Programmation Python

2022/2023

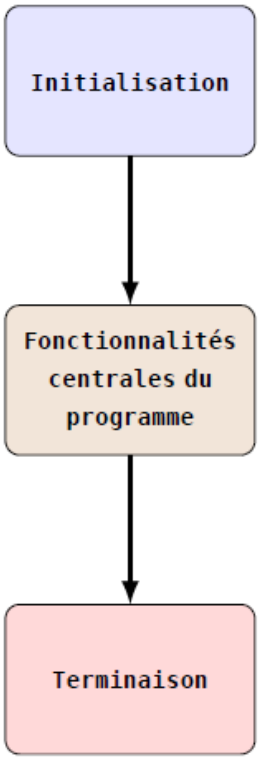
Prof : Amina BENGAG

Programmation Python

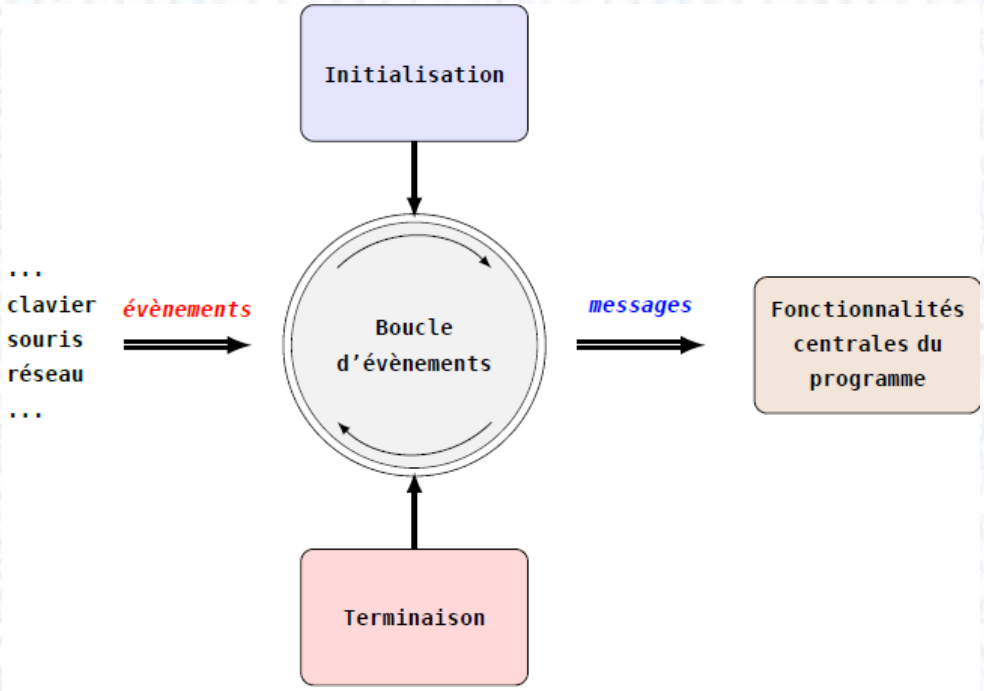
Chapitre 7: Des Interfaces graphiques avec Tkinter

I. Module Tkinter

- ✓ **Disponible par défaut, il offre un moyen pour créer des interfaces graphiques via Python.**
- ✓ **En programmation graphique objet, on remplace le déroulement séquentiel du script par une boucle d'événements**



(a) Programmation séquentielle



(b) Programmation pilotée par une boucle d'évènements

I. Module Tkinter

✓ **Exemple : l'affichage d'un Label**

```
"""Premier exemple avec Tkinter.
On crée une fenêtre simple qui souhaite la bienvenue à
l'utilisateur.
"""
# On importe Tkinter
from tkinter import *
# On crée une fenêtre, racine de notre interface
fenetre = Tk()
# On crée un label (ligne de texte) souhaitant la bienvenue
# Note : le premier paramètre passé au constructeur de Label est notre
# interface racine
champ_label = Label(fenetre, text="Bienvenue monde graphique!")
# On affiche le label dans la fenêtre
champ_label.pack()
# On démarre la boucle Tkinter qui s'interompt quand on ferme la fenêtre
fenetre.mainloop()
```

I. Module Tkinter

1. Widgets

✓ **Les objets graphiques (boutons, champs de texte, cases à cocher, barres de progression…) sont appelés des widgets.**

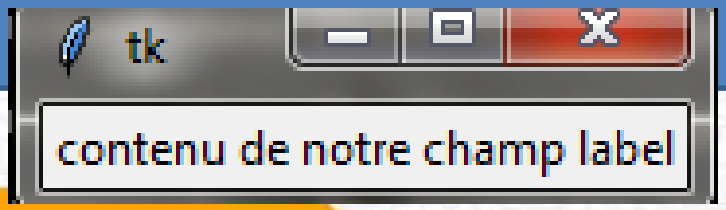
✓ **pour qu'un widget apparaisse, il faut :**

- **qu'il prenne, en premier paramètre du constructeur, la fenêtre principale ;**
- **qu'il fasse appel à la méthode pack.**

a. Les labels

Utilisés pour afficher du texte dans une fenêtre, du texte qui ne sera pas modifié par l'utilisateur.

```
champ_label = Label(fenetre, text="contenu de notre champ label")
champ_label.pack()
```



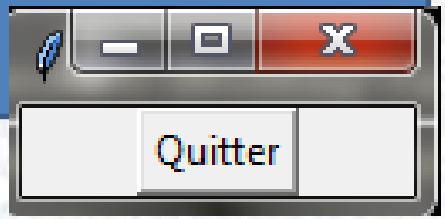
I. Module Tkinter

1. Widgets

b. Les boutons

Les boutons sont des widgets sur lesquels on peut cliquer et qui peuvent déclencher des **actions** ou **commandes**

```
bouton_quitter = Button( fenetre, text="Quitter", command=fenetre.quit)
bouton_quitter.pack()
```



I. Module Tkinter

1. Widgets

c. Une ligne de saisie

C'est une zone de texte dans lequel l'utilisateur peut écrire. En fait, il s'agit d'une ligne simple.

Peut **servir** à la récupération des données saisie par l'utilisateur afin les utiliser comme variables (formulaire d'inscription, formulaire d'identification...).

```
var_texte = StringVar()
ligne_texte = Entry( fenetre, textvariable=var_texte, width=90)
ligne_texte.pack()
```

I. Module Tkinter

1. Widgets

c. Une ligne de saisie

- Liste des options pour le widget **Entry** :
  - bg : définit la couleur d'arrière-plan du widget Entry.
  - bd : définit la taille des bordures. La valeur par défaut est 2 pixels.
  - command : définit et associe une commande
  - cursor : définit le motif du curseur de la souris
  - font : définit la police qui sera utilisée pour le texte.
  - fg : définit la couleur qui sera utilisée pour le texte
  - width : définit la largeur du widget
  - …

I. Module Tkinter

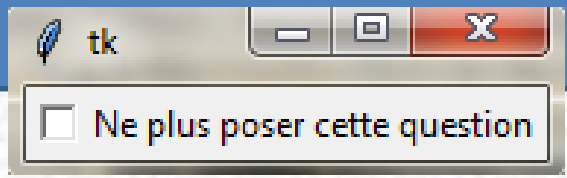
1. Widgets

d. Les cases à cocher

Les cases à cocher sont définies dans la classe **Checkbutton**.

Pour surveiller l'état d'une case à cocher (qui peut être soit active soit inactive), on préférera créer une variable de type **IntVar** plutôt que **StringVar**.

```
var_choix = IntVar()
choix = Checkbutton( fenetre, text="Ne plus poser cette question", variable=var_choix)
choix.pack()
```



Pour récupérer la valeur associée au bouton actuellement sélectionné :

```
var_choix.get()
```

I. Module Tkinter

1. Widgets

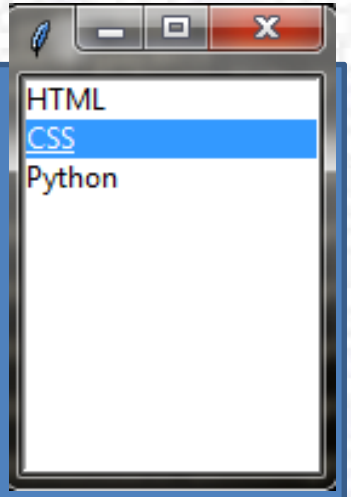
e. Les listes déroulantes

Ce widget permet de construire une **liste** dans laquelle on peut sélectionner un ou plusieurs éléments.

On insère ensuite des **éléments**. La méthode **insert** prend deux paramètres :

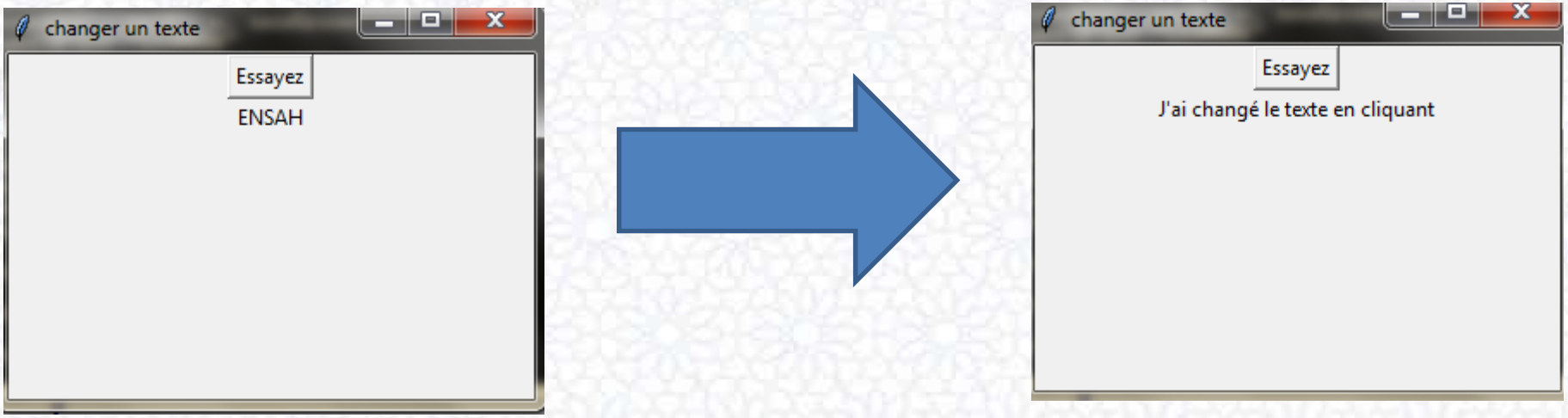
1. la position à laquelle insérer l'élément ;
2. l'élément même, sous la forme d'une chaîne de caractères.

```
liste = Listbox(fenetre)
liste.pack()
liste.insert(END, "HTML")
liste.insert(END, "CSS")
liste.insert(END, "Python")
```



I. Module Tkinter

✓ **Exemple** changer le texte via une action associé à un bouton de commande



I. Module Tkinter

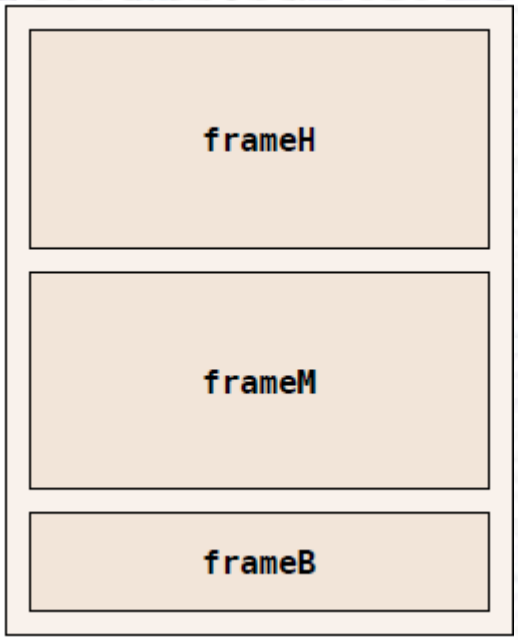
✓ **Exemple** changer le texte via une action associé à un bouton de commande

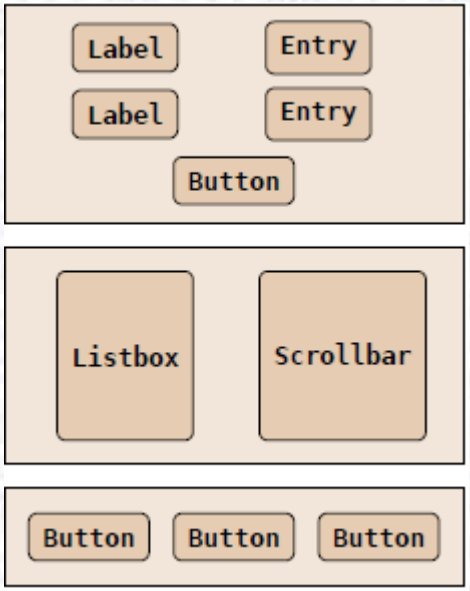
```
from tkinter import *
fenetre = Tk()
# le titre de la fenetre
fenetre.title("changer un texte")
# la taille de la fenetre
fenetre.geometry("400x400")

def action ():
    var.set("J'ai changé le texte en cliquant")

var = StringVar ()
label = Label ( fenetre , textvariable=var )
var .set( " ENSAH " )

b = Button( fenetre , text = "Essayez" , command=action )
b.pack()
label.pack()
fenetre.mainloop ()
```

I. Module Tkinter

2. Frame

Il existe plusieurs widgets qui peuvent contenir d'autres widgets. L'un d'entre eux se nomme **Frame**. C'est un cadre rectangulaire dans lequel on pout placer nos **widgets,** ainsi que d'autres **objets Frame**.





Syntaxe:

```
cadre =  Frame ( fenetre, option, ...)
```

I. Module Tkinter

2. Frame

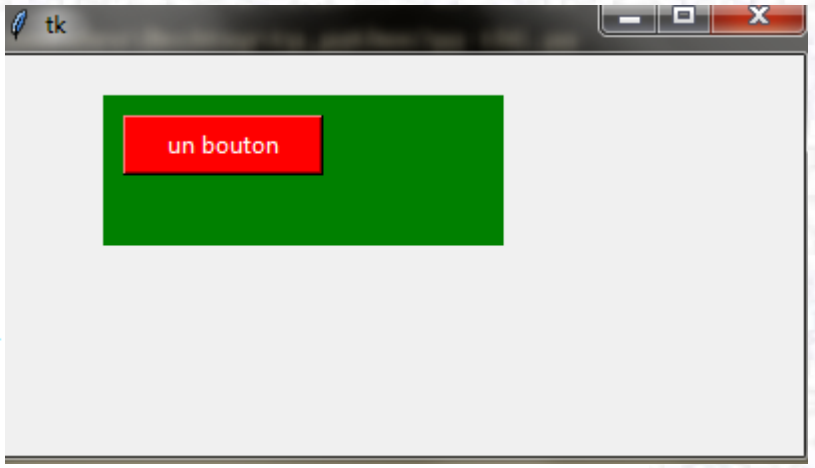
Liste des options d'un widget Frame:

- **bg : couleur d'arrière-plan du widget Frame**
- **bd : taille des bordures autour du Frame.**
- **cursor : permet de personnaliser le motif du curseur de la souris au moment du survole.**
- **height : définit la dimension verticale du Frame.**
- **highlightbackground : définit la couleur de la mise au point en surbrillance de l'objet Frame.**
- **width : définit la largeur du frame**

I. Module Tkinter

2. Frame

Si vous voulez qu'un widget apparaisse dans un cadre, utilisez le Frame comme parent à la création du widget :




```
from tkinter import *

 # Création de la fenêtre principale
master = Tk()
master.geometry( "400x200" )

 #Création d'un frame d'arrière plan
cadre = Frame(master , bg="green" )

 # Emplacement du frame
cadre.place (x=50, y=20, width=200,height=75)
# cadre.pack()

 # Création d 'un bouton au sein du frame
b = Button(cadre , text="un bouton" ,bg="red" , fg="white" )
b.place (x=10, y=10, width=100,height=30)
# b.pack()
master.mainloop()
```

I. Module Tkinter

3. Les méthodes de gestion des dispositions géométriques des widget

Tous les widgets Tkinter ont accès aux méthodes de gestion de géométrie spécifiques, qui ont pour but d'organiser les widgets dans la zone du widget parent.

Tkinter possède les classes de gestionnaire de géométrie suivantes :

1. **La méthode pack()** : Les widgets sont placés **l'un au dessous** de l'autre selon l'ordre d'application de la méthode pack() avec un emplacement **centré par défaut**.

2. **La méthode grid()** : organise les widgets dans une structure de **type table** dans le widget parent.

3. **La méthode place()** : organise les widgets en les plaçant à des positions spécifiques dans le widget parent suivant leurs **coordonnées** et leurs **dimensions**

I. Module Tkinter

3. Les méthodes de gestion des dispositions géométriques des widget

✓ Exemple : la méthode grid()

```
from tkinter import *

 #Création d 'une fenêtre
f = Tk()
f .geometry( "350x100" )

#Création des widgets
Nom = Label ( f , text = "Saisissez votre nom : " )
Prenom = Label ( f , text="Saisissez votre prénom : " )
champNom = Entry ( f )
champPrenom = Entry ( f )

#Application de la méthode grid () aux widget
Nom. grid (row=0, column=0)
Prenom. grid (row=1, column=0)
champNom. grid (row=0, column=1)
champPrenom. grid (row=1, column=1)
f .mainloop ()
```