

Partie 47 : Graphical User Interface (GUI)

Python : "tkinter"

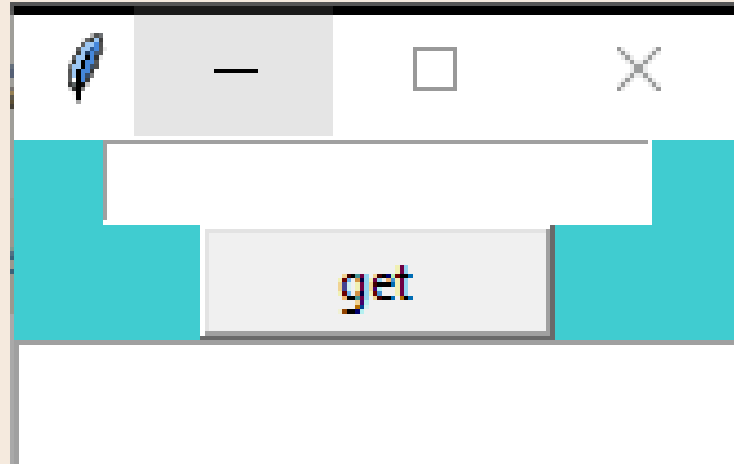
47. Graphical User Interface (GUI)

- 47-03 : widgets de base
- 47-05 : Model View Controller
- 47-07 : widgets avancés
- 47-13 : widgets animés
- 47-23 : widgets de base de données

Principes généraux

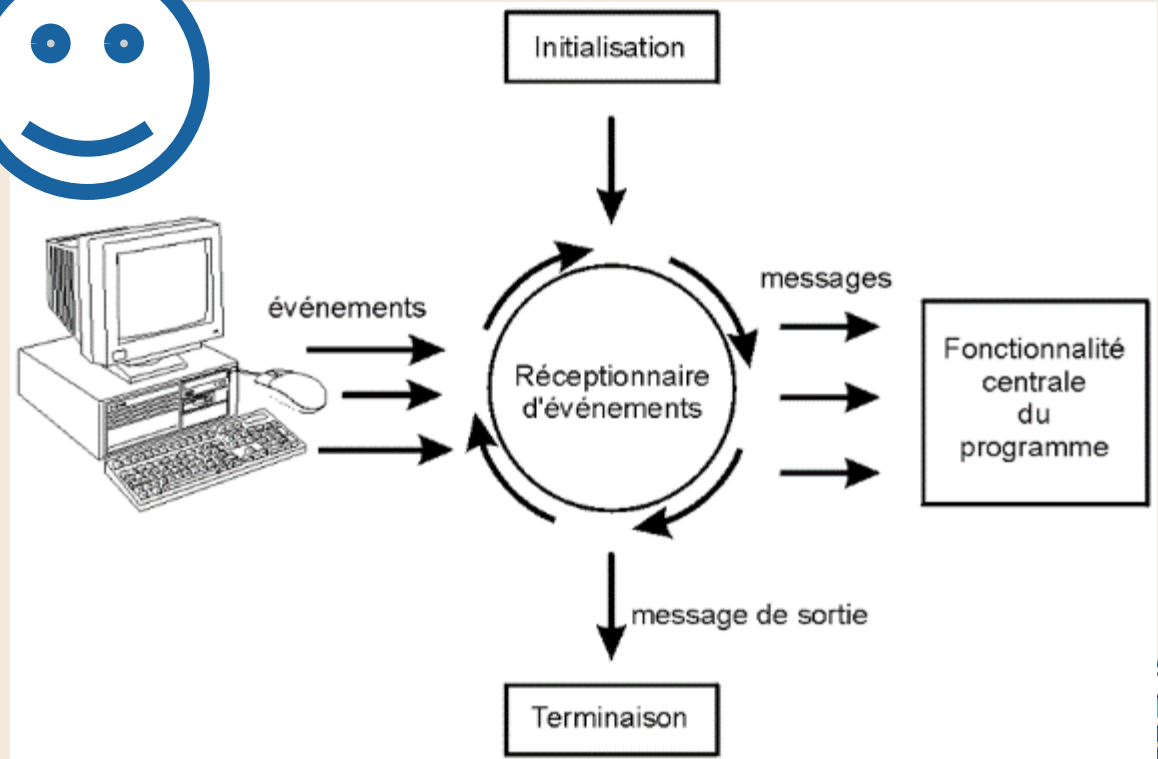
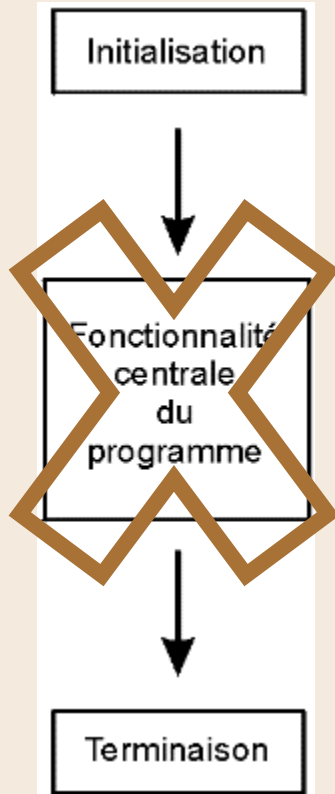
- Programmation événementielle ("event-driven")
- Widgets du GUI
- Callback
- Architecture du code
- Bibliographie :
 - <https://python.developpez.com/cours/TutoSwinnen/?page=Chapitre8>
 - <https://www.youtube.com/watch?v=HWxBtxPBCAc&t=119s>

Programmation événementielle



Fichier *07-01-sample.py*

Programmation événementielle



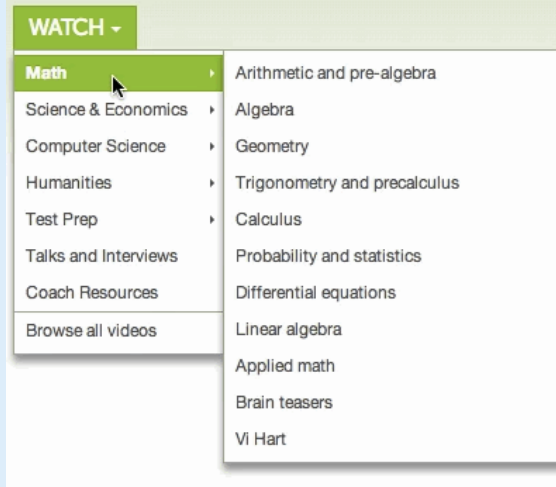
Callback : "Command"

- Callback = fonction appelée lorsque l'évènement arrive
 - càd lorsque le widget est activé
 - un bouton est pressé
 - la souris quitte une fenêtre, etc.
 - Python : le callback s'appelle **command**
- Procédure à suivre :
 1. Ecrire la fonction **py**
 2. L'associer au widget

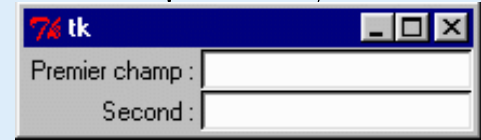
Chapitre 47-03 : widgets de base

Principaux Widgets de Tkinter

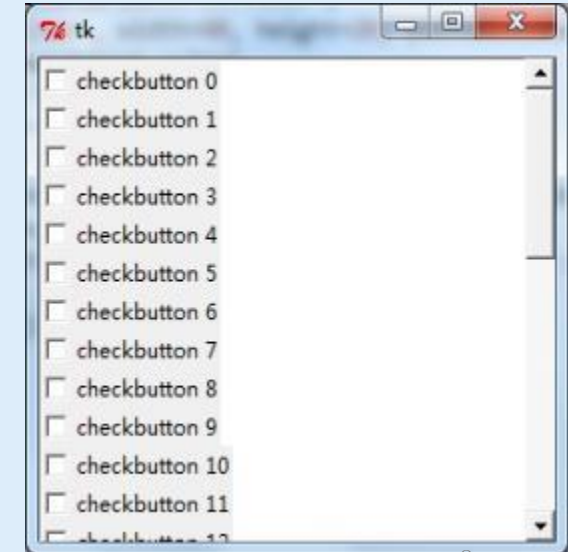
Menu, Menubutton



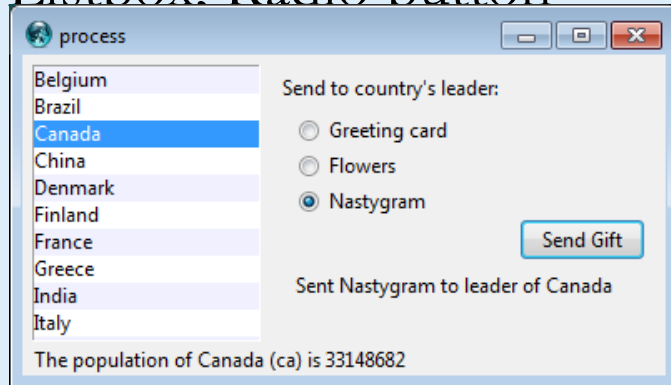
Label, Entry



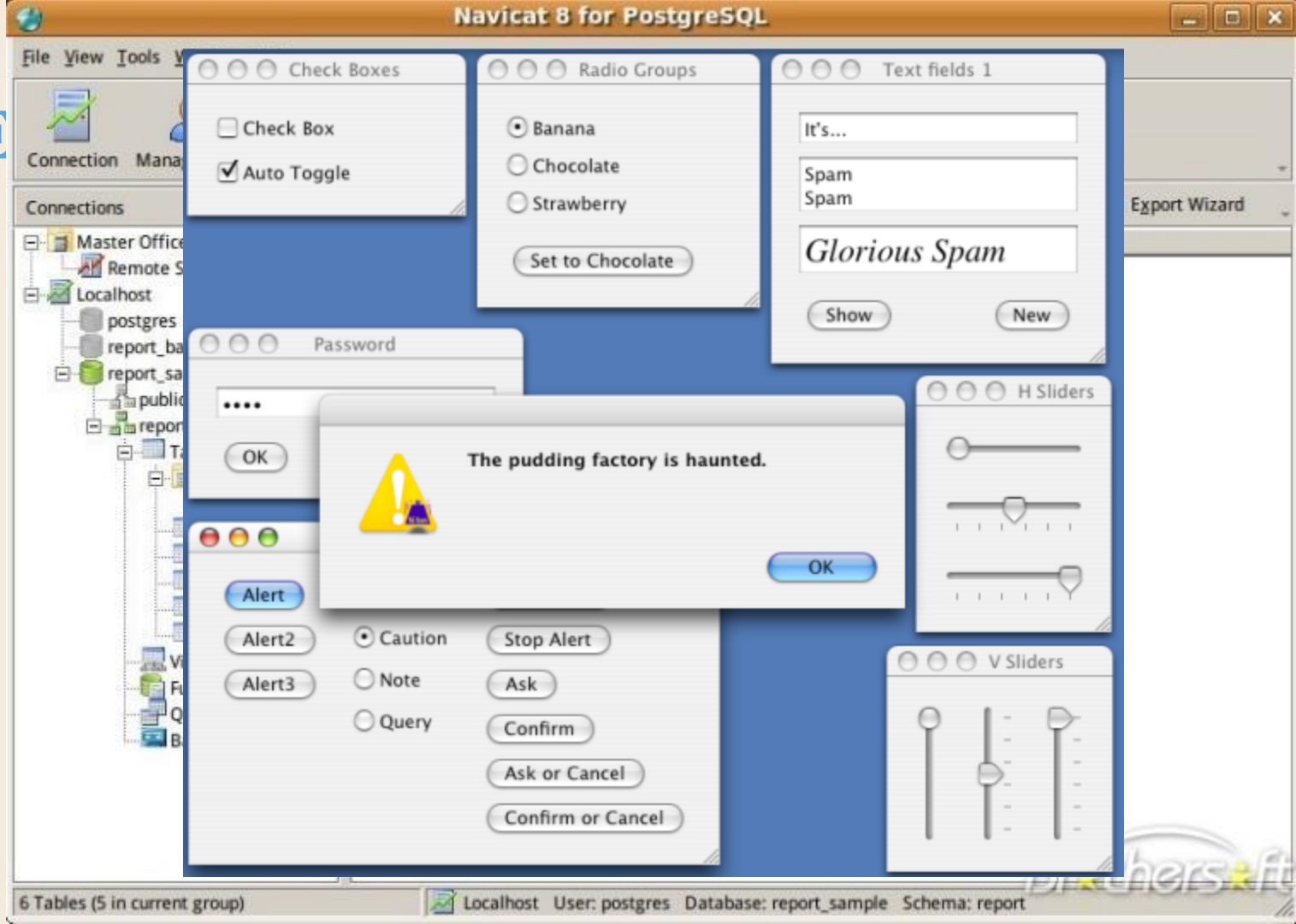
Check button



Listbox, Radio button

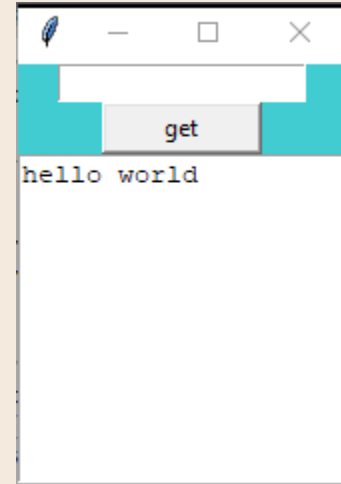
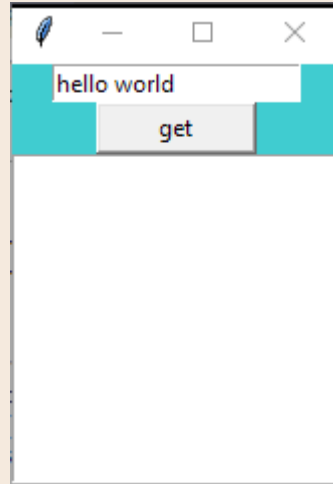
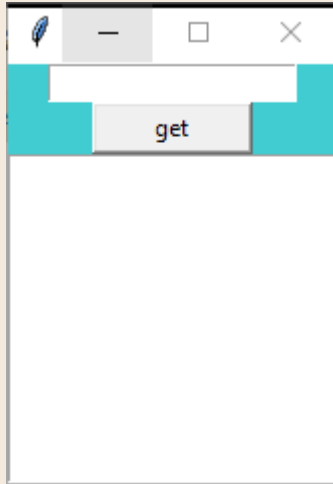


E



BUROTIX 0

Exo 47-03-01 : architecture générale



Fichier *47-03-01-sample.py*

Exo 47-03-01 : architecture générale

- importer le module tkinter

```
from tkinter import *
```

- ouvrir une fenêtre

```
window = Tk()
```

- créer une "entry", l'afficher et y placer le curseur de la souris

```
e = Entry(window)
e.pack()
e.focus_set()
```

- créer une zone pour afficher du texte

```
t = Text(window, height=10,
          width=20 )
t.pack()
```

- définir la commande pour le bouton

```
def callback():
    t.insert(END, e.get()+"\n" )
    e.delete(0, END)
```

- créer un bouton

```
b = Button( window, text="get",
            width=10, command=callback )
b.pack()
```

- que le programme reste à l'écoute des événements :

```
window.mainloop()
```

Fichier *07-01-sample.py*

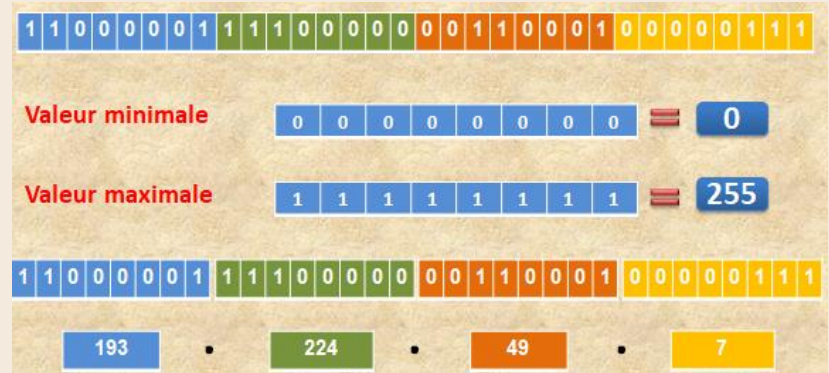
Exo 47-03-02 : la calculatrice

- Reproduisez les fonctionnalités de la calculatrice ci-contre
- Première étape : uniquement
 - Les chiffres 1, 2, 3
 - Les opérations +, = et C (clear)
- Deuxième étape
 - Les chiffres 4, 5, 6
 - Les opérations M+, M-, MR, MC
 - Les opérations $\frac{1}{x}$, et \sqrt{x}
- Troisième étape
 - Les chiffres 7, 8, 9
 - Les opérations -, *, /



Exo 47-03-03 : le convertisseur d'adresse IP

- Problème posé :
 - L'utilisateur entre une adresse IP en valeurs décimales
193.224.49.7
 - L'application affiche la même valeur en binaire
11000001.11100000.
00110001.00000111



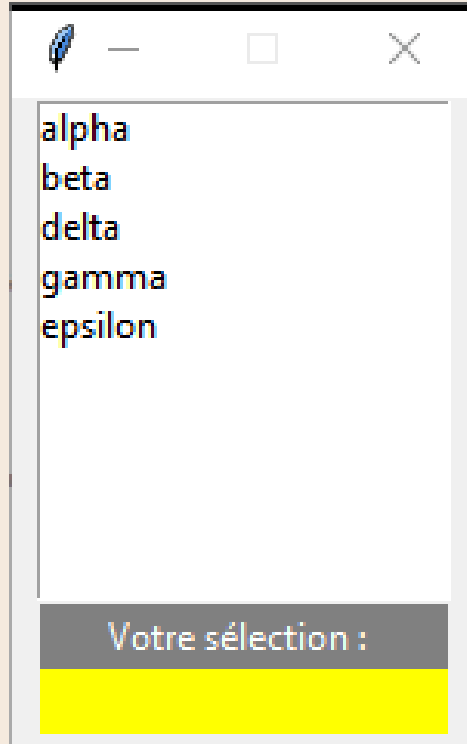
Exo 47-03-04 : combobox, sample



The screenshot shows a web browser window with the title "Combobox - sample". The page has a teal background and contains the following elements:

- A heading "choisissez votre QUIZZ" in white text.
- A dropdown menu (combobox) with the text "Les chats" and a downward arrow.
- A label "journal des évènements" in white text.
- A text box containing the text "Les chats".
- A button labeled "Réinitialiser".

Exo 47-03-05 : listbox, sample



Chapitre 47-05 : Model View Controller

Modèle

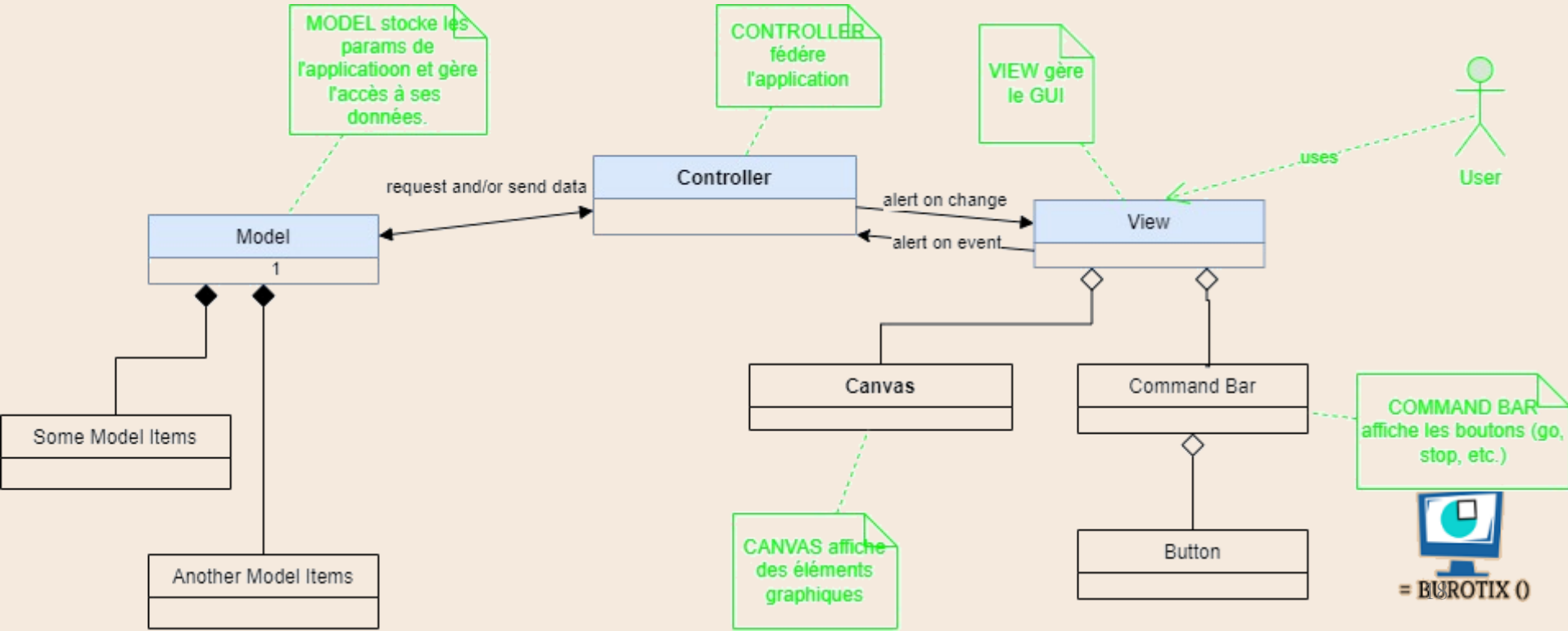
Avantages

Vue

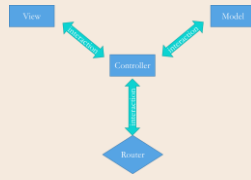
Inconvénients

Contrôleur

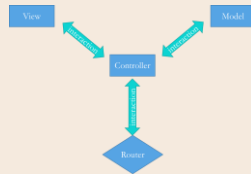
Architecture MVC (desktop)



Model

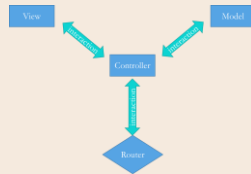


- Indépendant des autres modules
- Gérer toutes les données
- Gérer la logique des données
 - Validation
 - Lecture
 - Enregistrement
- Interagir avec **Controller**
 - **Controller** paramétrise les requêtes pour SELECT
 - **Controller** fournit les données pour UPDATE, INSERT, ..
- Exemple : application bancaire
 - Fichier des clients
 - Liste des dépôts
 - Vérification : les retraits ne dépassent pas la limite de crédit



View

- Indépendant des autres modules
 - Présenter les données via des éléments visuels
 - Texte, Table, Graphique, ...
 - Balises HTML, Javascript, CSS
 - Tkinter, Canvas
 - Préparer la mise à jour des données
- Interagir avec **Controller**
 - **Controller** envoie les données à **View**
 - **View** met en forme les données reçues



Contrôleur / Controller

- Dépendant de Model et View
- Traiter les actions de Router
 - Callbacks
 - Validation des données d'un formulaire
- Modifier Model
 - Mise à jour (UPDATE)
 - Ajout (INSERT)
 - Suppression (DELETE)
- Modifier View
 - Suite aux modifications de Model
 - Suite aux actions de Router

Avantages

■ Maintenance aisée du code

- code plus facile à étendre et à développer
- composant Model testable séparément de l'utilisateur
- prise en charge facilitée de nouveaux types de web clients
- différents composants développables en parallèle.
- migration de base de données facilitée

■ Réduction de la complexité

- division de l'application (en modèle, vue et contrôleur)
- router unique traitant les requêtes de l'utilisateur

- séparation de business logic et UI logic

■ Meilleur support pour le test-driven development (TDD)

- Modularité et indépendance des classes et des objets, donc testables séparément.

■ Bien adapté aux applications web complexes

- développement par de grandes équipes de concepteurs et de développeurs
- facilitation de Search Engine Optimization (SEO)
- exploitation et amélioration des fonctionnalités proposées par ASP.NET, JSP, Django, etc.

Inconvénients

- Code difficile à lire, à modifier, à tester unitairement et à réutiliser par un développeur extérieur.
 - augmentation du nombre de lignes de code
- Navigation dans le framework parfois complexe.
 - introduction de plusieurs couches d'abstraction obligeant les utilisateurs à s'adapter à l'architecture du MVC.
 - appropriation de l'application ralentie à cause de l'apprentissage
- Difficulté d'appliquer MVC dans un UI moderne
 - architecture non adaptée aux SPA
 - MVC orienté serveur, non orienté client
- Difficulté pour plusieurs programmeurs de mener une programmation parallèle.
- Connaissance de plusieurs technologies nécessaire.

Chapitre 47-07 : widgets avancés

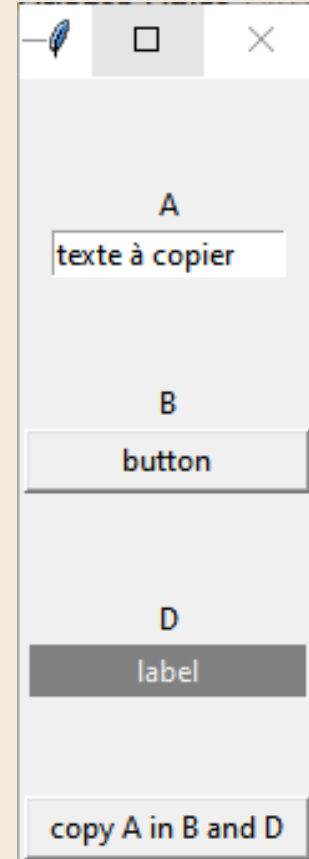
Pour les plus curieux ...

Autres évènements

- Pour lier des fonctions à d'autres évènements que le "click"
 - double-click, autres boutons de la souris, etc.
 - 47-07-12-event.py

GUI dynamique

- Pour rendre un GUI plus dynamique
 - modifier les labels
 - 47-07-13-gui_dynamique.py



Touches du clavier

- Pour associer une touche clavier à une fonction, de la même façon qu'un bouton.
 - 47-07-17-KeyBinding_sample.py

```
.bind( "<KeyPress>", command )
```

Insérer une image dans un form

- Pour insérer une image dans l'interface utilisateur
 - 47-07-21-insert_image.py
 - programmation.gif



Insérer une image animée dans un form

- Pour insérer une image dans l'interface utilisateur
 - 47-07-23-insert_anim.py
 - tenor.gif



Gestion de fenêtres multiples

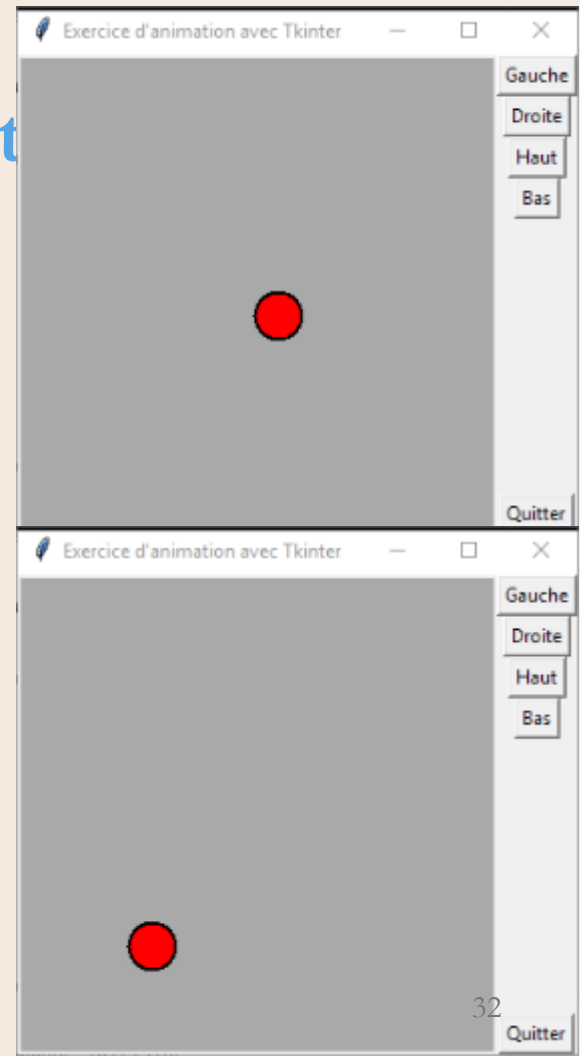
- approche par création et destruction
 - hard
 - 47-07-35-multiple_windows.zip
- approche par show/hide
 - soft et plus simple

Chapitre 47-13 : widgets animés

Jouons au ballon ... ☺

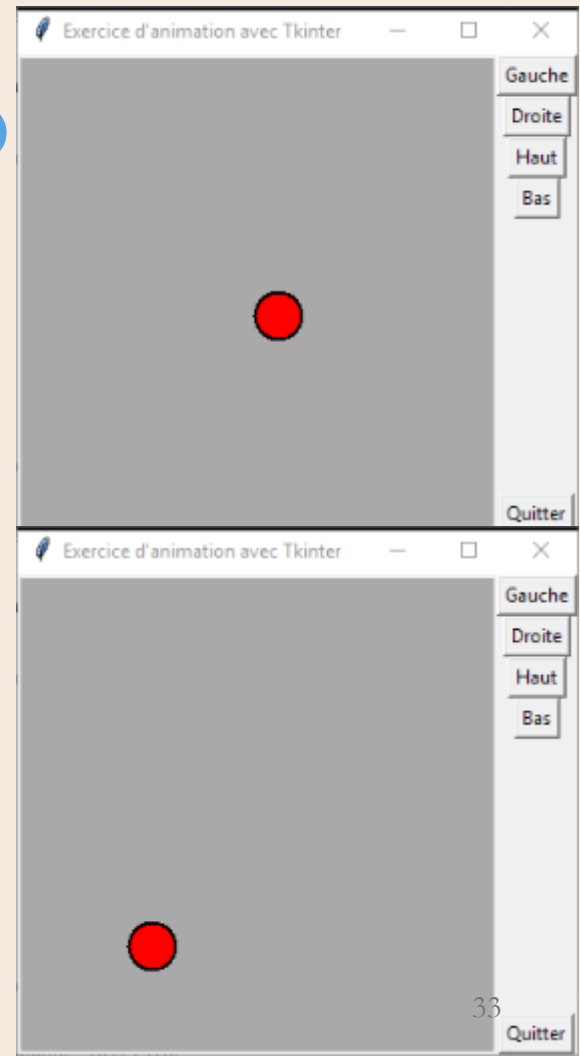
Exo 47-13-11 : la balle mouvante

- Fichier de départ
47-13-11_balle_start.py
- Ouvrez le fichier, faites tourner le programme et observez le code.
- Ajoutez les fonctionnalités suivantes :
 - Mouvoir la balle en diagonale
 - Paramétrer le saut et la couleur de la balle
 - Réinitialiser la balle en sa position centrale
- Pour les geeks :
 - Faire circuler la balle sur un carré
 - Faire circuler la balle sur un cercle



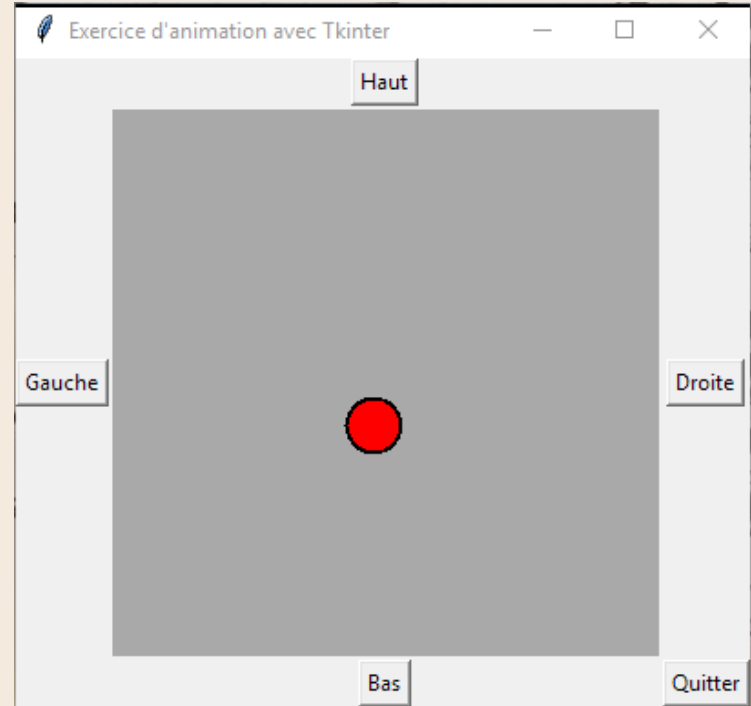
Exo 47-13-21 : la balle en POO

- Fichier de départ
47-13-11_balle_start.py
- Réécrivez le code sous forme de classe
 - un objet "ballon" est créé
 - les boutons "gauche", "droite", etc. donnent les ordres (callback) à cet objet
 - supprimez les variables globales



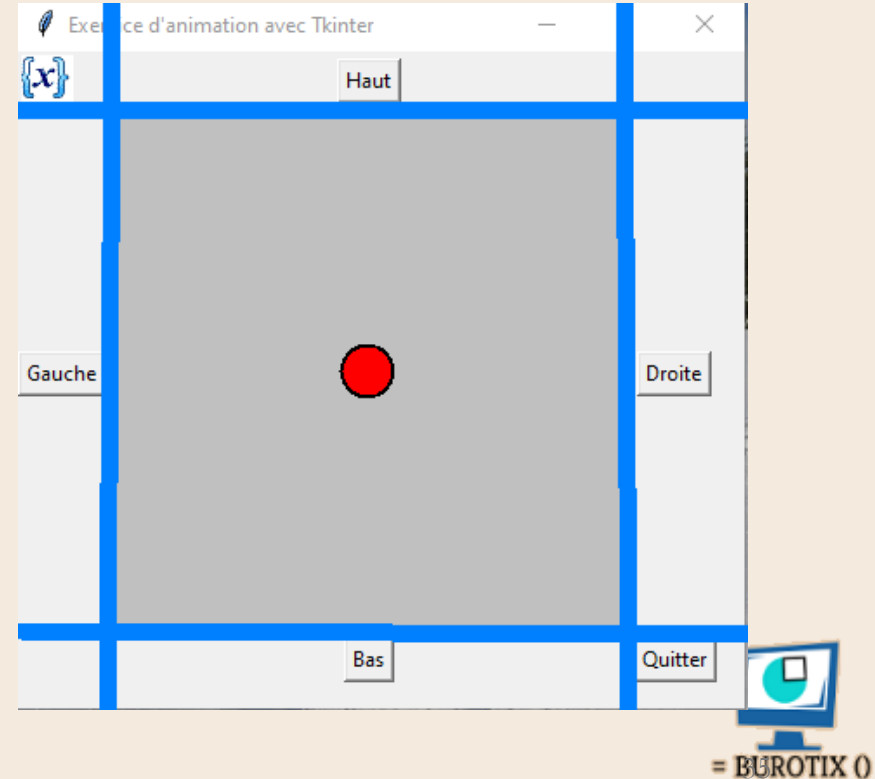
Exo 47-13-24 : la balle en mode "grille" (1/2)

- Reprenez l'exercice précédent
- Placez les boutons autour du canvas.
 - "gauche" à gauche
 - "droite" à droite, etc.
- Tuyaux:
 - Ne placez pas les boutons sur le canvas même.
 - Utilisez `grid()` au lieu de `pack()`



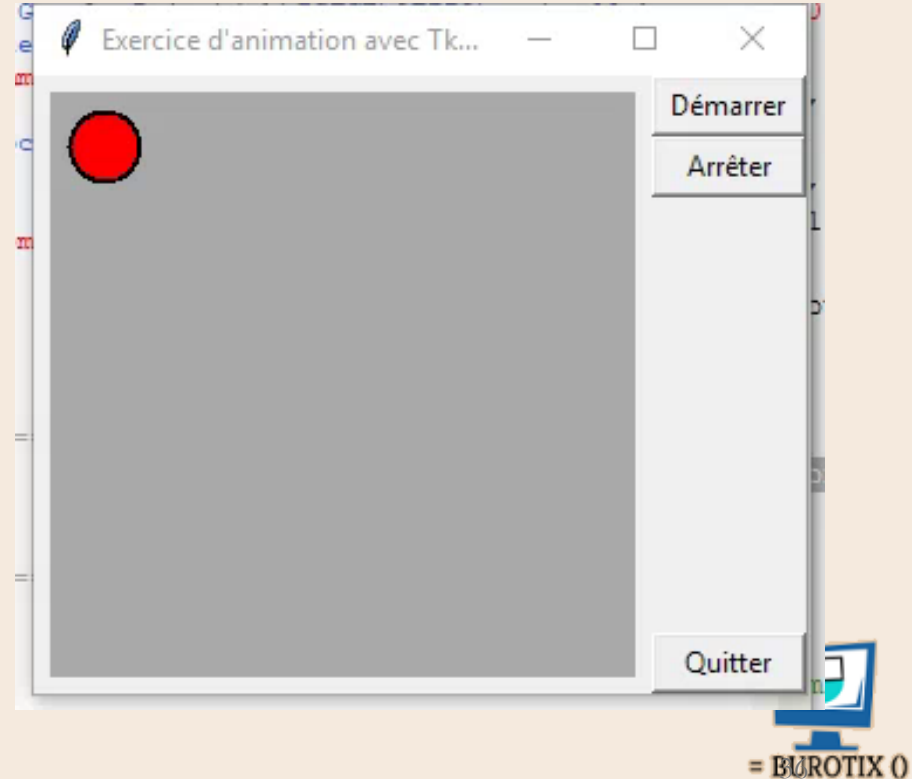
Exo 47-13-24 : la balle en mode "grille" (1/2)

- En mode "grille", on découpe le GUI en colonnes et rangées
- Exemple avec 3 colonnes et 3 rangées
 - L'image se trouve en haut à gauche
 - `row = 0`
 - `column = 0`
 - Le bouton "droite" se trouve en
 - `row = 1`
 - `column = 2`
- Code :
 - `.grid(row=1, column=2)`



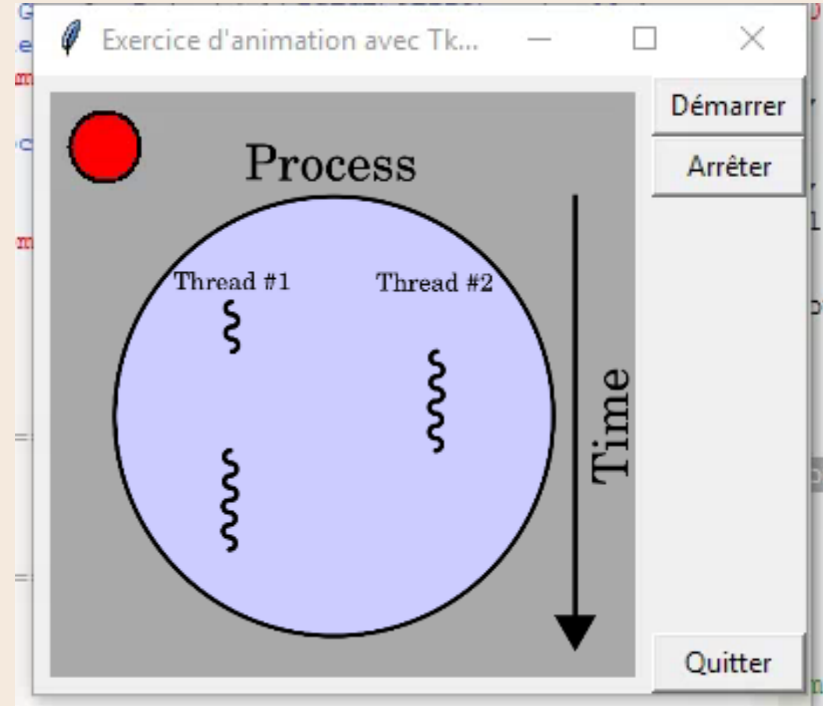
Exo 47-13-31 : la balle automatique (1/2)

- Reprenez l'exercice précédent
- Introduisez un mouvement automatique
 - démarrant au bouton "démarrer"
 - s'arrêtant au bouton "arrêter"



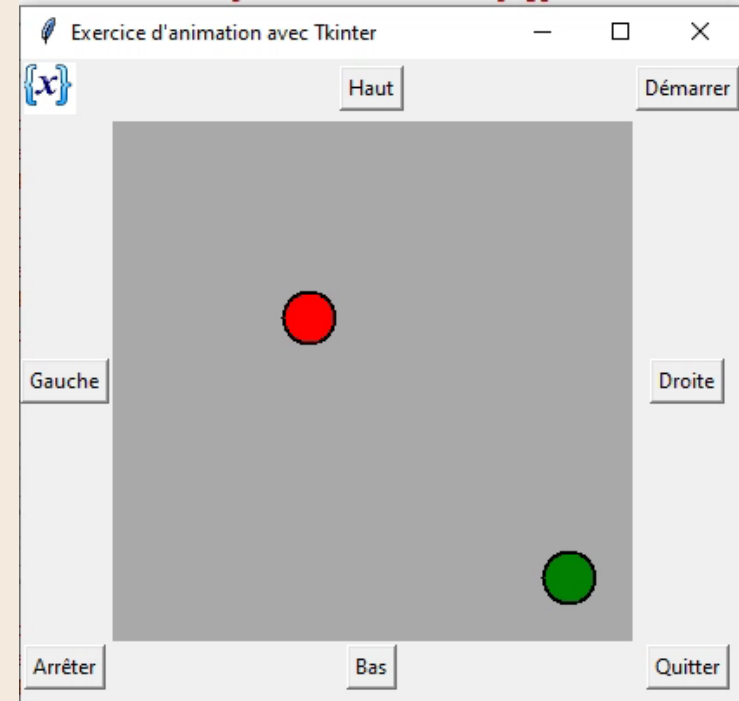
Exo 47-13-31 : la balle automatique (2/2)

- Considérez le corrigé fourni
 - Fichier "07-31-balle+auto.py"
 - Cliquez plusieurs fois sur "démarez".
 - Que constatez-vous ?
 - Expliquez ce bug.
 - [https://fr.wikipedia.org/wiki/Thread_\(informatique\)](https://fr.wikipedia.org/wiki/Thread_(informatique))
 - [https://en.wikipedia.org/wiki/Multithreading_\(computer_architecture\)](https://en.wikipedia.org/wiki/Multithreading_(computer_architecture))



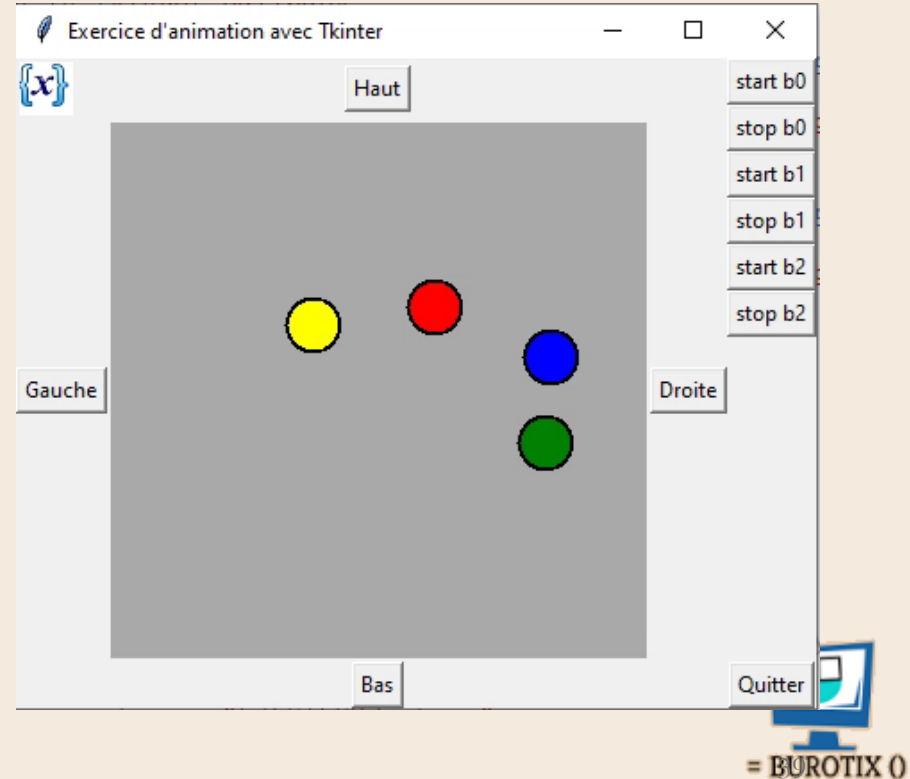
Exo 47-13-41 : deux balles

- Reprenez les exercices précédents
- Intégrez la balle manuelle et la balle automatique au sein de la même application. Cf video.



Exo 47-13-51 : plusieurs balles

- Un nombre indéterminé de balles : trois, quatre, ..., N.
- Détection de collision entre les balles et les murs
- Détection de collision des balles entre elles
- POO complète, avec héritage



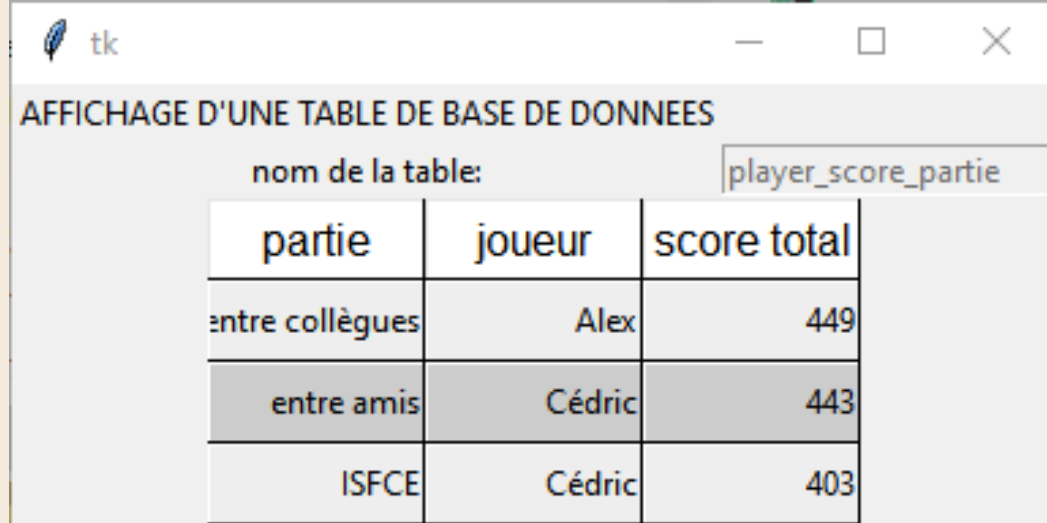
Exo 47-13-69 : la balle mouvante

- Reprenez l'exercice précédent.
- Ajoutez les fonctionnalités suivantes (de simple à complexe)
 - Paramétrer la couleur des balles
 - Réinitialiser les balles en leur position centrale
 - Faire circuler les balles sur un autre parcours de votre choix
 - Faire circuler la balle sur un cercle

Chapitre 47-23 : widgets base de données

widgets de base de données

- Approche par les widgets de base
 - combinaison de **Frame**, **Label**, **.pack()**, etc.
 - 47-23-11-database_label_grid.py



The screenshot shows a Tkinter window with the title bar 'tk'. The window content is titled 'AFFICHAGE D'UNE TABLE DE BASE DE DONNEES'. Below the title, there is a label 'nom de la table:' followed by a text entry field containing 'player_score_partie'. Below this, a table is displayed with three columns: 'partie', 'joueur', and 'score total'. The table has three data rows. The first row has 'entre collègues', 'Alex', and '449'. The second row has 'entre amis', 'Cédric', and '443'. The third row has 'ISFCE', 'Cédric', and '403'.

partie	joueur	score total
entre collègues	Alex	449
entre amis	Cédric	443
ISFCE	Cédric	403

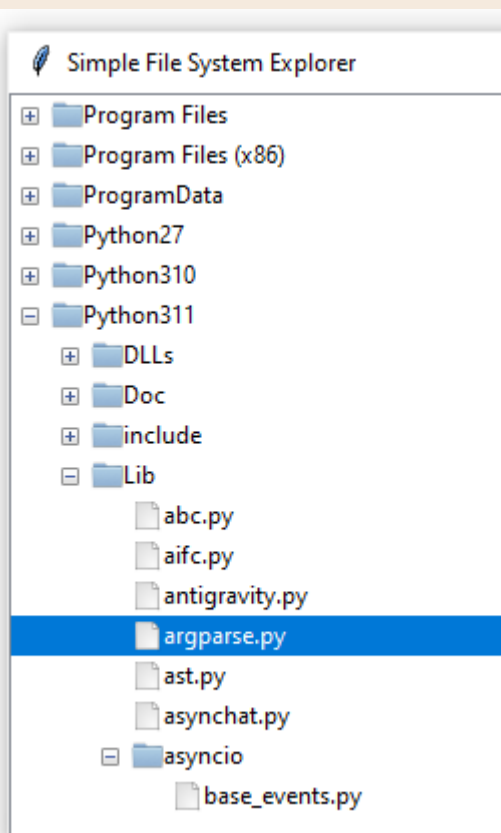
widgets de base de données

- Approche par les widgets avancés
 - utilisation de **Ttk** et **TreeView**
 - 47-23-13-database_treeview.py



LA PARTIE	LE JOUEUR	LE SCORE
entre collègues	Alex	449
entre amis	Cédric	443
ISFCE	Cédric	403
entre collègues	Bertrand	396
entre amis	Bertrand	375

Ttk TreeView



	Size	Modified
widgets	25KB	Yesterday
gallery	2KB	Two weeks ago
resources	220KB	Three weeks ago
tutorial	2.1MB	Ten minutes ago
canvas	18KB	Last week
tree	5KB	Ten minutes ago
text	12KB	Yesterday

Les couleurs en notation hexadécimale

Les couleurs en hexadecimal

- Le système décimal est un système de numération en base 10.
- Le système hexadécimal est un système de numération en base 16.

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Les couleurs en hexadecimal

- Que valent les nombres hexadécimaux suivants dans le système décimal ?
 - 4
 - B
 - A6
 - FF
 - 10

Les couleurs en hexadécimal

- Augmenter la valeur d'une couleur, augmente son intensité.

Vert
#00FF00
Rouge Bleu

Les noms de couleurs

- Couleurs standards :
 - **white** (#FFFFFF)
 - **black** (#000000)
 - **red** (#FF0000)
 - **green** (#00FF00)
 - **blue** (#0000FF)
- 140 noms (prédéfinis) de couleurs :
 - https://www.w3schools.com/colors/colors_names.asp