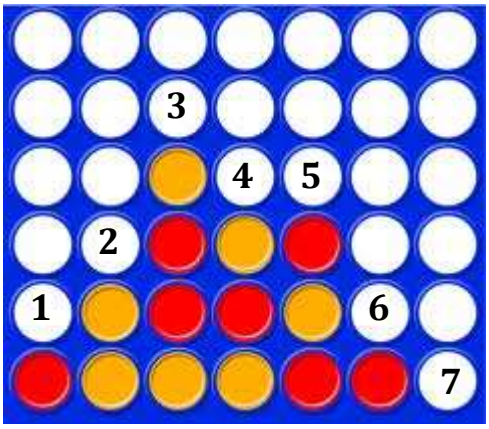
Le Puissance 4

<u>Présentation du sujet</u> :

Le puissance 4 est un jeu qui se joue à deux joueurs. Les joueurs jouent alternativement des pions par le haut de la grille. Les pions tombent alors au plus bas possible de la colonne choisie. Le premier joueur qui réussit l'alignement horizontalement ou verticalement de 4 de ses pions est vainqueur. Si la grille est pleine, sans qu'aucun des deux joueurs n'ait réussi l'alignement de quatre de ses pions, la partie est nulle. **<u>Pour simplifier l'étude, nous ne tiendrons pas compte des alignements en diagonale.</u>**



Dans l'exemple ci-contre, la prochaine case qui sera jouée doit-être choisie par le joueur parmi les 7 cases numérotées de 1 à 7.

Aucune des 7 colonnes n'étant pleine, le joueur a donc 7 choix.

On se propose d'écrire un algorithme qui permet l'affichage de la grille à chaque étape du jeu, contrôle les saisies des joueurs et vérifie si la partie est finie. C'est-à-dire si un des deux joueurs a gagné ou si la partie est nulle.

On utilise pour "stocker" la grille de jeu une matrice 6 lignes, 7 colonnes de lettres (**'X'** pour les pions rouges et **'O'** pour les jaunes) qui sera affichée à chaque étape du jeu. On utilise aussi un tableau de numéros de '1' à '7' désignant les colonnes à jouer, qui sera affiché sous la matrice à chaque étape du jeu afin de faciliter le choix des joueurs.

	0	1	2	3	4	5	6
0							
1							
2			'O'				
3			'X'	'O'	'X'		
4		'O'	'X'	'X'	'O'		
5	'X'	'O'	'O'	'O'	'X'	'X'	

0	1	2	3	4	5	6
'1'	'2'	'3'	'4'	'5'	'6'	'7'

Au fur et à mesure de la partie, la grille sera "remplie" alternativement avec des **'X'** ou des **'O'**, jusqu'à ce qu'un joueur gagne ou que la grille soit pleine. (Les **'X'** commencent.)

PARTIE A :

Utilisation d'un ordinateur interdite. Les réponses sont à rédiger sur les feuilles réponses

Durée : 30 minutes

1) On considère l'algorithme suivant:

```
Variables:
        compt, a : entiers
        tab: tableau de 7 chaines de caractères
début:
        tab ← [' _ ',' _ ',' _ ',' _ ',' _ ',' _ ',' _ ']
        compt ←0
        a ← 0
        tant que compt ≤ 3:
                a ← compt % 2          #compt % 2 renvoie le reste de la division de compt  par 2#
                si  a = 0 :
                        alors:
                                tab[compt] ← 'X'
                        sinon:
                                tab[compt] ← 'O'
                compt ←compt +1
        fin de tant que
fin
```

a) Déterminer le contenu des variables **a** et **tab** après le premier "passage" dans la boucle tant que, après le deuxième "passage" dans la boucle tant que, ainsi qu'à la fin de l'algorithme. Pour cela compléter le tableau de la page réponse 6.

b) Décrire le rôle de l'algorithme.

c) Corriger l'algorithme afin que la variable **tab** contienne à la fin du tant que la valeur suivante: **tab = ['X','O','X','O','X','O','X'].**

Corriger directement l'algorithme sur la feuille réponse, sans le réécrire.

2) On considère les variables **plat** et **numero** décrite ci-dessous.

plat = [['_', '_', '_', '_', '_', '_', '_'],['_', '_', '_', '_', '_', '_', '_'],['_', '_', '**O**', '_', '_', '_', '_'],['_', '_', '**X**', '**O**', '**X**', '_', '_'],['_', '**O**', '**X**', '**X**' , '**O**', '_', '_'],['**X**', '**O**', '**O**','**O**' , '**X**', '**X**', '_']]

numero = ['1','2','3','4','5','6','7']

a) Quelle est l'affichage correspondant à l'instruction suivante : afficher(plat[4])

b) Corriger et compléter la procédure **affplat,** sur la feuille réponse, **sans la réécrire**, afin qu'elle permette l'affichage suivant:

['_', '_', '_', '_', '_', '_', '_']
['_', '_', '_', '_', '_', '_', '_']
['_', '_', '**O**', '_', '_', '_', '_']
['_', '_', '**X**', '**O**', '**X**', '_', '_']
['_', '**O**', '**X**', '**X**' , '**O**', '_', '_']
['**X**', '**O**', '**O**','**O**' , '**X**', '**X**', '_']

['1', '2', '3', '4', '5', '6', '7']

```
procédure affplat():
        i : entier
        début procédure
                pour  i de 0 à 3:
                        afficher(plat[i])
                fin de pour
        fin de procédure
```

PARTIE B :
Durée : 30 minutes

Pour cette partie, une implémentation de vos solutions algorithmiques est demandée. Pour cela vous devez utiliser un ordinateur.

Toutefois, en cas de difficulté d'implémentation, un algorithme langage naturel sera pris en compte dans l'évaluation. Dans ce cas seulement, vous rédigerez votre algorithme sur la « feuille réponse » de la page 8.

Vous enregistrerez votre travail sur une clé USB dans un dossier à votre nom.

Pensez à sauvegarder régulièrement votre travail.

Vous trouverez sur le bureau de l'ordinateur qui vous a été désigné un dossier à votre nom dans lequel se trouve un fichier intitulé : Puissance 4 à compléter.py.

Commencez par enregistrer ce fichier sous **la clef USB**

Vous disposez également d'un code Python incomplet contenant:

- *Un début d'algorithme permettant de remplir la première ligne de la grille.*
- *La procédure* **affplat** *décrite dans les pages précédentes*
- *Une fonction* ***gagne(i , j)*** *qui retourne la valeur booléenne vraie si la case* **plat[i][j]** *qui vient d'être jouée, permet de gagner. (Comme indiqué en introduction,* ***pour simplifier l'étude, nous n'avons pas tenu compte des alignements en diagonale*****.**)

La matrice correspondant à la grille du jeu puissance 4 est remplie par empilement des jetons. C'est à dire qu'une case ne peut être modifiée tant que les case situées en dessous d'elle ne l'ont pas été.

Ecrire un code qui demande aux joueurs de choisir une colonne et qui permet le remplissage de la matrice **plat** alternativement avec des **'X'** ou des **'O'**, ainsi que son affichage, jusqu'à ce qu'un des deux joueurs gagne ou que la grille soit entièrement remplie.

***Vous utiliserez la procédure* affplat *et la fonction* gagne.**

FEUILLE REPONSE DE LA PARTIE A

1) a) Contenu des variables **a** et **tab** après le premier, après le deuxième "passage" dans la boucle " tant que", ainsi qu'à la fin de l'algorithme en complétant le tableau suivant:

Nom des variables	**compt**	**a**	**tab**
Valeurs initiales	0	0	[' _ ',' _ ',' _ ',' _ ',' _ ',' _ ',' _ ']
Après 1er Passage dans la boucle "tant que"	1		
Après 2eme Passage dans boucle "tant que"	2		
...			
Valeurs finales			

b) Rôle de l'algorithme.

c) Correction de l'algorithme

```
Variables:
        compt, a : entiers
        tab: tableau de 7 chaines de caractères

début:
        tab ← [' _ ',' _ ',' _ ',' _ ',' _ ',' _ ',' _ ']
        compt ←0
        a ← 0

        tant que compt ≤ 3:

                a ← compt % 2           #compt % 2 renvoie le reste de la division de compt  par 2#

                si  a = 0 :

                        alors:

                                tab[compt] ← 'X'

                        sinon:

                                tab[compt] ← 'O'

                compt ←compt +1

        fin de tant que

fin
```

2) a) Affichage correspondant à l'instruction : afficher(plat[4]):

b) Correction de la procédure **affplat:**

```
procédure affplat():
        i : entier
        début procédure

                pour  i de 0 à 3:

                        afficher(plat[i])

                fin de pour

        fin de procédure
```