TP PYTHON : INTERFACE GRAPHIQUE AVEC LE MODULE TKINTER

1 – MODULE TKINTER

*Le module Tkinter (**T**ool **K**it **inter**face) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Le module Tkinter est installé par défaut dans les différentes interfaces de développement Python.*

*Ce module propose de nombreux **composants graphiques** ou **widgets** : **fenêtre** (Tk), **bouton** (Button), **case à cocher** (Checkbutton), **étiquette** (Label), **zone de texte** simple (Entry), **menu** (Menu), **zone graphique** (Canvas), **cadre** (Frame)...*

*Il permet également de **gérer de nombreux événements** : clic sur la souris, déplacement de la souris, appui sur une touche du clavier, top d'horloge...*

*Le module Tkinter permet de réaliser de la **programmation événementielle**. Ce type de programme est basé sur une « **boucle infinie** » qui permet d'attendre **l'apparition des évènements**.*

Dans le cas de Tkinter, la boucle infinie est réalisée au moyen de l'instruction `mainloop()`(boucle principale).

La structure d'un programme utilisant le module Tkinter présente généralement la forme suivante :

- *Définition de **l'interface graphique**.*
- *Définition des **évènements**.*
- *Définition de la **boucle principale**.*

Exercice 1

1. **Editer** et **exécuter** le script suivant :

```
from tkinter import *
fen=Tk()
fen.mainloop()
```

2. **Commenter** chacune des lignes du script précédent.

2 – WIDGET

2.1 – Widgets Button et Label

*Un widget **bouton** (Button) permet de **proposer une action à l'utilisateur**. Un **label** est un espace prévu pour **afficher un texte**.*

Les widgets seront placés dans la fenêtre graphique. La méthode `pack()` *permet de **placer les widgets dans la fenêtre** et de réduire automatiquement la taille de la fenêtre afin qu'elle soit juste assez grande pour contenir le widget. Cette méthode accepte les attributs suivants :*

- ***side** : cet attribut peut prendre les valeurs TOP, BOTTOM, LEFT ou RIGHT, pour placer le widget du côté correspondant dans la fenêtre :*
- ***padx** et **pady** : ces attributs permettent de réserver un espace autour du widget. Cet espace est exprimé en nombre de pixels : **padx** réserve un espace à gauche et à droite du widget, **pady** réserve un espace au-dessus et au-dessous du widget.*

Syntaxe metode pack()

```
bouton.pack(side=RIGHT, padx =3, pady =3)
```

Exercice 2

1. **Editer** et **exécuter** le script suivant :

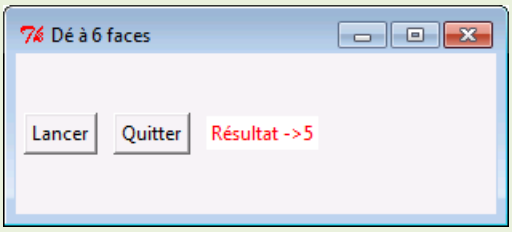
```
1   from tkinter import *
2
3   # Création de la fenêtre principale
4   Mafenetre=Tk()
5
6   # Création d'un widget Label
7   Label1 = Label(Mafenetre, text = 'Bienvenue au lycée Branly', fg = 'red')
8   Label1.pack()
9
10  # Création d'un widget Button
11  Bouton1 = Button(Mafenetre, text = 'Quitter', command = Mafenetre.destroy)
12  Bouton1.pack()
13
14  # Lancement de la boucle infinie (gestionnaire d'événements)
15  Mafenetre.mainloop()
```

2. **Commenter** les lignes 7 et 11 du script précédent.
3. **Modifier** le programme afin d'afficher le label à droite et le bouton à gauche avec un espace de 5 pixels par rapport aux bords de la fenêtre.
4. **Modifier** le programme afin d'afficher « Bienvenue en NSI » en bleu.

Exercice 3 : Dé à 6 faces

1. **Créer** un script de.py.

Ce script permet de simuler un dé à 6 faces.



*L'appui sur le **bouton « Lancer »** permet d'exécuter la fonction* `NouveauLance()` *qui doit permettre de **générer un nombre aléatoire compris entre 1 et 6**. Ce nombre sera affecté à la variable Texte qui est une chaîne de caractère variable (StringVar).*

2. **Ajouter** la ligne permettant de générer le nombre aléatoire.

```
nb = .....
```

3. **Justifier** la nécessité d'utilisation de l'instruction `str()`

4. **Ajouter** les lignes permettant de créer le bouton « Lancer » en respectant la forme donnée sur l'image ci-dessus.

```
BoutonLancer = Button(.....)
BoutonLancer.pack(.....)
```

5. **Ajouter** les lignes permettant de créer le bouton « Quitter » en respectant la forme donnée sur l'image ci-dessus.

```
BoutonQuitter = Button(.....)
BoutonQuitter.pack(.....)
```

6. **Exécuter** le script et **vérifier** le fonctionnement du programme.

7. **Modifier**-le afin d'obtenir d'autres couleurs d'affichage.

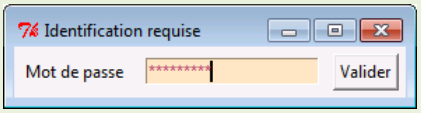
8. **Modifier**-le script afin d'obtenir une fenêtre placée au milieu de votre écran et intitulée « Jeu du dé ».

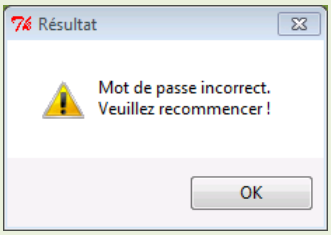
2.2 – Widgets Entry et boîtes de dialogue

Le widget « Entry » permet de saisir du texte.

Exercice 4 : Mot de passe

1. **Créer** un script mot_passe.py. Ce script doit permettre de réaliser une authentification.
   Importer le sous-module de tkinter nécessaire pour utiliser les boîtes de dialogue.





Le widget « Entry » permet de récupérer le mot de passe saisi par l'utilisateur. L'appui sur le ***bouton « Valider »*** *permet d'exécuter la fonction* `Verification()` *qui doit tester si le mot de passe est correct. Si le mot de passe est correct une boîte de dialogue « showinfo » apparaît, informant que le mot de passe est correct et la fenêtre Tkinter est fermée. Si le mot de passe est incorrect une boîte de dialogue « showwarning » apparaît, informant que le mot de passe est incorrect.*

2. **Ajouter**, dans une fonction nommée « Verification() », une ligne permettant de tester si le mot de passe est égal à « NSI_Branly » et qui ouvre la boîte de dialogue adéquate (cf. ci-dessus).

```
if Motdepasse.get() .....
```

3. **Ajouter** une ligne permettant de fermer la fenêtre Tkinter « Mafenetre » si le mot de passe est bon.

.....

4. **Ajouter**, dans le script principal, les lignes permettant de créer la fenêtre Tkinter « Mafenetre » et de lui donner le titre « Identification requise ».

.....

.....

5. **Ajouter** les lignes permettant de créer le label « Mot de passe » en respectant la forme donnée sur l'image ci-dessus.

```
Label1 = .....
Label1.pack(.....)
```

6. **Ajouter** les lignes pour créer le widget « Entry » en respectant la forme donnée ci-dessus.

```
Champ = Entry(.....)
Champ.pack (.....)
```

7. **Ajouter** les lignes pour créer le bouton « Valider » en respectant la forme donnée ci-dessus.

```
Bouton = (.....)
Bouton.pack(.....)
```

8. **Exécuter** le script et **vérifier** le fonctionnement du programme.

2.3 – Widget Canvas (canevas) et gestion des images.

*Un **canevas** est une **zone rectangulaire** intégrée à la fenêtre graphique et destinée à contenir des dessins, des figures complexes ou images. Il est possible d'y placer des graphiques, du texte, d'autres widgets ou des cadres (*frames*).*

L'objet « Canvas » peut accepter 4 paramètres :

- ***fen** : fenêtre tkinter dans laquelle sera integré le canevas*
- ***width** : largeur du canevas*
- ***height** : hauteur du canevas*
- ***background** : couleur de fond du canevas (par défaut, la couleur de fond est grise).*

Syntaxe

```
can=Canvas(fen,width=1000,height=1000,background='red')
```

*Création d'un canevas intégré à la **fenêtre Tkinter « fen »** de **dimensions 1000x1000** et de **fond d'écran rouge**.*

*Chaque élément affiché sera représenté par un **item placé dans une liste d'affichage**.*

La méthode `create_image(x,y,anchor=…,image=…)` *permet d'afficher une image dans un canevas :*

- ***x,y** : coordonnées du point par rapport auquel sera positionnée l'image.*
- ***anchor** : position de l'image par rapport aux coordonnées (x,y). L'image sera positionnée de sorte que le point* (x, y) *soit situé au milieu de son bord inférieur (sud) si* `anchor = S`*. Par défaut* `anchor = CENTER` *et dans ce cas l'image sera positionnée de sorte que le point* (x, y) *soit situé au centre de l'image.*
- *image : nom de l'image à afficher.*

NW	N	NE
W		E
SW	S	SE

Syntaxe

```
Item1=can.create_image(0,100,anchor=N,image=photo)
```

*Affichage de l'image « **photo** » dans le canevas « **can** », l'image sera positionnée de sorte que le point* (**0,100**) *soit situé au **milieu de son bord supérieur**.*

Exercice 5 : Afficher des images

1. **Créer** un script aff_images.py.

Ce script permet d'afficher deux images dans un canevas. Un bouton quitter permet de fermer la fenêtre Tkinter.

L'instruction `PhotoImage()` *permet de* ***charger le fichier contenant l'image*** *et de la rendre compatible avec Python. Tkinter ne reconnaît que les formats d'images GIF et PNM.*

2. **Ajouter** les lignes permettant de créer la fenêtre Tkinter « Mafenetre » et de lui donner le titre « Affichage Images ».

.....

.....

3. **Ajouter** la ligne permettant de créer un canevas de dimension **500x500** et de couleur de fond blanche.

```
Canevas = Canvas(.....)
```

4. **Ajouter** les lignes permettant de placer les deux images en respectant la disposition présentée sur l'image ci-dessus.

```
item1 = Canevas.create_image(.....)
item2 = Canevas.create_image(.....)
```

5. **Ajouter** les lignes permettant de créer le bouton « Quitter ».

```
BoutonQuitter = (.....)
BoutonQuitter.pack(.....)
```

6. **Exécuter** le script et **vérifier** le fonctionnement du programme.

3 – EVENEMENTS

Lorsqu'un évènement apparaît, une fonction spécifique appelée gestionnaire d'évènement doit être exécutée, pour cela il est nécessaire de lier l'évènement à son gestionnaire par l'intermédiaire de la méthode `bind()`.

Exemple

```
# Gestionnaire d'évènement
def clavier(event):
  touche = event.keysym
  print(touche)

canvas = Canvas(fenetre, width=500, height=500)
canvas.focus_set()

# L'appui sur une touche du clavier est lié au gestionnaire clavier
canvas.bind("<Key>",clavier)
canvas.pack()
```

3.1 – Gestion de la souris

Les principaux évènements dus à la souris sont :

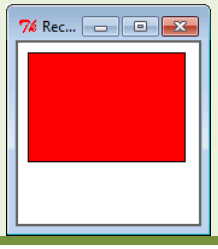
Evènement	*Description*
`<Button-1>`	*Clic gauche*
`<Button-2>`	*Clic droit*
`<Button-3>`	*Clic milieu*
`<Double-Button-1>`	*Double clic gauche*
`<Double-Button-2>`	*Double clic droit*
`<ButtonRelease>`	*Clic relaché*
`<Motion>`	*Mouvement de la souris*
`<B1-Motion>`	*Mouvement de la souris avec clic gauche*
`<MouseWheel>`	*Utilisation de la roulette*

La méthode `create_rectangle(x0,y0,x1,y1,outline=…,fill=…)` *permet de dessiner un rectangle dans un canevas :*

- ***x0,y0** : coordonnées du point supérieur gauche du rectangle.*
- ***X1,y1** : coordonnées du point inférieur droit.*
- ***outline** : couleur du contour du rectangle.*
- ***fill** : couleur de remplissage du rectangle.*

Exemple

```
Item1=can.create_rectangle(10,10,140,100,outline="black",fill="red")
```

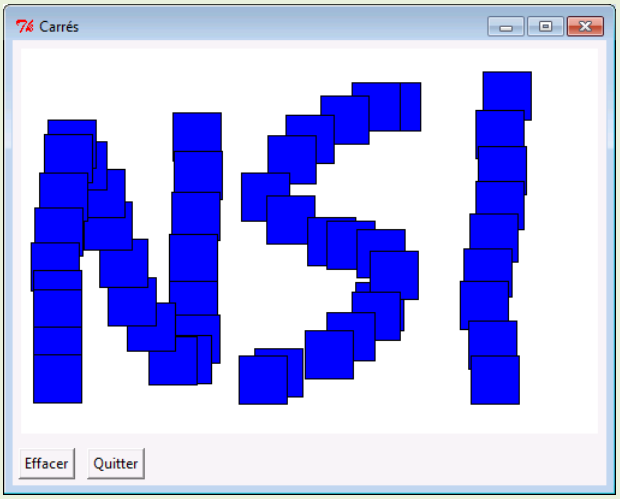


*Dessin d'un rectangle dans le canevas « **can** ». Le point supérieur gauche du rectangle a coordonnées* (**10,10**) *et le point inférieur droit* (**140,100**).
Le rectangle est **rouge** avec un bord **noir**.

Exercice 6 : Clic souris

1. **Créer** un script clic_souris.py.

Ce script permet de dessiner un carré bleu à l'endroit du clic gauche de la souris.



2. **Ajouter** les lignes permettant de créer la fenêtre Tkinter « Mafenetre » et de lui donner le titre « Carrés ».

.....

.....

3. **Ajouter** la ligne permettant de créer un canevas de dimension **480x320** et de couleur de fond blanche.

```
Canevas = Canvas(.....)
```

4. **Ajouter**, dans une fonction « Clic(event) », la ligne permettant de dessiner un carré bleu (contour noir), à partir des coordonnées X et Y du clic (centre du carré généré de taille 40).

```
Canevas.create_rectangle(.....)
```

5. **Ajouter** la ligne permettant de lier le clic sur le bouton gauche de la souris à au gestionnaire d'évènement « Clic ».

```
Canevas.bind(.....)
```

6. **Ajouter** les lignes permettant de créer le bouton « Effacer ».

```
BoutonEffacer = (.....)
BoutonEffacer.pack(.....)
```

7. **Ajouter** les lignes permettant de créer le bouton « Quitter ».

```
BoutonQuitter = (.....)
BoutonQuitter.pack(.....)
```

8. **Exécuter** le script et **vérifier** le fonctionnement du programme.

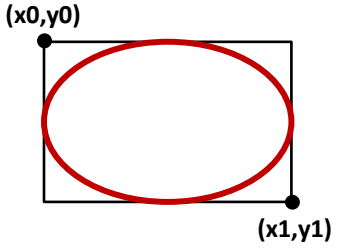
3.2 – Gestion du clavier

Les principaux évènements dus au clavier sont :

Evènement	*Description*
`<KeyPress>`	*Appui sur une touche*
`<KeyPress-a>`	*Appui sur la touche « a »*
`<KeyPress-A>`	*Appui sur la touche « A »*
`<Return>`	*Appui sur la touche « ENTREE »*
`<Escape>`	*Appui sur la touche « ECHAP »*
`<Up>`	*Pression sur la flèche directionnelle HAUT*
`<Down>`	*Pression sur la flèche directionnelle BAS*
`<Left>`	*Pression sur la flèche directionnelle GAUCHE*
`<Right>`	*Pression sur la flèche directionnelle DROITE*

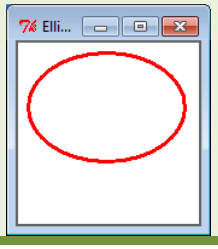
La méthode `create_oval(x0,y0,x1,y1,width=…, outline=…,fill=…)` *permet de dessiner une ellipse ou un cercle qui s'inscrit dans le rectangle (ou le carré) de coordonnées* (x0, y0), (x1, y1) :

- ***x0,y0*** *: coordonnées du point supérieur gauche du rectangle.*
- ***x1,y1*** *: coordonnées du point inférieur droit.*
- ***width*** *: largeur du contour de l'ellipse ou du cercle..*
- ***outline*** *: couleur du contour de l'ellipse ou du cercle.*
- ***fill*** *: couleur de remplissage de l'ellipse ou du cercle.*



Exemple

```
cercle=can.create_oval(10,10,140,100,width=3,outline="red",fill="white")
```



*Dessin d'une **ellipse** dans le canevas « **can** ». Le point supérieur gauche du rectangle dans lequel s'inscrit l'ellipse a coordonnées* (**10,10**) *et le point inférieur droit* (**140,100**). L'ellipse est **blanche** avec un **contour rouge** d'**épaisseur 3**.

La méthode `coords(item,x0,y0,x1,y1,x2,y2,……)` *permet de déplacer un item (figure, dessin, image…) en précisant ses nouvelles coordonnées :*

- ***x0,y0** : nouvelles coordonnées du premier point definissant l'item.*
- ***x1,y1** : nouvelles coordonnées du second point definissant l'item.*
- ***x2,y2** : nouvelles coordonnées du troisème point definissant l'item.*
- *…*

Exemple

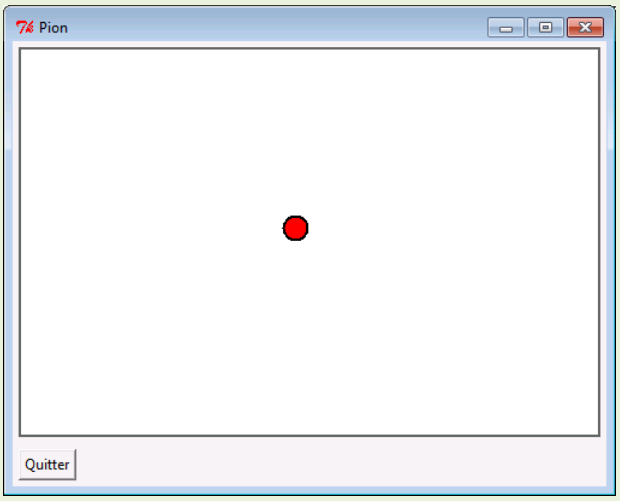
```
can.coords(carre,150,150,190,190)
```

Déplacement d'un carré de 50 pixels

Exercice 7 : Déplacer un cercle

1. **Créer** un script pion.py.

Ce script permet de déplacer un cercle de 20 pixels selon l'appui sur les touches h (haut), b (bas), g (gauche) d (droite) un carré bleu à l'endroit du clic gauche de la souris.



*Les variables **GLOBALES** PosX et PosY représentent les coordonnées du centre du cercle.*

2. **Ajouter** la ligne permettant de dessiner un cercle rouge nommé « Pion » (contour noir, épaisseur de 2) de rayon 20 à partir des coordonnées PosX et PosY du centre du cercle.

```
Pion = Canevas.create_oval(.....)
```

4. **Ajouter** la ligne permettant de lier l'appui sur une touche du clavier au gestionnaire d'évènement « Clavier ».

```
Canevas.bind(.....)
```

5. **Ajouter**, dans une fonction nommée « Clavier(event) », les lignes permettant de calculer la nouvelle valeur de PosY si la touche « h » est appuyée.

```
if touche.....
  PosY .....
```

6. **Ajouter** les lignes permettant de calculer la nouvelle valeur de PosY si la touche « b » est appuyée.

```
if touche.....
  PosY .....
```

7. **Ajouter** les lignes permettant de calculer la nouvelle valeur de PosX si la touche « d » est appuyée.

```
if touche.....
  PosX .....
```

8. **Ajouter** les lignes permettant de calculer la nouvelle valeur de PosX si la touche « g » est appuyée.

```
if touche.....
  PosX .....
```

9. **Ajouter** la ligne permettant de déplacer le cercle.

```
Canevas.coords(.....)
```

10. **Exécuter** le script et **vérifier** le fonctionnement du programme.