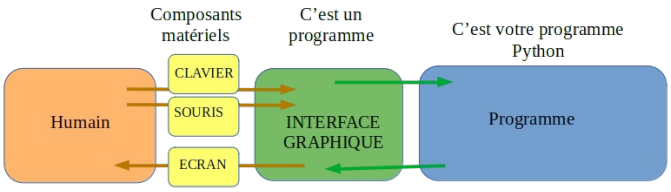
CC2.0 BY NC SA – www.infoforall.fr

17 -Interface et variables globales (Programmation)

I – Applications aux interfaces graphiques

1.1 Interface graphique

Une interface graphique est un programme qui permet l'interaction entre un utilisateur et votre propre programme.



1.2 Capteurs et actionneurs

L'interaction passe par deux types de composants physiques :

- **Capteur** : composant capable de transmettre une information issue du monde physique en signal électronique ou informatique.
- **Actionneur** : composant capable de provoquer ,à partir d'un signal électronique, une action réelle dans le monde physique

II – Application graphique via Tkinter

2.1 Pour créer une application graphique Tkinter avec le constructeur Tk()

```
import tkinter as tk ____________________________________ Importation du module contenant le code de Tkinter
application = tk.Tk() ___________________________________ Tk() renvoie la réf. d'une nouvelle application
application.geometry("600x400") _________________________ geometry() pour définir les dimensions de la fenêtre
application.title("Mon beau titre") _____________________ title() pour définir le titre sur la fenêtre
application.configure(bg="black") _______________________ configure() pour modifier un attribut, ici bg
. . .
application.mainloop()__________________________________ mainloop() lance la surveillance les événements
                                                          TOUJOURS LA DERNIERE INSTRUCTION
```

2.2 Créer et un widget Texte avec le constructeur Label()

En plusieurs étapes :

```
wL1C1 = tk.Label(application, text="L1C1") ______________ Label() renvoie la réf. D'une nouveau widget
wL1C1.configure(fg="white") _____________________________ configure() pour modifier la couleur d'écriture(fg)
wL1C1.configure(bg="white") _____________________________ configure() pour modifier la couleur de fond (bg)
wL1C1.configure(width=10) _______________________________ configure() pour modifier la largeur (width)
wL1C1.configure(height=5) _______________________________ configure() pour modifier la hauteur (height)
wL1C1.place(x=25, y=25) _________________________________ place() pour afficher réellement le widget en x, y
```

En une fois :

```
wL1C1 = tk.Label(application, text="L1C1", fg="white", bg= "red", width=10, height=5)
```

→ on doit fournir d'abord la fenêtre à laquelle attacher le widget
→ puis les attributs text pour le texte à afficher, fg pour la couleur d'écrite, bg pour le fond, width-height pour la taille.

```
wL1C1.place(x=25, y=25)__________________________________ place() pour afficher réellement le widget en x,y
```

2.3 Lier un événenement à un widget

```
wL1C1.bind('<Button-1>', modifier) _______________ bind() lie le clic-gauche ('<Button-1>') sur le
                                                   widget wL1C1 à la fonction modifier
```

2.4 Fonction événementielle

Ces fonctions ont comme particularités :
- d'être **appelées automatiquement** lorsque l'événement survient
- de recevoir les **informations sur l'événement** en cours via leur PREMIER PARAMETRE.

Exemple d'une fonction qui déplace le widget, change sa couleur et son texte :

```
def modifier(event):
    '''Fonction événementielle : event se remplit automatiquement'''
        widget_actif = event.widget           # permet de récupérer la référence du widget
        cx = widget_actif.winfo_x()         # méthode winfo_x renvoie la coordonnée x du widget
        cy = widget_actif.winfo_y()         # méthode winfo_y renvoie la coordonnée y du widget
        widget_actif.place(x=cx+10, y=cy)
        widget_actif.configure(text="Touché !")
```

III – Exemple commenté d'application graphique via Tkinter

```
# 1 - Importations
import tkinter as tk

# 2 - Déclaration des constantes
COULEUR_0 = "#666666"
COULEUR_1 = "#448800"
COULEUR_2 = "#884400"
COULEUR_3 = "#004488"
COULEUR_4 = "#888800"
FOND = "black"  # Couleur de fond

# 3 - Déclaration des fonctions

def modifier(event):
    '''Fonction événementielle : event se remplit automatiquement'''
    widget_clic = event.widget  # On récupère le widget ayant créé l'événement
    couleur_actuelle = widget_clic.cget('bg')  # méthode cget pour lire un attribut d'un widget
    if couleur_actuelle == COULEUR_0:
        widget_clic.configure(bg=COULEUR_1)  # méthode configure permet de modifier un attribut
    elif couleur_actuelle == COULEUR_1:
        widget_clic.configure(bg=COULEUR_2)

def deplacer(event):
    '''Fonction événementielle : event se remplit automatiquement'''
    for i in range( len(tableau_widgets) ):
        widget_en_cours = tableau_widgets[i]
        cx = widget_en_cours.winfo_x()  # méthode winfo_x renvoie la coordonnée x du widget
        cy = widget_en_cours.winfo_y()  # méthode winfo_y renvoie la coordonnée y du widget
        widget_en_cours.place(x=cx+10, y=cy)

# 4 - Programme

# 4-1 - Création et configuration de la fenêtre du logiciel
application = tk.Tk()
application.geometry("600x400")
application.title("Démonstration lecture des variables globales")
application.configure(bg=FOND)

# 4-2 - Création et affichage des Labels
wL1C1 = tk.Label(application, text="L1C1", fg="white", bg=COULEUR_0, width=10, height=5)
wL1C1.place(x=25, y=25)

wL1C2 = tk.Label(application, text="L1C1", fg="white", bg=COULEUR_0, width=10, height=5)
wL1C2.place(x=125, y=25)
wL1C3 = tk.Label(application, text="L1C2", fg="white", bg=COULEUR_0, width=10, height=5)
wL1C3.place(x=225, y=25)
wL2C1 = tk.Label(application, text="L2C3", fg="white", bg=COULEUR_0, width=10, height=5)
wL2C1.place(x=25, y=125)
wL2C2 = tk.Label(application, text="L2C2", fg="white", bg=COULEUR_0, width=10, height=5)
wL2C2.place(x=125, y=125)
wL2C3 = tk.Label(application, text="L2C3", fg="white", bg=COULEUR_0, width=10, height=5)
wL2C3.place(x=225, y=125)
wL3C1 = tk.Label(application, text="L3C1", fg="white", bg=COULEUR_0, width=10, height=5)
wL3C1.place(x=25, y=225)
wL3C2 = tk.Label(application, text="L3C2", fg="white", bg=COULEUR_0, width=10, height=5)
wL3C2.place(x=125, y=225)
wL3C3 = tk.Label(application, text="L3C3", fg="white", bg=COULEUR_0, width=10, height=5)
wL3C3.place(x=225, y=225)

tableau_widgets = [wL1C1, wL1C2, wL1C3, wL2C1, wL2C2, wL2C3, wL3C1, wL3C2, wL3C3]

# 4-3 - Création du gestionnaire d'événements

for i in range( len(tableau_widgets) ):               # Déclaration automatique, c'est plus joli non ?
    tableau_widgets[i].bind('<Button-3>', deplacer)

wL1C1.bind('<Button-1>', modifier)                  # ou Déclaration manuelle des événéments, c'est long...
wL1C2.bind('<Button-1>', modifier)
wL1C3.bind('<Button-1>', modifier)
wL2C1.bind('<Button-1>', modifier)
wL2C2.bind('<Button-1>', modifier)
wL2C3.bind('<Button-1>', modifier)
wL3C1.bind('<Button-1>', modifier)
wL3C2.bind('<Button-1>', modifier)
wL3C3.bind('<Button-1>', modifier)

# 4-4 - Surveillance des événements
application.mainloop()
```