FICHE PYTHON 7 : COMPLEMENT TKINTER (4 pages)

WIDGET Scale : Curseur pour récupérer une valeur numérique donnée par l'utilisateur

Création d'un objet-**Scale**	`curseur1 = Scale(fen_princ, from_=-20, to=300)` `curseur1.pack()`

Création d'un objet-**Scale** associé à un IntVar	`valeur2 = IntVar()`
	`curseur2 = Scale(fen_princ, from_=-20, to=300, variable=valeur2)`
	`curseur2.pack()`

Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattaché
Attribut **from** Attribut **to**	Cet attribut permet de définir la valeur minimale. Cet attribut permet de définir la valeur maximale.
Attribut **variable**	Permet d'associer la valeur du curseur à une variable.
Attribut **orient**	Permet de changer la direction par défaut : `orient = HORIZONTAL`

Lire le nombre sélectionné	Il faut utiliser la méthode **get**() soit sur le **IntVar**, soit sur l'**Entry** :
	`print(curseur2.get())`
	`print(valeur2.get())`

WIDGET Spinbox : Pour récupérer une valeur numérique tapée ou avant bouton

Création d'un objet-**Spinbox**	`s = Spinbox(fen_princ, from_=0, to=10)` `s.pack()`
Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattaché
Attribut **from_** Attribut **to**	Cet attribut permet de définir la valeur minimale. Cet attribut permet de définir la valeur maximale.

WIDGET Listbox : Pour récupérer un choix dans une liste prédéfinie

Création d'un objet-**Listbox**	`liste = Listbox(fen_princ)`
	`liste.insert(1, "Python")`
	`liste.insert(2, "C++")`
	`liste.insert(3, "HTML")`
	`liste.insert(4, "CSS")`
	`liste.insert(5, "Javascript")`
	`liste.pack()`

Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattach
Pour définir les choix	On peut faire comme dans l'exemple, utilisée la méthode **insert** des objets **Listbox**. Ou on peut utiliser un String avec la méthode **set** des objets **ListBox**: `liste.set('un deux trois')`
Pour lire le choix	`liste.get('active')` Cette commande permet de récupérer le choix de l'utilisateur.

Cas A : FILEDIALOG : Fenêtre de sélection des fichiers

Il faut importer les classes du module **filedialog** non importées directement avec tkinter.
Voilà en exemple un code qui crée un bouton sur lequel on peut cliquer pour récupérer le chemin d'un fichier dans une variable noté **filepath**.
On utilise **askopenfilename.**

```
# -*-coding:Utf-8 -*
from tkinter import *
from tkinter.filedialog import * #pour les gestions de fichiers

# --------------------------
# Définitions des fonctions utilisées
# --------------------------
def mise_a_jour():
      filepath = askopenfilename(title="Ouvrir une image",
filetypes=[ ('jpg files','.jpg') , ('all files','.*')])

# --------------------------
# Création de la fenêtre et des objets associés la fenêtre
# --------------------------
fen_princ = Tk()
# Création d'un Button lancant la fonction mise_a_jour
monBouton = Button(fen_princ, text="BOUTON 1", command=mise_a_jour)
monBouton.pack()
...
```

Ici le nom du fichier est dans une variable interne `filepath` de `mise_a_jour()`.
A vous d'en faire quelque chose d'utile. Par exemple :

Cas B : Afficher une image après l'avoir sélectionnée dans un répertoire

```
from tkinter import *
from tkinter.filedialog import * #pour les gestions de fichiers
from PIL import Image as Img
from PIL import ImageTk

filepath = askopenfilename(title="Ouvrir une image", filetypes=[ ('jpg files','.jpg') ,
('all files','.*')])

presentation = Img.open(filepath)
presentationTk = ImageTk.PhotoImage(presentation)
labelphoto = Label(fen_princ, image=presentationTk)
labelphoto.pack()
. . .
```

Cas A+B : Afficher une image après l'avoir sélectionnée dans une fonction

```
# -*-coding:Utf-8 -*
from tkinter import *
from tkinter.filedialog import * #pour les gestions de fichiers
from PIL import Image as Img
from PIL import ImageTk


# --------------------------
# Définitions des fonctions utilisées
# --------------------------
def mise_a_jour():
    # Recherche de l'adresse du fichier-image voulu
    filepath = askopenfilename(title="Ouvrir une image", filetypes=[
        ('jpg files','.jpg') , ('all files','.*')])
    # Mise à jour de monFichier
    monFichier.set(filepath)
    # Modification du Label image associé à l'adresse précédente
    presentationLocale = Img.open(filepath)
    presentationTkLocale = ImageTk.PhotoImage(presentationLocale)
    labelphoto.monEmplacement = presentationTkLocale
    labelphoto.config(image=presentationTkLocale)

# --------------------------
# Création de la fenêtre et des objets associés la fenêtre
# --------------------------
fen_princ = Tk()

monFichier = StringVar()
monFichier.set("Pas de fichier pour l'instant")

# Création d'un Label nommé monAffichage
monAffichage = Label(fen_princ, textvariable = monFichier, width=70)
monAffichage.pack()

# Création d'un Button lancant la fonction mise_a_jour
monBouton = Button(fen_princ, text="BOUTON 1", command=mise_a_jour)
monBouton.pack()

# Création d'un Label image associé à l'adresse précédente
presentation = Img.new("RGB", (20,20), (255,255,150))
presentationTk = ImageTk.PhotoImage(presentation)
labelphoto = Label(fen_princ, image=presentationTk)
labelphoto.pack()


# - - - - - - - - - - - - - - - - - - - -
# Bouclage de la fenêtre fen_princ
# - - - - - - - - - - - - - - - - - - - -
fen_princ.mainloop()
```

WIDGET Label Text : Complément

Largeur en nombre de caractères **width**	`monAffichage = Label(fen_princ, . . ., width=35)`
Hauteur en nombre de lignes **height**	`monAffichage = Label(fen_princ, . . ., height=3)`
Anchor pour gérer comment le texte va être affiché (centré, sur la gauche …)	La partie gauche : **anchor=W** (pour West, Ouest). La partie droite : **anchor=E** (pour East, Est) La partie centrale : **anchor=CENTER** (c'est la valeur par défaut si vous ne sélectionnez rien, voir au dessus). La partie haute : **anchor=N** (pour North, Nord) La partie basse : **anchor=S** (pour South, Sud).
Wraplength Pour fournir le nombre de pixels au bout desquels on peut passer à la ligne	`monAffichage = Label(fen_princ, . . ., wraplength = 800)`
Attribut **variable**	Permet d'associer la valeur du curseur à une variable.
Attribut **orient**	Permet de changer la direction par défaut : `orient = HORIZONTAL`

Lire le nombre sélectionné	Il faut utiliser la méthode **get**() soit sur le IntVar, soit sur l'Entry :
	`print(curseur2.get())`
	`print(valeur2.get())`

Gestion de la largeur et la hauteur des widgets

Largeur **width**	`monBouton = Button(fen_princ, . . ., width=35)`
Hauteur **height**	`monBouton = Button(fen_princ, . . ., height=3)`