Introduction à Tkinter – Partie 3 /

Table des matières

1. Fenêtres secondaires (Toplevel)....1
2. Couleurs, icônes, images....1
   a. Couleurs....1
   b. Icône de la fenêtre....1
   c. Images avec PhotoImage....1
3. Astuces & Conseils supplémentaires sur Tkinter....2
   a. Toujours structurer ton code avec une fonction principale....2
   b. Crée des fonctions claires pour les actions....2
   c. Fixer la taille des widgets avec .grid(sticky=...) ou .columnconfigure()....2
   d. Utiliser une police personnalisée....2
   e. Garder une référence à l'image....2
   f. Tester dans une fenêtre secondaire (pour outils internes, formulaires...)....3
   g. Pour le debug, affiche des messages dans un Text ou un Label....3
   h. Organise ton interface avec des Frame....3
   i. Ne bloque jamais mainloop()....3
   j. Utilise .after() pour mettre à jour des éléments régulièrement....3

1. Fenêtres secondaires (`Toplevel`)

Permet d'ouvrir une **nouvelle fenêtre** sans fermer la principale :

```
def ouvrir_fenetre():
    fenetre2 = tk.Toplevel()
    fenetre2.title("Fenêtre secondaire")
    tk.Label(fenetre2, text="Ceci est une autre fenêtre").pack()

tk.Button(fenetre, text="Ouvrir une fenêtre", command=ouvrir_fenetre).pack()
```

2. Couleurs, icônes, images

a. Couleurs

Utiliser les options `bg` (background, fond), `fg` (foreground, texte), `activebackground`, etc.

```
tk.Label(fenetre, text="Coloré", fg="white", bg="darkgreen").pack()
```

b. Icône de la fenêtre

```
fenetre.iconbitmap("mon_icone.ico")  # Format .ico requis
```

c. Images avec `PhotoImage`

```
image = tk.PhotoImage(file="image.png")
tk.Label(fenetre, image=image).pack()
fenetre.mainloop()
```

> ⚠️ `PhotoImage` ne gère que `.gif` et `.png`. Pour JPG ou d'autres formats, utiliser `PIL` (`Pillow`) qui est une bibliothèque python pour travailler les images (conversion, ...).

Oui, voici une **sélection d'astuces et de conseils utiles** pour utiliser Tkinter efficacement. Ces points rendent la création d'interfaces plus fluide, robuste et professionnelle.

3. Astuces & Conseils supplémentaires sur Tkinter

a. Toujours structurer ton code avec une fonction principale

Cela évite que la fenêtre s'ouvre automatiquement à l'importation (pratique pour les projets plus gros).

```
def main():
    fenetre = tk.Tk()
    # création des widgets...
    fenetre.mainloop()

if __name__ == "__main__":
    main()
```

b. Crée des fonctions claires pour les actions

Au lieu d'avoir tout dans un seul bloc, sépare bien :

- la création des widgets,
- les fonctions de traitement (par exemple : `calculer()`, `afficher_message()`).

Cela rend le code plus **lisible** et **facile à corriger**.

c. Fixer la taille des widgets avec `.grid(sticky=...)` ou `.columnconfigure()`

Pour un **meilleur alignement** avec `.grid()` :

```
fenetre.grid_columnconfigure(0, weight=1)
fenetre.grid_columnconfigure(1, weight=2)
```

Et pour étendre un widget :

```
entry.grid(row=0, column=1, sticky="ew")
```

d. Utiliser une police personnalisée

Tkinter permet de styliser les textes facilement :

```
import tkinter.font as tkFont

police = tkFont.Font(family="Arial", size=12, weight="bold")
label = tk.Label(fenetre, text="Bonjour", font=police)
```

e. Garder une référence à l'image

Tkinter supprime parfois les images si elles ne sont pas stockées dans une variable persistante :

```
image = tk.PhotoImage(file="monimage.png")
label = tk.Label(fenetre, image=image)
label.image = image  # 🔥 Ne pas oublier cette ligne !
label.pack()
```

f. Tester dans une fenêtre secondaire (pour outils internes, formulaires...)

Créer un outil dans une fenêtre secondaire permet d'éviter de mélanger la logique principale.

g. Pour le debug, affiche des messages dans un `Text` ou un `Label`

Plutôt que des `print()` invisibles, tu peux afficher directement dans l'interface :

```
zone_logs.insert(tk.END, "Opération terminée\n")
```

h. Organise ton interface avec des `Frame`

Tu peux grouper des widgets ensemble dans des **boîtes** pour mieux organiser ton interface :

```
cadre_gauche = tk.Frame(fenetre)
cadre_gauche.grid(row=0, column=0)

cadre_droit = tk.Frame(fenetre)
cadre_droit.grid(row=0, column=1)
```

i. Ne bloque jamais `mainloop()`

Si tu veux faire une opération longue (lecture série, téléchargement...), utilise :

- des `after()` (exécute une fonction plus tard),
- ou un **thread** si c'est nécessaire (à aborder plus tard).

j. Utilise `.after()` pour mettre à jour des éléments régulièrement

Par exemple, pour une horloge :

```
def mise_a_jour():
    heure.set(time.strftime('%H:%M:%S'))
    fenetre.after(1000, mise_a_jour)
```