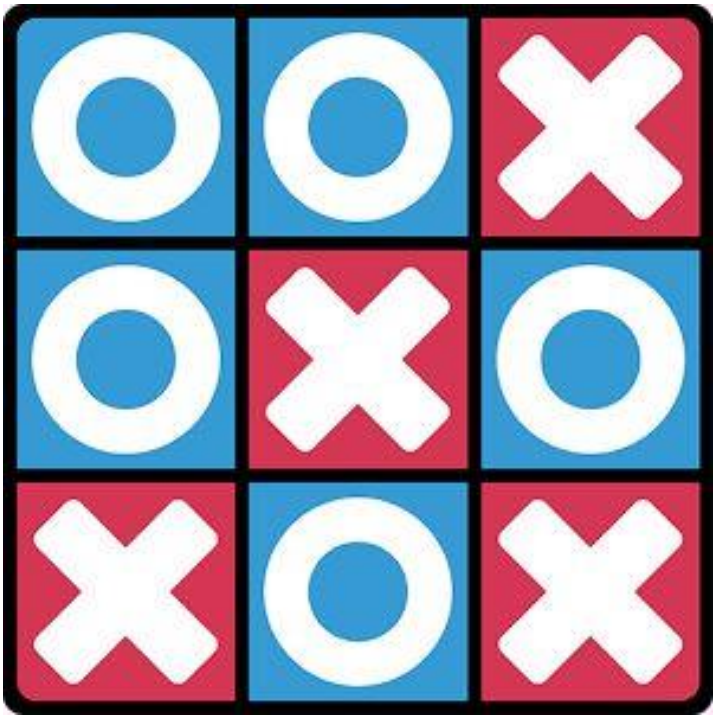
Coding Club

TicTacToe



Réalisation d'un morpion en Python avec Pygame

Réalisé par: ***Nathan et Timothée (Promo 2021)***

1. Introduction

L'objectif d'aujourd'hui sera de réaliser le célèbre jeu vidéo TicTacToe (le morpion en français).

Pour réaliser ce jeu en Python nous allons d'abord le faire en ligne de commande (donc sur le terminal) puis une fois la logique du jeu codée nous allons l'afficher grâce à Pygame.

C'est quoi Python ?

Python est un langage de programmation. Tout comme les êtres humains utilisent plusieurs langues, en informatique il existe plusieurs langages pour communiquer avec sa machine, et le Python en fait partie.

Quel est l'avantage d'utiliser Python ? C'est que c'est un langage rapide à prendre en main, en effet, il suffit d'écrire ce que tu veux faire en anglais et en principe ça devrait fonctionner.

Il comporte néanmoins certaines particularités comparé aux autres langages, comme le fait qu'il est très à cheval sur l'**indentation** (le nombre d'espace – tabulation) et qu'il ne nécessite pas de **compilation**. C'est un langage **interprété**. On n'est pas obligé de convertir le fichier qui contient ton code en un fichier exécutable comme les « .exe ».

C'est quoi Pygame ?

Pygame est une bibliothèque. En informatique une bibliothèque est un ensemble de fonctions, qui, dans notre cas nous permettrons de créer une fenêtre, d'afficher une image à tel endroit, etc…

Si nous n'avions pas cette bibliothèque, il nous faudrait écrire directement en binaire (succession de 0 et de 1) que nous voudrions tel pixel à tel endroit dans notre programme, ce qui serait beaucoup plus fatiguant. Utiliser Pygame vas donc nous simplifier la tâche !

1.1 Environnement de travail

Fais un dossier **My Game** sur le bureau de ton Ordinateur, c'est dans ce fichier que tu vas mettre tout ce dont tu auras besoin pour ton jeu.

Commence par créer un fichier Python qui te permettra d'écrire ton code (un fichier Python est un fichier dont l'extension est « .py », si tu vois toujours l'icône d'un fichier texte (.txt) ou que tu n'arrives pas à créer le fichier, **demande à un Cobra**, n'aies pas honte, ils sont gentils), évidemment tu peux nommer ton fichier python comme tu le souhaites tant qu'il finit bien par « .py ».

Si à un moment dans la journée ton jeu a besoin d'utiliser des images ou des musiques, choisis-les sur Internet puis enregistre les, et n'oublie pas de les mettre dans ton dossier **My Game**.

Pour le moment, ton dossier « My Game » ne comporte que ton fichier Python ! Mais prend garde à comment tu comptes remplir ce dossier, pour éviter d'avoir toutes tes futurs images / musiques / etc… en vrac dans ton dossier, commence dès maintenant à créer un dossier « Image », « Music » où tu mettras toute tes images, musiques, …

Pour modifier votre fichier Python, on te recommande d'utiliser « IDLE », il te permettra d'écrire dans ton fichier tout en lançant ton programme Python.

Pour l'ouvrir, fait un **Clic droit** sur ton fichier Python -> **Edit with IDLE** puis tu peux éditer ton fichier ! Pour le lancer tu n'auras qu'a appuyer sur **Run** -> **Run Module** (en haut d'IDLE).

Il existe plusieurs autres « éditeur de texte », si tu es plus familier avec Notepad++, Sublime Text, Visual Code voire même le Bloc Note de Windows, tu peux les utiliser.

2. Créer son jeu

2.1 La board du morpion

Pour commencer on veut avoir un board sur lequel placer nos croix et rond. En Python pour cela nous pouvons créer un tableau, dans lequel on peut placer des ‘x’ ou ‘o’. Pour accéder aux valeurs du tableau on utilise des crochets [ ]. Par exemple :

```
# Ici on crée un tableau de 3 par 3. (Ne cherchez pas à comprendre la syntaxe)
board = [[' '] * 3 for _ in range(0, 3)]

# Pour accéder au tableau il faut utiliser les [ ]
# le premier [ ] c’est les ordonnées (y) et le second c’est les abscisses (x)
# Voici comment insérer une croix en haut à gauche
board[0][0] = 'x'

# Voici comment insérer un rond en haut à droite
board[0][2] = 'o'

# Voici comment insérer une croix en bas à gauche
board[2][0] = 'x'

# Maintenant pour visualiser ton tableau il faut l’afficher sur la console !
# Pour ce faire un va utiliser une boucle, un for (« pour » en anglais)
for line in board:
    print(line)

# Donc cela veut dire : « Pour toute les lignes dans board, on affiche la ligne »
```

Pour bien comprendre le fonctionnement des tableaux, il est encouragé de tester ton programme en changeant les valeurs puis en le lançant !

Si tu lances ton programme, tu devrais avoir cet affichage :

```
Timocafé - TicTacToe $ python3 my_game.py
['x', ' ', 'o']
[' ', ' ', ' ']
['x', ' ', ' ']
Timocafé - TicTacToe $
```

Que se passe-t-il si tu mets des coordonnées plus grandes que 2 ? Et si tu mets des lettres à la place ?

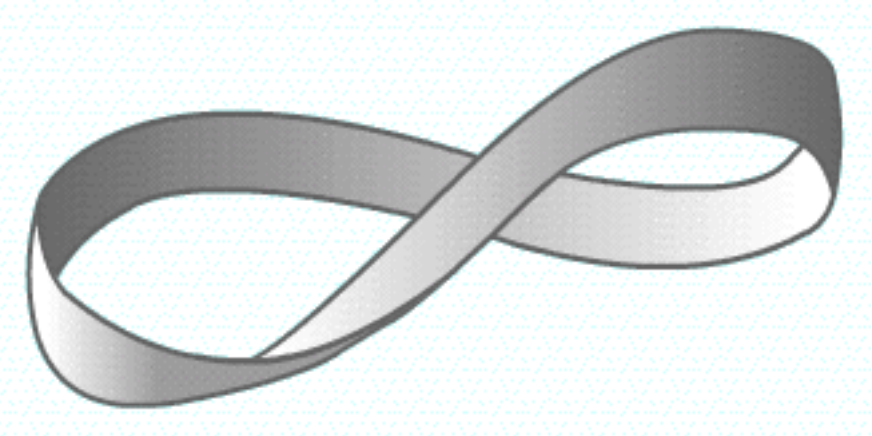
2.2 La boucle de jeu

Pour faire notre jeu, il faut une boucle qui continue tant que le jeu n'est pas terminé. Pour ce faire nous allons utiliser une variable « end » qui ne sera égale à « True » que quand le jeu est terminé.

Comme le morpion est un jeu au tour par tour, nous avons aussi besoin de savoir quand est le tour de quel joueur. Pour ce faire nous allons utiliser une autre variable !

```
# La variable end qui est un boolean (soit égale à « Vrai » soit à « Faux »)
end = False

# La variable turn qui est une string (qui est égale à une phrase)
# Elle est toujours entre quotes " "
turn = "Player1"

# Le tableau précédemment créé :
board = [[' '] * 3 for _ in range(0, 3)]

# L'affichage du tableau
for line in board:
    print(line)

# Le while veut dire « Tant que », donc on fait ce qu'il y a dans la boucle tant que
# le end n'est pas « True » (Vrai)
while not end:
    pass
# « pass » est un mot clé comme « while » et « for » qui permet de … ne rien faire !
```

Pour l'instant si tu copie/colle ce code puis que tu l'executes cela va juste faire une boucle infinie car il n'y a rien à faire dans le while.



2.3 Obtenir les entrées utilisateurs

Maintenant qu'on a une boucle, il faut qu'on puisse jouer !

Pour cela nous allons utiliser la fonction input() de Python. Elle permet demander à l'utilisateur de rentrer un chiffre. Le seul problème c'est que la fonction input() nous donne une « string » alors que nous voulons un « int » (un chiffre). On va donc le convertir en chiffre grâce à une fonction qui s'appelle int() !

Une fois que tu as demandé la position du x et la position du y, il va falloir placer soit un 'x' soit un 'o' en fonction du joueur ! Nous allons donc utiliser un if (« si » en français).

Si tu n'as pas tout compris, tu peux toujours demander à un Cobra de t'expliquer ! (Ils sont gentils et ne mordent pas)

```
# A partir de maintenant on va juste te donner des bouts de code.
# Donc par exemple dans ce cas il faut changer juste la partie de while
while not end:
    # La fonction « print » permet d'afficher sur la console
    # donc nous allons afficher qui doit jouer
    print(turn + " turn :")

    # Nous demandons à l'utilisateur de donner une position 'x' et 'y'
    # Puis on les sauvegarde dans des variables
    pos_x = int(input("Please enter x position: "))
    pos_y = int(input("Please enter y position: "))

   # « Si c'est au tour du Joueur 1 »
    if turn == "Player1":
        board[pos_y][pos_x] = 'x'
        # Comme le Player1 a joué il faut que ce soit au tour du Player2
        turn = "Player2"
    # Le mot clé « else » veut dire sinon
    else:
        board[pos_y][pos_x] = 'o'
        # Comme le Player2 a joué il faut que ce soit au tour du Player1
        turn = "Player1"
    # On print le board a chaque tour de boucle pour voir les changements
    for line in board:
        print(line)
```

Que se passe-t-il si tu donnes des positions incorrectes ? Comment empêcher le Joueur2 de jouer au même endroit que le Joueur1 ?

2.4 Les fonctions

Pour éviter que le code ne devienne trop compliqué à lire nous pouvons faire des fonctions !

En effet une fonction désigne un « sous-programme » permettant d'effectuer des actions répétitives, elles permettent aussi de structurer le code. A chaque fois qu'on appelle une fonction le code qu'il y a dedans s'exécute. La fonction permet aussi de renvoyer une valeur qui peut être un int (chiffre) ou une string (phrase).

En Python, on écrit une fonction grâce à def. Tu peux trouver le même code que tu as écrit mais avec des fonctions ci-dessous :

```
# Cette fonction permet d'afficher le board
def draw_board(board):
    for line in board:
        print(line)

# Cette fonction permet de demander à l'utilisateur les coordonnées du jeu
def ask_position(board, turn):
    print(turn + " turn :")
    pos_x = int(input("Please enter x position: "))
    pos_y = int(input("Please enter y position: "))
    if turn == "Player1":
        board[pos_y][pos_x] = 'x'
        turn = "Player2"
    else:
        board[pos_y][pos_x] = 'o'
        turn = "Player1"
    # On renvoi une string contenant le prochain player qui doit jouer
    return turn

end = False
turn = "Player1"
board = [[' '] * 3 for _ in range(0, 3)]

while not end:
    draw_board(board)
    # On récupère le prochain player qui doit jouer
    turn = ask_position(board, turn)
```

Tu as peut-être remarqué que l'on donne les variables board et turn à nos fonctions. En effet si on ne passe pas en « arguments » ces variables elles ne seront pas définies dans la fonction. Et cela donnera des erreurs comme :
« UnboundLocalError: local variable referenced before assigment »

2.5 La victoire ! (ou défaite :’( )

Pour te simplifier la tâche on va te donner une fonction qui gère la fin de la partie. Saches que ce code n’est ni optimisé ni très beau à regarder. Il te permet par contre de comprendre comment on vérifie que la partie est bel et bien terminée.

```
# Tu peux insérer cette fonction en dessous de la fonction « ask_position() »

def end_game(board):
    # On utilise un for pour p soit un ‘x’ puis un ‘o’
    # De cette façon on vérifie bien la fin pour les deux joueurs
    for p in ['x', 'o']:
        if board[0][0] == p and board[0][1] == p and board[0][2] == p:
            return True
        if board[1][0] == p and board[1][1] == p and board[1][2] == p:
            return True
        if board[2][0] == p and board[2][1] == p and board[2][2] == p:
            return True
        if board[0][0] == p and board[1][0] == p and board[2][0] == p:
            return True
        if board[0][1] == p and board[1][1] == p and board[2][1] == p:
            return True
        if board[0][2] == p and board[1][2] == p and board[2][2] == p:
            return True
        if board[0][0] == p and board[1][1] == p and board[2][2] == p:
            return True
        if board[0][2] == p and board[1][1] == p and board[2][0] == p:
            return True

    # La fonctionne renvoi donc Vrai si c’est la fin sinon elle renvoi Faux
    return False
```

Maintenant que tu as cette fonction il faut l’appeler dans ta boucle while !

N’oublie pas que la variable « end » doit être égale à ta fonction sinon ta fonction ne sert à rien.

Si tu n’arrives vraiment pas à appeler ta fonction end_game() demande à un Cobra, ils seront plus que content de t’aider !

N’oublie pas que cette fonction n’est pas du tout optimisée. Essaye de l’améliorer !

Un autre petit plus serait d’afficher quel est le joueur qui a gagné.

3. Afficher son jeu avec Pygame

3.1 Ouvrir sa première fenêtre

Maintenant on arrive à la question que vous vous posez tous : Comment ouvre-t-on une fenêtre avec Pygame?

Nous avons besoin d'ouvrir une fenêtre pour notre jeu car on ne peut pas coller et faire bouger des images ou du texte comme ça sur notre machine, il nous faut au préalable ouvrir une fenêtre où l'on pourra y faire tout ce que l'on veut !

Pour ouvrir une fenêtre copie et colle ce code dans un nouveau fichier Python.

```
# Import permet d'importer une bibliothèque pour utiliser ses fonctions, ici Pygame.
import pygame

# On initialise Pygame, chaque fonction de Pygame commence par « pygame. »
pygame.init()

# On fait une variable window de taille 600x700 pixels, ce seras notre fenêtre.
window = pygame.display.set_mode((600, 700))
# On assigne la phrase « TicTacToe » à notre fenêtre.
pygame.display.set_caption("TicTacToe")
# On fait une variable qui s'appelle « leave », et on lui assigne la valeur « False ».
leave = False

# While signifie « tant que », c'est une boucle, cela veut dire que tant que « leave »
# est égal à False, tout ce qui se trouve dans la boucle while va se ré-exécuté.
while not leave:
    # Ici Pygame cherche les évènements qui ont eu lieu lors de ce tour de boucle
    for event in pygame.event.get():
        # Si l'évènement est pygame.QUIT on met la variable « leave » a True
        if event.type == pygame.QUIT:
            leave = True

# Si on arrive ici c'est que la variable « leave » a été mise sur True
# On quitte Pygame et on termine le programme.
pygame.quit()
quit()
```

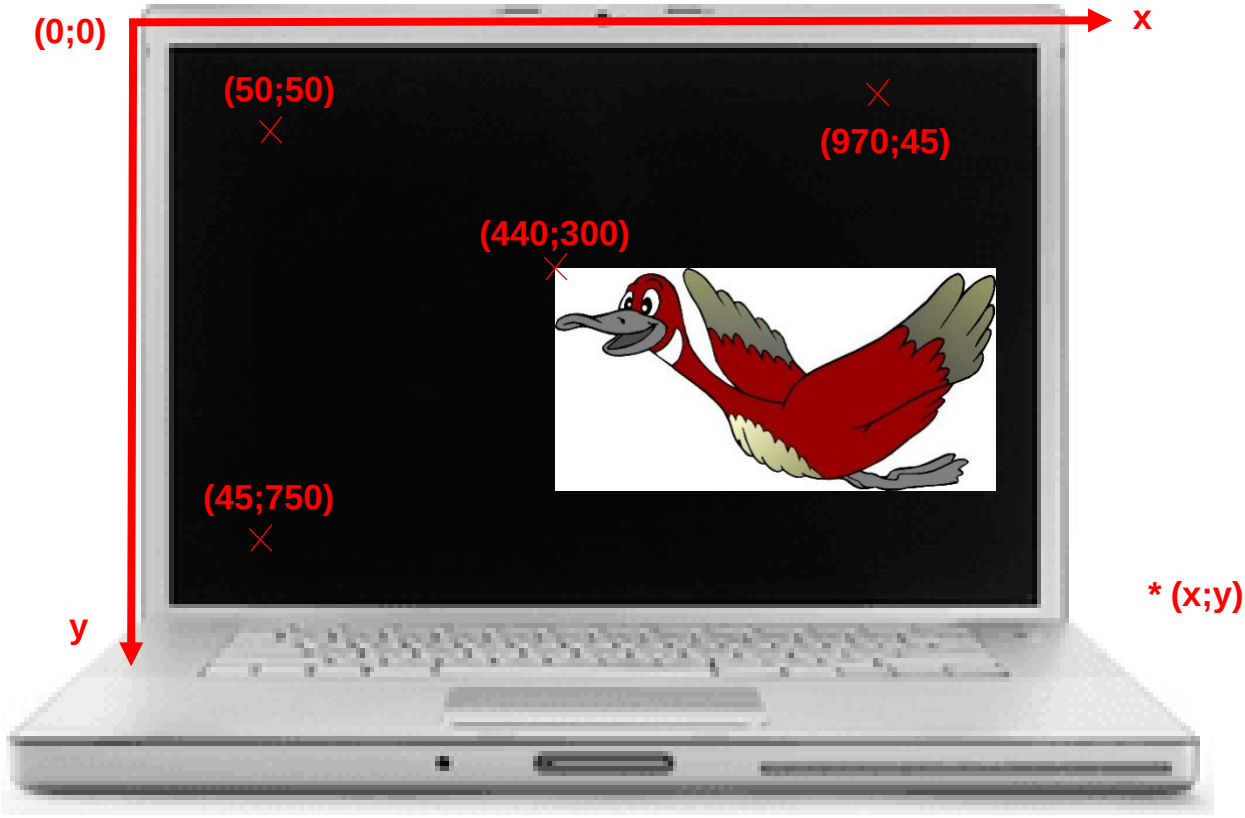
Attention ! Avec ce code tu devrais avoir seulement une fenêtre vide (noir ou blanche) qui ne fait rien d'autre que de se fermer quand tu cliques sur la croix.

3.2 Les Images

Nous voilà enfin à la partie que vous attendiez tous ! Ainsi que la partie la plus intéressante de Pygame, j'ai nommé : La gestion des images !

Avant d'en dire plus, voici un petit récapitulatif du système de coordonnées en informatique:

Vois-tu les graphs qu'on utilise pour représenter les fonctions en cours de maths ? (Si tu ne vois pas ou t'en souviens pas t'en fais pas, appelle un Cobra, **il sera ravi de t'éclairer**). Eh bien en informatique on utilise le même système de coordonnées, si ce n'est que l'axe des abscisses (y) est inversé ! Voici quelques coordonnées de point sur un écran d'ordinateur :



Attention ! En informatique les coordonnées d'une image ne sont représentées que par 1 seul point: celui le plus en haut à gauche de celle-ci (Soit (440 ; 300) dans notre exemple).

Ce qui signifie que pour savoir si un pixel se trouve dans la zone de notre image on doit vérifier si le point en x est entre la position en x de l'image et sa position en x + la taille de l'image en x (et pareille pour l'axe des abscisses) (Appelle un Cobra pour plus de détails si tu ne comprends pas)

Pour afficher une image sur ton écran, il te faudra passer par 2 étapes, la première sera de **blit** (Signifie « Apparaitre) ton image sur ta fenêtre, puis d'**actualiser** celle-ci.

Pour bien comprendre, imagine que ta fenêtre est une scène de théâtre. Quand tu blit quelque chose sur ta fenêtre, tu places quelque chose sur ta scène, mais pour que le public voit les changements, il te faudra ouvrir les rideaux (l'actualisation).

Si toi aussi t'as pas compris, appelle un cobra ou clic ici \'o'/

Entre chaque actualisation, l'image affichée restera la même, donc chaque blit a l'écran ne sera affiché qu'une fois la fenêtre actualisée, mais attention ! L'ancienne image seras toujours présente, donc pense bien à effacer l'ancienne avant de réafficher ton image (dans le cas d'un déplacement). Ça te permettra d'éviter l'effet de « trainé ».

Si tu blit plusieurs fois avant d'update, pense bien que si tu mets une image sur la position d'une autre, c'est la dernière blit qui sera par-dessus !

```
import pygame

pygame.init()

window = pygame.display.set_mode((600, 700))
pygame.display.set_caption("TicTacToe")
# Créer une variable contenant l'image Background.png (situé dans
# notre dossier « My Game/Image »).
background = pygame.image.load("Image/Background.png")
leave = False

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    # Remplie le fond de l'écran avec du gris
    window.fill([127, 114, 100])
    # Blit "background" en position (0;0).
    window.blit(background, (0, 0))
    # Rafraichis la fenêtre.
    pygame.display.update()

pygame.quit()
quit()
```

3.3 Afficher le TicTacToe

Maintenant que tu sais afficher une image on va pouvoir afficher tout le board.

Tout d'abord copie/colle tes fonctions draw_board(), ask_position() et end_game() entre le import pygame et le pygame.init(). Ensuite il faut aussi que tu instancies les variables end, turn et board comme nous avons fait dans la partie 2.

Nous allons maintenant changer la fonction draw_board() pour qu'elle puisse changer ce qu'on dessine avec Pygame.

```
def draw_board(board, window):
    # Fonctions qui était dans le main
    window.fill([127, 114, 100])
    window.blit(background, (0, 0))

   # On parcours toute les cases de la board
    for y in range(0, 3):
        for x in range(0, 3):
            # Si il y a un x on afficher l'image « cross » sinon c'est l'image « round »
            if board[y][x] == 'x':
                window.blit(cross,  (x * 200 + 50, y * 200 + 50))
            elif board[y][x] == 'y':
                window.blit(round,  (x * 200 + 50, y * 200 + 50))

    # Rafraichis la fenêtre.
    pygame.display.update()
```

Comme tu l'as remarqué parce que tu es observateur on utilise au dessus les images cross et round, il faut donc les instancier :

```
# Tu peux les mettre juste en dessous du background
round = pygame.image.load("Image/Round.png")
cross = pygame.image.load("Image/Cross.png")
```

N'oublie pas de mettre à jour ce que tu as dans ta boucle while.

Il faut en effet appeler les fonctions !

3.4 La souris !

Le seul truc qui nous manque maintenant c'est d'avoir le clic de la souris.

Pour ce faire nous allons utiliser les fonctions get_pos() et get_pressed() de la library pygame.

```
def ask_position(board, turn):
    # get_pressed() permet de savoir si on appuie sur la souris
    if pygame.mouse.get_pressed()[0]:

        # get_pos() permet d'obtenir la position de la souris
        mouse_x, mouse_y = pygame.mouse.get_pos()

        # Il faut diviser par 200 pour avoir la position sur le board
        pos_x = int(mouse_x / 200)
        pos_y = int(mouse_y / 200)

        if turn == "Player1":
            board[pos_y][pos_x] = 'x'
            turn = "Player2"
        else:
            board[pos_y][pos_x] = 'o'
            turn = "Player1"

    # On renvoi une string contenant le prochain player qui doit jouer
    return turn
```

3.5 Ajout de texte

Tu peux aussi rajouter du texte avec les fonctions suivantes :

```
# On déclare la font qui va etre utiliser pour tout les texts
# Tu peux le mettre avec les image.load()
font = pygame.font.SysFont("arial", 32)

# Cette function permet de créer ton objet de text pour l'afficher après avec blit()
# Tu peux les placer dans draw_board()
text = font.render("My text", 3, (255, 255, 255))
window.blit(text, (20, 630))
```

Félicitation tu es venu à bout de ce sujet. Tu sais maintenant comment créer des jeux-vidéo en Python avec PyGame.

Tu peux donc avec le temps qu'il te reste développer ton propre jeu ou améliorer celui que tu viens de créer, la seule limite c'est ton imagination.