TP noté	La bibliothèque Tkinter de Python	ISN

Le module Tkinter permet de créer des interfaces graphiques (GUI : graphical user interface).

Créer une fenêtre « maître »

Ecrire ce script sous IDLE , l'enregistrer sous le nom « fenetre.py » puis l'exécuter .

```
from tkinter import *                    # appel du module Tkinter
fenetre = Tk()                           #création de la fenêtre principale nommée fenetre
fenetre.title('ma première fenêtre')     # affichage du titre
fenetre.geometry('400x300')              # définition des dimensions de la fenêtre (400x300)
fenetre.mainloop()                       # lance le gestionnaire d'évènements
```

Ce qu'il faut retenir :

fenetre=**Tk()** crée la fenêtre qui s'appellera fenetre.

fenetre.**mainloop()** : c'est cette ligne qui provoque le démarrage du réceptionnaire d'événements associé à la fenêtre. Cette instruction est nécessaire pour que l'application soit « à l'affût » des clics de souris, des pressions exercées sur les touches du clavier, etc. C'est donc cette instruction qui la met en marche.

Remarque : *la fenêtre s'ajuste automatiquement au contenu si on ne précise pas ses dimensions.*

Créer une fenêtre « maître » et des widgets « esclaves »

Widget : contraction de window gadget

Ecrire ce script sous IDLE en complétant les lignes de commentaires, l'enregistrer sous le nom « fenetre2.py » puis l'exécuter .

```
from tkinter import*
fen1=Tk()
# ……. :
tex1=Label(fen1, text='Bonjour tout le monde !', fg='red')
tex1.pack(padx=10,pady=10)
# ……. :
bou1=Button(fen1, text='Quitter', command=fen1.destroy)
bou1.pack(side=RIGHT)
fen1.mainloop()
```

Appeler le professeur pour vérification

A retenir sur les widgets

a. **La classe « Label » :** *Label* permet un affichage simple de texte.

→ Paramètres du widget *Label*

Paramètres	Effet
Text	Précise le texte à afficher
fg	Précise la couleur du texte
bg	Précise la couleur de fond
heigt	Précise la hauteur du label
width	Précise la largeur du label

→ Modification du widget *Label*

Une fois le widget Label créé et affiché, il est possible de changer le texte à afficher à l'aide de la commande **Lab.config()** *(Vous en verrez un exemple dans l'exercice du lancer de dé ci-dessous).*

b. La classe Button :

Button définit un bouton sur lequel on peut cliquer et qui déclenche des actions ou commandes.

On retrouve les mêmes paramètres que pour *Label* avec en plus le paramètre **command qui permet de préciser la fonction à lancer lors d'un clic sur le bouton.**

Dans l'exemple précédent, la commande **.destroy permet de fermer la fenêtre** lorsque le bouton est cliqué.

Attention : Le paramètre command attend un nom de fonction mais sans parenthèses.

Encore d'autres widgets à explorer plus tard si besoin : Checkbutton (case à cocher qui prend deux états), **Menubutton** (menu déroulant), **Radiobutton** (ensemble de cases à cocher exclusives) ou encore **Frame** (des cadres dans lequel placer d'autres widgets).

Si ça vous intéresse, allez (plus tard) au lien suivant : http://fsincere.free.fr/isn/python/cours_python_tkinter.php

Gestion de l'espace dans la fenêtre

*Voir Lignes tex1.**pack()** ou bou1.**pack()** dans notre dernier exemple*

La méthode **pack()** fait partie d'un ensemble de méthodes qui sont applicables à la plupart des *widgets Tkinter*, et qui agissent sur leur disposition géométrique dans la fenêtre : la fenêtre maître est réduite automatiquement pour qu'elle soit juste assez grande pour contenir les widgets esclaves.

Options possibles pour la méthode pack():

- l'option **side** peut accepter les valeurs TOP, BOTTOM, LEFT ou RIGHT, pour « pousser » le widget du côté correspondant dans la fenêtre.
- les options **padx** et **pady** permettent de réserver un petit 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*.

D'autres méthodes à voir si besoin…

La méthode grid() *: La fenêtre est découpée en un quadrillage virtuel.*

La méthode place() : *On place un repère sur la fenêtre (origine en haut à gauche)*

Attention ! Dans un même script, on utilise une seule de ces méthodes à la fois.

Exercices :

1. Créer un programme « bonjour.py »

qui donne l'affichage suivant :

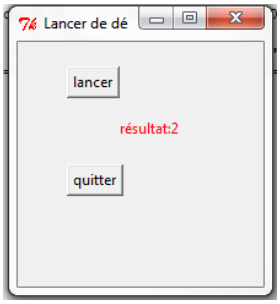
Lorsqu'on clique sur le bouton « Quitter » , l'application s'arrête

Appeler le professeur pour vérification

2. Lancer de dé : compléter le programme suivant

Programme	*Commentaires + travail à faire :*
from tkinter import * ***from random import ****	Importation des bibliothèques tkinter et random
……	**Compléter les 3 lignes pour créer une fenêtre « fen » , de dimensions « 200x200 » avec le titre « lancer de dé »**
……	
……	
def alea(): ***nb=randint(1,6)*** ***label.config(text='résultat:'+str(nb))***	nb est un nombre aléatoire entre 1 et 6 cette commande permet d'ajouter à label le texte « résultat » suivi du nombre aléatoire nb obtenu
……	**Créer un bouton « bou1 » qui commande la fonction alea()**
……	
……	**Créer un bouton « bou2 » qui permet de quitter le programme**
……	
……	**Créer le Label « label » utilisé dans alea()**
……	
fen.mainloop()	

Vous devez obtenir une fenêtre de ce type :



Appeler le professeur pour vérification

La commande Entry

Entry permet de saisir un texte. Il faut donc prévoir une variable permettant de récupérer le texte saisi :

→ ***Var=stringVar()*** permet de définir une variable qui recevra une chaîne de caractère.

→ ***Var=IntVar()*** permet de définir une variable qui recevra un entier.

Le widget *Entry* possède les même options que les autres widgets sauf text ou command.

→ **Entree.get()** permet de récupérer le texte entré par l'utilisateur

→ **Entree.delete(*i*)** permet d'effacer le contenu du champ de texte à la position *i*

→ **Entree.delete(*deb,fin*)** permet d'effacer le contenu du champ de texte entre les indices *deb* et *fin*

→ **Entree.focus()** permet d'obliger le curseur à se placer sur l'Entry

Exemple : Ecrire ce script sous IDLE en complétant les lignes de commentaires , l'enregistrer sous le nom « fenetre3.py » puis l'exécuter :

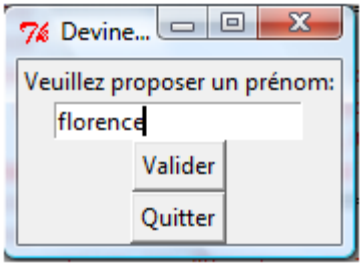
```
from tkinter import *
fen=Tk()
# ..... :
texte=Label(fen, text='Cliquer et saisir:', width=20, height=3, fg="black")
texte.pack()
# ..... :
saisie=StringVar()
# ..... :
entree=Entry(fen,textvariable=saisie, width=30)
entree.pack()
# ..... :
bou=Button(fen , text='Valider' , command=fen.destroy)
bou.pack()
fen.mainloop()
# ..... :
print (saisie.get())
```

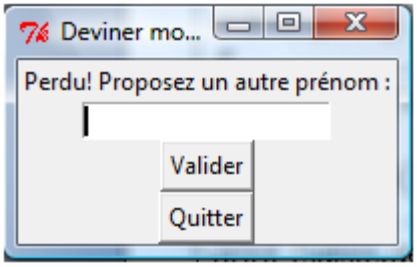
Appeler le professeur pour vérification

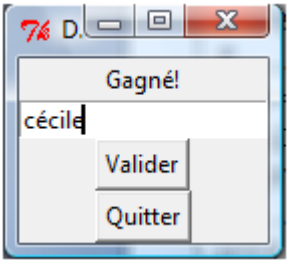
Exercice : Ecrire le programme **prenom.py** qui vous demande d'entrer un prénom (en minuscule) et affiche gagné si c'est le vôtre , perdu sinon. Le contenu doit s'effacer pour pouvoir rejouer .

Pour cela, créer la fonction **verif()** qui sera exécutée dès que le bouton « valider » est pressé .

Cette fonction récupère le contenu de la saisie et le compare avec votre prénom.







Appeler le professeur pour vérification