

## 2N6 Programmation 2



avec python™



# L'interface graphique ( GUI )

- Le *Graphical User Interface* (GUI) permet à un utilisateur d'interagir avec un programme à l'aide d'éléments visuels.
- Inclut icônes, boutons, fenêtres, zones de texte et plus.
- Permet de communiquer plus d'informations rapidement.
- Plus agréable à utiliser pour l'utilisateur novice.

# Les options de GUI



- > Selon que vous voulez faire quelque chose de basique ou développer une application complexe qui sera commercialisée, vous utiliserez différents modules pour développer votre interface GUI.
- > Quelques modules, du plus simple au plus complexe:
  - > Tkinter → inclut dans librairie standard
  - > Custom tkinter ... permet d'obtenir des gui stylisé rapidement à partir de Tkinter.
  - > Qt5 framework
  - > PySimpleGUI
  - > PyQt5.



- Fait partie de la librairie standard.
- Fournit des outils pour créer des éléments d'interface graphique pour des applications de bureau.
- Placement des éléments dans une grille.
- Peut être un peu compliqué mais offre beaucoup de possibilités.

# Module **customtkinter**



- Module ne faisant pas partie de la librairie standard.
- Doit être installé avec pip

```
pip install customtkinter
```

- Utilise les éléments de tkinter.
- Contient déjà des styles associés à chacun des contrôles.
- Facilite la création d'interfaces décentes rapidement.
- Pas recommandé pour les plus gros projets.



# Créer une fenêtre dans **customtkinter**



```
1 import customtkinter
2
3 customtkinter.set_appearance_mode("dark")..# Style de la fenêtre"
4 customtkinter.set_default_color_theme("blue")..# couleurs des éléments
5
6 app = customtkinter.CTk()
7 # Paramètres de l'app tel que la taille de la fenêtre.
8
9 # Tous les widgets qui formeront notre interface graphique.
10
11 app.mainloop() # Lance notre application..
```



# Frame

- On peut découper une fenêtre en différentes parties, appelées Frame.





- > Un widget est un élément d'interface graphique interactif.
- > L'assemblage de plusieurs widgets va former notre interface graphique.
- > Aujourd'hui, nous allons voir :
  - > Un **label** pour afficher de l'information.
  - > Une **zone de texte** pour entrer de l'information.
  - > Une **liste déroulante** pour faire des choix parmi une liste.
  - > Des **boutons** pour interagir avec l'interface.





- Petite zone pour afficher du texte.
- Une "étiquette" en français.
- Donne des informations sur les différentes sections et/ou widgets.
- Pas une source de saisie de données (Fait uniquement de l'affichage.)
- Peut être utilisé pour afficher un message de façon dynamique à l'aide de fonctions.





# CTkButton

- > Un bouton !
- > Ne fait rien par lui-même
- > On doit associer le bouton avec une méthode lors de sa création.

```
def appuyer_sur_button():  
    ...label_1.configure(text="Yay un bouton !")  
  
button_1 = customtkinter.CTkButton(master=frame_1, command=appuyer_sur_button, width=200)
```

- > **Notez :** le paramètre *command* prend la fonction elle-même et non son résultat, (sans les parenthèses).


# Associer du code



- > Il faut définir des fonctions pour associer du code à des éléments graphiques.

```
def appuyer_sur_button():  
    ...label_1.configure(text="Yay un bouton !")  
  
button_1 = customtkinter.CTkButton(master=frame_1, command=appuyer_sur_button, width=200)
```

Pas de parenthèses



- > Appuyer sur le bouton devient exactement la même chose que d'exécuter la commande.



- > Champs pour insérer du texte qui sera utilisé par le code.
- > Une fois que l'information est écrite :
  - > La méthode `get()` va nous retourner le texte entré par l'utilisateur.
  - > La méthode `delete()` va supprimer les caractères entre deux index donnés.

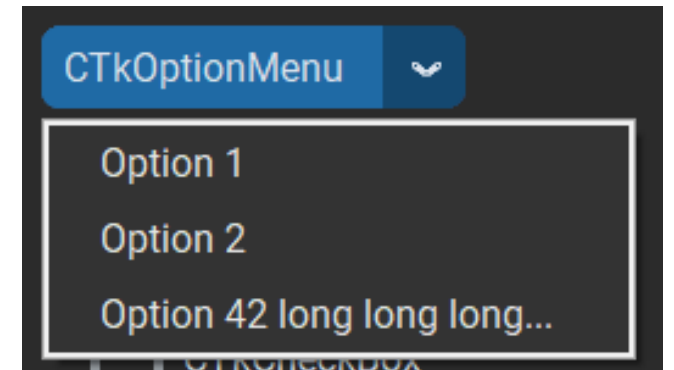
```
#Champs où écrire de l'information
entry_1 = customtkinter.CTkEntry(master=frame_1, placeholder_text="CTkEntry")
entry_1.grid(column=1,row=0,padx=30,pady=20,sticky="w")
```

```
def appuyer_sur_button():
    ...print(entry_1.get())
    ...entry_1.delete(0,len(entry_1.get()))
```

# CTkcombobox



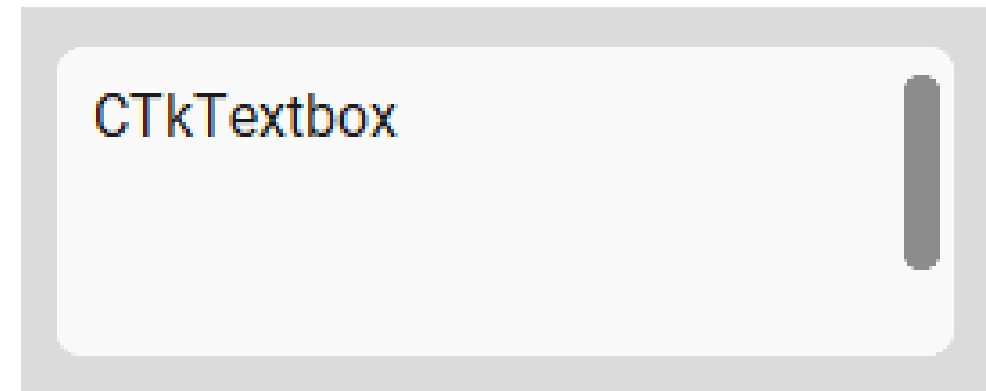
- > Listes de choix que l'utilisateur peut sélectionner.
- > On obtient l'élément choisi avec la méthode `get()`



```
combobox_1 = customtkinter.CTkComboBox(frame_1, values=["Option 1", "Option 2", "Option 42 long long long..."])
combobox_1.set("CTkComboBox") # donne une valeur par défaut
```



- Peut être utilisé pour entrer plus d'informations que le widget CtkEntry
- Peut aussi être utilisé pour afficher de l'information.



```
text_1 = customtkinter.CTkTextbox(master=frame_1, width=200, height=70)
text_1.insert("0.0", "CTkTextbox\n\n\n\n")
```



- On peut choisir si on veut que l'utilisateur puisse entrer du texte en changeant la valeur de l'attribut "state"

```
text_1 = customtkinter.CTkTextbox(master=frame_1, width=200, height=70, state="disabled")
```

```
text_1.configure(state="normal")
```

```
text_1.insert("0.0", "CTkTextbox\n\n\n\n")
```

```
text_1.configure(state="disabled")
```

- state="normal" veut dire que l'utilisateur peut entrer du texte
- state="disabled" veut dire que le texte est affiché uniquement