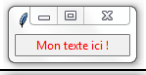
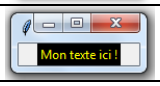
FICHE PYTHON 6 : INTERFACE GRAPHIQUE (3 pages)

Bibliothèque à importer pour créer des applications graphiques	`from tkinter import *` On importe toutes les classes.

Création d'une fenêtre	`fen_princ = Tk()`
	`# Insérer ici vos différents widgets`
	`fen_princ.mainloop()`

Afficher un widget rattaché à une fenêtre	`monWidget.pack()` Il existe d'autres façons d'afficher les widgets.

WIDGET Label : Pour afficher un texte

Création d'un objet-Label	`monAffichage = Label(fen_princ, text="Mon texte ici !")` `monAffichage.pack()`	
Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattaché	
Attribut **text**	`text="Mon texte ici !"` : On donne le contenu du Label.	
Attribut **foreground fg**	`fg="red"` : Couleur du texte	
Attribut **background bg**	`bg="black"` : Couleur du fond	
Attribut **font**	`font=("Helvetica", 32)` : Police taille 32 en Helvetica	
Attribut **height**	`height = 3` : Le Label s'étend sur 3 lignes	
Attribut **width**	`width = 15` : permet de définir la largeur du Label	
Attribut **anchor**	`anchor= W` : permet d'ancrer le texte à la bordure gauche(West) Les autres possibilités : CENTER, N, S, W, E, NE, NW, SEW, SW	
Attribut **wraplength**	`wraplength=800` : permet de fixer la longueur (en mesure-écran) à partir de laquelle on passe en mode multiligne.	

Attribut **textvariable**
Permet de donner le texte à afficher dans un objet-String particulier, un string variable, de classe StringVar.
	`monTexte = StringVar()` `# Création d'un objet StringVar`
	`monTexte.set("Hello World !")` `# Affectation via méthode set`
	`monAffichage = Label(fen_princ, textvariable=monTexte)`
	`monAffichage.pack()` `# Affichage du widget monAffichage`
	Pour modifier le contenu du Label : `monTexte.set("Bonjour le Monde !")`
	Attention, on utilise set avec StringVar et pas l'affectation des Strings avec =.
	Pour lire le contenu de monTexte, deux techniques :
	`lecture = monTexte.get()`
	`lecture = monAffichage.get()`	

Méthode cget() pour lire le contenu des attributs

Exemple 1 : Lecture de la couleur de fond	`monAffichage.cget('fg')`
Exemple 2 : Lecture du texte du Label Text	`monAffichage.cget('text')`

Méthode config ou configure pour modifier le contenu des attributs

Exemple 1 : Modification de la couleur de fond	`monAffichage.config(fg='red')`
Exemple 2 : Modification du texte du Label Text	`monAffichage.config(text='nouveau texte à afficher')`

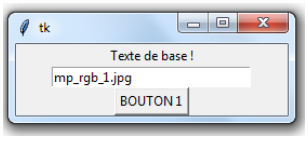
Remarque : Si les modifications sont courantes et si elles peuvent affecter plusieurs widgets, autant utiliser un StringVar.

WIDGET Button : Pour activer le code d'une fonction

Création d'un objet-Button	`monBouton = Button(fen_princ, text="Appuyez !", command=mise_a_jour)` `monBouton.pack()`
Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattaché
Attribut **text**	Cet attribut permet de définir le texte qui s'affichera : `text="Appuyer ici !"`
Attribut **command**	Lorsqu'on cliquera sur le bouton, cela enclenchera la fonction mise_a_jour `command=mise_a_jour  # Attention, on ne place pas les parenthèses()`

*Exemple de code intégrant un **Button** qui change le texte d'un Label ayant un texte variable :*	`def mise_a_jour():`
	`    monTexte.set("Et voilà !")`
	
	`fen_princ = Tk()`
	`monTexte = StringVar()`
	`monTexte.set("Hello World !")`
	`monAffichage = Label(fen_princ, textvariable=monTexte)`
	`monAffichage.pack()`
	`monBouton = Button(fen_princ, text="Changer !", command=mise_a_jour)`
	`monBouton.pack()`
	`fen_princ.mainloop()`

WIDGET Entry : Pour récupérer du texte tapé par l'utilisateur

Création d'un objet-**Entry** associé à un StringVar
 
	`nom_fichier = StringVar()`
	`nom_fichier.set( "mp_rgb_1.jpg" )`
	`entree_fichier = Entry(fen_princ, textvariable=nom_fichier, width=30)`
	`entree_fichier.pack()`

Première indication	`fen_princ` : Désigne la fenêtre à laquelle le widget est rattaché
Attribut **textvariable**	Cet attribut permet de définir le nom de l'objet StringVar associé à cet Entry.
Attribut **width**	`width = 30` : permet de définir la largeur du Entry, comme pour Label.

Lire le texte rentré	Il faut utiliser la méthode get() soit sur le StringVar, soit sur l'Entry :
	`print(nom_fichier.get())nom_fichier.set( "mp_rgb_1.jpg" )`
	`print(entree_fichier.get())`

WIDGET Label : Pour afficher une image

Classe **Image** et Classe **Tk** et Conversion à l'aide de la méthode **PhotoImage** de la classe ImageTk.	La classe **Image** de base de la bibliothèque PIL n'est pas interprétable par la bibliothèque Tk. Une fois l'image créée et modifiée via **Image** (la classe de l'obket-image du module PIL), on peut la transformer en **ImageTk** , la classe-image de Tk, qui sera alors utilisable dans Tkinter. Il faut utiliser la méthode **PhotoImage**.
	`from PIL import Image as Img`
	`from PIL import ImageTk`
	`presentation = Img.new("RGB", (20,20), (255,255,150))`
	`presentationTk = ImageTk.PhotoImage(presentation)`

Création d'un objet-Label	`monAffichage = Label(fen_princ, image=presentationTk)`
	`monAffichage.pack()`
	
	Attention : le nom de l'image doit correspondre à une image convertie avec `ImageTk.PhotoImage` () et pas une Image PIL directement.

Modifier l'image qui s'affiche avec une image préexistante dans le corps du programme	Si vous voulez afficher une image depuis un objet-image Tk pré-existant, c'est facile : `monAffichage.configure(image = presentationTk_nouvelle)` Bien entendu, `presentationTk_nouvelle` devra être le nom de la nouvelle image Tk.
Modifier l'image qui s'affiche avec une image créée dans une fonction liée à un bouton par exemple	C'est plus compliqué : il faudra attendre les activités suivantes.

Suppression de la console

Pour supprimer l'ouverture de la console lors de l'exécution d'un programme utilisant tkinter, il suffit de modifier l'extension du fichier **.py** en **.pyw**.

Méthode GEOMETRY

Impose initiale de la fenêtre lors du lancement	Pour créer une fenêtre de 800 pixels de long sur 600 pixels de haut : `fen_princ.geometry("800x600")`

Méthode PLACE

Permet le placement précis des WIDGETS	Pour placer un Widget à une position de 50 pixels en x et 100 pixels en y : `monWidget.place( x = 50, y = 100 )` ATTENTION : Il ne faut pas mélanger les méthodes de placements dans un même conteneur.