Créer une interface graphique avec Tkinter

Présentation

Python intègre par defaut le **module Tkinter** permettant de créer des interfaces graphiques en offrant une passerelle entre Python et la bibliothèque Tk.

Cela permet de créer des fenêtres, des boutons, et de faire réagir vos objets graphiques à certains événements.

Création d'une première fenêtre qui affiche « Bonjour »

```
# -*-coding:Latin-1 -*
 # fenetre1.py
from tkinter import *
# Création d'une fenêtre racine de l'interface
fenetre = Tk()

# Création d'un label (ligne de texte) souhaitant la bienvenue
# Note : le premier paramètre passé au constructeur de Label est l'interface racine
champ_label = Label(fenetre, text="Bonjour")

# Affichage du label dans la fenêtre
champ_label.pack()

# On démarre la boucle Tkinter qui s'interompt quand on ferme la fenêtre
fenetre.mainloop()
```

La méthode **pack** du Label permet de positionner et donc d'afficher l'objet dans la fenêtre.

La méthode **mainloop** de notre fenêtre racine ne se termine que lorsqu'on ferme la fenêtre.

Les objets graphiques (boutons, champs de texte, cases à cocher, barres de progression...) sont appelés des **widgets**.

Pour accéder à la valeur actuelle de l'option du widget ou pour modifier les options du widget après sa création, on le précise entre crochets.

Exemple : changer « Bonjour « par « Au revoir » :

```
>>> champ_label["text"] = "Au revoir"
```

Ajouter d'autres widgets

Démarche :

- créer le widget en précisant ses options ;
- il faut que premier paramètre du constructeur soit la fenêtre principale ;
- faire appel à sa méthode pack qui permet de positionner un objet dans une fenêtre ou dans un cadre.

Un bouton

```
bouton_quitter = Button(fenetre, text="Quitter", command=fenetre.destroy)
bouton_quitter.pack()
```

Le dernier paramètre passé au **constructeur** de **Button** est l'action liée à un clic sur le bouton. Ici, c'est la **méthode destroy** de la fenêtre racine qui est appelée.

Une ligne de saisie

Ce widget est une zone de texte (ligne simple) dans lequel l'utilisateur peut écrire.

Il est préférable de créer une variable associée au champ de texte. Cette variable est un objet. Il faudra utiliser ses méthodes **set()** pour définir sa valeur et **get()** pour lire sa valeur.

```
var_texte = StringVar()
var_texte.set("Saisissez une adresse MAC")
ligne_texte = Entry(fenetre, textvariable=var_texte, width=30)
ligne_texte.pack()
```

La variable var*texte va contenir le texte affiché ou saisi dans le **widget Entry**. Pour lire l'information saisie dans le **widget Entry** : <code> showinfo(“Alerte”, ligne*texte.get()) </code>

Comme le **widget Entry** n'est qu'une zone de saisie, il faut utiliser un **widget Label** ou un **widget Text** (champ de texte à plusieurs lignes) pour indiquer à l'utilisateur ce qu'il doit écrire.

Pour avoir des précisions sur la création des widgets cases à cocher, boutons radio, listes déroulantes, consultez la page http://www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-python/de-nombreux-widgets

Organiser ses widgets dans la fenêtre

Il existe plusieurs widgets qui peuvent contenir d'autres widgets comme le **widget Frame**. C'est un cadre rectangulaire dans lequel vous pouvez placer vos widgets... ainsi que d'autres objets Frame si besoin est.

Pour faire apparaître un widget dans un cadre, utilisez le Frame comme parent à la création du widget :

```
cadre = Frame(fenetre, width=768, height=576, borderwidth=1)
cadre.pack(fill=BOTH)
message = Label(cadre, text="Notre fenêtre")
message.pack(side="top", fill=X)
```

Les arguments de la méthode pack :

- side=“top” pour placer le widget en haut de son parent (ici, le cadre).
- fill=BOTH pour que le widget remplisse le widget parent, soit en largeur si la valeur est X, soit en hauteur si la valeur est Y, soit en largeur et hauteur si la valeur est BOTH.

Les commandes

Pour fermer la fenêtre quand on clique sur un bouton, on précise en dernier argument, l'argument command avec comme valeur la méthode quit de la fenêtre.

Sur ce principe, vous pouvez créer simplement des commandes personnalisées, en écrivant des méthodes. Petite dificulté : la méthode à créer ne peut prendre aucun paramètre. Pour qu'un clic sur le bouton modifie le bouton lui-même ou un autre objet, il faut placer les widgets dans un corps de classe.

Cette classe va hériter de Frame, ce qui signifie que cette classe sera un widget elle aussi.

```
# -*-coding:Latin-1 -*
from tkinter import *
class Interface(Frame):
    """fenêtre principale ; les widgets sont stockés comme attributs de cette fenêtre."""

    def __init__(self, fenetre, **kwargs):
        Frame.__init__(self, fenetre, width=768, height=576, **kwargs)
        self.pack(fill=BOTH)
        self.nb_clic = 0

        # Création des widgets
        self.message = Label(self, text="Vous n'avez pas cliqué sur le bouton.")
        self.message.pack()

        self.bouton_quitter = Button(self, text="Quitter", command=self.quit)
        self.bouton_quitter.pack(side="left")

        self.bouton_cliquer = Button(self, text="Cliquez ici", fg="red",
                command=self.cliquer)
        self.bouton_cliquer.pack(side="right")

    def cliquer(self):
```

```
        """clic sur le bouton donc changer la valeur du label message."""
        self.nb_clic += 1
        self.message["text"] = "Nb de clic : {0}".format(self.nb_clic)

# création et affichage de l'interface
fenetre = Tk()
interface = Interface(fenetre)

interface.mainloop()
interface.destroy()
```

Quelques précisions :

Création d'une classe qui contiendra toute la fenêtre et qui hérite de Frame, c'est-à-dire d'un cadre Tkinter. Dans le constructeur de la fenêtre, on appelle le constructeur du cadre et on pack (positionne et affiche) le cadre.

Dans le constructeur, on crée les différents widgets de la fenêtre (Position et les affichage).

Création d'une méthode bouton*cliquer, appelée quand on clique sur le bouton*cliquer. Elle ne prend aucun paramètre. Elle va mettre à jour le texte contenu dans le labelself.message, comptabiliser le nombre de clic et afficher ce nombre de clics.

On crée la fenêtre Tk qui est l'objet parent de l'interface que l'on instancie ensuite.

On rentre dans la boucle mainloop. Elle s'interrompra quand on fermera la fenêtre.

Ensuite, on détruit la fenêtre grâce à la méthode destroy.

Programme WoL à créer

Créer un programme Python avec interface graphique qui permet de saisir une adresse MAC pour ensuite réveiller un ordinateur éteint.

Des liens pour vous aider

http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel