**Pygame, un module de Python pour jouer**

Lors du lancement du logiciel Python, seules les fonctions principales sont chargées en mémoire. Les autres sont rangées dans des fichiers appelés librairies. Si on souhaite utiliser une fonction définie dans une librairie, il faut la charger en mémoire.

Par exemple, si vous souhaitez faire des mathématiques (si, si!), vous pouvez avoir besoin de la racine carrée. Vous débutez votre programme par

from math import sqrt #on n'importe que la racine carrée, définie dans la librairie math

ou

from math import \* #on importe toute la librairie de fonctions mathématiques

Certaines bibliothèques (library en anglais) sont éditées par les développeurs de Python (c'est le cas de math, random…) et installées automatiquement avec le logiciel. Vous pouvez en définir de nouvelles. Vous pouvez en utiliser d'autres développées par des gens ne travaillant pas pour Python. C'est le cas de Pygame. Vous devez vous assurer que la licence de la bibliothèque est conforme à l'utilisation que vous en ferez. Pygame est une bibliothèque libre de droit donc éprouvez de la reconnaissance pour ses développeurs qui vous font partager le fruit de leur travail gratuitement.

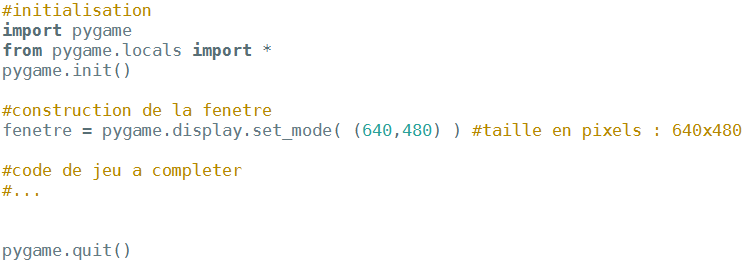
**Etape 0: télécharger pygame (il n'y en a normalement pas besoin au lycée)**

* Vérifier qu'on a les droits d'installation
* se rendre sur le site : <http://pygame.info/downloads/>
* télécharger la bonne version de Pygame en fonction de votre version de Python. Lors de l'installation, vous rencontrerez une fenêtre comme celle-ci. L'installeur détecte automatiquement l'endroit adéquat où installer la librairie.

**Pygame ne fonctionne au lycée qu'avec la version 3.2 de Python donc il faut peut être (re)configurer votre shell et le redemarrer!**

**Initialisation**

Voilà notre base de programme (il ne faut pas appeler son programme Pygame!) :



Implémentez-le et exécutez-le.

Vous devez avoir une fenêtre (de taille 640x480 pixels) qui s'ouvre brièvement, puis se ferme. C'est normal, puisque l'instruction suivant l'ouverture de cette fenêtre est la fermeture de pygame!

Pour la faire s'afficher 2 secondes (soit 2000 millisecondes) on doit compléter notre programme par :



Recopiez le code pour que la fenêtre soit affichée 2 secondes.

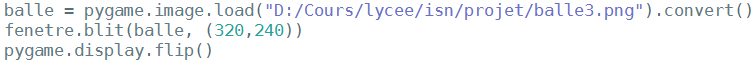
Modifiez votre code pour que la fenêtre soit affichée 5 secondes.

**IMAGES**

**Afficher une image dans notre fenêtre.**

récupérez l'image balle3.png et enregistrez-la en prenant soin de mémoriser son chemin d'accès.

Voilà le code permettant d'afficher l'image balle3.png dans notre fenêtre:



Pour faire simple, la première ligne indique où se trouve le fichier de l'image et la transforme en une variable ici appelée  *balle* utilisable dans le programme. Cette affectation n'est à faire qu'une fois dans le programme!

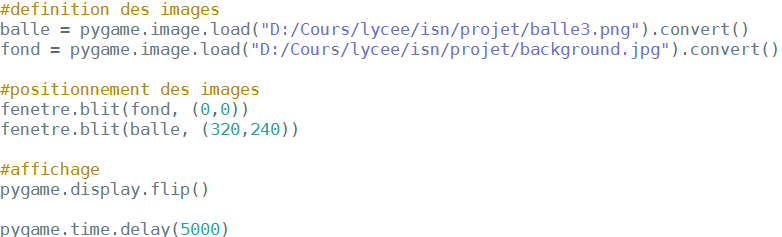
La deuxième ligne précise à quel endroit le coin en haut à gauche de *balle* sera placé dans la fenêtre.

La fenêtre est vue comme un tableau de pixels (ici de 640 de large sur 480 de haut). La case en haut à gauche est repérée par (0,0) et celle en bas à droite par (640,480). Ici, on place image en (320,240) soit au milieu!

La troisième ligne, sert à actualiser l'affichage avec nos modifications.

Recopiez ce code et modifier le pour placer 3 fois l'image sur l'écran: en haut à gauche, en haut à droite et en bas à droite.

On peut faire afficher une deuxième image de fond. Il suffit de définir une deuxième variable *fond*… la placer au bon endroit puis faire **un seul** flip()



Recopiez ce code et admirer votre oeuvre. L'ordre est important!

*Remarque:* notre balle semble entourée d'un fond blanc peu esthétique. Essayez :



la couleur des pixels du fond était en fait "transparente". Cette couleur n'existe pas dans tous les formats, nous creuserons la question si vous choisissez un projet Pygame…

On peut aussi juste vouloir un fond uni (couleur en RGB): 

**Faire se déplacer une image**

Pour faire "bouger" une image, il suffit de la faire s'afficher successivement à différents endroits.

Construire un programme qui déplace une image du bord gauche de l'écran vers le bord droit. Vous aurez donc besoin d'une boucle et d'une variable pour noter l'abscisse *x* de l'image.

Si vous réussissez, vous remarquerez que l'image laisse "une trainée" derrière elle car les images affichées précédemment sont restées. On ne peut pas les "effacer", il faut "écraser" votre précédente image avec le fond avant d'afficher l'image à sa nouvelle position.

**LES EVENEMENTS**

On sait maintenant insérer et faire bouger une image dans la fenêtre pygame. On souhaite maintenant que l'utilisateur du programme puisse interagir avec ce qu'il se passe dans la fenêtre avec réactivité, sans repasser par le shell. Par exemple, on aimerait pouvoir quitter proprement pygame en cliquant sur la croix de fermeture au lieu de simplement attendre la fin de l'exécution du script (ou de devoir l'écrire dans le shell car on a oublié de le notifier dans le script!).

Découvrons comment piloter le programme par des évènements (clic de souris, appui sur une touche comme flèche haut…). Chaque évènement qui se produit est envoyé sur une file. La méthode pygame.event.get() permet de les affecter les uns après les autres à une variable (que nous appelerons evenement).afin de pouvoir les traiter. Nous en analyserons ensuite le type (grâce à evenement.type) afin de déterminer ce que nous souhaitons faire.

**la croix de fermeture!**

Lorsque vous cliquez sur la croix de fermeture de la fenêtre, l'événement détecté a pour type : QUIT (en majuscules!).

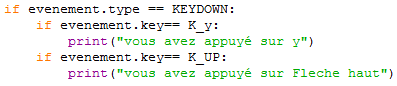
recopiez ce programme qui vous servira de base.



**les évènements "clavier"**

On veut savoir si une touche du clavier a été enfoncée ou relâchée. Il suffit de tester les types KEYDOWN ou KEYUP (rappel: key signifie touche ; et up/down signifient haut/bas). Une fois qu'on sait qu'une touche a été enfoncée (ou relâchée), il peut être utile de savoir laquelle. Pour cela on examine la valeur de evenement.key

exemple:



la valeur K\_y correspond à un enfoncement de la touche y, K\_UP pour flèche haut, K\_DOWN pour flèche bas, K\_RIGHT pour flèche droite, K\_LEFT pour flèche gauche.

*exercice 1:* écrire un programme qui :

- ferme lorsqu'on clique sur la croix de fermeture.

- affiche dans le shell (comme cela on continue d'utiliser la procédure print() ) "la touche A a été enfoncée", "la touche A a été relâchée" lorsque cela se produit.

Vous remarquez que le clavier est considéré comme en qwerty!

*exercice 2:* écrire un programme qui permet à l'utilisateur de déplacer la balle avec les flèches directionnelles.

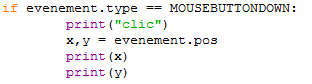
**événements "souris"**

Il y a trois types d'évènements pour la souris:

* MOUSEMOTION : la souris bouge
* MOUSEBUTTONDOWN: on appuie sur un bouton de la souris
* MOUSEBUTTONUP: on relâche un bouton de la souris.

Lorsqu'un de ces trois évènements est détecté, on peut accéder à la position de la souris à ce moment en faisant:

x,y = evenement.pos

exemple:

**TEXTE**

Il y a trois étapes pour écrire du texte:

* définir une police, au début du programme, grâce à :

police = pygame.font.SysFont("monospace" ,15)

où 15 est la taille et monospace est la police retenue.

* créer une image contenant le texte que l'on veut afficher grâce à:

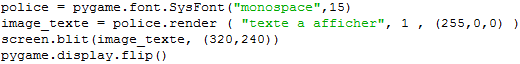
image\_texte = police.render ( "texte a afficher", 1 , (255,0,0) )

Vous avez reconnu à la fin le code RGB (ici rouge) de la couleur souhaitée pour le texte à afficher!

* Enfin, on affiche l'image à l'écran comme toutes les autres images!

screen.blit(image\_texte, (320,240))

pygame.display.flip()



exemple: