PYTHON PAR LA PRATIQUE

Présenté par :

Xavier TABUTEAU

16. Tkinter

Interface graphique avec Python

Le domaine des interfaces graphiques (ou GUI, Graphical User Interfaces) est extrêmement complexe. Chaque système d'exploitation peut en effet proposer plusieurs « bibliothèques » de fonctions graphiques de base, auxquelles viennent fréquemment s'ajouter de nombreux compléments, plus ou moins spécifiques de langages de programmation particuliers. Tous ces composants sont généralement présentés comme des classes d'objets, dont il vous faudra étudier les attributs et les méthodes. Avec Python, la bibliothèque graphique la plus utilisée jusqu'à présent est la bibliothèque Tkinter, qui est une adaptation de la bibliothèque Tk développée à l'origine pour le langage Tcl. Tkinter est un environnement graphique de type événementiel fourni par défaut avec python. Cela permet à l'utilisateur de saisir des informations de façon non séquentielle.

Plusieurs autres bibliothèques graphiques fort intéressantes ont été proposées pour Python : wxPython, pyQT, pyGTK, etc. Il existe également des possibilités d'utiliser les bibliothèques de widgets Java et les MFC de Windows.

Il est aisé de construire différents modules Python, qui contiendront des scripts, des définitions de fonctions, des classes d'objets, etc... On peut alors importer tout ou partie de ces modules dans n'importe quel programme, ou même dans l'interpréteur fonctionnant en mode interactif. Pour utiliser Tkinter donc, il suffit de l'importer dans votre fichier python.

16. Tkinter

Les 15 classes principales (widget) du module Tkinter

Button : Un bouton classique, à utiliser pour provoquer l'exécution d'une commande quelconque.

Canvas : Un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner, créer des éditeurs graphiques, et aussi pour implémenter des widgets personnalisés.

Checkbutton : Une « case à cocher » qui peut prendre deux états distincts (la case est cochée ou non). Un clic sur ce widget provoque le changement d'état.

Entry : Un champ d'entrée, dans lequel l'utilisateur du programme pourra insérer un texte quelconque à partir du clavier.

Frame : Un cadre rectangulaire dans la fenêtre, où l'on peut disposer d'autres widgets. Cette surface peut être colorée. Elle peut aussi être décorée d'une bordure.

Label : Un texte (ou libellé) quelconque (éventuellement une image).

Listbox : Une liste de choix proposés à l'utilisateur, généralement présentés dans une sorte de boîte. On peut également configurer la Listbox de telle manière qu'elle se comporte comme une série de « boutons radio » ou de cases à cocher.

16. Tkinter

Menu : Un menu. Ce peut être un menu déroulant attaché à la barre de titre, ou bien un menu « pop up » apparaissant n'importe où à la suite d'un clic.

Menubutton : Un bouton-menu, à utiliser pour implémenter des menus déroulants.

Message : Permet d'afficher un texte. Ce widget est une variante du widget Label, qui permet d'adapter automatiquement le texte affiché à une certaine taille ou à un certain rapport largeur/hauteur.

Radiobutton : Représente (par un point noir dans un petit cercle) une des valeurs d'une variable qui peut en posséder plusieurs. Cliquer sur un « bouton radio » donne la valeur correspondante à la variable, et « vide » tous les autres boutons radio associés à la même variable.

Scale : Vous permet de faire varier de manière très visuelle la valeur d'une variable, en déplaçant un curseur le long d'une règle.

Scrollbar : « ascenseur » ou « barre de défilement » que vous pouvez utiliser en association avec les autres widgets : Canvas, Entry, Listbox, Text.

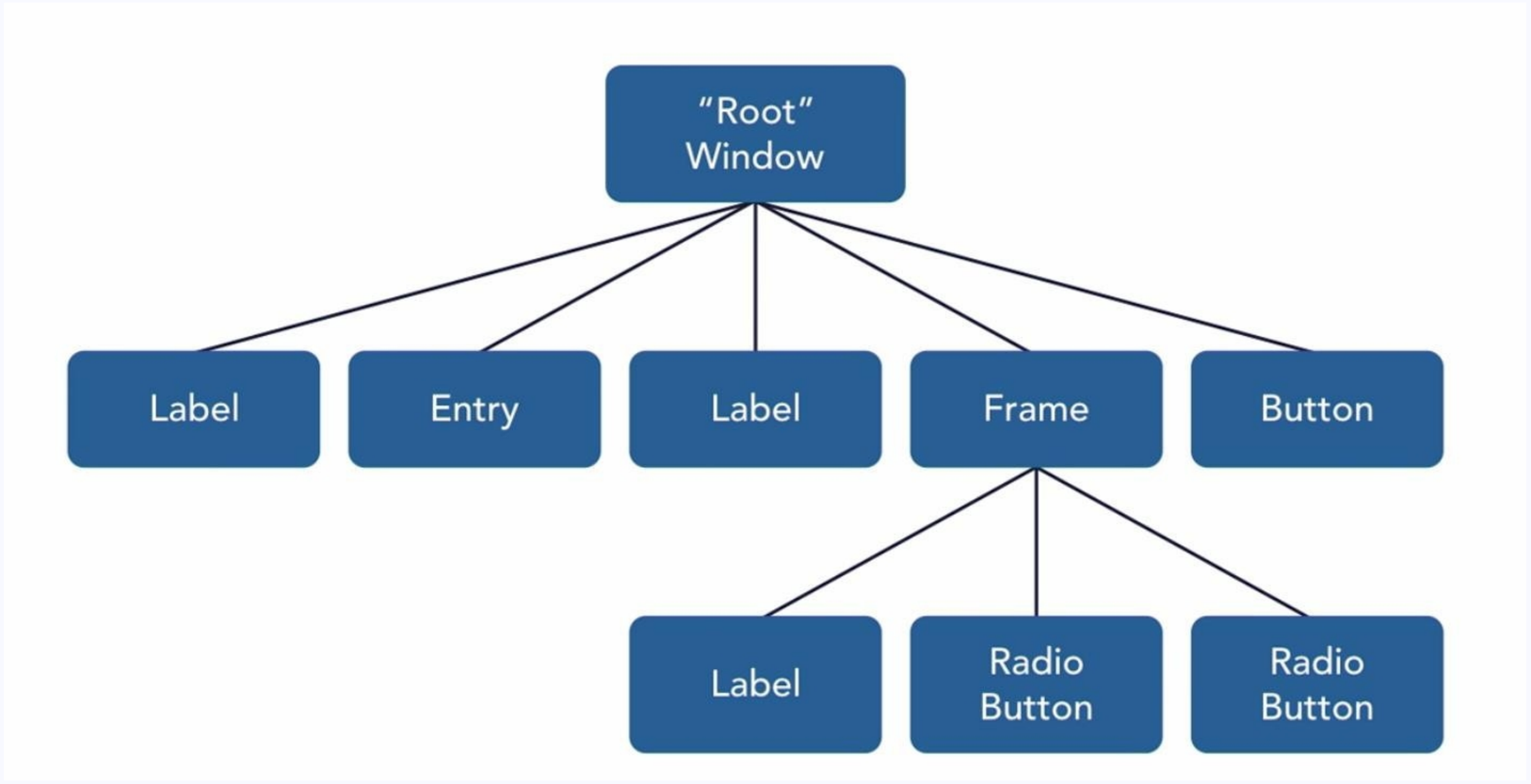
Text : Affichage de texte formaté. Permet aussi à l'utilisateur d'éditer le texte affiché. Des images peuvent également être insérées. Le texte peut être multi-ligne.

Toplevel : Une fenêtre affichée séparément, « par-dessus ».

16. Tkinter

Chaque type de widget est une classe

Les widgets sont les conteneurs visuels utilisés pour organiser ces contrôles, comme la fenêtre de l'application et les cadres qu'elle contient. Les widgets peuvent également être utilisés pour afficher des informations à l'utilisateur, allant d'une simple étiquette de texte à un graphique complexe sur un canevas.



16. Tkinter

Geometry Manager

Le simple fait de créer un widget Tkinter ne le rend pas visible. Pour qu'un widget soit affiché à l'écran, Tkinter doit savoir exactement où dessiner ce widget. C'est là que la gestion de la géométrie entre en jeu.

Le gestionnaire de géométrie de Tk prend les instructions du programme et fait de son mieux pour placer les widgets à l'endroit prévu. Pour ce faire, il utilise le concept de widgets maître et esclave.

Lorsque vous créez un nouvel objet widget et que vous spécifiez son parent, vous identifiez ce widget parent comme le maître du widget enfant. Ce widget maître utilisera un gestionnaire de géométrie pour contrôler le placement de son widget esclave.

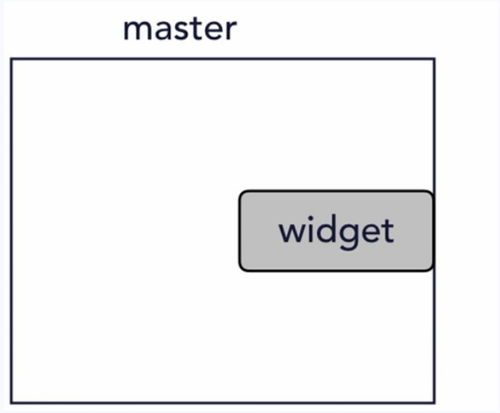
Les widgets maîtres sont généralement des conteneurs organisationnels tels que des fenêtres ou des cadres de niveau supérieur. Déterminer l'endroit exact où placer le widget esclave n'est pas toujours un problème simple.

16. Tkinter

tk_widget.py
tk_geometry_management.py
Exercices 18 à 20

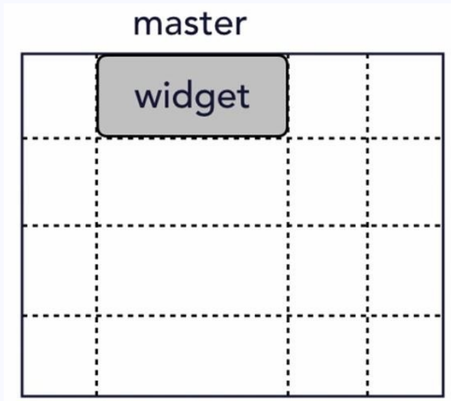
Pack

Widget.pack(side = RIGHT)



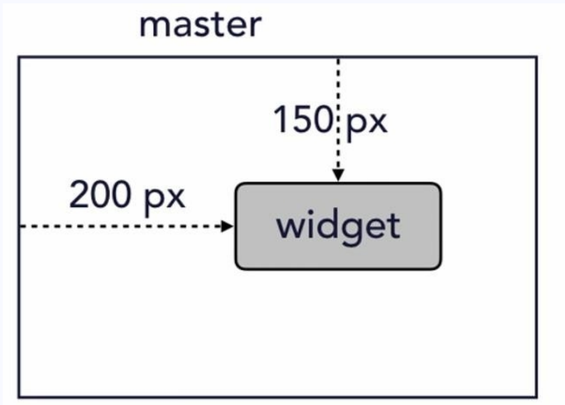
Grid

Widget.grid(row = 0, column = 1)



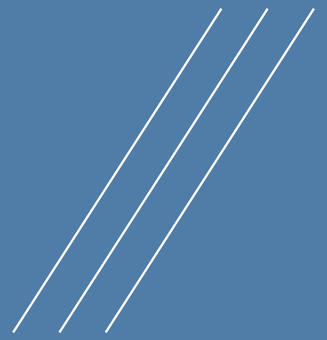
Place

Widget.place(x = 200, y = 150)



Il est possible d'utiliser plusieurs gestionnaires de géométrie au sein d'une même application mais vous devez toujours utiliser le même gestionnaire de géométrie pour organiser les widgets dans le même master.

PYTHON INITIATION



Présenté par
Xavier TABUTEAU