6

TP : FONCTIONS, INTERFACES GRAPHIQUES

Exercice 1 :

1. Les programmes suivants fonctionnent-ils ? Si oui, indiquer ce qui s'affiche. Si non, indiquer quelle est l'erreur.

`def produit(i,j):`
`  return i*j`
 	
`print(produit(2,3))`
`print(j)`
	`def concat(a,b):`
	`  return a+b+c`
	
	`a="Hello"`
	`b="world"`
	`c="!"`
	`print(concat(a,b))`
		`def mask():`
		`  p=20`
		`  print(p,q)`
		
		`p=15`
		`q=38`
		`mask()`
		`print(p,q)`

2. On donne ci-dessous un programme :

```
#définition des fonctions
def f1():
    print(x)

def f2():
    x=3
    print(x)

def f3():
    global x
    x=3
    y=12
    print(x)
    print(y)

#programme principal
x=5
print(x)
f1()
print(x)
f2()
print(x)
f3()
print(x)
print(y)
```

Indiquer à côté de chaque ligne du programme principal ce qui s'affiche ou s'il se produit une erreur.

Exercice 2 :

Dans cet exercice, nous allons découvrir brièvement le module tkinter qui permet de créer des interfaces graphiques. Pour pouvoir utiliser tkinter, il faut l'importer au début du programme.

1. Création d'une fenêtre

Taper le code suivant :

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

Le tester. Que se passe-t-il ? .....

Explications :

Ces 2 instructions sont obligatoires pour créer et utiliser une fenêtre graphique. La première instruction sert à créer une fenêtre graphique qui s'appelle fen(le nom peut évidemment être changé !). La deuxième instruction est indispensable pour afficher la fenêtre créée et pour pouvoir agir dans cette fenêtre (par exemple, cliquer, entrer un texte...). Il est possible de redimensionner la fenêtre en écrivant par exemple `fen.geometry("300x100")`, ce qui donnera une largeur de 300px et une hauteur de 100px à notre fenêtre.

Sur notre fenêtre, nous allons pouvoir placer différents objets, qu'on appelle des widgets. Nous allons en découvrir trois dans cet exercice.

2. Ajout d'un texte
   a) Compléter le code précédent par les lignes en gras :

```
fen=Tk()
texte1=Label(fen,text="Bonjour !")
texte1.grid(row=0,column=0)
fen.mainloop()
```

   Que se passe-t-il ? .....

   Explications :
   L'instruction `texte1=Label(fen,text="Bonjour !")` permet de créer une ligne de texte (mais pas de l'afficher !). Le widget Label prend comme premier argument le nom de la fenêtre dans laquelle on travaille. En 2e argument, on indique le texte à afficher.
   L'instruction `texte1.grid(row=0,column=0)` permet d'afficher la ligne de texte. La méthode `grid()` considère notre fenêtre comme étant une grille. Il faut donc lui indiquer dans quelle colonne et quelle ligne de notre fenêtre on souhaite placer notre widget. On peut numéroter les lignes et les colonnes comme on veut (généralement on commence à 0 ou à 1). Ici, on place notre texte à l'intersection de la ligne 0 et de la colonne 0 (remarque : tant qu'il n'y a pas d'autre widget, on peut indiquer n'importe quelles valeurs, cela ne change rien).

   b) Il est possible d'ajouter des options au widget Label : fg(couleur du texte), bg(couleur du fond), font(police, taille de la police, gras, italique...), height(hauteur), width(largeur)...
   Modifier la 2e ligne du programme en écrivant :

```
texte1=Label(fen,text="Bonjour !",fg="pink",bg="white",font="courier")
```

   Qu'obtient-on ? .....

   c) Compléter votre code en rajoutant un autre label contenant le texte "Saisir un nombre :". Ce label sera placé en-dessous du premier.

3. Création d'un bouton
   a) Compléter le code précédent par les lignes suivantes :

```
bou=Button(fen,text="Quitter",command=fen.destroy)
bou.grid(row=2,column=2)
```

   Que voit-on apparaître ? .....

   Explications :
   L'instruction `bou=Button(fen,text="Quitter",command=fen.destroy)` permet de créer un bouton sur lequel on peut cliquer. Le widget Button prend comme premier argument le nom de la fenêtre dans laquelle on travaille et comme deuxième argument le texte apparaissant sur le bouton. L'argument command permet de préciser quelle action sera effectuée lorsque l'utilisateur clique sur le bouton. Ici, la fenêtre sera fermée. Mais on peut également écrire une fonction pour préciser quelle action sera effectuée.

   b) Compléter le code précédent en ajoutant les lignes en gras suivantes :

```
def clic():
    print("J'ai cliqué sur le bouton !")

fen=Tk()
...
bou2=Button(fen,text="Cliquer ici",command=clic)
bou2.grid(row=0,column=2)
fen.mainloop()
```

   Qu'observe-t-on lorsqu'on clique sur le bouton "Cliquer ici" ? Expliquer.....
   .....

4. Création d'une ligne de saisie
   a) Compléter le code précédent en ajoutant les lignes suivantes :

```
def afficher():
    t=entree.get()
    print("Vous avez saisi : ",t)

fen=Tk()
...
entree=Entry(fen)
entree.grid(row=1,column=1)
bou3=Button(fen,text="Valider",command=afficher)
bou3.grid(row=1,column=2)
fen.mainloop()
```

   Que voit-on apparaître ? .....

   Entrer un texte ou un nombre, puis cliquer sur valider. Que se passe-t-il ? .....
   .....

   Explications :
   La ligne `entree=Entry(fen)` permet de créer une ligne de saisie.
   Lorsqu'on clique sur le bouton valider, la fonction afficher est exécutée. Dans cette fonction, la ligne `t=entree.get()` permet de récupérer le texte saisi et de l'affecter à une variable t. Ce texte est ensuite affiché dans la console.

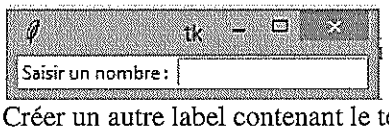
   b) On souhaite centrer le texte "Bonjour !" au-dessus du deuxième label et de la zone de saisie. Pour cela, rajouter l'argument `columnspan` au niveau du placement du texte 1 :
   `texte1.grid(row=0,column=0,columnspan=2)`

   Explications :
   L'argument `columnspan` permet d'indiquer sur combien de colonnes doit s'étendre le widget. Ici, on souhaite qu'il s'étende sur les deux premières colonnes, ainsi on indique `columnspan=2`.

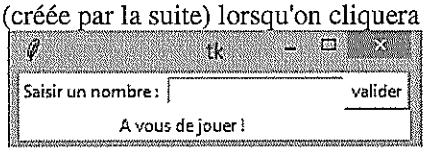
Exercice 3 :
Dans cet exercice, on veut créer une interface graphique pour le jeu du plus ou du moins vu dans l'exercice 3 du TP4 (inutile de reprendre le code car on va être obligé de le modifier en grande partie).
On considèrera ici que le joueur a droit à 5 essais.

Rappel des règles :
- L'ordinateur tire au sort un nombre entier entre 1 et 100.
- Ensuite, il demande à l'utilisateur de choisir un nombre entre 1 et 100.
- Si l'utilisateur a trouvé le même nombre que l'ordinateur, il affiche "vous avez gagné", sinon, il affiche "c'est plus" ou "c'est moins" et demande à l'utilisateur de choisir un nouveau nombre.
- Le jeu continue jusqu'à ce que l'utilisateur ait trouvé la bonne réponse ou qu'il ait fait 5 tentatives.

1. Créer une variable *n* contenant le nombre à deviner (un entier choisi aléatoirement entre 1 et 100).
   Rappel : pour obtenir un entier aléatoire, il faut importer le module `random` et utiliser la fonction `randint(a,b)` qui renvoie un entier aléatoire compris entre *a* et *b*
2. Créer une variable *nbEssais* contenant le nombre d'essais effectués. Initialiser cette variable à 0.
3. Créer un label et une zone de saisie de façon à obtenir la fenêtre suivante :



4. Créer un autre label contenant le texte "A vous de jouer !" et un bouton "valider" qui exécutera la fonction *valider* (créée par la suite) lorsqu'on cliquera dessus.



5. Ecrire la fonction *valider* qui :
   - incrémente le nombre d'essais de 1 et récupère le nombre saisi ;
   - si le nombre saisi est égal au nombre à deviner, modifie le texte du deuxième label en "Vous avez gagné !" et désactive le bouton valider ;
   - sinon si le nombre d'essais atteint 5, modifie le texte du deuxième label en "Vous avez perdu !" et désactive le bouton valider ;
   - sinon si le nombre saisi est supérieur au nombre attendu, modifie le texte du deuxième label en "C'est moins !" ;
   - sinon modifie le texte du deuxième label en "C'est plus !".

   Indications :
   - Le contenu de la zone de saisie est considéré comme une chaîne de caractères. Il faut donc le convertir.
   - Pour modifier le texte d'un label, il faut écrire : `nom_label.configure(text="nouveau texte")`
   - Pour désactiver un bouton, il faut écrire : `nom_bouton.configure(state=DISABLED)`

Exercice 4 :

1. Dans un nouveau programme, taper le code suivant :
```
from tkinter import *
fen=Tk()
can=Canvas(fen,bg="white",width=300,height=200)
can.grid(row=0,column=0)
bou=Button(fen,text="Quitter",command=fen.destroy)
bou.grid(row=1,column=0)
fen.mainloop()
```
   Que se passe-t-il ?

   Explications :
   Un canevas est une surface rectangulaire dans laquelle on peut mettre divers éléments graphiques (dessins, images...) à l'aide de fonctions spécifiques. Les options width et height permettent d'indiquer la largeur et la hauteur du canevas.

2. Compléter le code précédent en ajoutant la ligne :
```
can.grid(row=0,column=0)
dessin=can.create_line(0,0,50,100,fill="green")
bou=Button(fen,text="Quitter",command=fen.destroy)
```

   A quoi sert la fonction create_line ? .....
   A quoi sert l'option fill ? .....
   D'après vous, que représentent les nombres 0,0,50,100 ? .....
   .....
   Où est située l'origine du repère dans un canevas ? .....

3. Remplacer la fonction create_line par la fonction create_rectangle.
   A quoi sert cette fonction ? Que représentent les nombres 0,0,50,100 pour cette fonction ? .....
   .....
4. Dans la fonction create_rectangle, rajouter l'option outline="pink".
   Quelle est la différence entre les options outline et fill ? .....
   .....
5. Remplacer la fonction create_rectangle par la fonction create_oval.
   A quoi sert cette fonction ? Que représentent les nombres 0,0,50,100 pour cette fonction ? .....
   .....

6. a) Rajouter dans votre code la fonction suivante :
```
def bleu() :
    can.itemconfigure(dessin,fill="blue")
```
   b) Créez un bouton permettant d'exécuter cette fonction. Testez.

   Explications :
   La méthode `itemconfigure` permet de modifier n'importe quelle option d'un élément placé dans le canevas. Ici, on modifie la couleur de remplissage (option `fill`) de l'élément nommé dessin.