Projet Informatique,
CPBX MP, TD1

Paul Dorbec

1 Création de fenêtre

Il existe intégré dans python une bibliothèque qui permet de gérer des interfaces graphiques, appelée Tkinter.

On va ici expérimenter cette bibliothèque. Commencez par l'importer avec

```
from tkinter import *
```

Dans tkinter, les éléments graphiques sont des *widgets* qu'on crée avec un appel de fonction, et qu'on affecte dans une variable. Un récapitulatif des appels utilisé est fourni en fin de sujet.

Question 1.1 *Le premier widget est la fenêtre, qu'on crée avec la commande :*

```
fenetre=Tk()
```

Bien sûr, cette commande se charge déjà de stocker le widget dans la variable `fenetre` *(on aurait pu l'appeler autrement).*

Question 1.2 *On peut agir sur les éléments des widgets. On va par exemple donner un nom à la fenêtre avec*

```
fenetre.title("le nom que vous voulez")
```

Appelez par exemple votre fenêtre Mastermind.

Question 1.3 *Enfin, pour ouvrir effectivement la fenêtre, il faut faire un appel à* `mainloop` *avec :*

```
fenetre.mainloop()
```

Cela ouvre une fenêtre vide. On pourra interrompre la boucle avec `fenetre.quit()`*, ou détruire la fenêtre avec* `fenetre.destroy()`*. Selon que vous testez votre programme en l'exécutant dans un terminal (avec* `python3 nom_du.py`*) ou dans un interpréteur (e.g. idle3),* `quit()` *suffit à fermer la fenêtre ou pas.*

2 Quelques widgets d'affichage pur.

Le widget `Label` permet de créer un champ de texte.

Question 2.1 *Ajoutez à votre fenêtre un widget* `Label` *dont le texte est "Bienvenue dans le Mastermind". Il faut créer le widget avec*

```
mon_nom=Label(fenetre, text="blabla")
```

Il faut ensuite appeler `w.pack()` *pour placer le widget dans son conteneur, où* `w` *est la variable contenant votre widget, donc typiquement ici* `fenetre.pack()`*. Vous devriez voir apparaître votre texte.*

Avec `Frame`, on peut définir des cadres pour y placer les éléments.

Question 2.2 *Définissez deux champs* frame*, qui serviront à mettre le plateau de jeu et le menu. Placez (provisoirement) des widgets* `Label` *dans chaque* frame *pour les faire apparaître. Il faut pour cela que vous indiquiez dans vos widget que vous les placez dans la frame et non dans la fenêtre, au niveau du premier champ. N'oubliez pas de faire* `pack` *pour chaque widget. Observez que pour l'instant, ils sont empilés et qu'on ne décèle pas l'existence des* frame *!*

Question 2.3 *On va maintenant juxtaposer horizontalement nos éléments* frame *pour les révéler. Utilisez*

```
f.pack(side=LEFT)
```

où `f` *représente l'une de vos* frame*, même chose en plaçant la deuxième à droite. Normalement, vos* `Label` *se retrouvent côte à côte.*

Question 2.4 *Pour la* frame *de droite, remplacez le widget* `Frame` *par un widget* `LabelFrame`*, auquel vous ajouterez un attribut* `text="Menu"`*. Observez l'effet.*

Question 2.5 *Dans la frame de gauche, on voudrait placer un plateau de mastermind. Remplacez la frame de gauche par un canevas (*`Canvas`*) de largeur (width) 250 et de hauteur (height) 546. Ce sont les dimensions de l'image que nous placerons.*

Question 2.6 *Récupérez le fichier* `plateau.png`*. La méthode pour ouvrir l'image est différente dans tkinter que pour PIL. Utilisez*

```
img=PhotoImage(file="plateau.png")
```

puis insérez l'image dans le canevas avec

```
votre_canevas.create_image(0,0,anchor=NW,image=img)
```

où vous remplacez `votre_canevas` *par le nom de votre canevas. Testez un peu ce qui se passe si vous modifiez les paramètres* `0,0` *et* `anchor` *pour être sûr(e) de comprendre ce à quoi ça correspond.*

3 Des interactions

Nous allons maintenant utiliser des outils pour interagir avec l'utilisateur.

Question 3.1 *Dans la frame "Menu", nous allons d'abord ajouter un bouton "quitter". Le widget* `Button` *peut s'appeler avec des attributs* `text` *pour le texte du bouton et* `command` *pour l'action du bouton. Ce dernier attribut est directement le nom d'une fonction (qui s'appelle sans arguments). Ici, nous allons utiliser la commande* `fenetre.destroy` *qui ferme la fenêtre. Ajoutez le bouton et placez le à droite.*

Question 3.2 *Ajoutez un autre bouton à gauche du bouton quitter, que vous intitulerez "solution". On ne s'occupe pas de son action pour l'instant (en n'utilisant pas d'attribut* `command`*).*

Ces boutons seront placés en bas du menu, on va ajouter des éléments au dessus.

Question 3.3 *Tout d'abord, dans le menu, on va inviter à renseigner son nom. Définissez un widget* `Texte` *pour demander le nom, puis ajoutez un widget* `Entry` *pour que l'utilisateur puisse le renseigner. On ne se préoccupe pas pour l'instant de récupérer l'information dans une variable.*

Question 3.4 *Pour récupérer l'information saisie dans le champ, il faut créer votre première variable tkinter. On va utiliser une variable chaîne de caractères. Déclarez la variable avec*

```
nom=StringVar()
```

ela crée un conteneur de variable de type `string`*, que tkinter pourra affecter en fonction de ce que l'utilisateur fait. Vous pouvez alors ajouter dans le widget l'attribut* `variable=nom` *qui vous permettra de récupérer le nom.*

Question 3.5 *On va maintenant utiliser un système de puces radio pour sélectionner la couleur. Utilisez le widget* `RadioButton` *pour créer un bouton par couleur. Créez une variable de type* `string` *et indiquez la variable utilisée et la valeur correspondant à la puce avec un attribut comme par exemple* `value="green"`*.*

Question 3.6 *Alignez les puces radio, introduisez une couleur de fond mieux assorties, et essayez de donner une finition plus propre à l'ensemble. Nous attaquerons la gestion des clics souris la semaine prochaine.*

4 Rappel des fonctions utilisées

Vous pouvez trouver une liste assez complète de widget et de leurs options sur `https://www.tutorialspoint.com/python/python_gui_programming.htm`

Voici les plus utiles

`F=Tk()`	Ouvre une fenêtre et la stocke dans la variable `F`.
`F.title(nom)`	Donne un titre à une fenêtre.
`F.mainloop()`	Ouvre la fenêtre en lançant la boucle principale.
`F.quit()`	Interrompt la boucle principale.
`F.destroy()`	Ferme la fenêtre.
`L=Label(fenetre,text="bla")`	Ouvre un widget label, pour afficher du texte. La variable `fenetre` désigne le widget dans lequel se placera l'objet (typiquement une fenêtre ou une frame)
`Frame(fenetre)`	Définit un widget cadre (conteneur).
`LabelFrame(fenetre,text="bla")`	Un cadre avec un titre.
`Canvas(fenetre,...)`	Un widget pour afficher une image, que l'on peut modifier et où l'on peut tracer des éléments.
`L.pack()`	permet de placer le widget dans la fenêtre.
`L.pack(side=LEFT)`	On peut ajouter l'option `side` suivi d'une position (LEFT,RIGHT,TOP,BOTTOM)
`L.pack(padx=5)`	On peut ajouter les options `padx` et `pady` pour espacer l'objet.
`L.pack(fill=X)`	on peut indiquer de remplir le conteneur en largeur (X), en hauteur (Y) ou les deux (BOTH).
`Button(fenetre,...)`	Un bouton que l'on peut cliquer. Les attributs `text` pour le message du bouton et `command` pour l'action sont utiles.
`Checkbutton(fenetre,... )`	Une case à cocher. Le résultat (0 ou 1) est stocké dans la variable désignée par `variable=`...
`RadioButton(fenetre,...)`	Des puces à cocher, une seule puce peut être cochée à la fois. En plus du champ `variable`, un champ `value` permet de désigner ce que la variable contiendra.
`Entry(fenetre,...)`	Un champ à remplir. Il dispose d'un attribut `textvariable` de type `StringVar` pour stocker le contenu du champ.

Quelques attributs	

`L=Label(f,text="bla")`
`L.config(text="bla")` ou
`L["text"]="bla"`
	On peut définir les attributs ainsi à la création
	On peut aussi modifier les attributs après création avec l'une des deux méthodes.

`height` et `width`	Hauteur et largeur en pixels ou en nb de lettres pour les champs de texte.
`anchor`	Ancrage de l'élément, parmi N,S,E,W,NE,NW,SE,SW, et CENTER
`relief`	type de relief, parmi FLAT, RAISED, SUNKEN, GROOVE ou RIDGE
`fg` ou `foreground`	couleur du texte, recoit une couleur comme `"blue"` ou `"#RRGGBB"`
`bg` ou `background`	idem, couleur du fond.
`highlightbackground`, `highlightcolor`	les couleurs du cadre des éléments
`activebackground`, `activeforeground`	les couleurs quand la souris est placée sur l'élément.

Les actions sur `Canvas`	
`c.create_image(x,y,image=i)`	Insertion d'une image (chargée avec PhotoImage) dans le canevas. L'ancrage (par défaut `anchor=CENTER`) est placé en position (x,y).