UAA4 Projet de programmation impérative python en utilisant la librairie graphique tkinter

0 Introduction

Bibliographie

GUI et gestion d'évènements

Tkinter (Tool Kit INTERface) est le module graphique d'origine pour python permettant la création d'interfaces graphiques.

Lorsque l'on développe une **GUI** (**Graphical User Interface**), nous créons une fenêtre graphique contenant notre application, ainsi que des ***widgets*** inclus dans la fenêtre. Les ***widgets*** (***WInDow GadgET***) sont des objets graphiques permettant à l'utilisateur d'interagir avec le programme Python de manière conviviale.

L'utilisation d'une GUI va amener une nouvelle manière d'aborder le déroulement d'un programme, il s'agit de la programmation dite « **événementielle** ». Jusqu'à maintenant vous avez programmé « linéairement », c'est-à-dire que les instructions du programme principal s'enchaînaient les unes derrière les autres (avec bien sûr de possibles appels à des fonctions. Avec une GUI, **l'exécution est décidée par l'utilisateur en fonction de ses interactions avec les différents *widgets***. Comme c'est l'utilisateur qui décide quand et où il clique dans l'interface, il va falloir mettre en place ce qu'on appelle un « gestionnaire d'événements ».

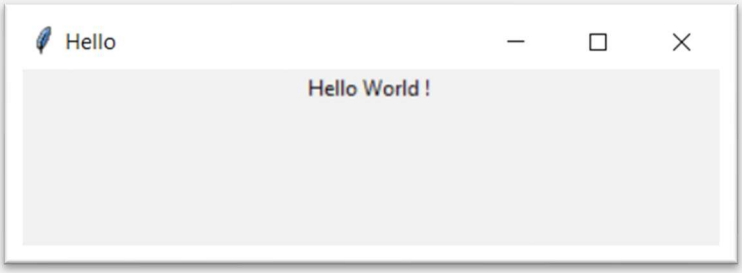
Le **gestionnaire d'événements** est une sorte de « **boucle infinie** » qui est à l'affût de la moindre action de la part de l'utilisateur. C'est lui qui effectuera une action lors de l'interaction de l'utilisateur avec chaque *widget* de la GUI. Ainsi, l'exécution du programme sera réellement guidée par les actions de l'utilisateur.

1 Prise en main de tkinter

1.1 Test1 - Hello world, affichage de texte: widget Label

Crée un fichier tkinter_1_hello_world_Text.py, ouvre-le avec visual studio et copie le code expliqué ci-dessous.

Résultat avec l'application de bureau



- ✓ Importe la librairie tkinter

```
from tkinter import *
```

- ✓ Crée une variable **fenetre** qui est une fenêtre graphique dans laquelle tu pourras placer différents éléments ou widgets : boutons (Button), étiquette (Label), zone de texte (Entry) …
  Tu pourras aussi gérer de nombreux évènements : clic ou déplacement de la souris, appuie sur une touche du clavier …

```
fenetre = Tk()
```

- ✓ Change le tite de la fenêtre et donne-lui une largeur de 400 pixelles et une hauteur de 100 pixelles

```
fenetre.title("Hello")
fenetre.geometry('400x100')
```

- ✓ Crée un widgets Label appelé message pour afficher à l'écran Hello world !

```
message = Label(fenetre, text="Hello World !")
```

- ✓ Positionne le label message dans la fenêtre.

```
message.pack()
```

- ✓ Lance le gestionnaire d'évènement

```
fenetre.mainloop()
```

Code complet

```
from tkinter import *

# Création de la fenêtre graphique
fenetre = Tk()
fenetre.title("Hello")
fenetre.geometry('400x100')

# Création des widgets
bonjour = Label(fenetre, text="Hello World !")

# Positionnement des widgets sur la fenêtre graphique avec pack()
bonjour.pack()

# Lancement du gestionnaire d'évènements
fenetre.mainloop()
```

Exercice

- ✓ Rajoute un 2ème label « Comment ça va ? » et observe où il se place.
- ✓ En t'aidant de la page 2 du memo « Enjoliver vos widgets », et de visual studio code (avec l'aide lorsque tape ton code), change la taille du texte en 14, bleu, avec un fond jaune.

1.2 Test2 – Positionnement des widgets sur la fenêtre graphique : pack, grid

Crée un fichier tkinter_2_Row_Column.py et écris 3 labels "Hello World !", "Comment ça va ?", "T'es sur que tu vas bien ?".

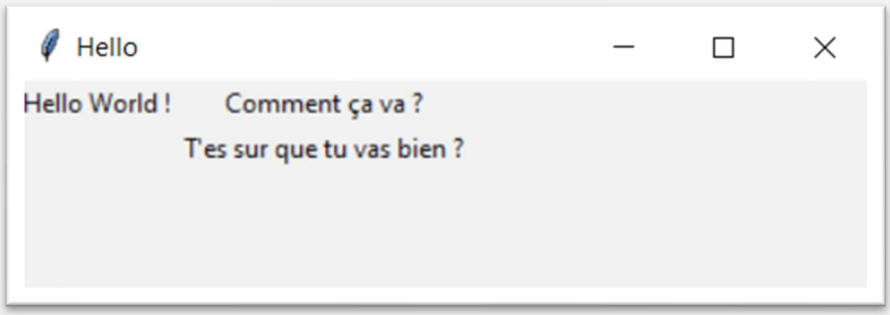
grid()

L'instruction **pack**() positionne les labels les uns sous les autres.
Change pack() par **grid**() : la fenêtre est divisée virtuellement en plusieurs zones correspondant aux cases d'une grille. Chaque case est repérée par sa ligne et sa colonne, en commençant la numérotation en haut à gauche en (0,0). La taille des cases s'adapte aux dimensions de l'objet qu'elle contient.
Mets le 1er label en haut à gauche, à côté à droite et le 3ème en dessous à gauche.

Résultat avec l'application de bureau

```
bonjour.grid(row=0,column=0)
politesse.grid(row=0,column=1)
inquietude.grid(row=1,column=1)
```

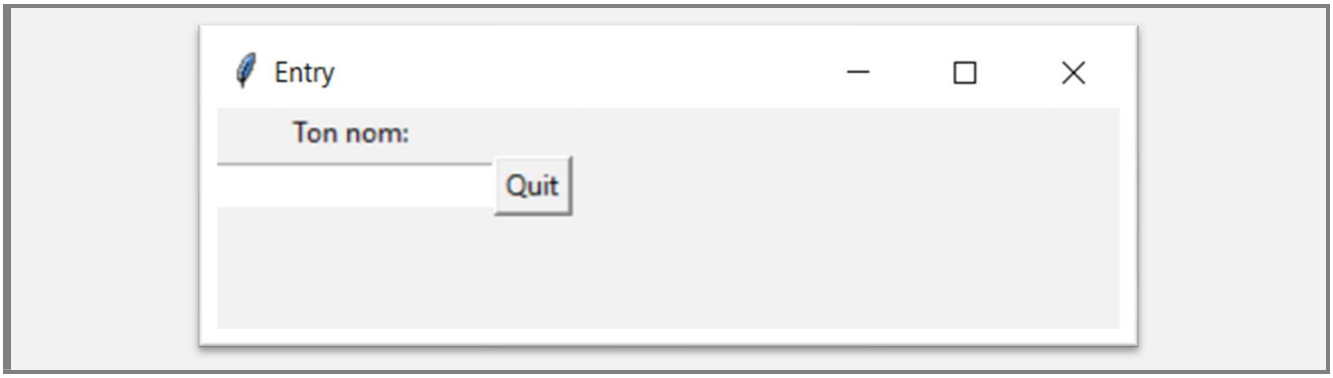
Résultat avec l'application de bureau



1.3 Test3 - Lecture clavier : widgets Entry et boutons Button

Crée un fichier tkinter_3_TextField_ElevatedBoutons.py, ouvre-le avec visual studio et copie le code expliqué ci-dessous.

Résultat avec l'application de bureau



- ✓ Crée un champ dans lequel tu pourras rentrer ton nom. Le widget **Entry()** est l'équivalent d'un **input()** dans une fenêtre graphique.

```
champ = Entry(fenetre)
```

Rajoute un bouton pour fermer la fenêtre. Le widget Button() permet de proposer une action à l'utilisateur.

```
bouton = Button(fenetre, text="Dis mon nom !", command=fenetre.destroy)
```

Code complet

```
from tkinter import *

# Création de la fenêtre graphique
fenetre = Tk()
fenetre.title("Entry")
fenetre.geometry('400x100')

# Création des widgets
consigne = Label(fenetre, text="Ton nom:")
champ = Entry(fenetre)
bouton = Button(fenetre, text="Dis mon nom !", command=fenetre.destroy)

# Positionnement des widgets sur la fenêtre graphique avec grid()
consigne.grid(row=0,column=0)
champ.grid(row=1,column=0)
bouton.grid(row=1,column=1)

# Lancement du gestionnaire d'évènements
fenetre.mainloop()
```

1.4 Test4 - Gestion d'évènement : click souris

Crée un fichier tkinter_4_click_souris.py, ouvre-le avec visual studio et copie le code expliqué ci-dessous. Le but est d'afficher « Clicked » lorsque l'on fait un clic souris sur le bouton.

Résultat avec l'application de bureau

Dans un 1er temps, contente-toi d'afficher le message dans le terminal

```
# Définition de la fonction:
def affiche_message():
    print("Clicked !")

# Création des widgets
bouton_clic = Button(fenetre, text="Click me !", command=affiche_message)
label_message=Label(fenetre)
```

Modification du Label avec config

```
# Définition de la fonction:
def affiche_message():
    label_message.config(text="Clicked !")

# Création des widgets
bouton_clic = Button(fenetre, text="Click me !", command=affiche_message)
label_message=Label(fenetre)
```

Modification du Label avec StringVar

```
# Définition de la fonction:
def affiche_message():
    message.set("Clicked !")

# Création des widgets
bouton_clic = Button(fenetre, text="Click me !", command=affiche_message)
message=StringVar()
label_message=Label(fenetre,textvariable=message)
```

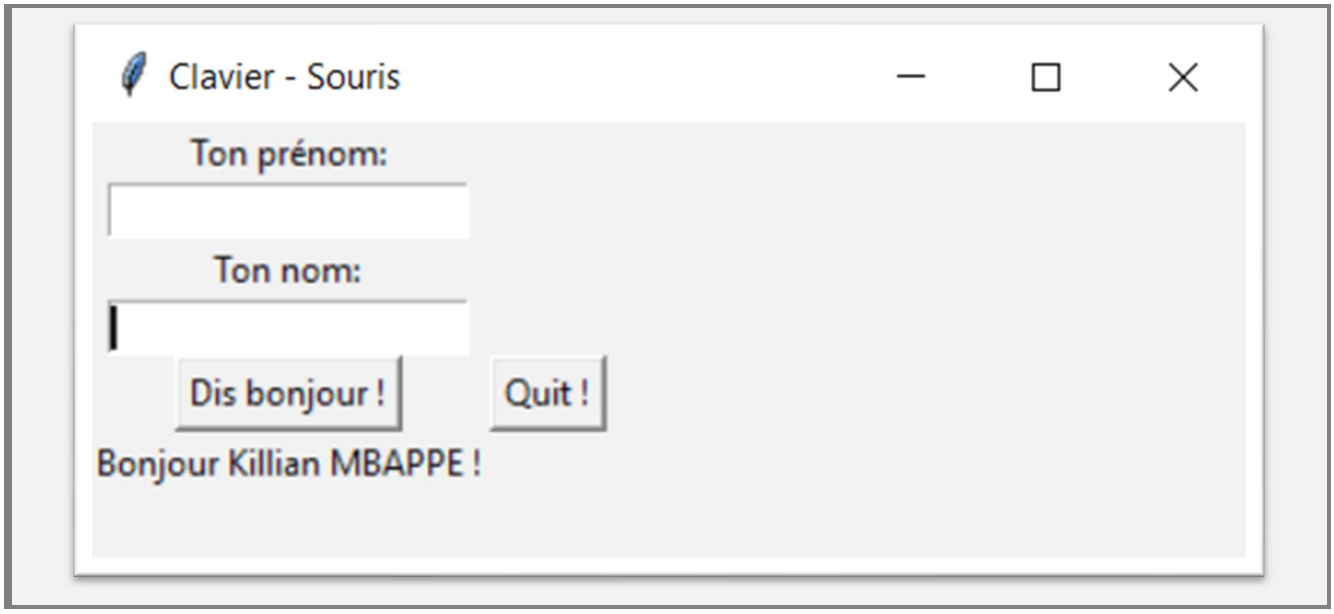
Exercice

Modifie le code de la fonction pour compter le nombre de clics.

1.5 Test5 – Exercice : lecture clavier, clic souris

Crée un fichier tkinter_5_clavier_souris.py, ouvre-le avec visual studio et fais l'exercice ci-dessous. Le but est d'afficher quand on clique sur le bouton les prénoms et noms tapés au clavier .

Résultat avec l'application de bureau



1.6 Test6 – Exercice : compteur - / +

Crée un fichier tkinter_6_compteur.py, ouvre-le avec visual studio et fais l'exercice ci-dessous. Le but est d'afficher un compteur qui va diminuer ou augmenter quand on clique sur le bouton – ou +.

- ✓ Crée 3 widgets: label_compteur, bouton_moins et boutons_plus

  Le compteur est centré, a 100px de largeur
  Les boutons affichent une icône moins / plus et appellent les fonctions minus_click ou plus_click

- ✓ Crée les fonctions minus_click ou plus_click qui vont diminuer ou augmenter la valeur du compteur.

- ✓ Rajoute un titre à la page

- ✓ Ajuste les contrôles dans une colonne, centrée au milieu de la page

1.7 Dessiner sur une toile: widget Canvas

Crée un fichier tkinter_7_canvas.py, ouvre-le avec visual studio et copie le code expliqué ci-dessous.

- ✓ Crée un widget **Canvas**() de 400x400: il s'agit d'une zone graphique dans lequel tu peux dessiner ou écrire ce que tu veux (lignes, rectangles, cercles …)

```
xmax, ymax=400,400
can1=Canvas(fenetre, bg='grey',height=ymax,width=xmax)
```

- ✓ Rajoute à côté un bouton pour fermer la fenêtre.
- ✓ Trace un trait horizontal à 10px du bas de la fenêtre avec **create_line()**.

```
can1.create_line(0,390,xmax,390,width=2,fill="red")
```

- ✓ Trace un carré bleu de 100px de côté en haut à gauche avec **create_rectangle()**.

```
rect=can1.create_rectangle(0,0,100,100, fill='blue')
```

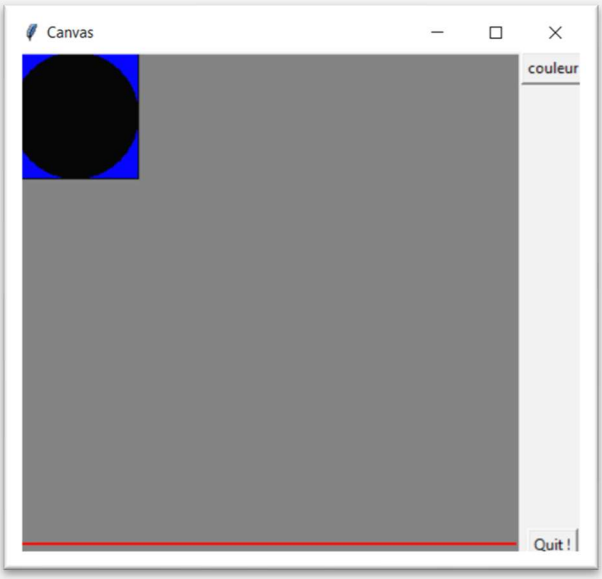
- ✓ Trace un cercle rempli en noir de même taille par-dessus avec **create_oval()**.

```
cercle=can1.create_oval(0,0,100,100, fill='black')
```

- ✓ Rajoute un bouton qui appelle la fonction change_couleur afin de changer la couleur du cercle en rouge avec **itemconfig()**

```
def change_couleur():
    can1.itemconfig(cercle,fill='red')
```

Résultat avec l'application de bureau



Code complet

```
from tkinter import *
# Création de la fenêtre graphique
fenetre = Tk()
fenetre.title("Canvas")
xmax, ymax=400,400

# Définition des fonctions
def change_couleur():
   can1.itemconfig(cercle,fill='red')

# Création des widgets
can1=Canvas(fenetre, bg='grey',height=ymax,width=xmax)
can1.create_line(0,390,xmax,390,width=2,fill="red")
rect=can1.create_rectangle(0,0,100,100, fill='blue')
cercle=can1.create_oval(0,0,100,100, fill='black')

bouton_couleur=Button(fenetre, text="couleur",command=change_couleur)

bouton_quit = Button(fenetre, text="Quit !", command=fenetre.destroy)

# Positionnement des widgets
can1.pack(side=LEFT)
bouton_couleur.pack()
bouton_quit.pack(side=BOTTOM)

# Lancement du gestionnaire d'évènements
fenetre.mainloop()
```