



avec python<sup>TM</sup>

- GUI -> principes et concepts
- Tkinter -> librairie standard
- Modules



# L'interface graphique ( GUI )

- Le *Graphical User Interface* (GUI) permet à un utilisateur d'interagir avec un programme à l'aide d'éléments visuels.
  - Utilisation facile
  - Communique plus d'informations rapidement.
  - Supports graphiques et organisation spatiale.
  - Moins d'options qu'une *Client line interface* (CLI)
  - Plus agréable à utiliser pour l'utilisateur novice.

# Les options de GUI



Module	Prix	Licence	Avantage	Désavantage
Tkinter	-	BSD	- <b>librairie standard</b> - Multiplateforme	- Compiqué - Pas très beau...
CustomTkinter	-	MIT	- donne des interfaces décentes rapidement - Basé sur Tkinter	- <b>Module obscur</b> - <b>Manque de documentation</b>
PyQt5 Framework	- / 550 \$*	GPL / Commercial	- Bonne documentation - Support professionnel	- Coûts - Licence gratuite problématique
PySimpleGUI	0** / 99\$	Commercial		- Coûts - Licence gratuite expire après 1 ans
PySide	- / ?	LGPL / Commercial	- Gratuit - Bonne documentation - Tutoriels	
Flask / Bottle	-	BSD / MIT	- Permet de faire des sites web !	- Plus gros projet. - Plus compliqué pour applications de bureau

\*550 USD par développeur sur le projet

\*\*Doit être renouvelé (gratuitement) chaque année

# modules **Tkinter** et **TTKinter**

- > Font partie de la **librairie standard**.
- > Sous module **Ttkinter** offre plus d'options de thèmes.

- > Compliquer mais offres beaucoup de possibilités.
- > Fonctionne sur tout système supportant Python
- > Permits de créer des GUI pour des applications de bureau.
- > Placement des éléments dans une grille.

# Élément GUI le plus facile : **messagebox**



- messagebox : sous module de tkinter
  - Permet d'ajouter des popups facilement pour ajouter de l'information
  - Prends 2 paramètres : **title** et **message**
- Utilisé pour :
  - Avertissements,
  - Ajout d'informations
  - Saisie de valeur booléenne

```
6  from tkinter import messagebox
7
8  messagebox.askquestion(title="En-tête", message="Texte ici !")
9
10
11
12
13
14
15
16
17
18
```

# Sous-module : **messagebox**

> 7 types de messagebox

> Chaque retourne une valeur qui peut être saisie

```
messagebox.showinfo("1","infos")
```

```
messagebox.showwarning("2","avertissement")
```

```
messagebox.showerror("3","erreur")
```

```
messagebox.askquestion("4","question")
```

```
messagebox.askokcancel("5","cancel?")
```

```
messagebox.askyesno("6","oui / non ?")
```

```
messagebox.askretrycancel("7","ré-essayer ?")
```

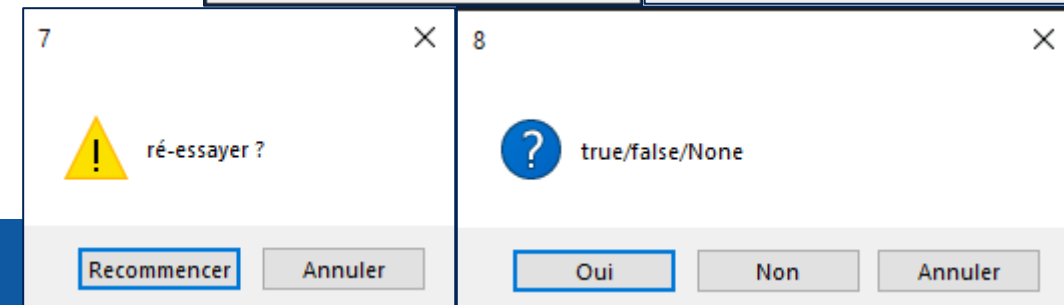
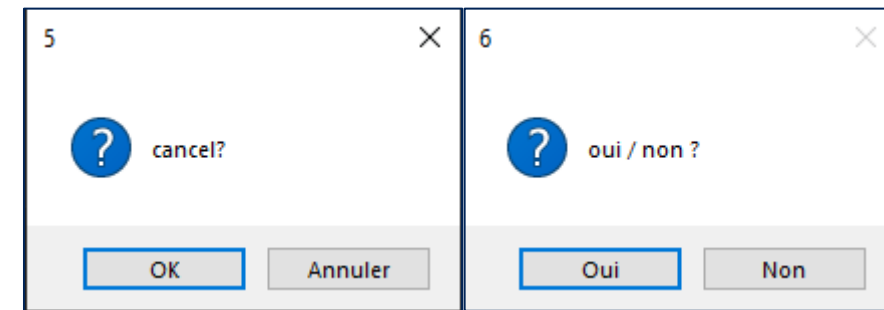
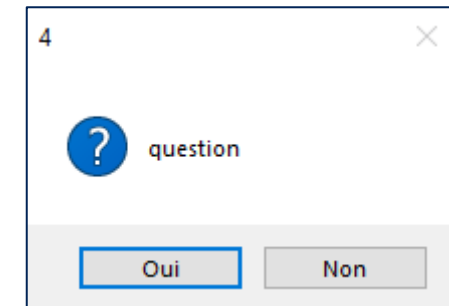
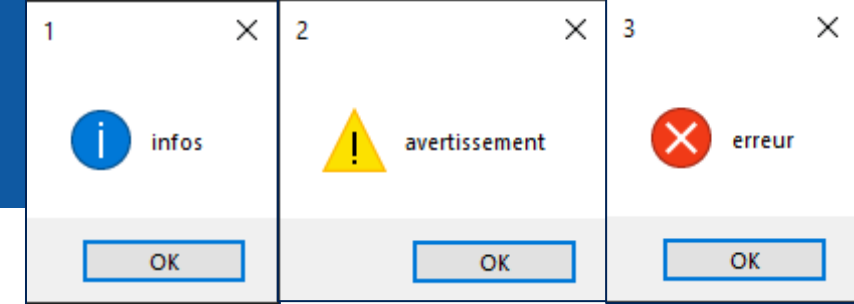
```
messagebox.askyesnocancel("8","true/false/None")
```

str : 'ok'

str : 'yes' ou 'no'

bool

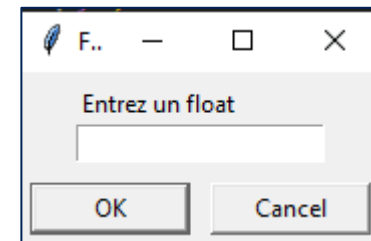
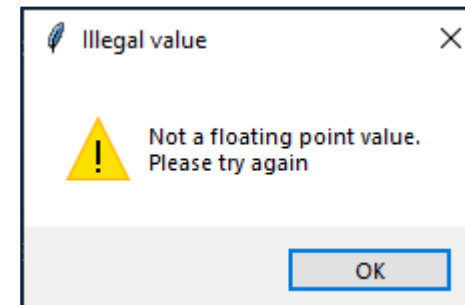
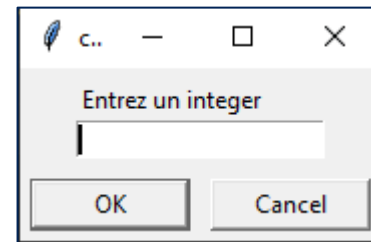
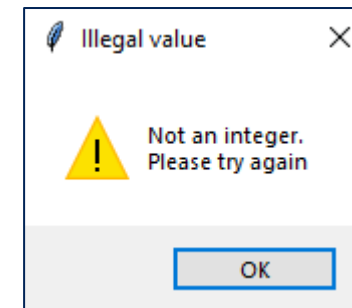
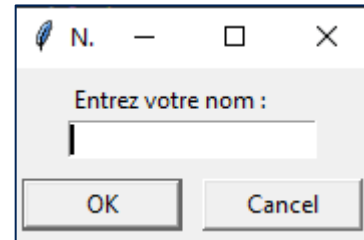
True, False, None



# sous-module **simpdialog**

- Permet la saisie de données avec une interface graphique simple.
- Évite les erreurs dues à un mix de CLI et de GUI

```
from tkinter import simpdialog  
  
repstr = simpdialog.askstring("Nom ?", "Entrez votre nom : ")  
repint = simpdialog.askinteger("chiffre", "Entrez un integer")  
repfloat = simpdialog.askfloat("Float", "Entrez un float")
```



# sous-module **filedialog**

- Permet l'interaction avec système de fichier.
- Ouvre une fenêtre d'ouverture / enregistrement

• **askopenfile()**->

Retourne un objet de type fichier ouvert en lecture

• **asksaveasfile()**->

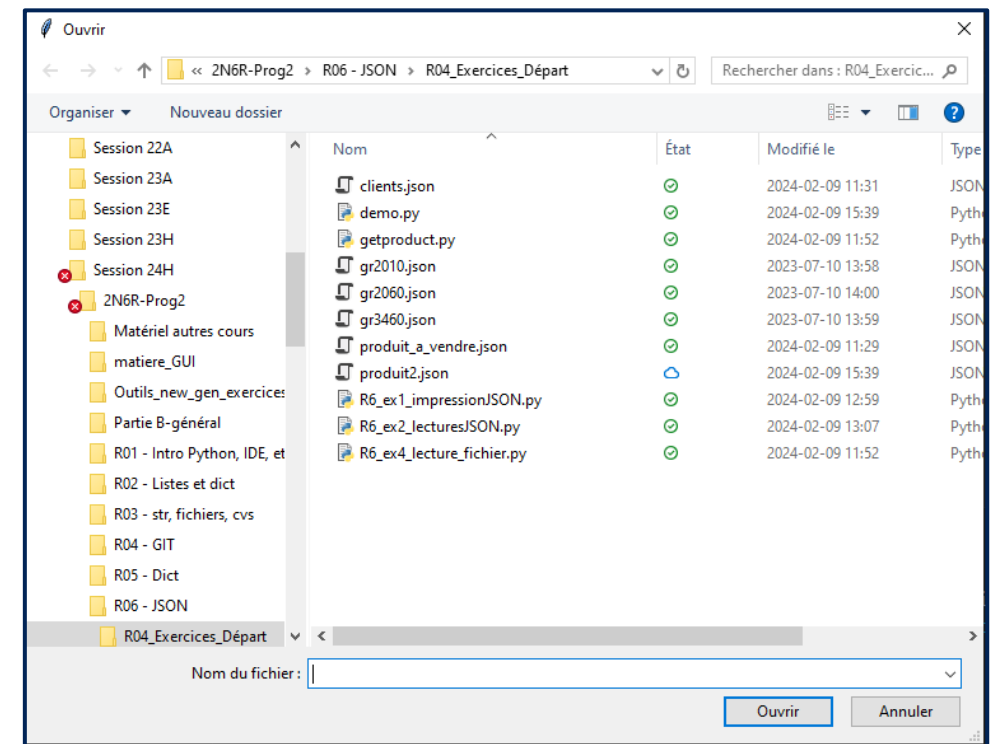
Retourne un objet de type fichier ouvert en écriture

• **askopenfilename()**->

• **asksaveasfilename()**->

Ils retournent le NOM du fichier choisi.

```
3_demo_file.py > ...
1  from tkinter import filedialog
2  import json
3
4  with filedialog.askopenfile() as f_lu :
5      text = f_lu.read()
6      data = json.loads(text)
7
8  print(data)
```





# Module **customtkinter**

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

```
pip install customtkinter
```

[customtkinter.tomschimansky.com/](https://customtkinter.tomschimansky.com/)

[pypi.org/project/customtkinter/0.3/](https://pypi.org/project/customtkinter/0.3/)

Documentation !

- > Utilise les éléments de tkinter.
- > Contiens 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 & Grid



- On organise généralement une fenêtre en différentes parties, appelées **Frame**.
- Ces Frame sont placés dans une grille et peuvent posséder leur propre grille.

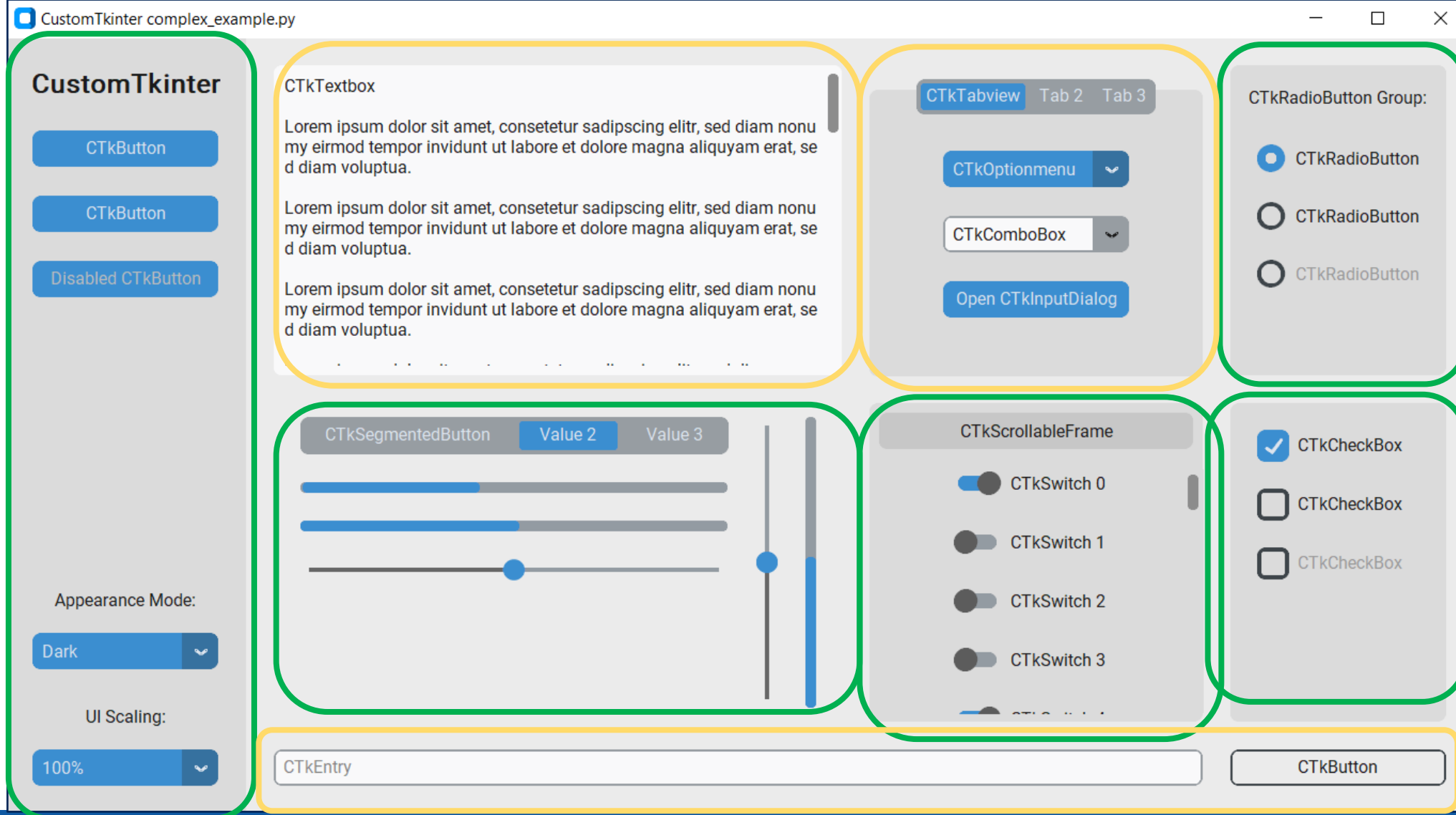


# Frames



Ici nous avons un mix de Frames et d'éléments poser directement sur la grille.

Une Frame est utilisée lorsqu'on veut ajouter plusieurs widgets dans une même "zone"



# Grid 4x4



0

1

2

3

La grille de base est de 4 rangées par 4 colonnes.

Mais dans ce cas-ci, une rangé n'est pas utilisée.

0

1

2

3

CustomTkinter complex\_example.py

<b>CustomTkinter</b>  CTkButton  CTkButton  Disabled CtkButton	<b>CTkTextbox</b>  Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonu my eirmod tempor invidunt ut labore et dolore magna aliquyam erat, se d diam voluptua.  Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonu my eirmod tempor invidunt ut labore et dolore magna aliquyam erat, se d diam voluptua.  Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonu my eirmod tempor invidunt ut labore et dolore magna aliquyam erat, se d diam voluptua.  ...	<b>CTkTabview</b> Tab 2 Tab 3  CTkOptionmenu  CTkComboBox  Open CtkInputDialog	<b>CTkRadioButton Group:</b>  <input checked="" type="radio"/> CtkRadioButton  <input type="radio"/> CtkRadioButton  <input type="radio"/> CtkRadioButton
Appearance Mode:  Dark	<b>CTkSegmentedButton</b> Value 2 Value 3  [Progress bars and sliders]  [Vertical slider]	<b>CTkScrollableFrame</b>  CTkSwitch 0 CTkSwitch 1 CTkSwitch 2 CTkSwitch 3	<input checked="" type="checkbox"/> CtkCheckBox  <input type="checkbox"/> CtkCheckBox  <input type="checkbox"/> CtkCheckBox
UI Scaling:  100%	CTkEntry		CTkButton

# Frames & Grid



`grid(row=0, column=0, rowspan=4,`

`.grid(row=0, column=1,`

`.grid(row=0, column=2,`

CustomTkinter

CTkButton

CTkButton

Disabled CTkButton

Appearance Mode:

Dark

UI Scaling:

100%

CTkTextbox

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonu my eirmod tempor invidunt ut labore et dolore magna aliquyam erat, se d diam voluptua.

CTkSegmentedButton

Value 2

Value 3

CTkTabview

Tab 2

Tab 3

CTkOptionmenu

CTkComboBox

Open CTkInputDialog

CTkRadioButton Group:

CTkRadioButton

CTkRadioButton

CTkRadioButton

CTkScrollableFrame

CTkSwitch 0

CTkSwitch 1

CTkSwitch 2

CTkSwitch 3

CTkCheckBox

CTkCheckBox

CTkCheckBox

CTkEntry

CTkButton

`.grid(row=1, column=2,`

`e.grid(row=1, column=1,`

`.grid(row=3, column=1, columnspan=2, p`

`.grid(row=3, column=3,`

# widget

- > Un widget est un élément d'interface graphique interactif.
- > L'assemblage de plusieurs widgets va former notre interface graphique.

- > Types principaux :
  - > **label** pour afficher de l'information.
  - > **zone de texte** pour entrer de l'information.
  - > **liste déroulante** pour faire des choix parmi une liste.
  - > **boutons** pour interagir avec l'interface.
  - > **checkBox** pour faire des choix True / False
  - > **radioButton** pour sélectionner une option parmi plusieurs.
  - > **image** pour communiquer plus d'informations et rendre le GUI plus beau.



- > 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.







- > Un bouton !
- > Ne fais 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.

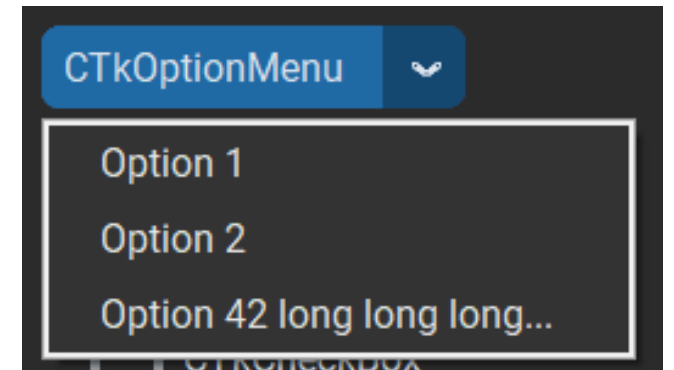
```
#Champs où écrire de l'information
entry_1 = customtkinter.CTkEntry(master=frame_1, placeholder_text="CTkEntry")
entry_1.grid(column=1,row=0,padx=30,pady=20,sticky="w")
```

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

# CTkcombobox



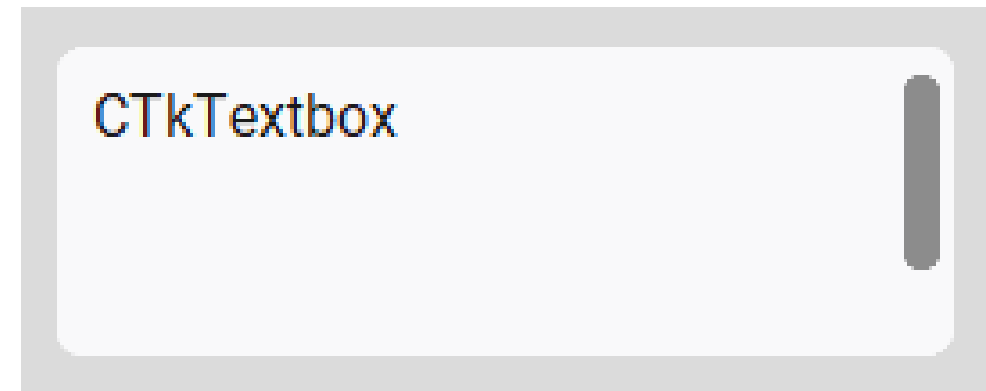
- > Listes de choix que l'utilisateur peut sélectionner.
- > On obtient l'élément choisi avec la méthode `get()`



```
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





# CheckBox (case à cocher)

- > Les cases à cocher permettent de faire une sélection booléenne, vraie ou faux

```
self.chk_1 = ctk.CTkCheckBox(master=frm_experience,  
.....text="Programmation Python",  
.....onvalue=1, offvalue=0)
```

- > onvalue : lorsque la case est cochée,
- > offvalue : lorsque la case n'est pas cochée
- > On obtient la valeur (onvalue ou offvalue) avec la méthode .get()

```
valeur_chkbox = self.chk_1.get()
```

<input type="checkbox"/>	Programmation Python
<input type="checkbox"/>	Programmation PowerShell
<input type="checkbox"/>	Maintenance des serveurs
<input type="checkbox"/>	Maintien des postes de travail
<input type="checkbox"/>	Maintien du réseau câblé et sans fil
<input type="checkbox"/>	Maintien des autres périphériques
<input type="checkbox"/>	Gestion des licences
<input type="checkbox"/>	Formation usagers
<input type="checkbox"/>	Gestion des accès usagers
<input type="checkbox"/>	Gestion des anti-virus
<input type="checkbox"/>	Gestion des télécommunications
<input type="checkbox"/>	Gestion des routeurs



# RadioButton (boutons radio)

- Permettent de choisir entre plusieurs options.
- Les groupes de boutons radio sont définis par la variable à laquelle ils réfèrent.

```
choix_pizza = tk.StringVar(value="nature")
```

```
pizza_nature = ttk.Radiobutton(frm_pizza_choix, text='Nature', variable=choix_pizza, value='nature')  
pizza_vege = ttk.Radiobutton(frm_pizza_choix, text='Végétarienne', variable=choix_pizza, value='végétarienne')  
pizza_garnie = ttk.Radiobutton(frm_pizza_choix, text='Toute garnie', variable=choix_pizza, value='toute garnie')
```

- On obtient la valeur du groupe de boutons radios à partir de la variable à laquelle ils réfèrent grâce à la méthode .get()

```
choix_pizza_val = choix_pizza.get()
```



# RadioButton (boutons radio)



```
choix_pizza = tk.StringVar(value="nature")
```

Classe importée de tkinter. Elle est similaire à la classe "str" mais fonctionne dans les widgets et possède la méthode .get()

```
pizza_nature = ttk.Radiobutton(frm_pizza_choix, text='Nature', variable=choix_pizza, value='nature')  
pizza_vege = ttk.Radiobutton(frm_pizza_choix, text='Végétarienne', variable=choix_pizza, value='végétarienne')  
pizza_garnie = ttk.Radiobutton(frm_pizza_choix, text='Toute garnie', variable=choix_pizza, value='toute garnie')
```

La même variable → ces boutons font partie du même groupe.  
Un seul RadioButton peut être sélectionné à la fois.

```
choix_pizza_val = choix_pizza.get()
```

La méthode retourne la valeur sous forme de "str" qui peut ensuite être utilisé.

# Images



- > Besoin d'un autre module :

```
t> pip install pillow
```

```
from PIL import ImageTk, Image
```

- > Permet d'ajouter des images dans des labels ou dans des boutons.
- > Par standard, les images sont situées dans un répertoire "images" dans le même emplacement que notre script.

```
R25_Ex2_MyOrder
├── R25_YourOrder.py
└── images
    ├── Logo.jpg
    ├── pizza.jpg
    ├── poutine.jpg
    └── sousmarin.jpg
```

# Images



- Par standard, les images sont situées dans un répertoire "images".
- On va chercher dans l'emplacement de l'image avec le module os.

```
R25_Ex2_MyOrder
├── R25_YourOrder.py
└── images
    ├── Logo.jpg
    ├── pizza.jpg
    ├── poutine.jpg
    └── sousmarin.jpg
```

```
self.image_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), "images")
```

```
self.logo_image = ImageTk.PhotoImage(Image.open(os.path.join(self.image_path, "logomatissoft.jpg")))
```

- On peut ensuite récupérer l'image et la mettre dans un objet. Qu'on utilisera dans un label ou un bouton.

```
# lbl pour le logo de l'entreprise
```

```
self.lbl_logo = tk.Label(master=frm_container, image=self.logo_image)
```