        array(0, 0, 0, 1, 0, 0, 2, 0, 5),
        array(1, 0, 9, 5, 0, 7, 4, 0, 8),
        array(4, 0, 0, 0, 0, 0, 0, 2, 0),
        array(0, 0, 2, 0, 6, 0, 0, 9, 3),
        array(9, 0, 0, 2, 0, 3, 0, 0, 6),
        array(0, 8, 1, 0, 0, 0, 0, 5, 0),
        array(7, 0, 5, 3, 0, 2, 0, 0, 4),
        array(6, 0, 0, 9, 0, 0, 1, 0, 0),
);
```

Echec et mat

Dans cette partie, vous allez vous inspirer de vos erreurs et enfin finaliser my_sudoku ! Comme vous avez pu le voir, l'algorithme est incapable de remplir certaines cases, car il ne peut prévoir ce qui se passera dans les cases futures au moment de remplir la case présente.

Pour cela, apprenez à votre script à voir toutes les combinaisons possibles avant de remplir une cellule. C'est ce qu'on appelle une résolution par arborescence.

L'objectif est :

- Pour chaque valeur possible testée pour une cellule n dans fill_cell, il faudra rappeler la fonction fill_cell en lui passant la cellule n+1.
- Si la grille a été parcourue entièrement, appelle la fonction draw.

Prototypes :

```
/**
 * [call fill_cell with x = 0 and y = 0]
 * @param  {$grid [array]}
 */
function fill_grid($grid);

/**
 * [fill a cell with a value]
 * [for each possible value of a cell n, call fill_cell with cell n+1]
 * [if x and y are bigger than the size of the grid, print the grind]
 * @param  {$grid [array]}
 * @param  {$y [integer]}
 * @param  {$x [integer]}
 * @return {$is_filled [boolean]}
 */
function fill_cell($grid, $y, $x);
```

```
/**
 * [get the next valide coordiantes of the grid]
 * @param  {$grid [array]}
 * @param  {$y [integer]}
 * @param  {$x [integer]}
 * @return {$coordinates [array]} [array with x and y coordinates]
 */
function get_next_coords($grid, $y, $x);
```

Obligation : Seul un usage raisonné et chirurgical de la récursivité pourra vous permettre de finaliser « my sudoku » et de lui donner la vie qu'il mérite.

Résultat attendu (ça et seulement ça) :

3	5	4	8	2	6	9	1	7
8	6	7	1	4	9	2	3	5
1	2	9	5	3	7	4	6	8
4	3	6	7	9	8	5	2	1
5	7	2	4	6	1	8	9	3
9	1	8	2	5	3	7	4	6
2	8	1	6	7	4	3	5	9
7	9	5	3	1	2	6	8	4
6	4	3	9	8	5	1	7	2

5	2	4	8	3	6	9	1	7
3	7	8	1	4	9	2	6	5
1	6	9	5	2	7	4	3	8
4	3	6	7	9	5	8	2	1
8	5	2	4	6	1	7	9	3
9	1	7	2	8	3	5	4	6
2	8	1	6	7	4	3	5	9
7	9	5	3	1	2	6	8	4
6	4	3	9	5	8	1	7	2

5	2	4	8	3	6	9	1	7
3	7	8	1	4	9	2	6	5
1	6	9	5	2	7	4	3	8
4	3	6	7	9	8	5	2	1
8	5	2	4	6	1	7	9	3
9	1	7	2	5	3	8	4	6
2	8	1	6	7	4	3	5	9
7	9	5	3	1	2	6	8	4
6	4	3	9	8	5	1	7	2