



Interfaces graphiques en Python

Définition

2

- Une interface graphique est un outil permettant d'interagir avec un programme informatique.
- La plupart des langages disposent de bibliothèques pour prendre en charge une partie de cette complexité
- Le programmeur doit simplement penser aux éléments graphiques à mettre en place (cadre, menu, boutons)
- Il doit également penser au fond de son programme c'est-à-dire les traitements qu'il doit faire

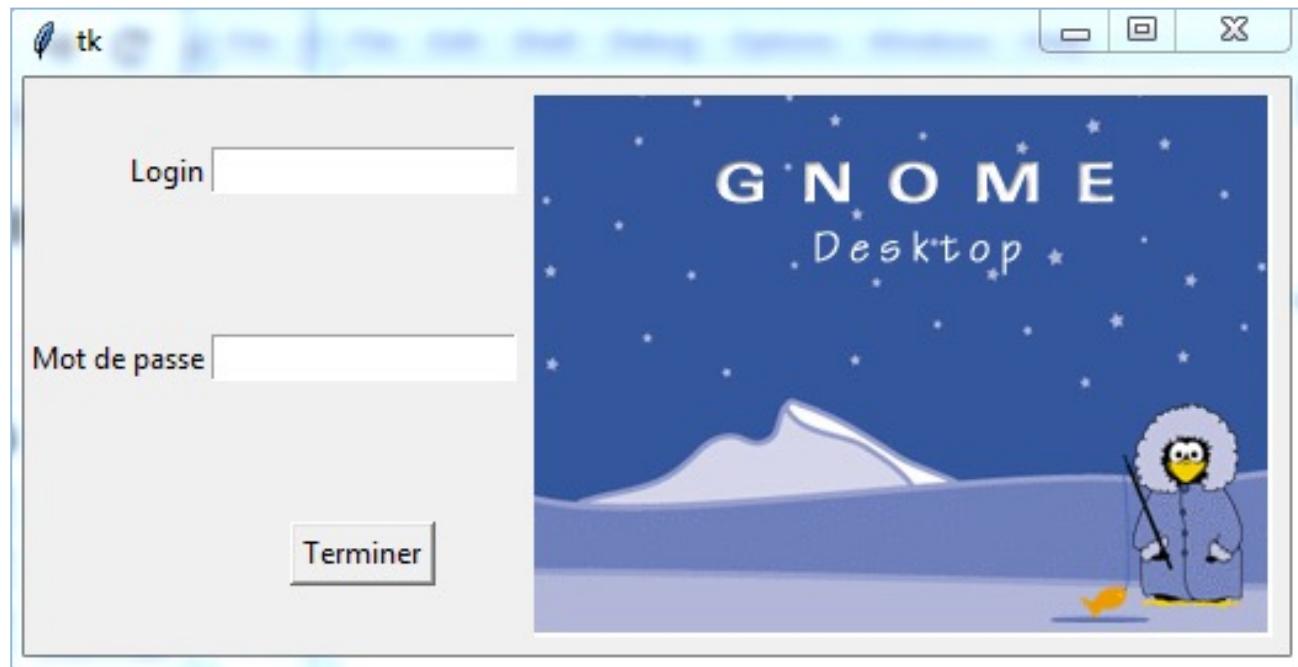
Interface graphique en python

3

- La bibliothèque la plus simple pour créer des interfaces est **Tkinter**
- Une autre bibliothèque classique est wxPython (repose sur wxWidgets)
- La bibliothèque wxWidgets est également disponible en C/C++
- Il en existe bien d'autres : pyQT, pyGTK, ...
- Dans ce cours on utilisera Tkinter
- Tkinter Permet de réaliser un **GUI (*Graphical User Interface*)**
- GUI : une interface homme-machine permettant à l'utilisateur d'interagir avec la machine .
- Le module Tkinter(**Tool Kit interface**) est une bibliothèque graphique libre pour le langage python. le module Tkinter est installé par défaut

Exemple

4

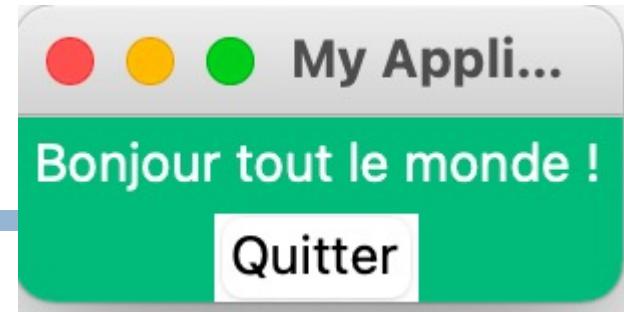


Les étapes à suivre pour créer une GUI

5

1. Importer le module Tkinter
2. Créer la fenêtre principale de l'application
3. Ajouter les widgets graphiques (bouton, label, Menu...)
4. Lancer la boucle d'événements principales pour réagir à chaque

Premier exemple



```
from tkinter import*
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

text1=Label(mafenetre, text='Bonjour tout le monde !',fg='white',bg='#41B77F')
#le premier argument est le nom de la fenêtre dans lequel
#sera le bouton (mafenetre est le widget maître de l'objet text1
#le troisième est la couleur d'avant plan(foreground).
text1.pack()
#nous activons ici la méthode pack() à l'objet tex1.
#Cette méthode agit sur la dispositiongéométrique,
#la fenêtre maître est réduite automatiquement pour qu'elle soit juste assez grande pour
#contenir les widgets esclaves.
bou1=Button(mafenetre, text='Quitter',command=mafenetre.destroy)
bou1.pack()
mafenetre.mainloop()

#fen1.mainloop() : c'est cette ligne qui provoque le démarrage du réceptionnaire
#d'événements associé à lafenêtre.
#Cette instruction est nécessaire pour que l'application soit « à l'écoute » des
#clics de souris, des pressions exercées sur les touches du clavier, etc.
#C'est donc cette instruction qui la met en marche
```

Les widgets : (1)

widget = window gadget : tous les composants de l'écran (fenêtres, boutons, ascenseurs, ...)

- **Button** : Un bouton classique, à utiliser pour provoquer l'exécution d'une commande quelconque.
- **Canvas**: Un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner
- **Checkbutton** : Une « case à cocher » qui peut prendre deux états distincts (la case est cochée ou non). Un clic sur ce widget provoque le changement d'état.
- **Entry** : Un champ d'entrée, dans lequel l'utilisateur du programme pourra insérer un texte quelconque à partir du clavier.
- **Frame** : Une surface rectangulaire dans la fenêtre, où l'on peut disposer d'autres widgets. Cette surface peut être colorée. Elle peut aussi être décorée d'une bordure.
- **Label** : Un texte (ou libellé) quelconque (éventuellement une image).
- **Listbox** : Une liste de choix proposés à l'utilisateur, généralement présentés dans une sorte de boîte.

Les widgets : (2)

- **Menu** : Un menu. Ce peut être un menu déroulant attaché à la barre de titre, ou bien un menu « pop up » apparaissant n'importe où à la suite d'un clic.
- **Menubutton** : Un bouton-menu, à utiliser pour implémenter des menus déroulants.
- **Message** : Permet d'afficher un texte. Ce widget est une variante du widget Label, qui permet d'adapter automatiquement le texte affiché à une certaine taille ou à un certain rapport largeur/hauteur.
- **Radiobutton** : Représente (par un point noir dans un petit cercle) une des valeurs d'une variable qui peut en posséder plusieurs.
- **Scale** : Vous permet de faire varier de manière très visuelle la valeur d'une variable, en déplaçant un curseur le long d'une règle.
- **Scrollbar** : « ascenseur » ou « barre de défilement » que vous pouvez utiliser en association avec les autres widgets : Canvas, Entry, Listbox, Text.
- **Text** : Affichage de texte formaté. Permet aussi à l'utilisateur d'édition le texte affiché. Des images peuvent également être insérées.
- **Toplevel** : Une fenêtre affichée séparément, « par-dessus ».

Création de widgets

9

- Chaque widget est un objet python.
- Création de widget : passer le parent de la hiérarchie en premier argument de la fonction de création.
- Les widgets ont beaucoup d'options de configuration (couleur, position,...)

Widget: Label

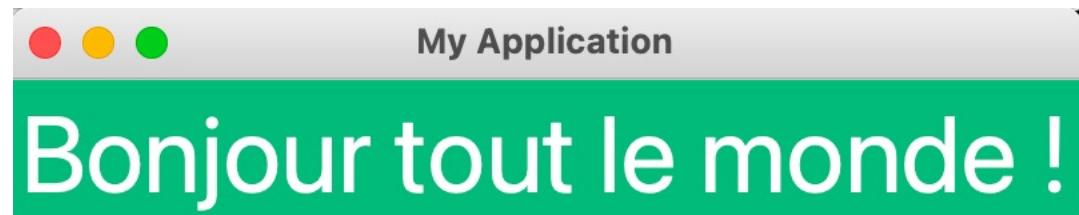
10

- Sert à insérer dans une fenêtre graphique un texte

```
from tkinter import*
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

text1=Label(mafenetre, text='Bonjour tout le monde !',fg='white',bg="#41B77F",font=("Courier", 40))
text1.pack()

mafenetre.mainloop()
```



Pour changer le text du label, on fait appel à la méthode config comme suit :

```
text1.config (text = "second texte")
```

Widget: Button

11

- Un bouton a pour but de faire le lien entre une fonction et un clic de souris. Un bouton correspond à la classe Button

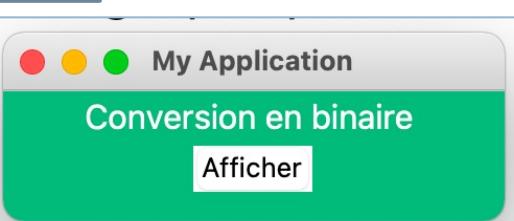
```
from tkinter import*
def afficher():
    print("Bouton cliqué")

mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

text1=Label(mafenetre, text='Conversion en binaire ',fg='white',
           bg='#41B77F',font=("Courrier", 16))
text1.pack()
boutton_afficher = Button(text="Afficher",command=afficher)
boutton_afficher.pack()

mafenetre.mainloop()
```

Bouton cliqué



Widget: Button

12

- Il est possible également d'associer une image à un bouton en créant un objet PhotoImage

```
from tkinter import*
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

text1=Label(mafenetre, text='Conversion en binaire ',fg='white',bg="#41B77F",font=("Courrier", 30))
text1.pack()
bouton_calculer = Button()
image_calculer = PhotoImage(file="check.png")
bouton_calculer.config(image=image_calculer)
bouton_calculer.pack()

mafenetre.mainloop()
```

Conversion en binaire



Widget: Zone de saisie

13

- Une zone de saisie a pour but de recevoir une information entrée par l'utilisateur. Une zone de saisie correspond à la classe Entry

`nom_zone_text = tkinter.Entry ()`

- Pour modifier le contenu de la zone de saisie, il faut utiliser la méthode insert qui insère un texte à une position donnée.

`nom_zone_text.insert (pos, "contenu")`

- *le premier paramètre est la position où insérer le texte*
 - *le second paramètre est le texte à insérer*

- Pour obtenir le contenu de la zone de saisie, il faut utiliser la méthode get :

`contenu = nom_zone_text.get ()`

- Pour supprimer le contenu de la zone de saisie, il faut utiliser la méthode delete. Cette méthode supprime le texte entre deux positions.
- *# supprime le texte entre les positions pos1, pos2*

`nom_zone_text.delete (pos1, pos2)`

- Par exemple, pour supprimer le contenu d'une zone de saisie, on peut utiliser l'instruction suivante :

`nom_zone_text.delete (0, len(saisie.get()))`

Exemple

14

```
from tkinter import*
def afficher():
    global zone_nom
    print(zone_nom.get())
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

text1=Label(mafenetre, text='Votre nom ',fg='white',bg='#41B77F',
           font=("Courrier", 12))
text1.pack()
zone_nom = Entry ()
zone_nom.pack()

bouton_afficher = Button(text="Afficher",command=afficher)
bouton_afficher.pack()

mafenetre.mainloop()
```

oumaira

Votre nom

oumaira

Afficher

My Application

Widget: Case à cocher

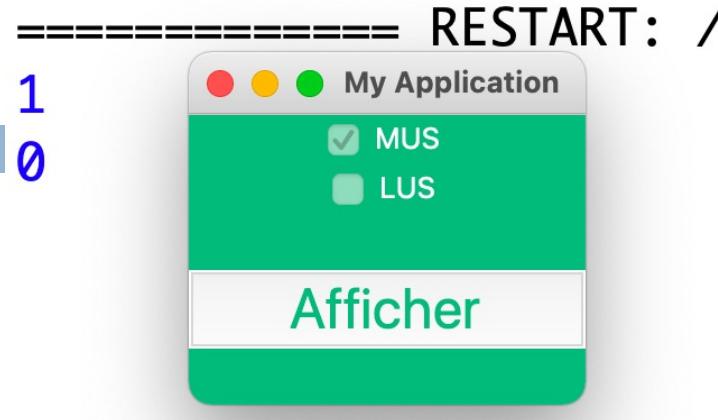
15

```
from tkinter import*
def afficher():
    # v1.get() égal à 1 si la case est cochée, 0 sinon
    print(v1.get())
    print(v2.get())
```

```
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")
```

#On déclare une variable de type IntVar, qui va mémoriser la valeur de la case à cocher.

```
v1= IntVar ()
#on déclare un objet, de type Checkbutton, qui gère l'apparence au niveau de l'interface graphique
case1 = Checkbutton (variable = v1,text="MUS" , bg='#41B77F',fg='white')
#pour cocher la case1
case1.select ()
case1.pack()
v2= IntVar ()
case2 = Checkbutton (variable = v2,text="LUS" , bg='#41B77F',fg='white')
case2.pack()
#le clique sur le bouton affiche les valeurs de v1 et v2
button = Button(mafenetre, text="Afficher", font=("Courrier", 25), bg='white', fg='#41B77F',
                command=afficher)
button.pack(pady=25, fill=X)
```



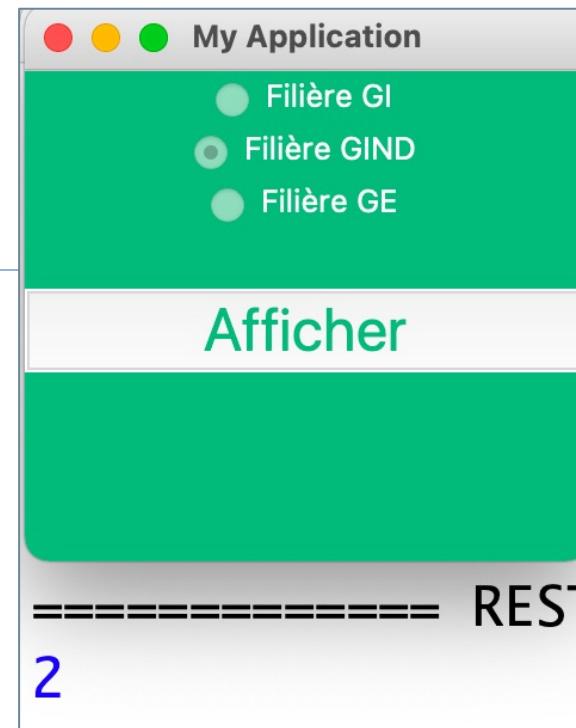
Widget : bouton radio

```
from tkinter import*
def afficher():
    # v1.get() égal à 1 si la case est cochée, 0 sinon
    print(v.get())

mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")

#On déclare une variable de type IntVar, qui va mémoriser la valeur du bouton radio sélectionné.
v=IntVar()
case1 = Radiobutton(variable = v, value = 1, text="Filière GI", bg="#41B77F", fg='white')
case2 = Radiobutton(variable = v, value = 2, text="Filière GIND", bg="#41B77F", fg='white')
case3 = Radiobutton(variable = v, value = 3, text="Filière GE", bg="#41B77F", fg='white')
case2.select()
case1.pack()
case2.pack()
case3.pack()
#le clique sur le bouton affiche la valeur sélectionnée
button = Button(mafenetre, text="Afficher", font=("Courrier", 25), bg='white', fg="#41B77F",
                command=afficher)
button.pack(pady=25, fill=X)

mafenetre.mainloop()
```



Tracer dans un canevas

17

Un canevas (toile, tableau en français) dans Tkinter est une surface rectangulaire délimitée, dans laquelle on peut insérer ensuite divers dessins, textes et images à l'aide de méthodes spécifiques

Les méthodes relatives au Canvas

18

- **create_rectangle**: les deux premiers entiers représentent les coordonnées du point en haut à gauche du rectangle, les deux suivants celles du point en bas à droite
- **create_oval**: il s'agit des coordonnées du rectangle circonscrit à l'ovale (pour un cercle, largeur et hauteur doivent être égales)
- **fill**= définit la couleur de remplissage de la forme géométrique (transparente par défaut)
- **outline** = Couleur de la bordure. Par défaut, outline='black'.
- **width**= définit l'épaisseur du contour de la forme géométrique (1 par défaut)

Tracer dans un canevas

19

```
from tkinter import *
fenetre = Tk()
fenetre.title('Dessiner et Ecrire dans un canvas')
can1 = Canvas(fenetre,width=500,height=300,bg ='#27ae60')
can1.pack()
can1.create_oval(40,20,120,200 ,outline ='#2980b9',fill ='#3498db' , width=2)
can1.create_rectangle(230,50,320,100,outline ='#d35400',fill ='#e67e22' ,width=10)
can1.create_line(40,20,120,200)
can1.create_text(250,150,text='Python is the best !!!',font=('monaco',36),fill ='#c0392b')
fenetre.mainloop()
```

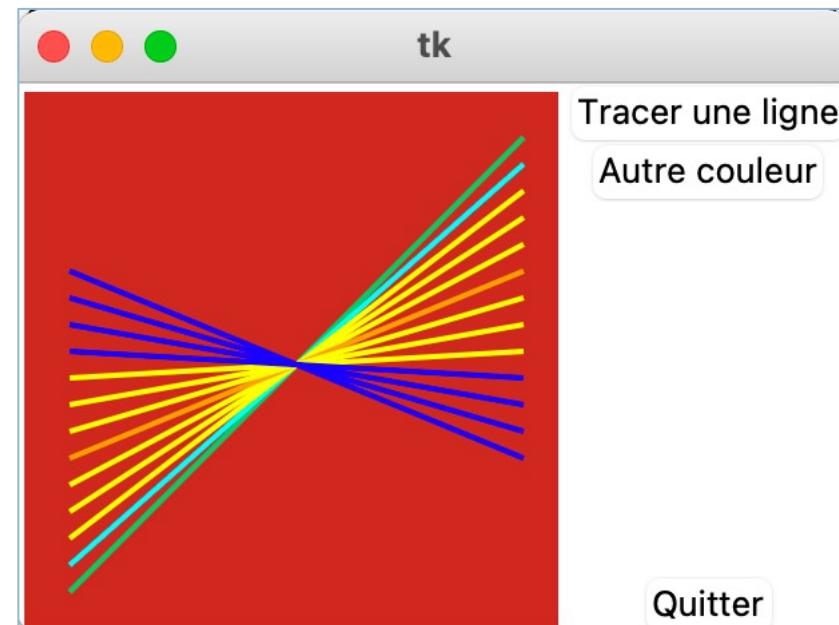
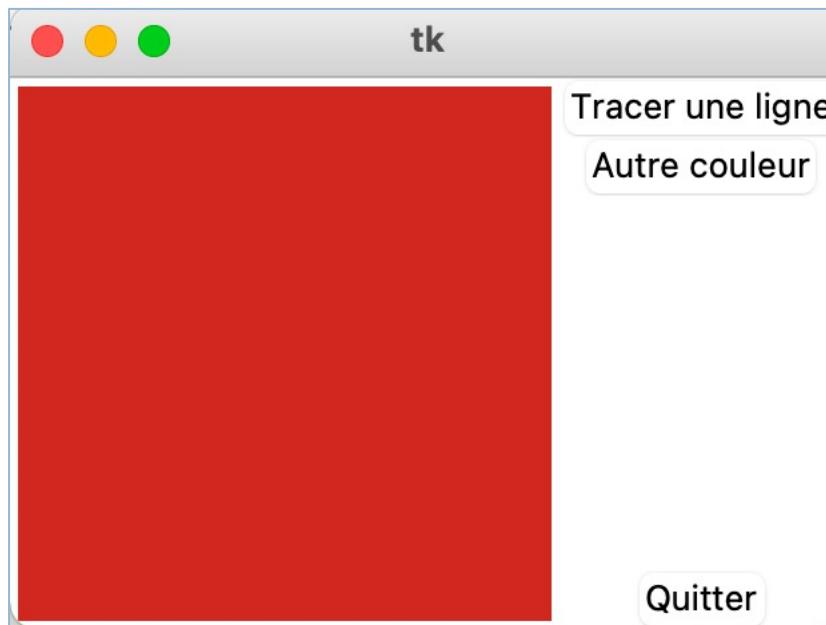
Résultat

20



Tracer dans un canevas

21



Code source

22

- [exemples/interface/lignes.py](#)

Associer un événement à un objet

23

- La méthode **bind** permet d'exécuter une fonction lorsqu'un certain événement donné est intercepté par un objet donné. La fonction exécutée accepte un seul paramètre de type **Event** qui est l'événement qui l'a déclenchée. Cette méthode a pour syntaxe :

w.bind(ev, fonction)

- w est l'identificateur de l'objet devant intercepter l'événement désigné par la chaîne de caractères ev .
- fonction est la fonction qui est appelée lorsque l'événement survient. Cette fonction ne prend qu'un paramètre de type Event.

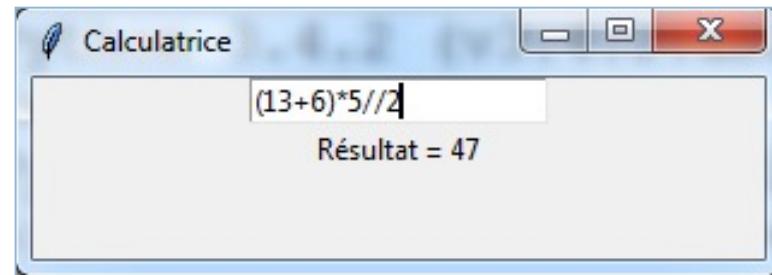
Liste des événements

<Key>	Intercepter la pression de n'importe quelle touche du clavier.
<Button-i>	Intercepter la pression d'un bouton de la souris. <i>i</i> doit être remplacé par 1,2,3.
<ButtonRelease-i>	Intercepter le relâchement d'un bouton de la souris. <i>i</i> doit être remplacé par 1,2,3.
<Double-Button-i>	Intercepter la double pression d'un bouton de la souris. <i>i</i> doit être remplacé par 1,2,3.
<Motion>	Intercepter le mouvement de la souris, dès que le curseur bouge, la fonction liée à l'événement est appelée.
<Enter>	Intercepter un événement correspondant au fait que le curseur de la souris entre la zone graphique de l'objet.
<Leave>	Intercepter un événement correspondant au fait que le curseur de la souris sorte la zone graphique de l'objet.

Calculatrice

25

```
from tkinter import*
from math import*
# définition de l'action à effectuer si l'utilisateur actionne
# la touche "enter" alors qu'il édite le champ d'entrée :
def evaluer(event):
    resultat.configure(text='Résultat = '+str(eval(zonne_text.get())))
##### Programme principal #####
fenetre = Tk()
fenetre.title('Calculatrice')
zonne_text = Entry(fenetre, background='white')
zonne_text.bind("<Return>", evaluer)
resultat = Label(fenetre)
zonne_text.pack()
resultat.pack()
fenetre.mainloop()
```



Explication

26

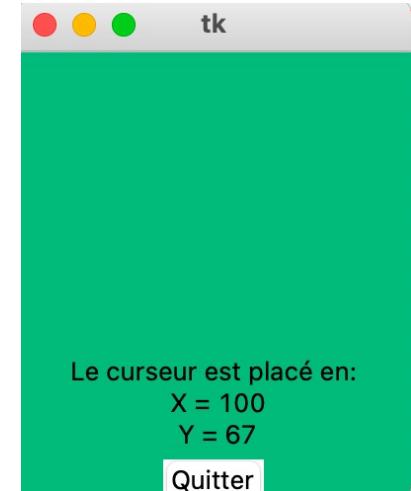
- La fonction evaluer() sera la commande exécutée par le programme lorsque l'utilisateur actionnera la touche Return. cette fonction utilise la méthode configure() du widget resultat, pour modifier son attribut text.
- eval() évalue (fait le calcul) la chaîne de caractères.
- str() transforme une expression numérique en chaîne de caractères.
- get() est une méthode qui permet d'extraire du widget zonne_text la chaîne de caractères qui lui a été fournie par l'utilisateur.
- zonne_text est un widget de la « classe » Entry. Afin que ce widget puisse transmettre au programme l'expression que l'utilisateur y aura encodée, il faut lui associer un événement à l'aide de la méthode bind() (bind signifie « lier » en anglais).
- zonne_text.bind("<Return>", evaluer) signifie : « lier l'événement 'pression sur la touche <Return>' à l'objet zonne_text , le gestionnaire de cet événement étant la fonction evaluer ».
- L'argument event fourni à la fonction evaluer est obligatoire dès que l'on utilise la méthode bind().

Détection et positionnement d'un clic de souris

```
# Détection des cliques de souris
from sys import *
# définition des fonctions gestionnaires d'événements
def clique(event):
    "On vient de cliquer"
    textMessage="Le curseur est placé en:\n X = "+str(event.x)+"\n Y = "+str(event.y)
    message.configure(text = textMessage)

# Programme principal
from tkinter import *
fen = Tk()
fen.config(background="#41B77F")

# Zone de clique
cadre = Frame(fen, width = 200, height = 150, bg = "#41B77F")
cadre.bind("<Button-1>", clique)
# Bouton de fin
bou = Button(fen, text="Quitter", bg = "#41B77F", command=fen.destroy)
# Message initial
message = Label(fen, text = 'Vous n\'avez pas encore cliqué',bg = "#41B77F")
cadre.pack()
message.pack()
bou.pack()
fen.mainloop()
# démarrage du récepteur d'événements
```



Explications

28

- La méthode bind() du widget cadre associe l'événement <clic à l'aide du premier bouton de la souris> au gestionnaire d'événement « pointeur ».
- Ce gestionnaire d'événement peut utiliser les attributs x et y de l'objet event généré automatiquement par Python, pour construire la chaîne de caractères qui affichera la position de la souris au moment du clic

Une fenêtre Toplevel

29

- Tk est la racine absolue de l'application, c'est le premier widget qui doit être instancié et l'interface graphique s'arrête lorsqu'elle est détruite.

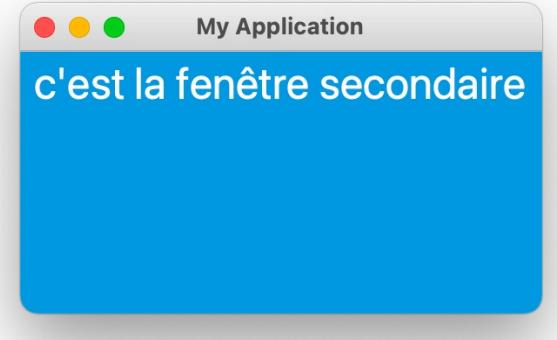
- Toplevel est une fenêtre dans l'application, la fermeture de la fenêtre détruira tous les widgets enfants placés sur cette fenêtre mais ne fermera pas le programme.

Exemple

30

```
from tkinter import*
def afficher():
    fenetre = Toplevel(root)
    fenetre.config(background="#3498DB")
    fenetre.geometry("300x150")
    label2 = Label(fenetre, text="c'est la fenêtre secondaire",
                   font=("Courrier", 25), bg="#3498DB", fg='white',)
    label2.pack()
    fenetre.mainloop()

root=Tk() #crée la fenêtre qui s'appellera mafenetre
root.title("My Application")
root.config(background="#41B77F")
#Ajouter une label à la tk
label1 = Label(root, text="c'est la fenêtre principale",font=("Courrier", 25),
               bg='white', fg="#41B77F")
label1.pack()
#le clique sur le boutton affiche la fenetre 2
button = Button(root, text="Nouvelle fenêtre", font=("Courrier", 25), bg='white', fg="#41B77F",
                command=afficher)
button.pack(pady=25, fill=X)
```



Menu Tkinter

- Ajout d'une barre de menu
- Widget Menu

```
from tkinter import *
```

```
fenetre = Tk()
```

```
menubar = Menu(fenetre)
fenetre.config(menu=menubar)
```

Menu Tkinter

- Ajout d'un onglet dans une barre de menu
- Widget Menu ajouté en cascade à la barre de menu

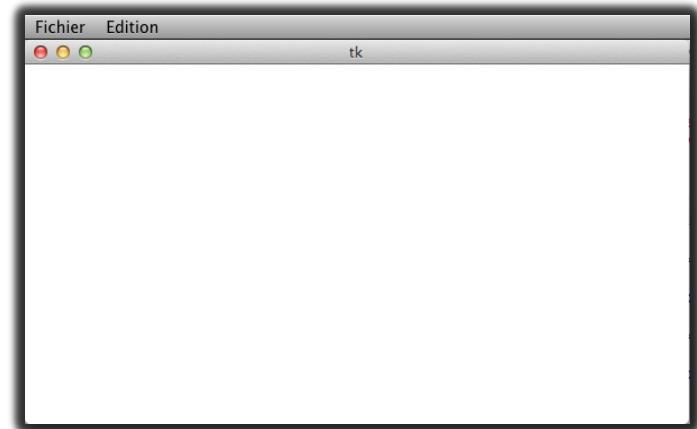
```
from tkinter import *
```

```
fenetre = Tk()
```

```
menubar = Menu(fenetre)
fenetre.config(menu=menubar)
```

```
menufichier = Menu(menubar, tearoff=0)
menubar.add_cascade(label="Fichier", menu=menufichier)
```

```
menuedition = Menu(menubar, tearoff=0)
menubar.add_cascade(label= "Edition", menu=menufichier)
```



Menu Tkinter

- Ajout d'un bouton dans un onglet de la barre de menu
- Méthode `add_command` de la classe `Menu`

```
from tkinter import *
```

```
fenetre = Tk()
```

```
menubar = Menu(fenetre)
fenetre.config(menu=menubar)
```

```
menufichier = Menu(menubar, tearoff=0)
menubar.add_cascade(label="Fichier", menu=menufichier)
```

```
menufichier.add_command(label="Ouvrir ")
menufichier.add_command(label="Enregistrer")
menufichier.add_command(label="Enregistrer sous")
menufichier.add_command(label="Quitter")
```



Menu Tkinter

- Ajout d'un séparateur dans un onglet de la barre de menu
- Méthode `add_separator` de la classe `Menu`

```
from tkinter import *
```

```
fenetre = Tk()
```

```
menubar = Menu(fenetre)
```

```
fenetre.config(menu=menubar)
```

```
menufichier = Menu(menubar, tearoff=0)
```

```
menubar.add_cascade(label="Fichier", menu=menufichier)
```

```
menufichier.add_command(label="Ouvrir ")
```

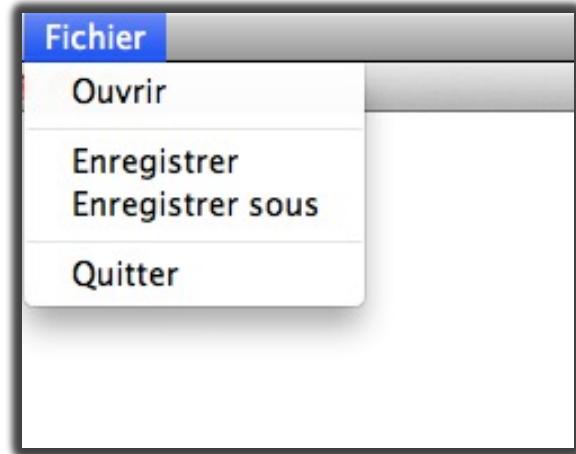
```
menufichier.add_separator()
```

```
menufichier.add_command(label="Enregistrer")
```

```
menufichier.add_command(label="Enregistrer sous")
```

```
menufichier.add_separator()
```

```
menufichier.add_command(label="Quitter")
```



Menu Tkinter

- Paramètres de `add_command`

```
from tkinter import *

fenetre = Tk()

menubar = Menu(fenetre)
fenetre.config(menu=menubar)

menufichier = Menu(menubar,tearoff=0)
menubar.add_cascade(label="Fichier", menu=menufichier)

def ouvrir():
    print ("Bonjour")

menufichier.add_command(label="Ouvrir", command=ouvrir)
menufichier.add_command(label="Quitter", command=fenetre.destroy)
```



Menu Tkinter

- Paramètres de add_command

```
from tkinter import *

fenetre = Tk()

menubar = Menu(fenetre)
fenetre.config(menu=menubar)

menufichier = Menu(menubar,tearoff=0)
menubar.add_cascade(label="Fichier", menu=menufichier)

def ouvrir():
    print ("Bonjour")

menufichier.add_command(label="Ouvrir", command=ouvrir)
menufichier.add_command(label="Quitter", command=fenetre.destroy)

=====
===== RESTART =====
>>>
>>> Bonjour
```



```

from tkinter import *
from tkinter import messagebox

def openFile():
    file = askopenfilename(title="Choose the file to open",
                           filetypes=[("PNG image", ".png"), ("GIF image", ".gif"), ("All files", "*")])
    print(file)

def doSomething():
    print("Menu clicked")

def doAbout():
    messagebox.showinfo("My title", "My message")

mafenetre = Tk()
mafenetre.config(background="#41B77F")
mafenetre.geometry('400x400')
mafenetre.title(" Mon application ")
mafenetre.iconbitmap("logo.ico")
menuBar = Menu()

menuFile = Menu(menuBar, tearoff=0)
menuFile.add_command(label="New", command=doSomething)
menuFile.add_command(label="Open", command=openFile)
menuFile.add_command(label="Save", command=doSomething)
menuFile.add_separator()
menuFile.add_command(label="Exit", command=quit)
menuBar.add_cascade( label="File", menu=menuFile)

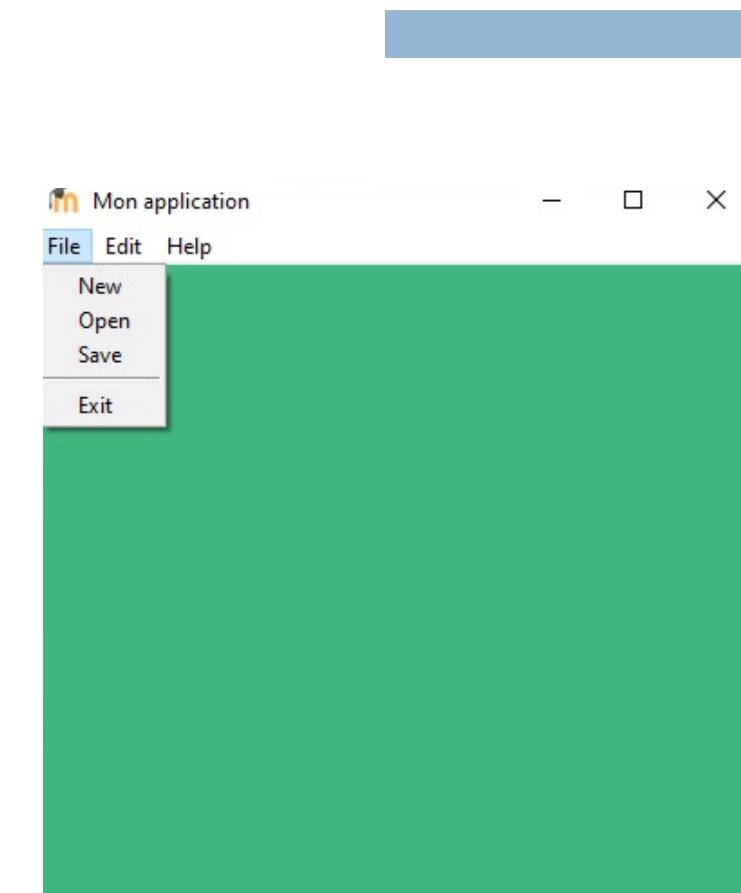
menuEdit = Menu(menuBar, tearoff=0)
menuEdit.add_command(label="Undo", command=doSomething)
menuEdit.add_separator()
menuEdit.add_command(label="Copy", command=doSomething)
menuEdit.add_command(label="Cut", command=doSomething)
menuEdit.add_command(label="Paste", command=doSomething)
menuBar.add_cascade( label="Edit", menu=menuEdit)

menuHelp = Menu(menuBar, tearoff=0)
menuHelp.add_command(label="About", command=doAbout)
menuBar.add_cascade( label="Help", menu=menuHelp)

mafenetre.config(menu=menuBar)

mafenetre.mainloop()

```



démo

38

The screenshot shows a Windows desktop environment. In the center, there is a code editor window titled "menu4.py - C:/Users/Amine/Desktop/PYTHON/menu4.py (3.9.1)". The code is written in Python and defines a menu system for a Tkinter application. The menu includes File, Edit, Format, Run, Options, Window, and Help menus. It includes functions for opening files, doing something, and displaying about information. The main window is titled "Mon application" and has a logo icon. The code uses the Tkinter library to create the interface.

```
menuFile = Menu(menuBar, tearoff=0)           [ ]
menuFile.add_command(label="New", command=doSomething)
menuFile.add_command(label="Open", command=openFile)
menuFile.add_command(label="Save", command=doSomething)
menuFile.add_separator()
menuFile.add_command(label="Exit", command=quit)
menuBar.add_cascade( label="File", menu=menuFile)

menuEdit = Menu(menuBar, tearoff=0)
menuEdit.add_command(label="Undo", command=doSomething)
menuEdit.add_separator()
menuEdit.add_command(label="Copy", command=doSomething)
menuEdit.add_command(label="Cut", command=doSomething)
menuEdit.add_command(label="Paste", command=doSomething)
menuBar.add_cascade( label="Edit", menu=menuEdit)

menuHelp = Menu(menuBar, tearoff=0)
menuHelp.add_command(label="About", command=doAbout)
menuBar.add_cascade( label="Help", menu=menuHelp)

mafenetre.config(menu=menuBar)

mafenetre.mainloop()
```

At the bottom of the screen, there is a taskbar with several icons: Start button, Search bar, File Explorer, Google Chrome, Settings, Task View, and others. The status bar at the bottom right shows "Ln: 32".

Positionnement des widgets

39

- Les widgets intègrent chacune un grand nombre de méthodes. On peut aussi leur associer (lier) des événements.
- Tous ces widgets peuvent être positionnés dans les fenêtres à l'aide de trois méthodes différentes :
 - **la méthode grid(),**
 - **la méthode pack()**
 - **La méthode place().**

1 - La méthode pack()

40

- Ce gestionnaire organise les widgets en blocs avant de les placer dans le widget parent
- Cette méthode empile les objets les uns à la suite des autres
- La méthode pack possède trois options :
 - Side : prend 4 valeurs TOP (par défaut), LEFT, BOTTOM, RIGHT
 - expand : prend 2 valeurs : True ou False (valeur par défaut), si cette option est vraie, l'objet occupe tout l'espace.
 - fill : égale à None (valeur par défaut), X, Y, BOTH, l'objet s'étend selon un axe (X ou Y ou les deux).

Exemple

41

*pack2.py - /Users/oumaira/Documents/Python 2019/cours formation continu/cours 2020 -2021/exemp

```
from tkinter import*
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")
mafenetre.geometry('400x200')

text1=Label(mafenetre, text='Votre nom ',fg='white',bg="#41B77F",
            font=("Courrier", 12))
text1.pack()
zone_nom = Entry ()
zone_nom.pack()

text2=Label(mafenetre, text='Votre Prénom ',fg='white',bg="#41B77F",
            font=("Courrier", 12))
text2.pack()
zone_prenom = Entry ()
zone_prenom.pack()

boutton_calculer = Button(mafenetre,text="Enregistrer",
                           font=("Courrier", 12))
boutton_calculer.pack(pady=25)

mafenetre.mainloop()
```



La méthode pack() avec l'option expand=True

42

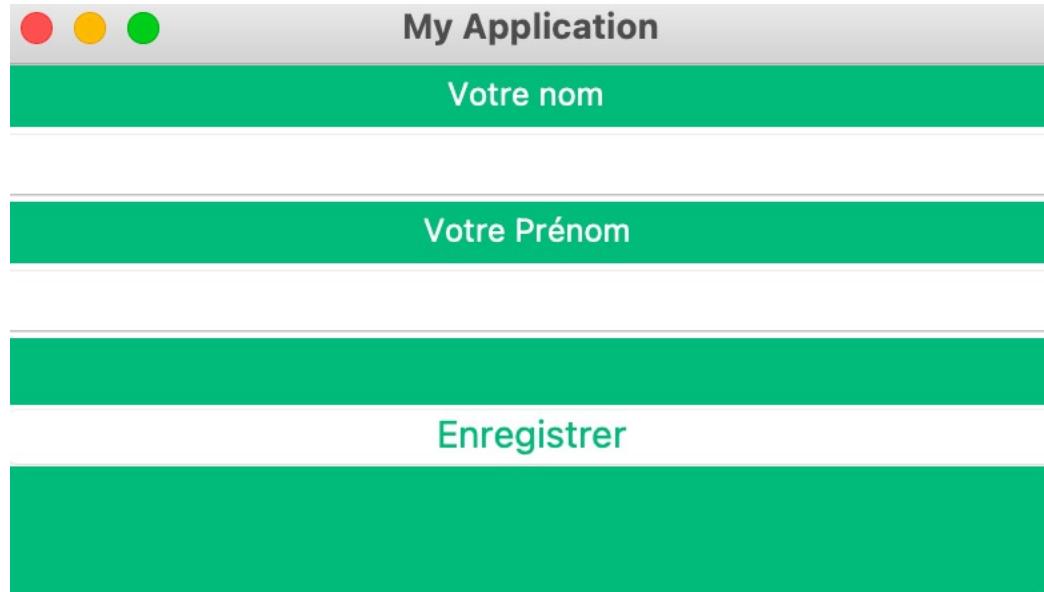
s.pack (expand=True)



La méthode pack() avec l'option fill=X

43

```
bouton_calculer.pack(fill=X,pady=25)
```



2 - La méthode *grid()*

- La méthode *grid()* considère la fenêtre comme un tableau, avec des lignes (*row*) et des colonnes(*column*).
 - La méthode *grid* possède plusieurs options, en voici cinq :
-
- **column** : Cette option est utilisée pour placer le widget dans une colonne qui est la colonne la plus à gauche. La valeur par défaut est 0.
 - **columnspan** : Normalement un widget occupe seulement une cellule. Cependant, vous pouvez regrouper plusieurs cellules d'une ligne en indiquant via **columnspan** le nombre de cellules à regrouper.
 - **row** : Cette option est lorsque le widget est à mettre dans une ligne, par défaut la première ligne est vide.
 - **rowspan** : Cette option sera utilisée pour dire combien de rowwidgets sont occupés et la valeur par défaut est 1.

L'option sticky de La méthode *grid()*

45

- L'option **sticky** indique ce qu'il faut faire lorsque la case est plus grande que l'objet qu'elle doit contenir.
- Par défaut, l'objet est centré mais il est possible d'aligner l'objet sur un ou plusieurs bords en précisant que `sticky="N" ou "S" ou "W" ou "E"`.
- Pour aligner l'objet sur un angle, il suffit de concaténer les deux lettres correspondant aux deux bords concernés.
- Il est aussi possible d'étendre l'objet d'un bord à l'autre en écrivant `sticky="N+S" ou sticky="E+W"`.

Exemple

46

```
mafenetre=Tk() #crée la fenêtre qui s'appellera mafenetre
mafenetre.title("My Application")
mafenetre.config(background="#41B77F")
mafenetre.geometry('400x200')

text1=Label(mafenetre, text='Votre nom ',fg='white',bg="#41B77F",
            font=("Courier", 12))
text1.grid(row=0, column=0)
zone_nom = Entry ()
zone_nom.grid(row=0, column=1)

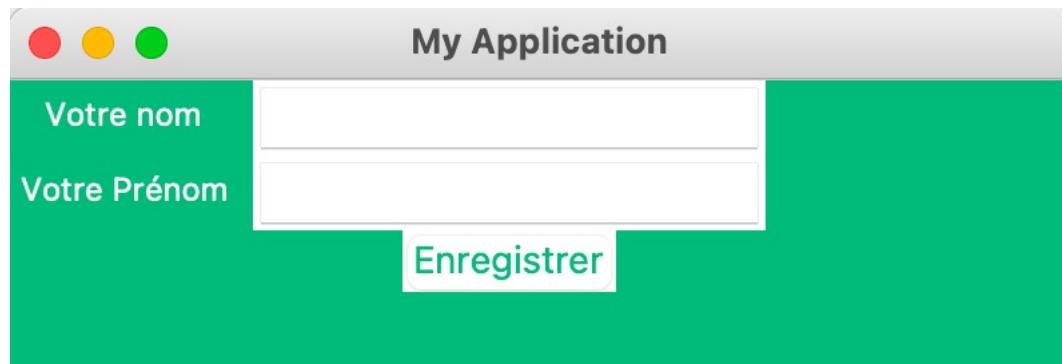
text2=Label(mafenetre, text='Votre Prénom ',fg='white',bg="#41B77F",
            font=("Courier", 12))
text2.grid(r|ow=1, column=0)
zone_prenom = Entry ()
zone_prenom.grid(row=1, column=1)

boutton_calculer = Button(mafenetre,text="Enregistrer",
                           font=("Courier", 14),fg='#41B77F')
boutton_calculer.grid(row=2, column=1)

mafenetre.mainloop()
```

Résultat

47



3- la méthode place()

- Ce gestionnaire de géométrie organise les widgets en les plaçant dans une position spécifique définie par des coordonnées dans le widget parent.

- L'inconvénient de cette méthode survient lorsqu'on cherche à modifier l'emplacement d'un objet : il faut en général revoir les positions de tous les autres éléments de la fenêtre. On procède souvent par tâtonnement pour construire une fenêtre et disposer les objets. Ce travail est beaucoup plus long avec la méthode place.

Exemple

49

```
from tkinter import *

mafenetre = Tk()

nom = Label(mafenetre, text = "Nom")
nom.place(x = 10, y = 10)
zone_nom = Entry(mafenetre)
zone_nom.place(x = 90, y = 10)

prenom = Label(mafenetre, text = "Prénom")
prenom.place(x = 10, y = 50)
zone_prenom = Entry(mafenetre)
zone_prenom.place(x = 90, y = 50)

btn = Button(mafenetre, text = "Cliquez ici!")
btn.place(x = 100, y = 100)
mafenetre.geometry("300x200")
mafenetre.mainloop()
```

