Interface graphique

A : le module Tortue

B : le module Tkinter

A : le module Tortue

Python possède de nombreuses bibliothèques regroupant des catégories de fonctions. Elles sont appelées des modules, au même titre que les fichiers ".py" que vous créez et qui deviennent vos propres modules.

Dans EduPython, si vous souhaitez réaliser une nouvelle figure à l'aide de la tortue, cliquer sur le bouton Nouveau Fichier... et choisissez le mode Tortue. Un nouveau programme comportant déjà quelques lignes de code est généré :

```
# Créé par ... , le ... avec EduPython
    from lycee import *
    import turtle as tortue
    ... .
    tortue . mainloop()
```

Pour utiliser le module "Tortue", écrivez au début de votre programme :
from tortue import * (sans oublier "*")

Cette tortue est une interface graphique dans laquelle vous avez un traceur que l'on va apprendre à commander.
Une documentation très complète est disponible sur : docs.python.org
Au départ la tortue (le traceur) est au centre de l'écran (de coordonnées (0,0)) et orienté vers l'est.

Les déplacements :

- fd(dist) : fait avancer la tortue de dist
- bk(dist) : la même chose en reculant
- left(angle) : change la direction du mouvement en la faisant tourner de angle vers la gauche(en degrés par défaut)
- right(angle) : de même vers la droite

- goto(x,y) : déplace la tortue au point de coordonnées (x,y)
- circle(r, angle) : trace un cercle de rayon r.

Le tracé du cercle commence à l'endroit où est la tortue, puis tourne vers la gauche. Si aucun angle n'est précisé, trace le cercle un entier, sinon un arc de cercle.

- undo() : annule la dernière action (peut être répété).
- reset() : remet la tortue à l'origine et efface le tracé
- home() : remet la tortue à l'origine mais sans effacer le tracé
- dot(r,"couleur") : trace un point de rayon r et de la couleur choisie (ex : "red", "blue" etc...)

Propriétés du traceur

On peut évidemment changer les paramètres du traceur :

- pu() (ou penup()) : lève le "stylo" , la tortue se déplace, mais sans tracé
- pd() (ou pendown()) : redescend le "stylo"
- pensize(nb) : fixe la largeur du "stylo" à nb
- color("coul") : fixe la couleur du stylo ("red", "blue" etc..)
- speed(n) : règle la vitesse. Si n = 0, le tracé est quasi instantané, si n = 10, il est rapide, si n = 1 il est très lent

La fenêtre

Les propriétés de la fenêtre peuvent également être modifiées :

- setup(largeur,hauteur) : règle la largeur et la hauteur (en pixels) de la fenêtre
- bgcolor("coul") : fixe la couleur de l'arrière-plan
- bye() : permet de fermer la fenêtre

S'il n'y a aucune boucle dans votre programme, aucune pause, vous n'aurez guère le temps de voir votre tracé

- mainloop() : permet d'entrer dans une boucle d'attente

ceci vous donne le temps de voir votre tracé, la fenêtre se ferme en cliquant sur la croix

- exitonclick() : permet de fermer la fenêtre avec un clic
- clear() : efface la fenêtre, sans bouger la tortue

I) Avancer, reculer, tourner

tortue.forward(n) et tortue.back(n)

Fait respectivement avancer ou reculer la tortue dans la direction où elle regarde de n pas (n pouvant être entier ou non)

tortue.left(a) et tortue.right(a)

Fait respectivement tourner la tortue vers la gauche ou la droite de a degrés. (Aucun tracé n'est effectué, juste une rotation de la tête)

Code: Une maison.

```
from lycee import *

tortue . forward(100)
tortue . right (90)
tortue . forward(100)
tortue . right (90)
tortue . forward(100)
tortue . right (90)
tortue . forward(100)
tortue . right (30)
tortue . forward(100)
tortue . right ( 120)
tortue . forward(100)
tortue . mainloop()
```

Code: Une église.

```
from lycee import *
import turtle as tortue tortue . left(90)
tortue . forward(100)
B = acos(40 / 100)
tortue . right (90 - B)
tortue . forward(100) C = 180 - 2 * B
tortue . right ( 180 - C)
tortue . forward(100)
tortue . right (90 - B)
tortue . forward(20)
tortue . left(90)
tortue . forward(100)
tortue . right (50)
FG = 20 / cos(40)
tortue . forward(FG)
tortue . right (40)
HG=80 - sqrt(FG * FG - 20 * 20)
tortue . forward(HG)
tortue . right (90)
tortue . forward(200)
tortue . mainloop()
```

Code: Tracer un polygone régulier à n côtés.

```
from lycee import *
import turtle as tortue

n = demande(" Nombre de côtés (au moins 3)")
for i in range(n):
tortue . forward(50)
tortue . left( 360/n)

tortue . mainloop()
```

III) Tracer des cercles

tortue.circle(rayon) ou tortue.circle(rayon,angle)

– Si rayon>0 : Trace un cercle de rayon rayon à partir de la position de la tortue et en tournant dans le sens trigonométrique.

– Si rayon < 0 : Trace un cercle de rayon |rayon| dans le sens anti-horaire.

– Si angle est précisé, trace un arc de cercle de rayon |rayon| avec une ouverture de angle

(en degré). Si angle n'est pas précisé, le cercle est tracé dans son intégralité.

La spirale est obtenue en traçant bout à bout des quarts de cercle de centres successifs A, B, C et D.

Code: Tracer une spirale.

```
from lycee import *
import turtle as tortue

n = 10
for i in range(n):
tortue . circle (5*i,90)

tortue . mainloop()
```

Remarque:
Attention, on ne connaît a priori pas le centre de ce cercle... c'est d'ailleurs ce qui peut faire l'intérêt de l'algorithme !
On veut tracer l'œuf de Pâques ci-contre.
Données : OA = OB = OC = r, C1 est un demi-cercle de diamètre [AB], C2 est un arc de cercle de centre A et passant par B et E, C3 est un arc de cercle de centre C et passant par E et D, enfin C4 est un arc de cercle de centre B et passant par D et A.

Code: Tracer un œuf de Pâques.

```
from lycee import *
import turtle as tortue
r = 100
tortue . right (90)
tortue . circle (r, 180)
tortue . circle (2 * r, 45)
tortue . circle (r * (2 - sqrt(2)), 90)
tortue . circle (2 * r, 45)

tortue . mainloop()
```

Code: Le yin et le yang

```
from lycee import *
import turtle as tortue

r = 100
tortue .up ()
tortue . forward(r)
tortue . down ()
tortue . left(90)
tortue . circle (2 * r)
tortue . circle (r, 180)
tortue . circle (-r, 180)
tortue . hideturtle()

tortue . mainloop()
```

Code: Le yin et le yang.

```
from lycee import *
import turtle as tortue

r = 100
tortue .up ()
tortue . forward(r)
tortue . down ()
tortue . left(90)
tortue . circle (2 * r)
tortue . circle (r, 180)
tortue . circle (-r, 180)
tortue . hideturtle()

tortue . mainloop()
```

Code: Un soleil avec 120 rayons.

```
from lycee import *
import turtle as tortue
i = 0
while i < 120:
tortue . right(90)
tortue . forward(100)
tortue . right( 180)
tortue . forward(100)
tortue . right(90)
tortue . circle (50 , 3)
i = i + 1
tortue . mainloop()
```

IV) La tortue : Afficher, Cacher, Vitesse

tortue.showturtle() tortue.hideturtle()

A pour effet de respectivement cacher ou montrer la tortue à l'écran. Pour des questions d'esthétisme, on peut par exemple vouloir cacher la tortue en fin de tracé.

tortue.speed(v)

Permet de régler la vitesse de la tortue. v est un nombre entier entre 1 et 10, 1 étant la vitesse la plus lente et 10 la plus rapide.

V) Le crayon : lever, baisser, taille, couleur

On peut pour certains dessins avoir besoin de déplacer la tortue sans laisser de trace.

tortue.up() tortue.down()

A pour effet de respectivement lever et baisser le crayon

tortue.pencolor(texte) ou tortue.pencolor(rouge,vert,bleu)

Définit la couleur du crayon.

– On peut entrer un texte entre guillemets parmi (entre autres) : 'aqua', 'beige', 'black', 'blue', 'brown',

'chocolate', 'fuchsia', 'gold', 'gray', 'green', 'indigo', 'khaki', 'maroon', 'orange', 'red', 'white', ...

– On peut aussi définir sa propre couleur en paramétrant les composantes rouge, vert et bleu de la couleur

(chaque composante étant un nombre entre 0 et 1).

Code: Le drapeau européen.

```
from lycee import *
import turtle as tortue

tortue . pensize(2)
tortue . pencolor(0.9 , 0.9, 0.2)
tortue . bgcolor( 'blue ')
for etoile in range (12 ):
tortue . down ()
for branche in range (5):
tortue . forward(30)
tortue . left(144)
tortue .up()
tortue . forward(50)
tortue . left(30)

tortue . mainloop()
```

VI) L'écran : effacer, colorer le fond, afficher un texte

tortue.reset()

Efface l'écran et repositionne la tortue dans sa position initiale

tortue.clear()

Efface l'écran mais la position du crayon reste inchangée

tortue.write(texte)

Affiche le texte texte à l'emplacement de la tortue. Celle-ci ne se déplace pas lors de l'affichage.

De nombreuses autres fonctionnalités sont disponibles sur le site officiel :

http ://docs.python.org/3.2/library/turtle.html

Un exemple simple :

```
from turtle import*
colormode(255)  #pour utiliser les couleurs en mode RGB (sinon colormode(1))
pu()          #on deplace le point de départ pour centrer la figure (stylo levé)
bk(100)       #on recule de 100
pd(          )#on descend le stylo
for i in range(1,7):
    color((155,255-30*i,200))#pour changer de couleur quand i change
    fd(10*i)
    left(90)
    fd(10*i)
    left(90)
```

```
    fd(10*i)
    left(90)
    fd(10*i)
    left(90)
    pu()
    fd(10*i)
    pd()
pu()
fd(40)
pd()            #à ce stade le traceur est orienté vers l'est
left(90)         #on veut que le cercle se trace vers le haut
circle(150)
mainloop()
```

TP: réaliser un jeu de pendu

Réaliser à l'aide de la tortue un dessin de gibet avec son pendu.

Ecrire "mainloop()" à la fin de votre programme pour pouvoir facilement fermer la fenêtre graphique

B : INTERFACE GRAPHIQUE AVEC LE MODULE TKINTER

Le module Tkinter ("Tk interface") de Python permet de créer des interfaces graphiques

(GUI : graphical user interface).

De nombreux composants graphiques (ou widgets) sont disponibles : fenêtre (classe Tk), bouton (classe Button), case à cocher (classe Checkbutton), étiquette (classe Label), zone de texte simple (classe Entry), menu (classe Menu), zone graphique (classe Canvas), cadre (classe Frame)...

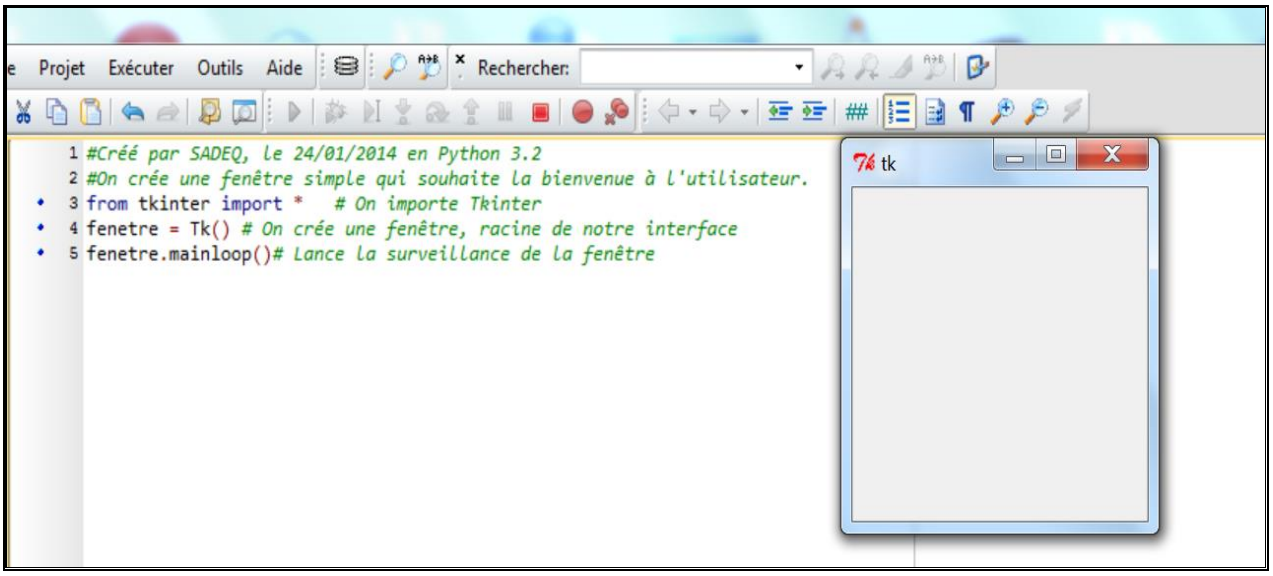
On peut gérer de nombreux événements : clic sur la souris, déplacement de la souris, appui sur une touche du clavier, top d'horloge...

Logiciels utilisant Python et sa bibliothèque graphique Tkinter

Tkinter est l'interface graphique des logiciels IDLE (environnement de développement intégré pour le langage Python) etPyMOL (logiciel libre de visualisation de structures chimiques en 3D) :

1) Avant de se lancer :

```
#On crée une fenêtre simple qui souhaite la bienvenue à l'utilisateur.
```



Comme tous les autres objets, la fenêtre aura un nom (ici **fenetre**) qu'on peut évidemment changer.

Comment régler ses dimensions ?

Méthode	Effet
fenetre.geometry(‘’ 500x150 ‘’)	Redimensionne la fenêtre **fenetre** en 500 pixels de large et 150 pixels de haut.
fenetre.title(T)	Affiche le titre T dans la fenêtre
fenetre.winfo_width()	Renvoie la largeur(interne) de la fenêtre **fenetre**
fenetre.winfo_height()	Renvoie la hauteur (interne) de la fenêtre fenetre
fenetre.resizable(width=False)	Empêche le redimensionnement en largeur de la fenêtre **fenetre**
fenetre.resizable(height=False)	Empêche le redimensionnement en hauteur de la fenêtre **fenetre**
	Remarque : On peut placer ces deux arguments dans les parenthèses en les séparant par une virgule pour n'utiliser qu'une seule ligne : fenetre.resizable(width=False , height=False)

Les fenêtres sont donc des objets sur lesquels on peut definer des fonctions(on dit méthodes)

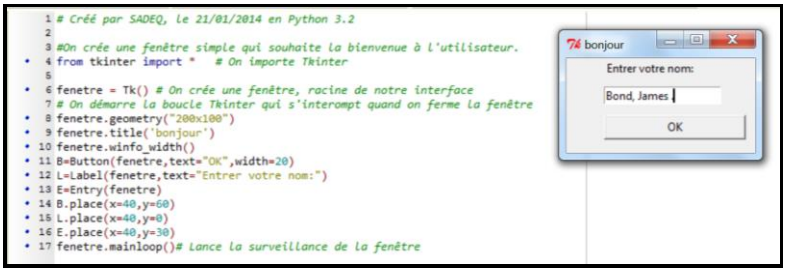
2) Premiers widgets.

Sur la fenêtre ,nous allons donc pouvoir placer différents objets que l'on appelle des « widgets » (ce mot peut être traduit en français par machin)

Il en existe de nombreux, en voici déjà trois pour commencer :

- Le widget Button : il permet d'afficher un bouton avec un texte(OUI, NON,Confirmer…)
- Le widget Label :pour afficher un texte sur votre fenêtre.
- Le widget Entry : il permet au joueur d'entrer du texte

Exemple :



```
#On crée une fenêtre simple qui souhaite la bienvenue à l'utilisateur.
1.from tkinter import *    # On importe Tkinter
2.fenetre = Tk() # On crée une fenêtre, racine de notre interface
# On démarre la boucle Tkinter qui s'interompt quand on ferme la fenêtre
4.fenetre.geometry("200x100")
5.B=Button(fenetre,text="OK",width=20)
6.L=Label(fenetre,text="Entrer votre nom:")
7.E=Entry(fenetre)
8.B.place(x=40,y=60)
9.L.place(x=40,y=0)
10.E.place(x=40,y=30)
11.fenetre.mainloop()# Lance la surveillance de la fenêtre
```

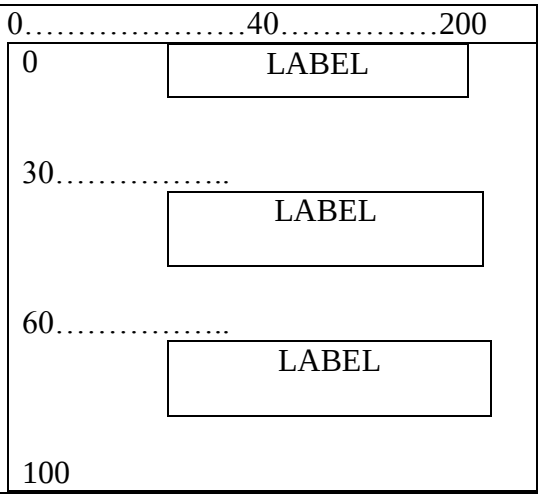
- ❖ Les lignes 1 à 4 initialisent la fenêtre avec une dimension de 200x100 pixels et un titre
- ❖ Les lignes 5,6 et 7 créent respectivement un Button, Label et Entry que nous avons choisi d'appeler B,L et E dans notre programme.Quel que soit le widget, l'appel se fait selon le même principe : Le premier argument est le nom de l'objet sur lequel il est déposé (ici fenetre) suivi d'un certain nombre de paramètres.
- ❖ Les lignes 8,9 et 10 sont importantes, car elles demandent à Python de dessiner les 3 composants.
- ❖ La ligne 11 lance le processus de mise en place de la fenêtre que l'on appelle gestionnaire d'événements.

a- Placement des widgets avec la méthode place

Il existe différentes méthodes pour placer les composants sur une fenêtre.La methode place est la plus simple.

Pour placer les widgets, on utilise donc un système de coordonnées sur la fen

- Le point de coordonnées (0 ;0) se trouve
  En haut à gauche de l'écran.
- Les valeurs de y augmentent lorsqu'on descend sur l'écran



On peut aussi avoir besoin de cacher un widget par l'opération inverse : B.place_forget() aura donc pour effet de cacher l'objet B

b- Label : Afficher un texte

Le widget Label sert donc à afficher un texte à l'écran. Au moment de la création, vous pouvez spécifier un certain nombre de paramètres :

Paramètres	Effet
text	Précise le texte à afficher
fg	Précise la couleur du texte
bg	Précise la couleur de fond
heigt	Précise la hauteur du Label
width	Précise la Largeur du Label
font	

Une fois le widget Label créé et affiché, un certain nombre d'actions(méthodes) sont possibles.Lappel se fait de la manière suivante :

Méthode	Effet
L.config(…)	Permet de modifier les paramètres du widget,par exemple L.config(text= '' Il ne vous reste qu'une vie'' permet de modifier l'affichage du Label L.Vous pouvez modifier plusieures options d'un coup en les séparant par une virgule
L.cget(…)	Renvoie la valeur de l'option demandée sous forme d'une chaine. Par exemple print (L.cget(''fg'') affiche la couleur du texte du Label L.

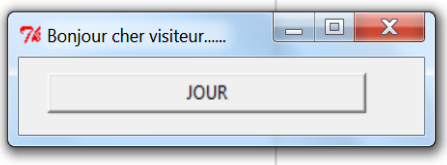
c- Button :Un bouton

Ce widget est un bouton sur lequel on peut définir le déclenchement d'une action lors d'un clic.
Au niveau des paramètres ,on va retrouver les mêmes que pour le Label(text, fg , bg , height, font,…) et en particulier :

Paramètre	Effet
command	Permet de préciser la fonction à lancer lors d'un clic sur ce bouton. Par exemple command= go , indique qu'il faut exécuter la fonction go Attention : la fonction ne doit pas comporter de paramètre .On ne fait que figurer le nom de la fonction go et non go()

Exemple :

```
1
2  from tkinter import *
3  def change():
4      if B.cget('text')=='JOUR':
5          B.config(text='Nuit')
6      else:
7          B.config(text='JOUR')
8  fenetre = Tk()
9  fenetre.title('Bonjour cher visiteur.....')
10 fenetre.geometry('280x50')
11 B= Button(fenetre,text="JOUR",width=30,command=change)
12 B.place(x=20,y=10)
13 fenetre.mainloop()
```



d- Entry : Saisir un texte

Le widget Entry permet au joueur de saisir un texte court sur une ligne.Il possède lui aussi des paramètres comme les deux autres (fg, bg, height, width, font) mais pas les paramètres **text** ou **command par exemple.**

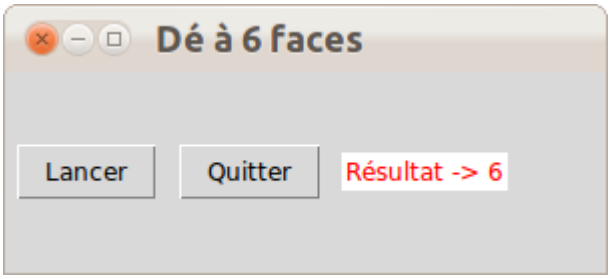
Pour accéder au contenu du texte saisi, voici différentes méthodes :

Méthodes	Effet
E. get()	Renvoie le texte entré dans l'Entry E.
E. insert(i,T) E. insert (INSERT,T) E. insert(END,T)	Insère le texte T dans l'Entry E respectivement à la position i, à la place du curseur, à la fin du contenu existant
E. delete(i) E. delete(deb,fin) E. delete(0,END)	Efface un morceau du contenu de l'Entry E. Efface respectivement le caractère à la position i , les caractères placés entre les indices deb et fin et l'intégralité du champ texte.

Remarque : On peut forcer le curseur à se placer sur l'Entry en exécutant la commande E.focus() pour éviter au joueur d'avoir à cliquer dessus.

Exemple n°1 : widgets Button et Label

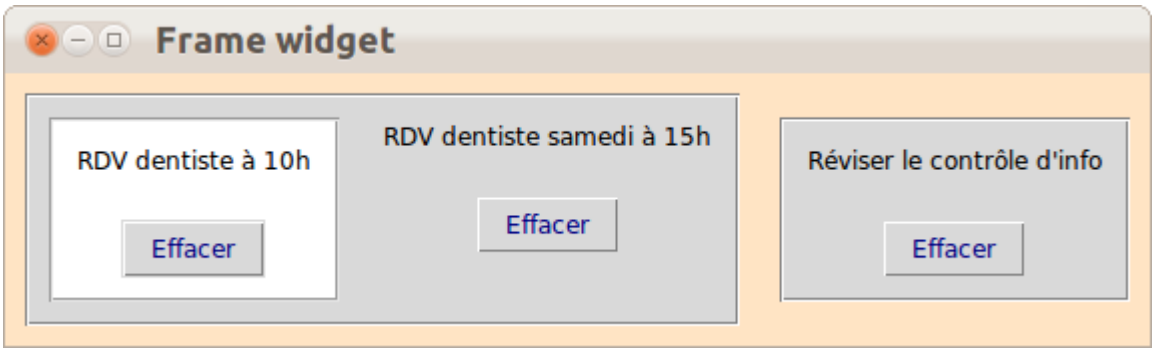
Ce script simule un dé à 6 faces :



widgets Button et Label.py

Exemple n°2 : widgets Frame, Label et Button

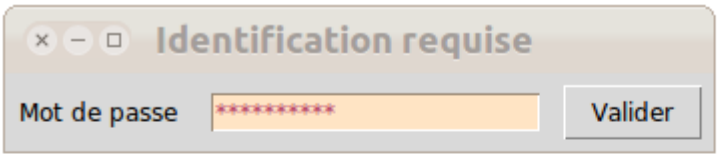
Un widget Frame est une zone rectangulaire qui peut contenir d'autres widgets.

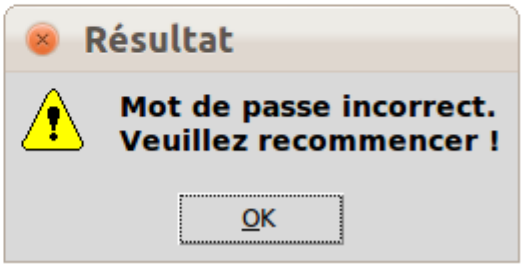


widgets Frame, Label et Button.py

Exemple n°3 : widgets Entry, Label, Button et boîte de dialogue MessageBox

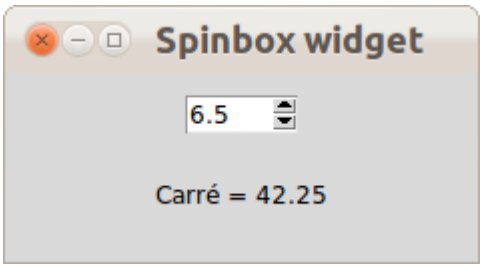
Un script d'authentification :





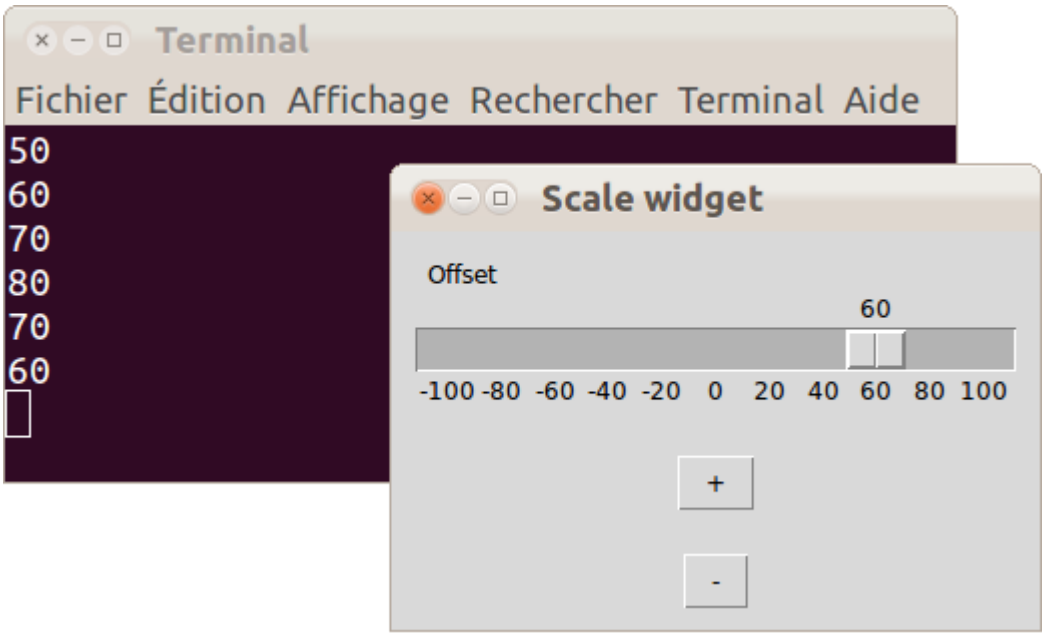
widgets Entry, Label, Button et boîte de dialogue MessageBox.py

Exemple n°4 : widgets Spinbox et Label



widgets Spinbox et Label.py

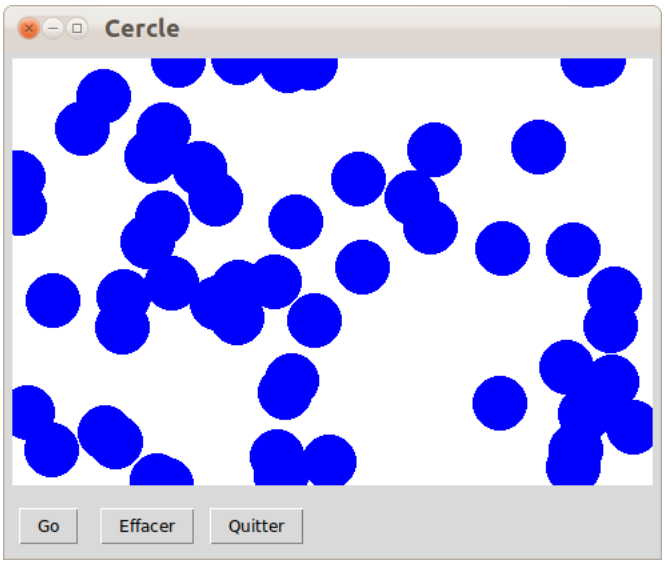
Exemple n°5 : widgets Scale et Button



widgets Scale et Button.py

Exemple n°6 : widgets Canvas et Button

Le script cercle.py dessine, à chaque clic sur le bouton Go, un disque de rayon 20 pixels à une position aléatoire :

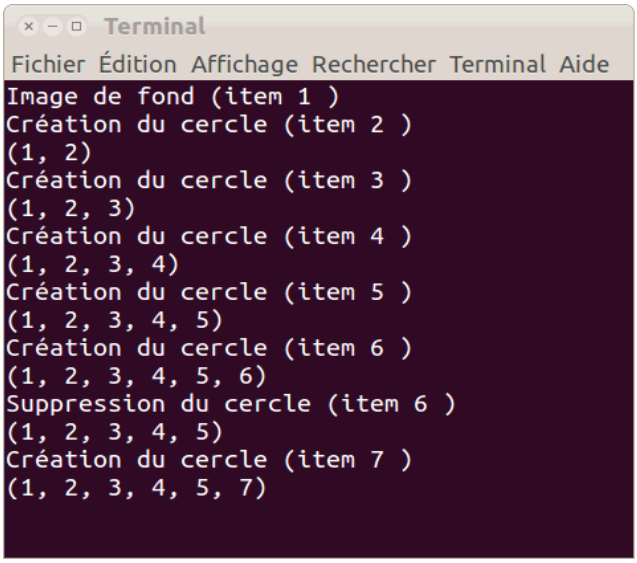


widgets Canvas et Button.py

Exemple n°7 : widgets Canvas et Button ; gestion des images

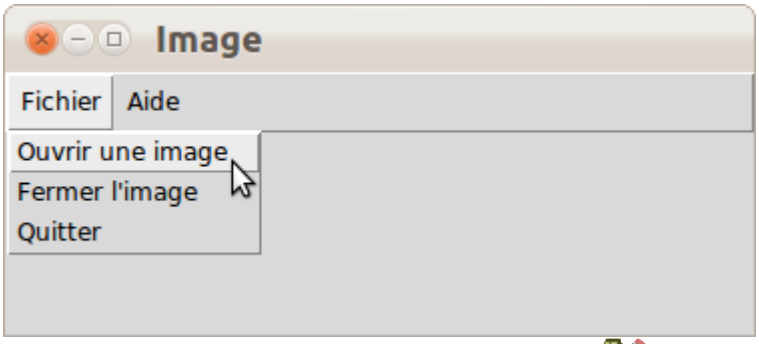
Ce script reprend le script cercle.py avec une image de fond (méthode create_image() de la classe Canvas) et la possibilité d'effacer la dernière action (pour cela, on se sert du numéro identifiant de chaque item d'un widget Canvas) :

widgets Canvas et Button ; gestion des images.py



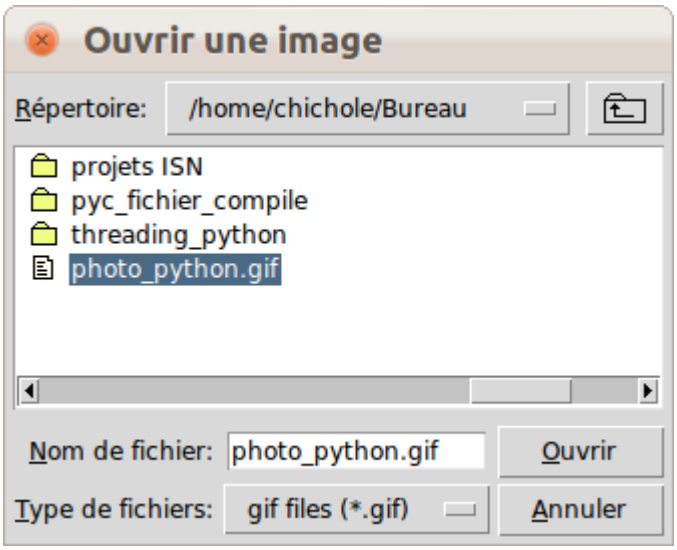
Exemple n°8 : widgets Menu et Canvas ; gestion des images ; boîtes de dialogueFileDialog et MessageBox

Le script suivant est un browser d'images (formats .gif .ppm .pgm), avec un widget Menu :

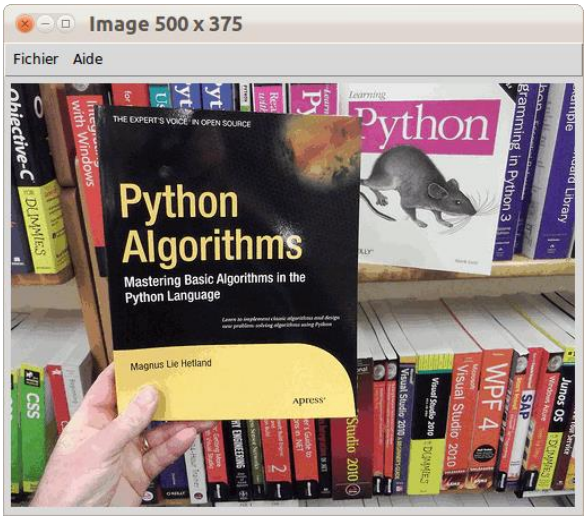


widgets Menu et Canvas ; gestion des images ; boîtes de dialogue FileDialog et MessageBox.py

une boîte de dialogue FileDialog pour rechercher un fichier :

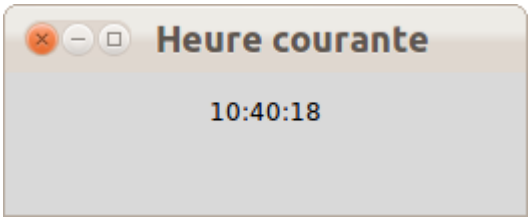


et un widget Canvas dans lequel sera affiché l'image :



Exemple n°9 : gestion du temps

L'heure courante est mise à jour toutes les secondes :

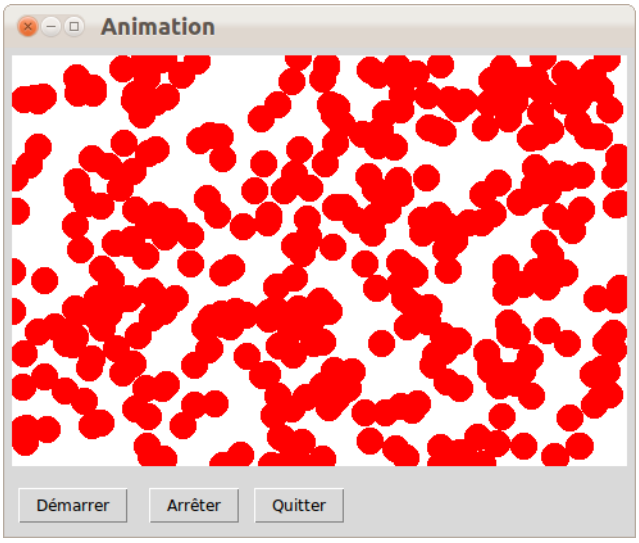


widgets Canvas et Button ; gestion du temps.py

Pour cela, on utilise la méthode after() qui appelle une fonction après une durée donnée en millisecondes :

Exemple n°10 : widgets Canvas et Button ; gestion du temps

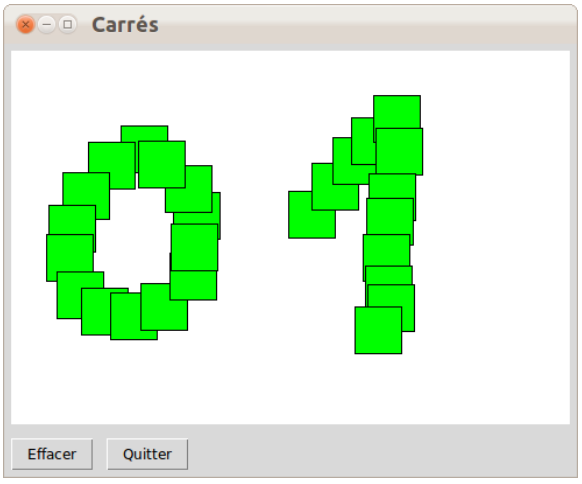
Le script animation.py est un exemple d'animation (environ 20 disques par seconde) :



On se sert de la méthode after() pour actualiser la zone graphique toutes les 50 ms :

Exemple n°11 : widgets Canvas et Button ; gestion de la souris

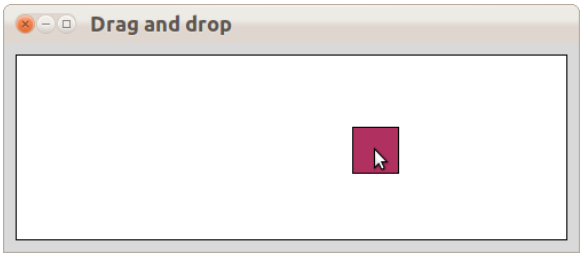
Le script carre.py dessine un carré à l'endroit du clic de la souris.
Pour cela, on utilise l'événement associé au clic gauche de la souris.



widgets Canvas et Button ; gestion de la souris.py

Exemple n°12 : widget Canvas ; gestion de la souris

Nous allons voir comment déplacer un objet graphique avec la souris (clic, drag and drop) :

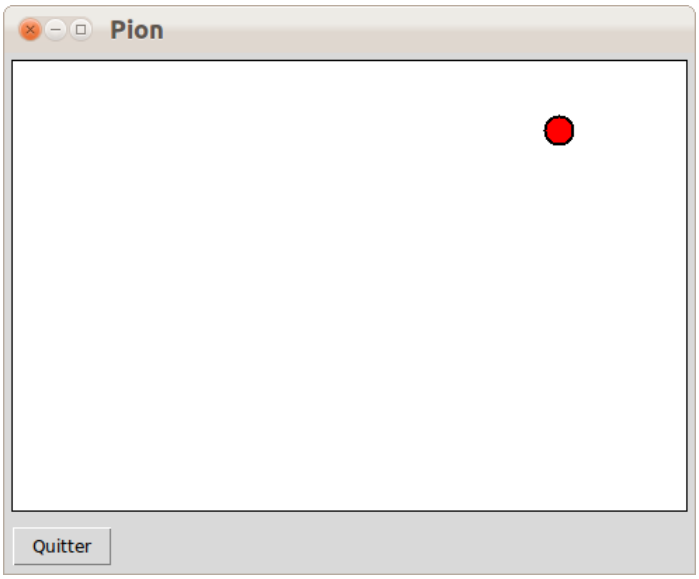


widget Canvas ; gestion de la souris.py

Exemple n°13 : widgets Canvas et Button ; gestion du clavier

Le script pion.py gère le déplacement d'un pion avec le clavier.
Pour se faire, on utilise l'événement associé à l'appui d'une touche du clavier.

- touche a déplacement vers le haut
- touche q déplacement vers le bas
- touche l déplacement vers la gauche
- touche m déplacement vers la droite



widgets Canvas et Button ; gestion du clavier.py

Symboles des quelques touches spéciales

'Up', 'Down', 'Left', 'Right' (flèches directionnelles haut, bas, gauche, droite), 'Return' (touche Entrée), 'space' (barre Espace)...

Exemple n°14 : widgets Checkbutton et Button ; musiques et sons avec pygame

widgets Checkbutton et Button ; musiques et sons avec pygame.py

Le module pygame est un module externe de création de jeux vidéo en 2D.
pygame contient un sous module pygame.mixer qui permet de charger et de lire des musiques ou des sons dans plusieurs formats (mp3, ogg, wav...).
La procédure d'installation de pygame se trouve ici.

Télécharger le son death1.wav
Télécharger le son balla1.ogg
Télécharger la musique chavmusic7.mp3

D'AUTRES EXEMPLES D'UTILISATION

Vous trouverez en annexes d'autres exemples d'utilisation de tkinter pour faire des interfaces graphiques.
Il est conseillé de débuter par l'exemple 1, puis le 2, ... car les commentaires présents dans les premiers exemples ne sont pas reproduits dans les suivants.
(Cependant, l'exemple 2 avec barres de défilement pourra attendre de passer après l'exemple 8).

Exemple 1 : Button, Entry et Label.
Exemple 2 : Canevas.
Exemple 3 : Radiobutton.
Exemple 4 : Menu.
Exemple 5 : Utilisation de PIL et de tkinter.filedialog : images et fichiers
Exemple 5bis : Amélioration du précédent avec traitement d'erreurs.
Exemple 6 : Toplevel (ouverture d'une deuxième fenêtre)
Exemple 7 : Frame (sous-fenêtre)
Exemple 8 : ListBox et Scrollbar (liste avec ascenseur)
Exemple 9 exemple 2 amélioré : exemple 2 avec ascenseur.
Exemple 10 : NoteBook de tkinter.tix (des onglets)
Exemple 11 : Un chronomètre avec plusieurs Frame et Button
Exemple 12 : Nommer plusieurs éléments (boutons, frames, labels...) d'un coup
Exemple 13 : La méthode 'bind' plusieurs fois sur un même bouton

On trouvera ici la librairie PIL

Annexes

Ex1fenetreTexteVariable.py	Ex7fenetreListe_BarreDefi.py
Ex2fenetreCanevas.py	Ex8frames.py
Ex2fenetreCanevasBarreDefil.py	Ex9tixOnglets.py
Ex3fenetreRadioBouton.py	Ex10Chrono.py
Ex4fenetreMenu.py	Ex11multinommage.py
Ex5ouvertureImagePIL.py	Ex12toBindBoundBound.py
Ex5ouvertureImagePILbis.py	
Ex6deuxiemeFenetre.py	

WEBOGRAPHIE

Autour de Python :
Apprendre à programmer avec Python 3 de G Swinnen Débuter avec Python au lycée de K Naroun Les Pythonneries (en vidéo) Mémento pour la création d'un exécutable avec cx_Freeze

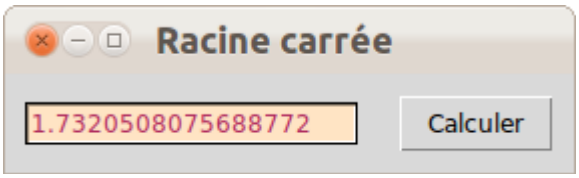
Autour de Tkinter :
http://www.python-course.eu/python_tkinter.php (en anglais) http://infohost.nmt.edu/tcc/help/lang/python/tkinter.pdf (en anglais) http://www.pythonware.com/library/tkinter/introduction/ (en anglais) http://www.dil.univ-mrs.fr/~chris/Documents/Tkinter.pdf (en français) http://www.ferg.org/thinking_in_tkinter/languages/france/penser_en_tkinter.html(en français) http://www.jchr.be/python/tkinter.htm (en français) http://www.fil.univ-lille1.fr/~marvie/python/chapitre6.html# (en français) http://python.mesexemples.com/plus-de-python/gui/gui-ajouter-un-composant-a-un-frame/ (en français) http://www.tutorialspoint.com/python/python_gui_programming.htm (en français) http://python.developpez.com/faq/?page=Tk (en français)

Autour du Web :
Start Your Dev (des infos bien structurées sur le html5 et le CSS3) Openclassooms (ex- Site du zéro) (Une multitude de tutoriels avec en particulier celui de Mathieu Nebra) Développez.com (Le tutoriel de Josselin Willette) Html Elements et attributes (tout à partir d'une liste...)

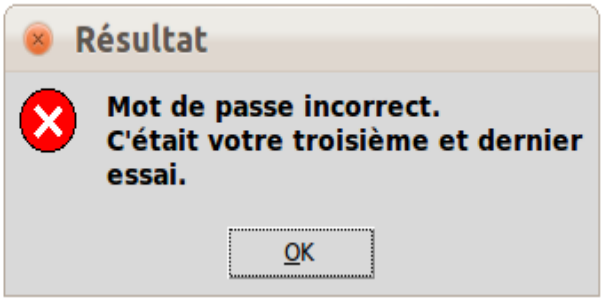
Exercices

Exercice 1 En s'inspirant des scripts de.py et mot_de_passe.py, écrire une application avec interface graphique qui calcule la racine carrée d'un nombre.

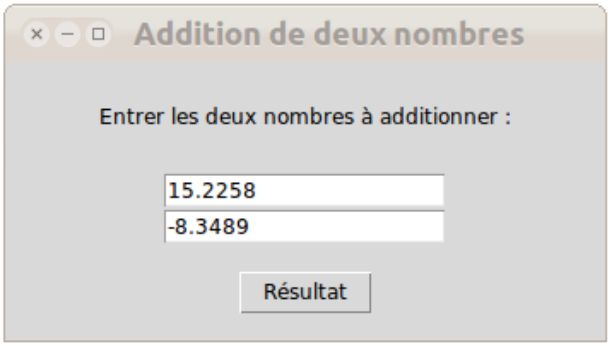
Par exemple, le calcul de √3 donne :

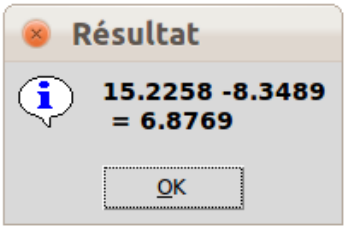


Exercice 2 Reprendre le script mot_de_passe.py de manière à limiter le nombre d'essais à trois.

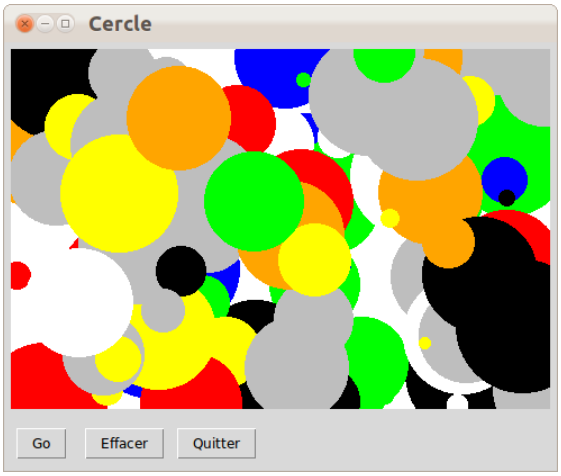


Exercice 3 En s'inspirant des scripts de.py et mot_de_passe.py, écrire une application avec interface graphique qui calcule l'addition ou la soustraction de deux nombres :





Exercice 4 A partir du script cercle.py, dessiner des disques de positions, rayons et couleurs aléatoires :



Exercice 5 A partir du script animation.py, faire une animation avec des disques de positions, rayons et couleurs aléatoires.

Exercice 6 :

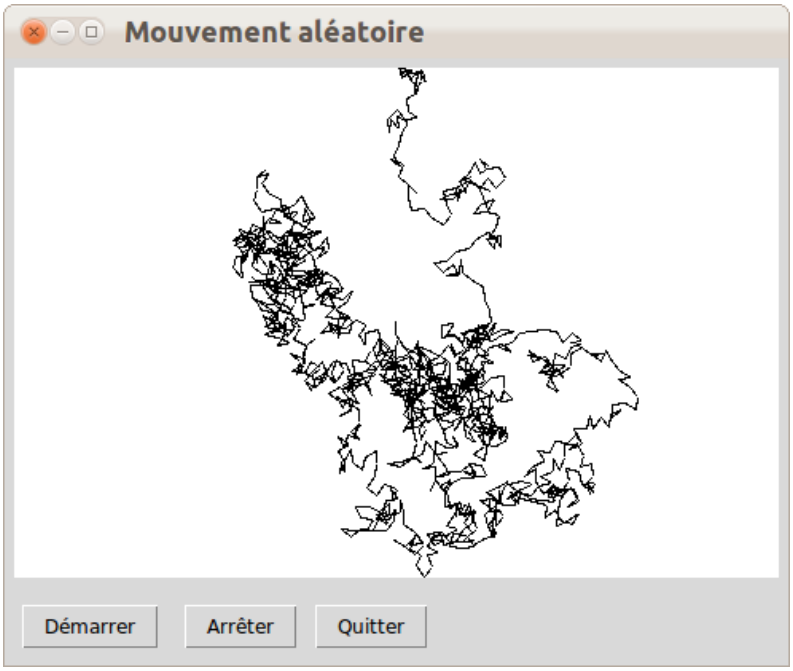
1) Reprendre le script cible.py et remplacer le disque noir par une image :

Télécharger l'image impact.gif
Remarque : l'image de l'impact doit avoir un fond transparent.

2) En s'inspirant du script sons_pygame.py, ajouter un effet sonore (tk_coup_fusil.wav).

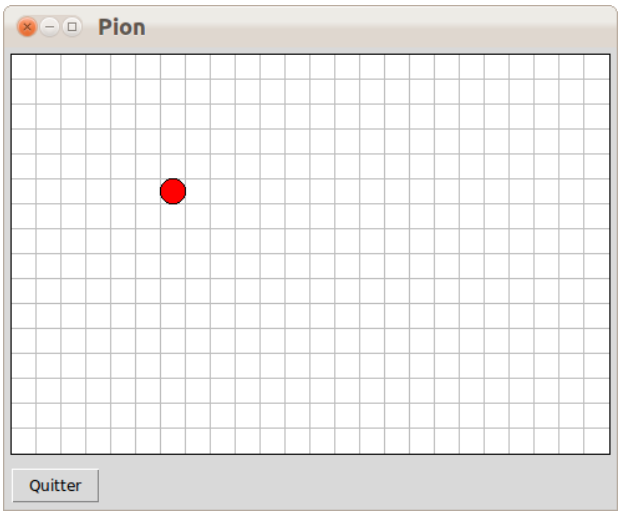
Exercice 7 : En s'inspirant du script animation.py, faire l'animation d'un mouvement aléatoire brownien.

On utilisera la méthode create_line() de la classe Canvas.



Exercice 8 : Compléter le script pion.py de manière à dessiner une grille.

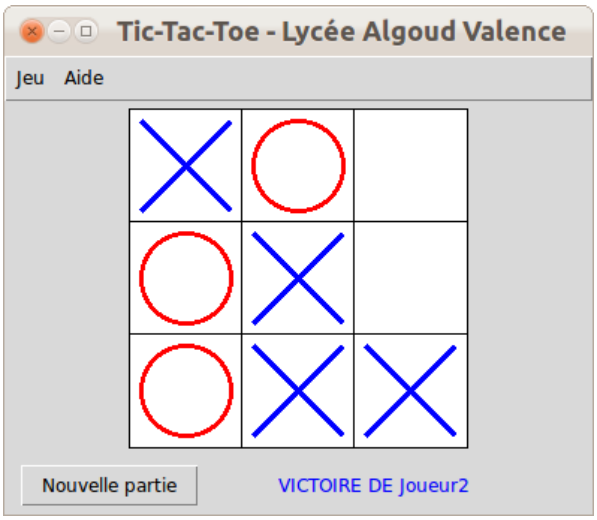
On utilisera la méthode create_line() de la classe Canvas.



Quelques idées de projets

Projet n°1 : Jeu Tic-Tac-Toe (jeu du morpion)

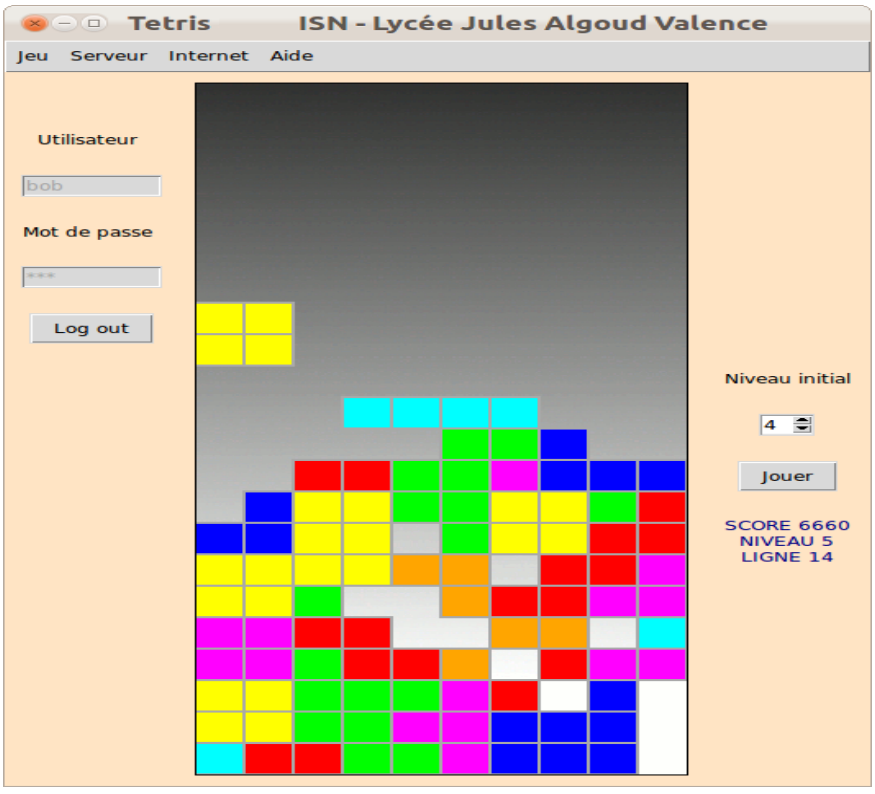
Un projet relativement simple pour un travail en binôme.



Projet n°2 : Jeu de Tetris avec classement en ligne

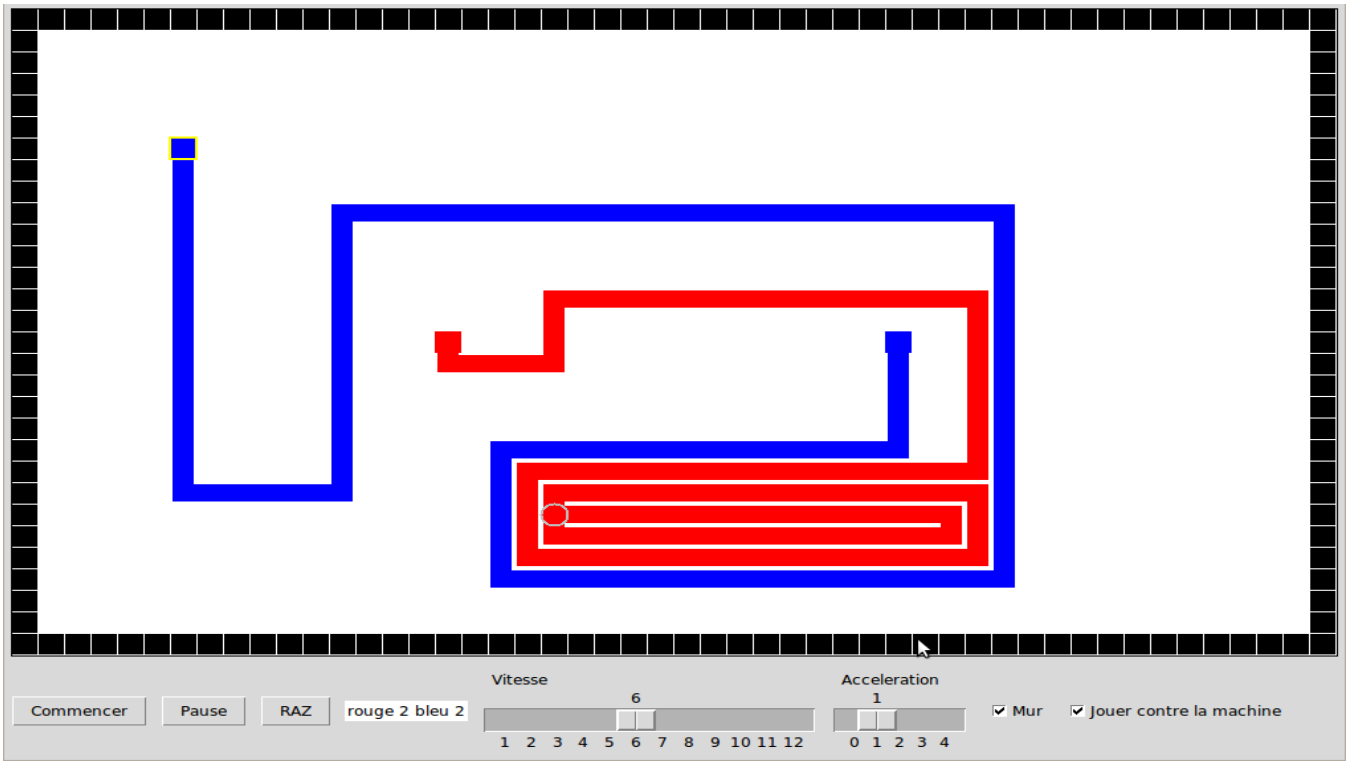
Un gros projet à décomposer en plusieurs tâches :

- jeu en local avec Python
- applications Web (en PHP ou CGI-Python, base de données MySQL)
  - nombre d'inscrits
  - inscription en ligne (essayez !)
  - classement en ligne
  - record
  - dernières parties
  - dernière version



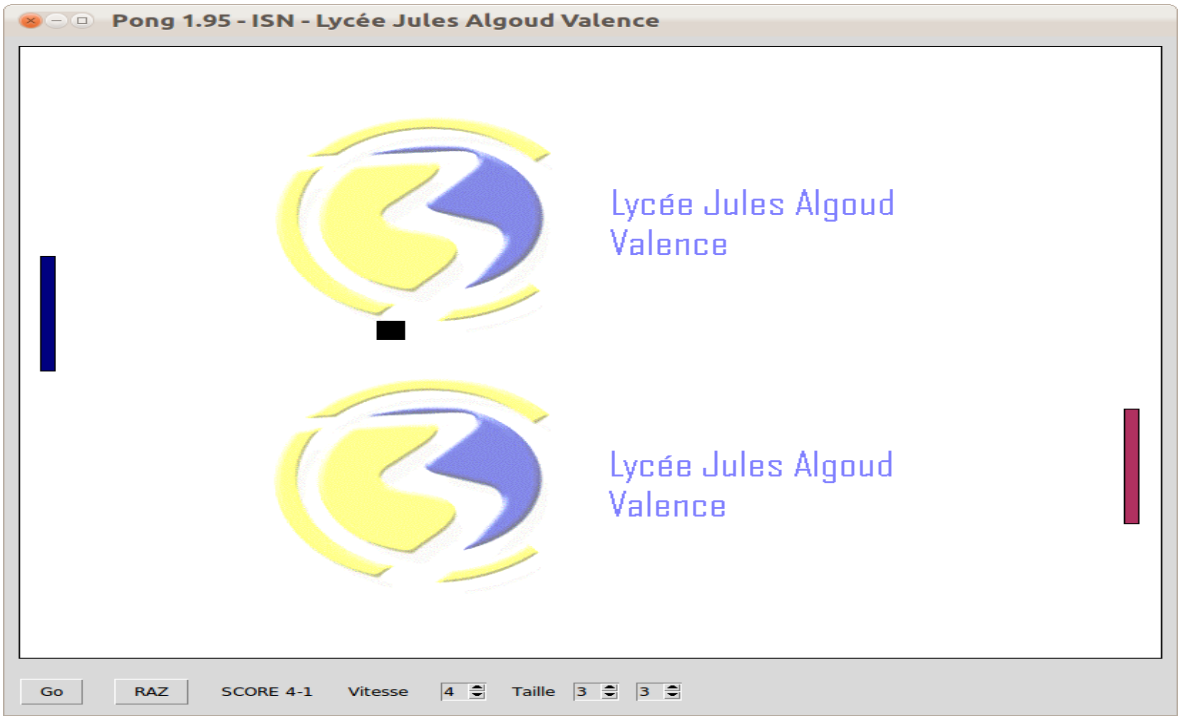
Projet n°3 : Jeu de SnakeDuel

Un jeu qui se joue à deux, ou seul contre l'ordinateur.



Projet n°4 : Jeu de Pong

Un jeu qui se joue à deux.



Programmes exécutables pour Windows

Pas besoin d'avoir Python sur votre machine !

Les programmes exécutables (extension .exe) des exercices et projets sont téléchargeables ici (7 Mo).
Décompresser ensuite l'archive.

Pour jouer à Tetris (par exemple), lancer le programme tk_Tetris.exe
Plus d'informations sur les jeux ici.
Have fun !

Remarques

- Testé avec succès sous Windows XP, Windows 7 et ... Linux/Ubuntu (avec l'émulateur Wine).

Webographie

- fr.wikibooks.org
- python.developpez.com
- docs.python.org
- infohost.nmt.edu
- effbot.org
- epydoc.sourceforge.net
- www.tutorialspoint.com
- lmgtfy.com
- Module pygame.mixer.Sound (documentation)
- Module pygame.mixer.music (documentation)