Richard Gomez

Le petit Python graphique avec Tkinter

Interfaces graphiques et programmation événementielle avec Python et Tkinter



ellipses

Le petit Python graphique avec Tkinter

Interfaces graphiques et programmation événementielle avec Python et Tkinter

Références sciences

Le petit Python graphique avec Tkinter

Interfaces graphiques et programmation événementielle avec Python et Tkinter

Richard Gomez

ellipses

Collection Références sciences

dirigée par Paul de Laboulaye
paul.delaboulaye@editions-ellipses.fr

Retrouvez tous les livres de la collection et des extraits sur www.editions-ellipses.fr

ISBN 9782340-084322
© Ellipses Édition Marketing S.A., 2023
8/10 rue la Quintinie 75015 Paris

DANGER LE PHOTOCOPILLAGE TUE LE LIVRE

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5.2° et 3°a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit constituerait une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

À Pepe Belinchón Cantarero (mi bisabuelo), Martina Martínez Roldán (mi bisabuela), Pilar Belinchón Martínez (mi abuela), Pedro Sáez Martínez (mi abuelo), Brígida Sáez (mi madre), Juliana Sáez (mi tía), Manolo Sáez (mi tío), Pedrito Sáez (mi tío), Federico Gómez Rosell (mi abuelo), Antonio Gómez Minaya (mi padre), David Gomez (mon frère), Didier Calas, Farid Belkalem, Ivan Revel, Stéphane Jammes, Stéphane Baseilhac, Ludovic Jacquet, les Reeds, Alexandre Trijoulet, Ysé Commenges, Yanis Riguet (mon fils) et Estelle Chaumont.

À la mémoire de Rufino Jimenez Rosell, Bautista Villanueva Sáez, Cédric Darolles et Nicolas Guilloux.

Prologue

À propos de Tkinter

Tkinter est une bibliothèque standard du langage Python. En vérité, il s'agit d'un paquet de bibliothèques (*package*). Cette bibliothèque est présente dans tous les environnements de développement Python (IDE, *integrated development environment*, en anglais), nul besoin donc de la télécharger et de l'installer à part.

Tkinter permet de créer des interfaces graphiques facilement et efficacement. Autrement dit, Tkinter permet de créer des applications possédant des fenêtres, des menus, des champs de saisie, des boutons, des sélecteurs en tous genres et bien d'autres éléments graphiques. On peut, grâce à elle, dessiner, créer des jeux et des animations.

L'environnement de développement intégré pour le langage Python (IDLE PYTHON), par exemple, est entièrement conçu avec Tkinter (on peut le télécharger sur le site officiel `www.python.org`).

Tkinter n'est pas la seule bibliothèque d'interfaces graphiques du langage Python, il y a aussi PyGTK, PyQt, wxPython, PyGObject, etc. Tkinter n'est même pas la meilleure bibliothèque graphique (son visuel a un peu vieilli), mais elle présente de nombreux avantages :

- elle n'est pas trop mal documentée (quand on compare avec les autres) ;
- elle est très répandue ;
- elle est portable et compatible avec tous les systèmes d'exploitation ;
- elle est fournie avec Python (pas besoin de l'installer) ;
- elle est plutôt facile à utiliser.

Ajoutons à cela qu'elle possède une sous-bibliothèque (TTK) apportant une touche un peu plus moderne et esthétique. Il serait idiot de se priver d'un tel outil.

À propos de ce livre

Le présent ouvrage propose une description complète de toutes les fonctionnalités offertes par la bibliothèque Tkinter. On montre comment créer une interface graphique (fenêtres, menus, boutons, curseurs, images, dessins, etc.) et comment faire de la programmation événementielle, c'est-à-dire écrire des programmes qui réagissent à des évènements (provoqués par l'utilisateur, par exemple).

À l'origine, ce document était prévu pour être la seconde partie du livre *Le petit Python objet* [2]. Malheureusement, son volume nous a contraint à scinder ce corpus en deux livres.

Comme d'habitude, nous avons privilégié :

- une présentation claire, simple et directe ;
- une rigueur mathématique ;
- et la volonté de présenter chaque notion à fond.

Chaque point est illustré par de nombreux exemples (des programmes plus ou moins longs).

On peut lire ce livre dans l'ordre, comme un cours de programmation graphique-événementielle, mais on peut aussi l'utiliser comme un aide-mémoire permettant de retrouver la syntaxe de n'importe quel widget, méthode ou fonction Tkinter.

On peut également l'utiliser pour approfondir des points particuliers en étudiant les chapitres de son choix dans l'ordre que l'on préfère : les références et les renvois sont nombreux et facilitent une telle démarche. La table des matières est écrite dans cet esprit : on y détaille chaque section, chaque sous-section et chaque sous-sous-section. Un index riche de 752 entrées se trouve à la fin du livre.

Le premier chapitre pose les bases de Tkinter. Il décrit le fonctionnement général des widgets et les notions de hiérarchie et de boucle principale. Il définit entre autre la notion de fenêtre principale, d'option et de méthode universelle.

Le deuxième chapitre propose une introspection visant à bien connaître le contenu du module et la taxonomie des classes de widgets, de variables et d'images offertes par la bibliothèque. Il offre des outils permettant d'explorer Tkinter en profondeur et de générer automatiquement (grâce à un programme que nous proposons) une documentation complète de la bibliothèque et de la sauvegarder sous la forme d'un fichier texte.

Le troisième chapitre décrit chaque type de widgets du point de vue de l'utilisateur d'une interface graphique. Nous verrons que Tkinter fournit une palette large de widgets : champs de saisie, boutons, menus, etc.

Le chapitre 4 est consacré à la notion de focus.

Les chapitres 5, 6, 7, 8, 9 et 10 étudient des options courantes, c'est-à-dire des options que presque tous les widgets possèdent : longueur, dimensions, coordonnées, ancrage (alignement et étirement), type de bordure, type de pointeurs de souris, couleurs et fonte (police de caractères).

Le chapitre 11 est consacré aux images. Tkinter implémente les images sous la forme d'objets de type `Image` ou plus précisément de type `BitmapImage` ou `PhotoImage` (ces deux derniers dérivent du premier). Ces types possèdent un certain nombre de méthodes. On peut utiliser les images pour décorer des widgets, mais on peut aussi afficher des images dans un canevas (widget de type `Canvas`) pour faire du graphisme, du traitement d'image, un jeu et bien d'autres choses.

Le chapitre 12 étudie les méthodes universelles. Il s'agit des méthodes qui existent chez tous les widgets, quel que soit leur type. Ces méthodes permettent, entre autre, d'éditer les options des widgets, d'éditer le gestionnaire des évènements, de fermer l'application, de donner le focus à un élément, de récupérer des informations sur les widgets, de contrôler le texte sélectionné dans une application, de contrôler le temps, etc.

Les chapitres 13, 14 et 15 sont consacrés aux gestionnaires de position `grid`, `pack` et `place`. N'oublions pas qu'avec Tkinter, chaque widget instancié doit être « positionné » pour être visible. Les gestionnaires de position permettent également de contrôler des options d'alignement, d'espacement et de redimensionnement.

Le chapitre 16 est consacré aux variables Tkinter. Une telle variable est un objet qui émule une variable au sens habituel du terme et qui a vocation à contrôler certains widgets. La bibliothèque fournit plusieurs types de variables : `Variable`, `IntVar`, `DoubleVar`, `BooleanVar` et `StringVar`.

Les chapitres de 17 à 38 sont consacrés aux différents types de widget (éléments graphiques, fenêtres, pop-ups, etc.) Par exemple le chapitre 17 traite des étiquettes (`Label`), le 19 des champs de saisie (`Entry`) et le 22 des boutons (`Button`).

Le chapitre 39 est consacré à la programmation évènementielle. Lorsque l'utilisateur d'un programme clique sur un widget, par exemple, Tkinter génère un objet de type `Event`. Cet objet est un enregistrement de différentes données relatives à l'évènement ayant eu lieu : lieu du clic (coordonnées), touche pressée, widget concerné, etc. Il est possible de lier un type spécifique d'évènements à une fonction donnée (*callback*). De cette manière, chaque fois qu'un tel évènement a lieu, la fonction correspondante est appelée. Finalement, un programme écrit avec Tkinter est une description de ce qu'il faut faire en fonction des évènements. Il en résulte une application interactive.

Le chapitre 40 est consacré à la programmation orientée objet avec Tkinter. Les outils fournis par Tkinter sont très orientés objets. On tirera le meilleur de Tkinter en suivant le paradigme objet.

Le chapitre 41 est consacré à la base des options. Il s'agit en quelque sorte d'une feuille de style définissant l'aspect visuel de l'interface graphique créée. La base des options permet de factoriser la partie esthétique d'un programme.

Les chapitres 42, 43, 44 et 45 proposent des projets graphiques complets illustrant tous les chapitres précédents ainsi que notre livre consacré à la POO [2]. On propose un traceur de courbes mathématiques, un simulateur de *boids* (banc de poissons, groupe d'étourneaux), un simulateur d'attraction universelle (avec des corps qui s'attirent, entrent en orbite et s'entrechoquent) et un serveur de chat multi-usagers, ainsi que son client.

Le chapitre 46 est une annexe sur la partialisation des fonctions et la notion de *curryfication* (notion informatique introduite par Moses Schönfinkel et Haskell Curry). On est souvent amené à opérer une curryfication sur une fonction quand on utilise Tkinter.

La deuxième partie de ce livre est consacrée au sous-module TTK de Tkinter. Le module TTK, abréviation de *Tkinter Themed Widget Set*, apporte de nouveaux widgets comme la barre de progression ou la fenêtre d'arborescence. Il redéfinit aussi la plupart des widgets de Tkinter en les simplifiant et leur apportant une touche de modernité. Les outils de TTK permettent également de séparer les fonctionnalités de l'aspect visuel de chaque élément graphique.

Les chapitres 47 et 48 donnent une présentation générale de TTK et montrent comment générer une documentation assez complète de l'ensemble de la bibliothèque.

Le chapitre 49 présente les nouveaux widgets apportés par TTK du point de vue de l'utilisateur.

Le chapitre 50 montre comment écrire les feuilles de style qui détermineront l'aspect visuel de nos applications.

Le chapitre 51 introduit la notion de *mapping*. Grâce à elle, on peut définir l'aspect d'un widget en fonction de son état de manière beaucoup plus simple et précise qu'avec Tkinter.

Le chapitre 52 montre comment éditer la structure d'un widget et le chapitre 53 comment créer et éditer des thèmes.

Les chapitres de 54 à 73 sont consacrés aux différents types de widget offerts par TTK (20 types en tout).

Remerciements

Ce livre est le fruit d'une longue compilation d'informations récoltées et mûries au fil des années, à l'occasion de différents projets de programmation et de différents programmes d'enseignement. Je tiens à remercier chaleureusement mon ami et collègue Jocelyn Etienne pour son soutien, sa finesse, ses remarques et l'aide apportée dans la correction de cet ouvrage. Je veux aussi remercier mon collègue et ami Stéphane Lévy avec lequel j'ai pu échanger régulièrement sur Python. Lui aussi a soutenu ce projet et contribué à corriger les coquilles et les programmes de ce livre. Je remercie Joris van der Hoeven, Massimiliano Gubinelli et Giovanni Piredda de l'équipe TEXMACS pour les améliorations qu'ils ont apportées à ce document (voir [17]). Je remercie ma compagne Estelle Chaumont pour son soutien et la patience dont elle a fait preuve pendant la rédaction de ce texte. Sans sa bienveillance et son amour, ce livre n'aurait probablement pas existé. Je remercie mes parents, Antoine Gómez Minaya et Brigitte Sáez Belinchón, pour leur soutien et leur bienveillance, surtout quand je travaille chez eux à Torreilles (Pyrénées Orientales). Je remercie le petit village de Guitera-Les-Bains (Corse) qui m'a offert la paix et la tranquilité nécessaires pour réfléchir aux notions présentées dans cet ouvrage (sans oublier les fromages de la famille Casamarta). Je remercie Paul de Laboulaye, Corinne Baud et les éditions Ellipses pour avoir eu la gentillesse de s'être intéressés à ce document et pour avoir accepté de le publier.

Les programmes exposés dans ce livre ont été testés avec Python 3.8.5.

Ce livre a été tapé avec TEXMACS 1.99.16 (éditeur de textes scientifiques, logiciel libre GNU, `www.texmacs.org`).

À Montpellier,

à Villeneuve-Lès-Maguelone,

à Torreilles,

à Guitera-Les-Bains,

2020-2023



Conventions

1. Chaque programme en langage Python est présenté dans un encart gris (police de caractères à chasse fixe).

2. Le nom d'une méthode spéciale commence et termine toujours par deux underscores, comme par exemple `__getattr__`, ce qui peut alourdir le texte. Il arrive parfois que nous écrivions tout simplement *getattr* (en italique) au lieu de `__getattr__`. Bien sûr, on ne confondra pas la méthode *getattr* avec la fonction interne `getattr` qui elle s'écrit sans underscores.

3. Quand on renvoie par exemple à la section 5 du chapitre 3 de ce livre, on écrit 3.5. De même, quand on renvoie à la sous-section 7.2 du chapitre 3, on écrit 3.7.2. Dans toute référence de ce type, le premier nombre indique le numéro du chapitre où se trouve la cible.

4. Certains énoncés sont numérotés (définitions, remarques, etc.) La définition 12.3, par exemple, désigne le 3-ème énoncé numéroté du chapitre 12. Dans la référence d'un tel énoncé, le premier nombre indique le numéro du chapitre où se trouve la cible. Même convention avec les figures, les tableaux et les équations.

5. Quand on écrit une classe, il arrive parfois, dans un souci de clarté, que l'on ne présente pas tout son contenu d'emblée. Par exemple, la classe `A` ci-dessous possède deux méthodes nommées `ceci` et `cela` :

```
class A:
    def ceci(self): print("Reeds rocks")
    def cela(self): print("SpacePopDave")
```

et on pourrait présenter `A` en commençant par `ceci` :

```
class A:
    def ceci(self): print("Reeds rocks")
```

et en terminant par `cela` :

```
def cela(self): print("SpacePopDave")
```

en gardant à l'esprit que le corps de `cela` est écrit dans celui de `A` avec, bien entendu, une indentation de 4 cadratins (nous avons omis l'indentation dans `cela` dans un souci de lisibilité).

6. Il arrive que la description d'une méthode ne donne pas les vrais noms des paramètres mais une adaptation plus facile à comprendre. C'est le cas par exemple de la méthode `tag_bind` d'un canevas pour laquelle nous avons écrit `tag_bind(u, sequence, function, add)` alors qu'il s'agit en fait de `tag_bind(tagOrId, sequence, func, add)`. Pour connaître la signature d'un appelable, on peut utiliser la fonction `signature` du module `inspect`.

Table des matières

I Tkinter . . . 39

1 Généralités . . . 41

1.1 Origines de Tkinter . . . 41
1.2 Grands principes . . . 42
1.3 Importer Tkinter . . . 42
1.4 Fenêtre et boucle principales . . . 43
1.5 Widgets conteneurs . . . 44
1.6 Instancier un widget . . . 45
1.7 Placer un widget . . . 46
1.8 Détruire un widget . . . 46
1.9 Nom Tkinter d'un widget . . . 47
1.9.1 Nom . . . 47
1.9.2 Nom complet . . . 49
1.9.3 Remarques . . . 50
1.10 Numéro d'identité Tkinter . . . 50
1.11 Représentations formelle et informelle d'un widget . . . 52
1.12 Hiérarchie . . . 53
1.13 Contrôler les options d'un widgets . . . 55
1.13.1 Méthode `configure` . . . 55
1.13.2 Méthode `cget` . . . 55
1.13.3 Dictionnaire des options . . . 56
1.14 Aspect dictionnaire des widgets . . . 58
1.15 État d'un widget . . . 59
1.16 Instancier sans nommer . . . 60
1.17 Méthodes universelles . . . 61
1.18 Programmation orientée objet . . . 62
1.19 Modules accompagnant `tkinter` . . . 62
1.19.1 Enrichissements . . . 62
1. Module `tkinter.ttk` . . . 62
2. Module `tkinter.messagebox` . . . 63
3. Module `tkinter.filedialog` . . . 63
4. Module `tkinter.colorchooser` . . . 63
5. Module `tkinter.scrolledtext` . . . 63

1.19.2 Utilitaires . . . 63
6. Module `tkinter.font` . . . 63
7. Module `turtle` . . . 63
8. Module `tkinter.constants` . . . 63
1.19.3 Autres . . . 64
9. Module `tkinter.simpledialog` . . . 64
10. Module `tkinter.commondialog` . . . 65
11. Module `tkinter.dnd` : bientôt déprécié. . . . 65
12. Module `tkinter.tix` : déprécié. . . . 65
1.20 Un exemple en langage TCL-TK . . . 65

2 Introspection . . . 67

2.1 Contenu du module `tkinter` . . . 67
2.2 Taxonomie des widgets . . . 68
2.3 Classe `Variable` . . . 70
2.4 Classe `Image` . . . 73
2.5 Options d'un widget . . . 75
2.6 Alias d'une option . . . 77
2.7 Méthodes universelles . . . 79
2.8 Méthodes d'un widget . . . 82
2.9 Générer la documentation de tous les widgets . . . 84
2.10 Interpréteur TCL . . . 89
2.10.1 Classe `tkapp` . . . 89
2.10.2 Fonction `Tcl` . . . 89
2.10.3 Méthode `eval` . . . 90
2.10.4 Méthode `call` . . . 90
2.10.5 Créer une commande TCL . . . 91
2.10.6 Variables TCL . . . 92
2.10.7 Émuler une variable TCL avec un objet . . . 93
2.10.8 Une autre manière d'émuler une variable TCL . . . 93
2.10.9 Demander à TCL de créer une variable Python . . . 95
2.10.10 Signature . . . 96

3 Point de vue utilisateur . . . 97

3.1 Champ de saisie (`Entry`) . . . 97
3.2 Zone de sélection numérique (`Spinbox`) . . . 97
3.3 Curseur (`Scale`) . . . 98
3.4 Bouton (`Button`) . . . 98
3.5 Case à cocher (`Checkbutton`) . . . 98
3.6 Bouton de radio (`Radiobutton`) . . . 98
3.7 Zone de liste (`Listbox`) . . . 99
3.8 Sélecteur d'options (`OptionMenu`) . . . 99
3.9 Canevas (`Canvas`) . . . 99
3.10 Zone de texte (`Text`) . . . 100
3.11 Fenêtre à pans (`PanedWindow`) . . . 100
3.12 Barre de défilement (`Scrollbar`) . . . 100
3.13 Menu (`Menu`) . . . 101
3.14 Bouton-menu (`Menubutton`) . . . 101

3.15 Fenêtres pop-up . . . 102

4 Focus . . . 103

4.1 Notion de focus . . . 103
4.2 Donner le focus . . . 103
4.3 Cycle du focus . . . 104
4.4 Widgets acceptant le focus . . . 104
4.5 Widgets sous focus . . . 104
4.6 Aspect d'un widget sous focus . . . 105
4.7 Méthodes . . . 105
4.7.1 Donner le focus . . . 105
4.7.2 Connaître le cycle de focus . . . 106
4.7.3 Savoir qui a le focus . . . 106
4.8 Expérience . . . 106

5 Longueurs et coordonnées . . . 109

5.1 Longueurs . . . 109
5.1.1 Unités . . . 109
5.1.2 Résolution d'écran . . . 109
5.1.3 Facteur d'échelle (*scaling factor*) . . . 110
5.1.4 Unités par défaut chez Tkinter . . . 111
5.1.5 Unités physiques chez Tkinter . . . 112
5.1.6 Méthodes pour convertir des longueurs en pixels . . . 112
5.1.7 Erreur de résolution . . . 113
5.1.8 Communiquer la résolution à Tkinter . . . 115
5.1.9 Petite expérience . . . 115
5.1.10 Dimensions de l'écran . . . 116
5.1.11 Connaître les dimensions d'un widget . . . 117
5.2 Coordonnées . . . 119
5.2.1 Systèmes de coordonnées . . . 119
5.2.2 Donne-moi un widget, je te dirai ses coordonnées . . . 119
5.2.3 Donne-moi des coordonnées, je te dirai le widget . . . 122
5.3 Géométrie d'une fenêtre (*Geometry string*) . . . 124
5.3.1 Méthode `geometry` . . . 124
5.3.2 Format général d'une *geometry string* . . . 125
5.3.3 Cas particuliers . . . 126
5.3.4 Expérience . . . 126
5.3.5 Récupérer l'information géométrique . . . 127

6 Ancrage . . . 131

6.1 Notion d'ancrage . . . 131
6.1.1 Ancrage d'un texte dans un widget . . . 131
6.1.2 Ancrage d'un widget dans un espace . . . 132
6.1.3 Ancrage d'une grille dans un conteneur . . . 132
6.1.4 Point d'ancrage d'un objet . . . 133
6.2 Valeurs possibles d'un ancrage . . . 134
6.3 Étirement . . . 135
6.4 Widgets possédant une option `anchor` . . . 136
6.5 Méthodes possédant une option d'ancrage . . . 136

6.5.1 Gestionnaires de position . . . 136
6.5.1.1 Méthode `grid` . . . 136
6.5.1.2 Méthode `grid_anchor` . . . 137
6.5.1.3 Méthode `pack` . . . 137
6.5.1.4 Méthode `place` . . . 137
6.5.2 Méthodes de canevas . . . 138
6.5.3 Méthode d'insertion d'une fenêtre à pans . . . 139
6.6 Cas particulier de `LabelFrame` . . . 140

7 Bordure . . . 141

7.1 Introduction . . . 141
7.2 Styles de bordure . . . 141
7.3 Widgets particuliers . . . 142
7.3.1 Champ de saisie (`Entry`) . . . 142
7.3.2 Zone de sélection numérique (`Spinbox`) . . . 143
7.3.3 Curseur (`Scale`) . . . 144
7.3.4 Bouton (`Button`) . . . 145
7.3.5 Case à cocher (`Checkbutton`) . . . 146
7.3.5.1 Case avec coche . . . 147
7.3.5.2 Case sans coche (push-push) . . . 147
7.3.5.3 Case avec image qui change . . . 147
7.3.5.4 Un programme pour tester les trois . . . 148
7.3.6 Bouton de radio (`Radiobutton`) . . . 150
7.3.6.1 Bouton de radio classique . . . 150
7.3.6.2 Bouton de radio push-push . . . 150
7.3.6.3 Bouton de radio icône . . . 151
7.3.6.4 Un programme pour tester les trois . . . 151
7.3.7 Zone de liste (`Listbox`) . . . 153
7.3.8 Zone de texte (`Text`) . . . 155
7.3.9 Fenêtre à pans (`PanedWindow`) . . . 157
7.3.10 Barre de défilement (`Scrollbar`) . . . 159
7.3.11 Menu (`Menu`) . . . 161

8 Pointeur de la souris . . . 165

8.1 Introduction . . . 165
8.2 Pointeurs offerts par Tkinter . . . 165
8.3 Pointeurs chez MAC OS . . . 166
8.4 Pointeurs chez PC WINDOWS . . . 166
8.5 Fabriquer un pointeur . . . 166
8.6 Un petit programme . . . 166

9 Couleurs . . . 171

9.1 Système RGB . . . 171
9.2 Tkinter et RGB . . . 172
9.3 Tkinter et noms de couleurs . . . 174
9.4 Méthode `winfo_rgb` . . . 179
9.5 Transparence . . . 180
9.6 Contrôle global des couleurs . . . 180
9.7 Quelques programmes . . . 181

9.7.1 Palette des couleurs . . . 181
9.7.2 Zone de sélection de couleurs . . . 182
9.7.3 Conversions . . . 184

10 Fontes . . . 185

10.1 Introduction . . . 185
10.2 Propriétés d'une fonte . . . 185
10.2.1 Police . . . 185
10.2.2 Taille . . . 186
10.2.3 Style . . . 186
10.3 Différentes manières de déterminer une fonte . . . 186
10.3.1 Avec un nom propre . . . 186
10.3.2 Avec un uplet . . . 186
10.3.3 Avec un uplet mis en chaîne de caractères . . . 187
10.3.4 Avec une chaîne de propriétés nommées . . . 187
10.3.5 Avec un objet de type `Font` . . . 188
10.4 Définition partielle . . . 188
10.5 Module `font` . . . 189
10.5.1 Fonction `families` . . . 190
10.5.2 Fonction `names` . . . 190
10.5.3 Fonction `nametofont` . . . 190
10.6 Classe `Font` . . . 191
10.6.1 Options . . . 191
10.6.2 Instanciation . . . 191
10.6.3 Attributs . . . 192
10.6.4 Méthodes . . . 192
10.6.5 Exemples . . . 192
10.7 Contrôler les options d'une fonte . . . 193
10.8 Récupérer les propriétés de la fonte d'un widget . . . 193
10.9 Copier une fonte . . . 194
10.10 Métriques . . . 194
10.11 Exemples de polices . . . 195
10.12 Programme pour éditer une fonte . . . 197

11 Images . . . 199

11.1 Introduction . . . 199
11.1.1 Option `bitmap` . . . 199
11.1.2 Option `image` . . . 200
11.2 Bitmaps fournis par Tkinter . . . 201
11.3 Format XBM . . . 202
11.4 Type `Image` . . . 207
11.4.1 Introduction . . . 207
11.4.2 Signature . . . 207
11.4.3 Attributs . . . 207
11.4.4 Options . . . 208
11.4.5 Aspect dictionnaire d'une image . . . 208
11.4.6 Méthodes . . . 208
11.5 Type `BitmapImage` . . . 208
11.5.1 Introduction . . . 208

11.5.2 Signature . . . 209
11.5.3 Attributs . . . 209
11.5.4 Options . . . 209
11.5.5 Méthodes . . . 209
11.5.6 Option `file` . . . 209
11.5.7 Option `data` . . . 210
11.5.8 Choix des deux couleurs . . . 210
11.5.9 Masque . . . 211
11.6 Type `PhotoImage` . . . 214
11.6.1 Introduction . . . 214
11.6.2 Signature . . . 214
11.6.3 Attributs . . . 215
11.6.4 Options . . . 215
11.6.5 Méthodes . . . 215
11.6.6 Option `file` . . . 215
11.6.7 Option `data` . . . 216
11.6.8 Méthode `put` . . . 217
11.6.9 Autres méthodes . . . 220
11.6.9.1 Connaître la valeur des pixels . . . 220
11.6.9.2 Contrôler les options . . . 220
11.6.9.3 Remettre l'image à zéro . . . 221
11.6.9.4 Extraire une sous-image . . . 221
11.6.9.5 Agrandir l'image . . . 221
11.6.9.6 Copier . . . 221
11.6.9.7 Sauvegarder . . . 221
11.6.9.8 Exemples . . . 221
11.7 Attention au ramasse-miettes . . . 223
11.8 Récupérer le nom de toutes les images . . . 224
11.9 Quelques outils du module Pillow . . . 224
11.9.1 Méthode `resize` . . . 224
11.9.2 Classe `ImageTk.PhotoImage` . . . 225
11.10 Annexe : formats PBM, PGM et PPM . . . 226
11.10.1 Format PBM . . . 226
11.10.2 Format PGM . . . 226
11.10.3 Format PPM . . . 227
11.10.4 Exemple . . . 227
11.11 Annexe : module `os` . . . 230

12 Méthodes universelles . . . 231

12.1 Introduction . . . 231
12.2 Méthodes relatives à l'écran . . . 233
12.3 Méthodes relatives au texte sélectionné . . . 233
12.4 Méthodes relatives au presse-papier . . . 238
12.5 Méthodes relatives au temps . . . 238
12.6 Méthodes pour bloquer un fil . . . 239
12.7 Méthodes relatives au pointeur de la souris . . . 241
12.8 Méthodes pour convertir . . . 241
12.9 Méthodes diverses . . . 242

13 Gestionnaire de position `grid` . . . 243
13.1 Introduction . . . 243
13.2 Mode d'emploi . . . 243
13.3 Options de `grid` . . . 244
13.4 Étalement sur plusieurs cellules . . . 245
13.5 Ancrage et étirement dans une cellule . . . 245
13.6 Ancrage de la grille . . . 246
13.7 Propagation des dimensions . . . 248
13.8 Marges . . . 249
13.8.1 Marges extérieures . . . 249
13.8.2 Marges intérieures . . . 249
13.8.3 Un programme pour tester . . . 249
13.9 Propriétés des colonnes de la grille . . . 251
13.9.1 Méthode `grid_columnconfigure` . . . 251
13.9.2 Option `minsize` . . . 251
13.9.3 Option `pad` . . . 251
13.9.4 Option `weight` . . . 251
13.10 Propriétés des lignes de la grille . . . 252
13.10.1 Méthode `grid_rowconfigure` . . . 252
13.10.2 Exemple . . . 252
13.11 Changer le père du widget (`in_`) . . . 253
13.12 Retirer un widget de la grille . . . 253
13.13 Informations sur la grille . . . 254
13.13.1 Taille de la grille . . . 254
13.13.2 Contenu de la grille . . . 254
13.13.3 Coordonnées des éléments d'une grille . . . 254
13.14 Informations sur le placement d'un widget . . . 255
13.14.1 Connaître les options *grid* d'un widget . . . 255
13.14.2 Savoir si un widget a été placé . . . 255
13.14.3 Savoir quel gestionnaire a placé un widget . . . 255
13.15 Autres gestionnaires de position . . . 256

14 Gestionnaire de position `pack` . . . 257
14.1 Fonctionnement : option `side` . . . 257
14.2 Options de `pack` . . . 260
14.3 Ancrage . . . 260
14.4 Étirement . . . 261
14.5 Propagation des dimensions . . . 263
14.6 Marges . . . 264
14.7 Changer le père d'un widget (`in_`) . . . 265
14.8 Retirer un widget de l'écran . . . 266
14.9 Informations sur un conteneur . . . 266
14.10 Information sur le placement d'un widget . . . 266

15 Gestionnaire de position `place` . . . 269
15.1 Mode d'emploi . . . 269
15.2 Options de `place` . . . 270
15.3 Point d'ancrage . . . 270
15.4 Taille du widget . . . 271
15.5 Non propagation des dimensions . . . 271

15.6 Option `bordermode` . . . 272
15.7 Changer le père d'un widget (`in_`) . . . 273
15.8 Retirer un widget de l'écran . . . 274
15.9 Informations sur un conteneur . . . 274
15.10 Informations sur le placement d'un widget . . . 274

16 Variables Tkinter . . . 277

16.1 Introduction . . . 277
16.2 Types de variables . . . 277
16.3 Instanciation . . . 278
16.4 Lecture et écriture . . . 279
16.5 Séquences et autres types . . . 280
16.6 Widgets associés à une variable . . . 281
16.6.1 Étiquette (`Label`) . . . 281
16.6.2 Affichage (`Message`) . . . 281
16.6.3 Champ de saisie (`Entry`) . . . 281
16.6.4 Zone de sélection numérique (`Spinbox`) . . . 281
16.6.5 Curseur (`Scale`) . . . 282
16.6.6 Bouton (`Button`) . . . 282
16.6.7 Case à cocher (`Checkbutton`) . . . 282
16.6.8 Bouton de radio (`Radiobutton`) . . . 283
16.6.9 Zone de liste (`Listbox`) . . . 283
16.6.10 Sélecteur d'options (`OptionMenu`) . . . 283
16.6.11 Bouton-menu (`Menubutton`) . . . 284
16.6.12 Contenu d'une option `variable`, `textvariable` ou `listvariable` . . . 284
16.7 Variable comme attribut d'un widget . . . 285
16.8 On peut se passer de `Variable` et ses dérivées . . . 286
16.9 Programmation évènementielle . . . 286
16.9.1 Introduction . . . 286
16.9.2 Créer un binding . . . 286
16.9.3 Arguments passés au callback . . . 287
16.9.4 Récupérer la liste des bindings sur une variable . . . 288
16.9.5 Désactiver un binding . . . 289
16.9.6 Application aux sélecteurs d'options . . . 289
16.10 Exploration . . . 290
16.10.1 Affichage . . . 290
16.10.2 Copier une variable en utilisant `name` . . . 291
16.10.3 Attributs . . . 292

17 Étiquettes (`Label`) . . . 293

17.1 Introduction . . . 293
17.2 Liste des options . . . 293
17.3 Liste des méthodes . . . 293
17.4 Message affiché . . . 293
17.5 Image affichée . . . 294
17.6 Visuel . . . 294
17.7 Focus . . . 295
17.8 État du widget . . . 295
17.9 Widget survolé par la souris . . . 295

17.10 Exemple presque complet . . . 296

18 Affichages (`Message`) . . . 299

18.1 Introduction . . . 299
18.2 Liste des options . . . 299
18.3 Liste des méthodes . . . 299
18.4 Message affiché . . . 299
18.5 Visuel . . . 300
18.6 Focus . . . 300
18.7 Widget survolé par la souris . . . 300

19 Champs de saisie (`Entry`) . . . 301

19.1 Introduction . . . 301
19.2 Liste des options . . . 302
19.3 Liste des méthodes . . . 302
19.4 Visuel . . . 302
19.5 Focus . . . 302
19.6 État du widget . . . 303
19.7 Widget survolé par la souris . . . 303
19.8 Texte . . . 303
19.9 Texte sélectionné (surligné) . . . 304
19.10 Défilement (scrolling) . . . 304
19.10.1 Mode d'emploi . . . 304
19.10.2 Approfondissement . . . 304
19.11 Défilement rapide (fast scrolling) . . . 305
19.12 Procédure de validation . . . 306
19.12.1 Principe . . . 306
19.12.2 Fonction de validation . . . 307
19.12.3 Fonction de correction . . . 307
19.12.4 Option `validate` . . . 307
19.12.5 Codes de substitution . . . 308
19.12.6 Faire passer des codes à la fonction de validation . . . 308
19.12.7 Faire passer des codes à la fonction de correction . . . 309
19.12.8 Exemple simple . . . 309
19.12.9 Exemple complet . . . 310
19.13 Indices spéciaux . . . 313
19.14 Méthodes pour traiter le texte . . . 314
19.15 Méthodes pour traiter du texte sélectionné . . . 314
19.16 Insertion du curseur . . . 315
19.17 Utilisation classique . . . 315

20 Zones de sélection numérique (`Spinbox`) . . . 317

20.1 Utilisation simple . . . 317
20.2 Spinbox avec variable de contrôle . . . 318
20.3 Sélection non numérique . . . 318
20.4 Liste des options . . . 319
20.5 Liste des méthodes . . . 319
20.6 Zone de saisie . . . 319
20.7 Visuel . . . 320

20.8 Réactivité des flèches . . . 320
20.9 Focus . . . 321
20.10 État du widget . . . 321
20.11 Widget survolé par la souris . . . 321
20.12 Texte sélectionné (surligné) . . . 321
20.13 Défilement (scrolling) . . . 322
20.14 Défilement rapide (fast scrolling) . . . 322
20.15 Procédure de validation . . . 322
20.16 Indices spéciaux . . . 323
20.17 Nom des éléments composant un spinbox . . . 324
20.18 Méthodes pour l'édition . . . 324
20.19 Méthodes pour gérer les coordonnées . . . 324
20.20 Méthodes pour la sélection . . . 325
20.21 Méthode `invoke` . . . 325

21 Curseurs (`Scale`) . . . 327

21.1 Mode d'emploi . . . 327
21.2 Curseur avec variable de contrôle . . . 327
21.3 Liste des options . . . 328
21.4 Liste des méthodes . . . 328
21.5 Visuel . . . 328
21.6 Texte . . . 329
21.7 Réactivité des flèches . . . 329
21.8 Focus . . . 329
21.9 État du widget . . . 329
21.10 Widget survolé par la souris . . . 330
21.11 Description des méthodes . . . 330

22 Boutons (`Button`) . . . 331

22.1 Mode d'emploi . . . 331
22.2 Liste des options . . . 331
22.3 Liste des méthodes . . . 331
22.4 Légende . . . 332
22.5 Icône . . . 332
22.6 Visuel . . . 332
22.7 Focus . . . 333
22.8 État du widget . . . 333
22.9 Bouton survolé par la souris . . . 333
22.10 Répétitions . . . 334
22.11 Description des méthodes . . . 334

23 Cases à cocher (`Checkbutton`) . . . 335

23.1 Introduction . . . 335
23.1.1 Mode d'emploi . . . 335
23.1.2 Valeurs de la variable de contrôle . . . 335
23.2 Types de cases à cocher . . . 335
23.2.1 Case avec coche . . . 336
23.2.2 Case sans coche (push-push) . . . 336
23.2.3 Case avec image qui change . . . 336

23.2.4 Mélange . . . 336
23.3 Liste des options . . . 336
23.4 Liste des méthodes . . . 336
23.5 Légende . . . 337
23.6 Icône . . . 337
23.7 Visuel . . . 337
23.8 Visuels *on* et *off* . . . 338
23.8.1 Case avec coche . . . 338
23.8.2 Case sans coche (push-push) . . . 338
23.8.3 Case avec image qui change . . . 338
23.9 Focus . . . 338
23.10 État du widget . . . 338
23.11 Case survolée par la souris . . . 339
23.12 Description des méthodes . . . 339

24 Boutons de radio (`Radiobutton`) . . . 341

24.1 Mode d'emploi . . . 341
24.2 Types de boutons de radio . . . 342
24.2.1 Bouton de radio classique . . . 342
24.2.2 Bouton de radio push-push . . . 342
24.2.3 Image qui change . . . 342
24.2.4 Mélange . . . 342
24.3 Liste des options . . . 342
24.4 Liste des méthodes . . . 343
24.5 Légende . . . 343
24.6 Icône . . . 343
24.7 Visuel . . . 343
24.8 Visuels *on* et *off* . . . 344
24.8.1 Bouton classique . . . 344
24.8.2 Bouton push-push . . . 344
24.8.3 Image qui change . . . 344
24.9 Focus . . . 344
24.10 État du bouton . . . 344
24.11 Bouton survolé par la souris . . . 345
24.12 Description des méthodes . . . 345

25 Zones de liste (`Listbox`) . . . 347

25.1 Introduction . . . 347
25.2 Liste des options . . . 347
25.3 Liste des méthodes . . . 348
25.4 Mode de sélection . . . 348
25.5 Apparence des items . . . 348
25.6 Visuel . . . 348
25.7 Focus . . . 349
25.8 État du widget . . . 349
25.9 Zone survolée par la souris . . . 349
25.10 Sélection de texte . . . 350
25.11 Défilement (scrolling) . . . 350
25.11.1 Mode d'emploi . . . 350

25.11.2 Méthode `see` . . . 350
25.11.3 Approfondissement . . . 351
25.12 Défilement rapide (fast scrolling) . . . 352
25.13 Indices spéciaux . . . 353
25.14 Méthodes pour gérer la sélection d'items . . . 353
25.15 Méthodes pour éditer la liste . . . 354
25.16 Méthodes pour éditer les options des items . . . 354
25.17 Méthodes pour gérer les coordonnées . . . 355
25.18 Programmation évènementielle . . . 355

26 Sélecteurs d'options (`OptionMenu`) . . . 357

26.1 Mode d'emploi . . . 357
26.2 Liste des options . . . 358
26.3 Liste des méthodes . . . 358
26.4 Texte affiché . . . 358
26.5 image . . . 359
26.6 Visuel . . . 359
26.7 focus . . . 360
26.8 État du widget . . . 360
26.9 Widget survolé par la souris . . . 360
26.10 Associer un menu . . . 360
26.11 Option `command` versus méthode `trace_add` . . . 361

27 Canevas (`Canvas`) . . . 363

27.1 Introduction . . . 363
27.2 Tags . . . 364
27.3 Liste des options . . . 364
27.4 Liste des méthodes . . . 365
27.5 Visuel . . . 365
27.6 Focus . . . 365
27.7 État du widget . . . 365
27.8 Canevas survolé par la souris . . . 365
27.9 Sensibilité des items . . . 366
27.10 Sélection d'items . . . 366
27.11 Défilement (scrolling) . . . 366
27.11.1 Mode d'emploi . . . 366
27.11.2 Exemple . . . 367
27.11.3 Approfondissement . . . 367
27.12 Défilement rapide (fast scrolling) . . . 368
27.13 Défilement et coordonnées . . . 369
27.14 Item de type `rectangle` . . . 370
27.14.1 Mode d'emploi . . . 370
27.14.2 Options de `create_rectangle` . . . 371
27.14.3 Trait . . . 372
27.14.4 Remplissage . . . 372
27.14.5 État . . . 372
27.14.6 Motif survolé par la souris . . . 372
27.15 Item de type `polygon` . . . 372
27.15.1 Mode d'emploi . . . 372

27.15.2 Options de `create_polygon` . . . 373
27.15.3 Trait . . . 373
27.15.4 Remplissage . . . 373
27.15.5 État . . . 374
27.15.6 Motif survolé par la souris . . . 374
27.16 Item de type `oval` . . . 374
27.16.1 Mode d'emploi . . . 374
27.16.2 Options de `create_oval` . . . 374
27.16.3 Trait . . . 374
27.16.4 Remplissage . . . 375
27.16.5 État . . . 375
27.16.6 Motif survolé par la souris . . . 375
27.17 Item de type `arc` . . . 375
27.17.1 Mode d'emploi . . . 375
27.17.2 Options de `create_arc` . . . 376
27.17.3 Trait . . . 376
27.17.4 Remplissage . . . 376
27.17.5 État . . . 377
27.17.6 Motif survolé par la souris . . . 377
27.18 Item de type `line` . . . 377
27.18.1 Mode d'emploi . . . 377
27.18.2 Options de `create_line` . . . 377
27.18.3 Extrémités . . . 377
27.18.4 Avec ou sans flèche . . . 378
27.18.5 Trait . . . 378
27.18.6 État . . . 379
27.18.7 Motif survolé par la souris . . . 379
27.19 Item de type `bitmap` . . . 379
27.19.1 Mode d'emploi . . . 379
27.19.2 Options de `create_bitmap` . . . 379
27.19.3 Couleurs . . . 379
27.19.4 État . . . 380
27.19.5 Motif survolé par la souris . . . 380
27.20 Item de type `image` . . . 380
27.20.1 Mode d'emploi . . . 380
27.20.2 Options de `create_image` . . . 380
27.20.3 État . . . 380
27.20.4 Motif survolé par la souris . . . 380
27.21 Item de type `text` . . . 381
27.21.1 Mode d'emploi . . . 381
27.21.2 Options de `create_text` . . . 381
27.21.3 Fonte, couleur et texture . . . 381
27.21.4 État . . . 381
27.21.5 Motif survolé par la souris . . . 382
27.22 Item de type `window` . . . 382
27.22.1 Mode d'emploi . . . 382
27.22.2 Options de `create_window` . . . 382
27.22.3 Dimensions . . . 382
27.22.4 État . . . 382

27.23 Jonction entre segments . . . 382
27.24 Pointillés . . . 383
27.25 Remplissage . . . 383
27.26 Gérer les options d'un item . . . 384
27.27 Gérer les coordonnées . . . 384
27.28 Retrouver l'identité à partir des coordonnées . . . 385
27.29 Gérer les évènements . . . 386
27.30 Gérer les tags . . . 388
27.31 Retrouver l'identité grâce aux tags . . . 389
27.32 Gérer l'ordre des items dans la pile . . . 390
27.33 Gérer les items de type `"text"` . . . 390
27.34 Gérer les items de type `"window"` . . . 392
27.35 Autres méthodes . . . 392
27.36 Exemples . . . 393

28 Zones de texte (`Text`) . . . 397

28.1 Mode d'emploi . . . 397
28.2 Liste des options . . . 398
28.3 Liste des méthodes . . . 398
28.4 Visuel . . . 398
28.5 Lignes du texte . . . 399
28.6 Focus . . . 399
28.7 État du widget . . . 400
28.8 Widget survolé par la souris . . . 400
28.9 Curseur d'insertion . . . 400
28.10 Texte sélectionné . . . 401
28.11 Défilement (scrolling) . . . 401
28.11.1 Mode d'emploi . . . 401
28.11.2 Approfondissement . . . 402
28.12 Défilement rapide (fast scrolling) . . . 403
28.13 Indices spéciaux . . . 404
28.14 Éditer le texte . . . 406
28.15 Images . . . 408
28.16 Widgets . . . 409
28.17 Tabulations . . . 411
28.18 Mécanisme undo . . . 411
28.18.1 Opération *undo* . . . 412
28.18.2 Opération *redo* . . . 413
28.18.3 Délimiter une action (séparateurs) . . . 413
28.18.4 Vider les piles . . . 413
28.18.5 Exemple . . . 414
28.19 Méthode `search` et expressions régulières . . . 415
28.20 Marqueurs . . . 416
28.21 Tags . . . 418
28.21.1 Introduction . . . 418
28.21.2 Options contrôlables par un tag . . . 419
28.21.3 Création et destruction de tags . . . 420
28.21.4 Appliquer un tag . . . 420
28.21.5 Permuter les tags dans la pile . . . 421

28.21.6 Récupérer les lieux d'application d'un tag . . . 421
28.21.7 Programmation évènementielle avec les tags . . . 423
28.21.8 Exemple . . . 425
28.22 Autres méthodes . . . 425
28.23 Widget `ScrolledText` . . . 426

29 Cadres (`Frame`) . . . 427

29.1 Introduction . . . 427
29.2 Liste des options . . . 427
29.3 Liste des méthodes . . . 427
29.4 Visuel . . . 428
29.5 Focus . . . 428
29.6 Widget survolé par la souris . . . 428

30 Cadres avec légende (`LabelFrame`) . . . 429

30.1 Introduction . . . 429
30.2 Liste des options . . . 429
30.3 Liste des méthodes . . . 429
30.4 Légende . . . 430
30.5 Visuel . . . 430
30.6 Focus . . . 431
30.7 Cadre survolé par la souris . . . 431

31 Fenêtres à pans (`PanedWindow`) . . . 433

31.1 Introduction . . . 433
31.2 Liste des options . . . 434
31.3 Liste des méthodes . . . 434
31.4 Visuel . . . 434
31.5 Widget survolé par la souris . . . 434
31.6 Redimensionnement . . . 435
31.7 Insertion . . . 435
31.8 Options d'insertion . . . 435
31.8.1 Options `after` et `before` . . . 436
31.8.2 Option `sticky` . . . 436
31.8.3 Options `width` et `height` . . . 436
31.8.4 Options `padx` et `pady` . . . 436
31.8.5 Options `hide` . . . 436
31.8.6 Options `stretch` . . . 436
31.9 Éditer les options d'insertion . . . 436
31.10 Suppression . . . 437
31.11 Autres méthodes . . . 437

32 Barres de défilement (`Scrollbar`) . . . 439

32.1 Introduction . . . 439
32.2 Mode d'emploi . . . 440
32.3 Liste des options . . . 441
32.4 Liste des méthodes . . . 441
32.5 Widgets concernés . . . 441
32.6 Visuel . . . 441

32.7 Focus . . . 442
32.8 Widget survolé par la souris . . . 442
32.9 Répétitions . . . 442
32.10 Callback . . . 442
32.11 Mécanisme du scrolling . . . 442
32.12 Autres méthodes . . . 444

33 Menus (`Menu`) . . . 445

33.1 Introduction . . . 445
33.2 Items d'un menu . . . 445
33.2.1 Types (*kinds*) . . . 445
33.2.2 Méthodes *add* . . . 446
33.2.3 Indices . . . 446
33.2.4 Indices spéciaux . . . 447
33.3 Options d'un item (*coptions*) . . . 447
33.3.1 Liste des coptions . . . 447
33.3.2 Méthodes pour éditer les coptions . . . 448
33.3.3 Visuel d'un item . . . 449
33.3.4 Icône accompagnant un item . . . 449
33.3.5 État d'un item . . . 449
33.3.6 Item survolé par la souris . . . 449
33.3.7 Callback associé à un item . . . 450
33.3.8 Choix de type case à cocher ou bouton de radio . . . 450
33.3.9 Choix de type case à cocher . . . 450
33.3.10 Choix de type bouton de radio . . . 450
33.3.11 Choix ouvrant un sous-menu . . . 450
33.3.12 Agencement des items . . . 450
33.4 Options d'un menu . . . 451
33.5 Liste des méthodes . . . 451
33.6 Callback associé à l'apparition du menu . . . 451
33.7 Menu détachable . . . 451
33.8 Visuel . . . 451
33.9 Pas d'état pour un menu . . . 452
33.10 Menu survolé par la souris . . . 452
33.11 Où placer un menu ? . . . 452
33.11.1 Cas possibles . . . 452
33.11.2 Menu posté sous forme de pop-up . . . 453
33.11.3 Sous-menu d'un menu (cascade) . . . 454
33.11.4 Menu principal d'une fenêtre . . . 454
33.11.5 Menu associé à un bouton-menu . . . 456
33.12 Autres méthodes . . . 457

34 Boutons-menu (`Menubutton`) . . . 459

34.1 Introduction . . . 459
34.2 Liste des options . . . 460
34.3 Liste des méthodes . . . 460
34.4 Légende . . . 460
34.5 Icône . . . 460
34.6 Visuel . . . 461

34.7 Focus . . . 461
34.8 État du widget . . . 461
34.9 Bouton-menu survolé par la souris . . . 462

35 Fenêtre principale et fenêtres secondaires (`Tk` et `Toplevel`) . . . 463

35.1 Introduction . . . 463
35.2 Différences entre `Tk` et `Toplevel` . . . 463
35.3 Liste des options . . . 464
35.4 Liste des méthodes . . . 464
35.5 Visuel . . . 464
35.6 Dimensions . . . 465
35.7 Dépendance . . . 466
35.8 Bloquer la fenêtre . . . 466
35.9 Focus . . . 467
35.10 Fenêtre survolée par la souris . . . 467
35.11 Menu principal . . . 467
35.12 Connaître la fenêtre d'un widget . . . 467
35.13 Cacher, rabattre, iconifier, recouvrir . . . 468
35.13.1 Introduction . . . 468
35.13.2 Cacher et rabattre . . . 468
35.13.3 Monter et descendre . . . 469
35.14 Capter tous les évènements . . . 469
35.15 Icônes . . . 471
35.16 Introspection . . . 471

36 Pop-ups pour dialogues simples (`messagebox`) . . . 473

36.1 Introduction . . . 473
36.2 Information ou alerte : *show* . . . 474
36.3 Question : *ask* . . . 474
36.4 Options . . . 475
36.4.1 Constantes . . . 475
36.4.2 Option `type` . . . 475
36.4.3 Option `default` . . . 475
36.4.4 Option `icon` . . . 476
36.4.5 Option `parent` . . . 476

37 Pop-ups pour sauvegarder ou charger (`filedialog`) . . . 477

37.1 Fonctions de dialogue . . . 477
37.2 Description des fenêtres . . . 478
37.3 Options . . . 478
37.4 Exemple . . . 479

38 Pop-up pour choisir une couleur (`colorchooser`) . . . 481

39 Programmation évènementielle . . . 483

39.1 Notion de *binding* . . . 483
39.2 Binding concernant un widget . . . 483
39.2.1 Methode `bind` . . . 483
39.2.2 Chaîne-séquence des callbacks . . . 485

39.2.3 Méthode `unbind` . . . 486
39.2.4 Dysfonctionnement de `unbind` . . . 487
39.2.4.1 Premier remède . . . 488
39.2.4.2 Deuxième remède . . . 489
39.2.4.3 Troisième remède . . . 491
39.3 Binding concernant une classe de widgets . . . 491
39.3.1 Methode `bind_class` . . . 491
39.3.2 Methode `unbind_class` . . . 492
39.4 Binding concernant toute l'application . . . 493
39.4.1 Methode `bind_all` . . . 493
39.4.2 Methode `unbind_all` . . . 494
39.5 Séquences . . . 494
39.5.1 Syntaxe générale . . . 494
39.5.2 Modificateurs . . . 494
39.5.3 Types . . . 495
39.5.4 Détails . . . 496
39.5.5 Touche régulière et focus . . . 499
39.6 Séquences virtuelles . . . 500
39.7 Objets de type `Event` . . . 502
39.7.1 Touche pressée . . . 502
39.7.2 Action de la souris . . . 502
39.7.3 Redimensionnement . . . 503
39.7.4 Évènement quelconque . . . 503
39.7.5 Exemples . . . 503
39.8 Priorités . . . 507
39.8.1 Priorité à la séquence la plus précise . . . 507
39.8.2 Priorité au widget le plus proche . . . 508
39.9 Notion de *tag binding* . . . 509
39.10 Évènements générés artificiellement . . . 511
39.11 Captation forcée des évènements . . . 511
39.12 Évènements associés à des variables de contrôle . . . 513
39.13 Gestionnaire de protocole . . . 513
39.14 Introspection . . . 515

40 Tkinter orienté objet . . . 519

40.1 Introduction . . . 519
40.2 Spécialiser un widget existant . . . 519
40.3 Créer un widget composite . . . 522
40.4 Créer une application . . . 523
40.5 Classe Tkinter . . . 523
40.6 Tkinter fonctionnel versus Tkinter orienté objet . . . 525
40.6.1 Introduction . . . 525
40.6.2 Version orientée objet . . . 526
40.6.3 Version fonctionnelle . . . 530

41 Base des options (*option database*) . . . 535

41.1 Introduction . . . 535
41.2 Noms et classes Tkinter (rappels) . . . 535
41.3 Nom *database* et classe *database* d'une option . . . 536

41.4 Spécification de style (syntaxe d'un pattern) . . . 539
41.4.1 Syntaxe d'une spécification . . . 539
41.4.2 Syntaxe d'un pattern . . . 539
41.4.3 Exemples de patterns . . . 540
41.4.4 Erreur fréquente . . . 541
41.5 Ajouter une spécification avec `option_add` . . . 541
41.6 Utiliser un fichier de spécifications . . . 542
41.7 Hiérarchie dans les spécifications . . . 543
41.8 Autres méthodes . . . 544

42 Projet : traceur . . . 545

42.1 Point de vue de l'utilisateur . . . 545
42.2 Présentation générale du programme . . . 546
42.3 Imports et fonctions . . . 549
42.4 Classe `Traceur` . . . 550
42.4.1 Initialisation . . . 550
42.4.2 Mises à jour de la fenêtre graphique . . . 552
42.4.3 Zoom . . . 555
42.4.4 Courbes $y = f(x)$ . . . 557
42.4.5 Courbe de Bézier . . . 559
42.4.6 Sauvegarder, charger et exporter . . . 564
42.4.7 Revenir au mode normal . . . 571
42.4.8 Métadonnées . . . 571
42.5 Classe `Modes` . . . 572
42.6 Classe `Homotopie` . . . 572
42.7 Classe `Front` . . . 573
42.8 Classe `Paramètres` . . . 574
42.9 Classe `Fonctions` . . . 577
42.10 Classe `Bézier` . . . 579
42.11 Problème éventuel avec `unbind` . . . 580
42.12 À faire . . . 582

43 Projet : Boids . . . 583

43.1 Introduction . . . 583
43.2 Classe `Vecteur` . . . 584
43.3 Classe `Display` . . . 586
43.4 Classe `Boid` . . . 588
43.5 Classe Population . . . 593
43.6 Lancer le programme . . . 593
43.7 Procédé d'animation . . . 594
43.8 À faire . . . 594

44 Projet : attraction universelle . . . 595

44.1 Introduction . . . 595
44.2 Classe `Vecteur` . . . 595
44.3 Classe `Display` . . . 596
44.4 Classe `Planète` . . . 598
44.5 Classe `Population` . . . 602
44.6 À faire . . . 606

44.7 Annexe : collisions . . . 607

45 Projet : chat . . . 609

45.1 Programme serveur . . . 609
45.1.1 Importations . . . 609
45.1.2 Classe `Socket` . . . 609
45.1.3 Classe `Serveur` . . . 611
45.1.4 Classe `Client` . . . 611
45.1.5 Programme principal . . . 613
45.2 Programme Client . . . 613
45.2.1 Importations . . . 613
45.2.2 Classe `Socket` . . . 613
45.2.3 Classe `Display` . . . 613
45.2.4 Classe `Récepteur` . . . 616
45.2.5 Programme principal . . . 617

46 Annexe : Curryfication . . . 619

46.1 Introduction . . . 619
46.2 Solution 1 : fonction lambda . . . 620
46.3 Solution 2 : fonction d'ordre supérieur . . . 620
46.4 Solution 3 : partialisation . . . 621
46.5 Partialisation dans le cas général . . . 622
46.6 Fonction `partial` du module `functools` . . . 623
46.7 Curryfication . . . 624

II TTK . . . 627

47 Généralités . . . 629

47.1 Introduction . . . 629
47.2 Importer TTK . . . 629
47.2.1 Importer le contenu . . . 629
47.2.2 Importer le contenant . . . 630
47.2.3 Convention . . . 631
47.3 Thèmes . . . 631
47.4 Widgets proposés par TTK . . . 632
47.5 Feuilles et options de style . . . 634
47.6 Apparence associée à un état (`map`) . . . 635
47.7 Structure d'un widget (*layout*) . . . 636
47.8 Méthodes ttk-universelles . . . 637
47.9 Options particulières . . . 637
47.9.1 Texte . . . 637
47.9.2 Images . . . 638
47.9.3 Largeur (`width`) . . . 640
47.9.4 Espacement (`padding`) . . . 641
47.9.5 Cycle du focus . . . 641
47.9.6 État . . . 641
47.9.7 Pointeur de souris . . . 642
47.10 Bizarrerie chez `Scrollbar` . . . 642

48 Introspection . . . 647

48.1 Contenu du module `tkinter.ttk` . . . 647
48.2 Taxonomie des widgets . . . 647
48.3 Méthodes universelles . . . 648
48.4 Instances de `Style` . . . 650
48.5 Options de style d'un widget . . . 652
48.6 Générer la documentation de tous les widgets . . . 655

49 Point de vue utilisateur . . . 661

49.1 Séparateur (`Separator`) . . . 661
49.2 Poignée de redimensionnement (`Sizegrip`) . . . 661
49.3 Barre de progression (`Progressbar`) . . . 661
49.4 Boîte combinée (`Combobox`) . . . 662
49.5 Calepin (`Notenook`) . . . 662
49.6 Arborescence (`Treeview`) . . . 663

50 Feuilles de style . . . 665

50.1 Introduction . . . 665
50.2 Notion de style . . . 665
50.3 Style par défaut . . . 666
50.4 Créer et éditer un style . . . 668
50.4.1 Éditer un style existant . . . 668
50.4.2 Créer un style . . . 668
50.4.3 Feuille commune . . . 669
50.4.4 Dépendance au thème . . . 669
50.4.5 Bilan . . . 670
50.5 Options de style communes . . . 671
50.5.1 Sous `"aqua"` . . . 671
50.5.2 Sous `"clam"` . . . 671
50.5.3 Sous `"alt"` . . . 671
50.5.4 Sous `"default"` . . . 671
50.5.5 Sous `"classic"` . . . 671
50.6 Interface de style . . . 671

51 États (*mapping*) . . . 673

51.1 États et drapeaux . . . 673
51.2 Spécification d'état . . . 673
51.3 Connaître l'état d'un widget . . . 674
51.4 Changer l'état d'un widget . . . 674
51.5 Callback associé à `instate` . . . 675
51.6 Lien avec l'option `state` . . . 676
51.7 Écrire des *mappings* (`map`) . . . 678
51.8 Lire les mappings écrits dans une feuille de style . . . 680
51.9 Valeur d'une option en fonction d'un état . . . 681
51.10 Méthodes . . . 681
51.10.1 Méthode `state` . . . 681
51.10.2 Méthode `instate` . . . 681
51.10.3 Méthode `map` . . . 682

51.11 Exemple 682

52 Éléments et structure (*layout*) 687

52.1 Notion de structure (*layout*) 687
52.2 Syntaxe d'une structure 689
52.3 Notre classe `Layout` 690
52.4 Créer ou éditer une structure 695
52.5 Options de style attachées à un élément 697
52.6 Trouver les options de style d'un widget 698
52.7 Générer la documentation complète des widgets 701
52.8 Éléments existant 706
52.9 Créer des éléments 709
52.9.1 À partir d'images 709
52.9.2 À partir d'éléments existant 710
52.9.3 À partir du moteur propre à WINDOWS 710
52.9.4 Bilan 710

53 Thèmes 711

53.1 Introduction 711
53.2 Choisir le thème 712
53.3 Créer un thème 712
53.3.1 Introduction 712
53.3.2 Clé `"configure"` 713
53.3.3 Clé `"map"` 713
53.3.4 Clé `"layout"` 714
53.3.5 Clé `"element_create"` 714
53.3.6 Bilan 715
53.4 Éditer un thème 716

54 Étiquettes (`Label`) 719

54.1 Options 719
54.2 Options disparues 719
54.3 Options de style 719
54.3.1 Sous `"aqua"`, `"default"` et `"classic"` 719
54.3.2 Sous `"clam"` 719
54.3.3 Sous `"alt"` 719
54.4 Méthodes 719

55 Champs de saisie (`Entry`) 721

55.1 Options 721
55.2 Options disparues 721
55.3 Options de style 721
55.3.1 Sous `"aqua"` 721
55.3.2 Sous `"clam"` 721
55.3.3 Sous `"alt"` 721
55.3.4 Sous `"default"` 722
55.3.5 Sous `"classic"` 722
55.4 Méthodes 722
55.5 Méthode `validate` 722

56 Zones de sélection numérique (`Spinbox`) . . . 725
56.1 Options . . . 725
56.2 Options disparues . . . 725
56.3 Options de style . . . 725
56.3.1 Sous `"aqua"` . . . 725
56.3.2 Sous `"clam"` . . . 725
56.3.3 Sous `"alt"` . . . 726
56.3.4 Sous `"default"` . . . 726
56.3.5 Sous `"classic"` . . . 726
56.4 Méthodes . . . 726
56.5 Description des nouvelles méthodes . . . 726
56.6 Programmation évènementielle . . . 727

57 Curseurs (`Scale`) . . . 729
57.1 Options . . . 729
57.2 Options disparues . . . 729
57.3 Deux feuilles de style . . . 729
57.4 Options de style . . . 729
57.4.1 Sous `"aqua"` . . . 729
57.4.2 Sous `"clam"` . . . 729
57.4.3 Sous `"alt"` . . . 730
57.4.4 Sous `"default"` et `"classic"` . . . 730
57.5 Méthodes . . . 730
57.6 Nouveautés . . . 730

58 Curseurs avec légende (`LabeledScale`) . . . 733
58.1 Introduction . . . 733
58.2 Mode d'emploi . . . 733
58.3 Options . . . 735
58.4 Options de style . . . 735
58.4.1 Sous `"aqua"`, `"default"` et `"classic"` . . . 735
58.4.2 Sous `"clam"` . . . 735
58.4.3 Sous `"alt"` . . . 736
58.5 Méthodes . . . 736

59 Boutons (`Button`) . . . 737
59.1 Options . . . 737
59.2 Options disparues . . . 737
59.3 Options de style . . . 737
59.3.1 Sous `"aqua"` . . . 737
59.3.2 Sous `"clam"` . . . 737
59.3.3 Sous `"alt"` . . . 738
59.3.4 Sous `"default"` . . . 738
59.3.5 Sous `"classic"` . . . 738
59.4 Méthodes . . . 738

60 Cases à cocher (`Checkbutton`) . . . 739
60.1 Options . . . 739
60.2 Options disparues . . . 739

60.3 Options de style . . . 739
60.3.1 Sous "aqua" . . . 739
60.3.2 Sous "clam" . . . 739
60.3.3 Sous "alt" . . . 740
60.3.4 Sous "default" . . . 740
60.3.5 Sous "classic" . . . 740
60.4 Méthodes . . . 740

61 Boutons de radio (Radiobutton) . . . 741

61.1 Options . . . 741
61.2 Options disparues . . . 741
61.3 Options de style . . . 741
61.3.1 Sous "aqua" . . . 741
61.3.2 Sous "clam" . . . 741
61.3.3 Sous "alt" . . . 742
61.3.4 Sous "default" . . . 742
61.3.5 Sous "classic" . . . 742
61.4 Méthodes . . . 742

62 Sélecteurs d'options (OptionMenu) . . . 743

62.1 Introduction . . . 743
62.2 Mode d'emploi . . . 743
62.3 Options . . . 744
62.4 Options disparues . . . 744
62.5 Options de style . . . 744
62.5.1 Sous "aqua" . . . 745
62.5.2 Sous "clam" . . . 745
62.5.3 Sous "alt" . . . 745
62.5.4 Sous "default" . . . 745
62.5.5 Sous "classic" . . . 745
62.6 Méthodes . . . 745

63 Cadres (Frame) . . . 747

63.1 Options . . . 747
63.2 Options disparues . . . 747
63.3 Options de style . . . 747
63.3.1 Sous "aqua", "default" et "classic" . . . 747
63.3.2 Sous "clam" . . . 747
63.3.3 Sous "alt" . . . 747
63.4 Méthodes . . . 747

64 Cadres avec légende (Labelframe) . . . 749

64.1 Options . . . 749
64.2 Options disparues . . . 749
64.3 Options de style . . . 749
64.3.1 Sous "default" et "classic" . . . 749
64.3.2 Sous "clam" . . . 749
64.3.3 Sous "alt" . . . 749
64.4 Méthodes . . . 749

65 Fenêtres à pans (`Panedwindow`) . . . 751

65.1 Options . . . 751
65.2 Options disparues . . . 751
65.3 Options de style . . . 751
65.4 Méthodes . . . 751
65.5 Options d'insertion . . . 752
65.6 Méthode `insert` . . . 752
65.7 Méthode `sashpos` . . . 752
65.8 Méthode `pane` . . . 753

66 Barres de défilement (`Scrollbar`) . . . 755

66.1 Options . . . 756
66.2 Options disparues . . . 756
66.3 Options de style . . . 757
66.3.1 Sous `"aqua"` et `"default"` . . . 757
66.3.2 Sous `"clam"` . . . 757
66.3.3 Sous `"alt"` . . . 757
66.3.4 Sous `"classic"` . . . 757
66.4 Méthodes . . . 757

67 Boutons-menus (`Menubutton`) . . . 759

67.1 Introduction . . . 759
67.2 Options . . . 759
67.3 Options disparues . . . 759
67.4 Options de style . . . 759
67.4.1 Sous `"aqua"` . . . 759
67.4.2 Sous `"clam"` . . . 760
67.4.3 Sous `"alt"` . . . 760
67.4.4 Sous `"default"` . . . 760
67.4.5 Sous `"classic"` . . . 760
67.5 Méthodes . . . 760

68 Séparateurs (`Separator`) . . . 761

68.1 Introduction . . . 761
68.2 Liste des options . . . 761
68.3 Visuel . . . 761
68.4 Options de style . . . 761
68.5 Méthodes . . . 761

69 Poignées (`Sizegrip`) . . . 763

69.1 Introduction . . . 763
69.2 Liste des options . . . 764
69.3 Style . . . 764
69.4 Méthodes . . . 764

70 Barres de progression (`Progressbar`) . . . 765

70.1 Introduction . . . 765
70.2 Liste des options . . . 766

70.3 Visuel . . . 766
70.4 Valeur de la progression . . . 766
70.5 Modes *determinate* versus *indeterminate* . . . 766
70.6 Trois types de progression . . . 767
70.6.1 Barre absolue . . . 767
70.6.2 Barre non absolue . . . 768
70.6.3 Barre va-et-vient . . . 769
70.7 Variable associée . . . 769
70.8 Méthodes . . . 770
70.8.1 Méthode `step` . . . 770
70.8.2 Méthode `start` . . . 770
70.8.3 Méthode `stop` . . . 771
70.9 Style . . . 771
70.9.1 Sous `"aqua"` . . . 771
70.9.2 Sous `"clam"` . . . 771
70.9.3 Sous `"alt"` . . . 771
70.9.4 Sous `"default"` et `"classic"` . . . 771
70.10 Exemple complet . . . 771

71 Boîtes combinées (`Combobox`) . . . 775

71.1 Mode d'emploi . . . 775
71.2 Liste des options . . . 776
71.3 Méthodes . . . 776
71.4 Visuel . . . 776
71.5 Défilement (scrolling) . . . 777
71.6 Procédure de validation . . . 777
71.7 État . . . 777
71.8 Description des méthodes . . . 777
71.8.1 Méthodes héritées de `Entry` . . . 777
71.8.2 Autres méthodes . . . 778
71.9 Options de style . . . 778
71.9.1 Sous `"aqua"` . . . 778
71.9.2 Sous `"clam"` . . . 778
71.9.3 Sous `"alt"` . . . 778
71.9.4 Sous `"default"` . . . 779
71.9.5 Sous `"classic"` . . . 779
71.10 Programmation évènementielle . . . 779

72 Calepins (`Notebook`) . . . 781

72.1 Mode d'emploi . . . 781
72.2 Liste des options . . . 782
72.3 Visuel . . . 782
72.4 Créer un onglet . . . 782
72.4.1 Options . . . 782
72.4.2 Ajouter . . . 782
72.4.3 Insérer . . . 783
72.4.4 Récupérer la liste des onglets . . . 783
72.5 Indices spéciaux . . . 783
72.6 Éditer les options d'un onglet . . . 784

72.7 Suppression . . . 784
72.8 Raccourcis . . . 785
72.9 Options de style . . . 785
72.9.1 Sous "clam" . . . 785
72.9.2 Sous "aqua", "default" et "classic" . . . 785
72.10 Méthodes spécifiques . . . 785
72.11 Programmation évènementielle . . . 785

73 Arborescences (Treeview) . . . 787

73.1 Mode d'emploi . . . 787
73.2 Liste des options . . . 788
73.3 Liste des méthodes spécifiques . . . 788
73.4 Visuel . . . 788
73.5 Défilement (scrolling) . . . 789
73.6 Options de style . . . 789
73.6.1 Sous "aqua", "default" et "classic" . . . 789
73.6.2 Sous "clam" . . . 790
73.6.3 Sous "alt" . . . 790
73.7 Sélection d'items . . . 790
73.8 Focus . . . 791
73.9 Indices des items . . . 791
73.10 Éditer l'arbre . . . 792
73.10.1 Options d'un item . . . 792
73.10.2 Insérer . . . 792
73.10.3 Déplacer . . . 792
73.10.4 Cacher . . . 793
73.10.5 Effacer . . . 793
73.10.6 Modifier les enfants . . . 793
73.11 Éditer les options d'un item . . . 793
73.12 Éditer les attributs d'un item . . . 794
73.13 Éditer les options d'une colonne . . . 794
73.13.1 Options d'une colonne . . . 794
73.13.2 Nom d'une colonne . . . 794
73.13.3 Édition . . . 795
73.14 Éditer l'entête d'une colonne . . . 795
73.15 Lire la structure de l'arbre . . . 796
73.16 Tags . . . 796
73.16.1 Options d'un groupe . . . 797
73.16.2 Édition des t-options . . . 797
73.16.3 Binding . . . 797
73.16.4 Connaître les membres d'un groupe . . . 798
73.16.5 Exemple . . . 798
73.17 Coordonnées . . . 799
73.17.1 Parties . . . 799
73.17.2 Lignes . . . 799
73.17.3 Colonnes . . . 800
73.17.4 Éléments . . . 800
73.17.5 Méthode bbox . . . 801
73.18 Programmation évènementielle . . . 801

Bibliographie 803

Index 805

Partie I

Tkinter

Chapitre 1
Généralités

1.1 Origines de Tkinter

Le module Tkinter est une caisse à outils permettant de créer des interfaces graphiques (*graphical user interfaces*, en anglais, ou plus simplement GUI).

Pour raconter l'histoire de Tkinter, il faut remonter à la fin des années 80. En 1988, John Ousterhout invente le langage Tcl (Tool Command Language). En 1990, il développe une extension de ce langage appelée Tk. Il s'agit d'une bibliothèque permettant de créer des interfaces graphiques (*tool kit* signifie boîte à outils en anglais). Depuis, une importante communauté de développeurs utilise le langage Tcl-Tk.

Plus tard, Steen Lumholt et Guido van Rossum (à qui l'on doit Python) créent le module Tkinter (Tk interface). Il s'agit d'une adaptation de Tk pour Python (un enrobage). La documentation officielle [4] cite également Fredrik Lundh pour sa contribution (les noms sont cités à la section 24.1.2 de la documentation de la vieille version 2.7).

Aujourd'hui, Tkinter est l'interface graphique standard de Python.



Figure 1.1. Autoportrait tramé de John Ousterhout, inventeur de Tcl et Tk, illustrant l'option `dither` de la commande `canvas` de Tk. Source : Wikipedia

Il n'existe pas de documentation complète sur Tkinter et malheureusement on doit parfois aller chercher des informations dans la documentation de Tcl-Tk [9] pour ensuite les traduire en langage Python-Tkinter (les syntaxes sont légèrement différentes). Heureusement pour nous, cela arrive rarement. Normalement, nous n'aurons pas besoin de connaître Tcl-Tk pour apprendre à utiliser Tkinter correctement.

Note. La documentation officielle (mais incomplète) de Tkinter se trouve sur le site officiel de Python, voir [3] et [4].

1.2 Grands principes

Tkinter permet de créer un environnement graphique dans lequel un utilisateur peut, de manière agréable et intuitive, entrer des données, cocher des options, déclencher des calculs, dialoguer, admirer des animations, etc. On peut grâce à ses outils programmer un jeu vidéo, réaliser une simulation graphique et bien d'autres choses encore. Les programmes ainsi créés ressemblent (plus ou moins) aux applications habituellement utilisées sur un ordinateur.

Un programme écrit avec Tkinter crée une *fenêtre principale* (*main window* ou *root window*, en anglais) et éventuellement des fenêtres secondaires (*toplevel window* ou *pop-up windows*). La fenêtre principale est une instance de `Tk`. Les fenêtres secondaires sont des instances de `Toplevel`. On munit bien sûr les fenêtres de plusieurs *widgets* (contraction, semblerait-t-il de *window* et *gadget*). Tkinter offre toutes sortes de widgets : zones de saisie, zones de sélection numérique, boutons, zones de listes, sélecteurs d'options, canevas, etc. Les widgets sont les éléments de base d'une interface graphique.

Notes.

1. Les widgets sont aussi appelés *composants d'interface graphique* ou encore *éléments visuels d'interface graphique*.
2. Les fenêtres sont elles-mêmes des widgets. C'est ainsi, en tout cas, que nous comprenons le mot widget. Par conséquent, quand nous écrivons : « les widgets », nous sous-entendons la plupart du temps « les widgets et les fenêtres ». Dans la documentation Tkinter et de nombreux manuels écrits en anglais, il arrive que le mot *window* désigne à la fois fenêtre et widget. On retrouve cela dans le « langage Tkinter » lui-même. En effet, pour insérer un widget dans un canevas, par exemple, on utilise la méthode `create_window`, et l'item ainsi créé est de type `"window"` (chapitre 27). Quand la documentation parle d'une fenêtre (pas un widget), elle dit « *toplevel window* ». C'est dans ces termes que [4] décrit la fenêtre principale (instance de `Tk`) : « `root` is a toplevel window ».

1.3 Importer Tkinter

Il recommandé de ne pas importer le contenu de `tkinter` mais le module lui-même, afin d'éviter des problèmes d'écrasement et de saturation de l'espace global. De plus, il est d'usage de donner l'alias `tk` à `tkinter` :

```
import tkinter as tk
racine = tk.Tk()
(...)
racine.mainloop()
```

ainsi, tout ce qui vient de `tkinter` porte un nom qualifié commençant par `tk.` Dans le présent document, en revanche, nous ne suivrons presque jamais cette consigne pour des raisons de confort de lecture. Nous importerons « bestialement » le contenu du module :

```
from tkinter import *
```

On peut consulter `TclVersion` et `TkVersion` pour savoir quelle est notre version de Tkinter :

```
>>> from tkinter import TclVersion, TkVersion
>>> TclVersion
8.6
>>> TkVersion
8.6
```

Si on veut connaître la version dans le format *majeur.mineur.correctif* (en anglais *major.minor.patch*), on procède à la manipulation ci-dessous :

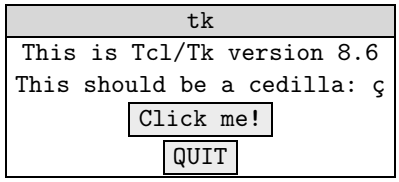
```
>>> racine = Tk()
>>> racine.call("info", "patchlevel")
'8.6.8'
```

(On explique la méthode `call` au chapitre 2.)

La commande « `python3 -m tkinter` » exécutée à partir du `Terminal` de MAC OS, par exemple, permet également de connaître la version utilisée :

```
$ python3 -m tkinter
```

Cette dernière déclenche la création d'une fenêtre ressemblant à ceci :



1.4 Fenêtre et boucle principales

Nous avons dit que la fenêtre principale est une instance de `Tk`. Pour créer un environnement graphique, nous commençons donc par instancier `Tk` :

```
>>> from tkinter import *
>>> racine = Tk()
```

La deuxième instruction fait apparaître une fenêtre ressemblant plus ou moins à ceci :

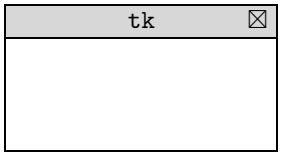


Figure 1.2. La fenêtre principale

Tout programme Tkinter suit le schéma suivant : on importe le module `tkinter`, on instancie `Tk`, on instancie des widgets, on les *place* et on lance la *boucle principale* grâce à la méthode `mainloop` :

```
from tkinter import *
racine = Tk()
(...)
racine.mainloop()
```

Attention. On n'instancie la classe `Tk` qu'une fois.

La boucle principale consiste en ceci : tant que la fenêtre principale n'est pas fermée (détruite), Tkinter est en écoute, ce qui signifie qu'il attend qu'un évènement (au clavier ou à la souris) advienne pour réagir en fonction de qui est écrit dans le programme. On parle ici de *programmation évènementielle*.

Les lignes de code écrites éventuellement après l'appel de `mainloop` sont exécutées lorsque la fenêtre principale est fermée (ou lorsque la méthode `quit` est appelée).

Tous les widgets possèdent des méthodes `mainloop` et `quit`. Nous décrivons ces dernières ci-dessous, sur un widget `w` quelconque.

`w.mainloop()` :

- effet : une fois créés les widgets, on appelle `w.mainloop` afin de lancer la boucle principale. La plupart du temps, `w` est la fenêtre principale, mais cela n'est pas obligatoire. Il est par ailleurs possible d'appeler `mainloop` à l'intérieur d'un gestionnaire d'évènements pour relancer la boucle principale (voir chapitre 39).

`w.quit()` :

- effet : le programme sort de la boucle principale. La plupart du temps, `w` est la fenêtre principale, mais cela n'est pas obligatoire.

Note. Si le programme a lancé `mainloop`, tant que la fenêtre principale n'est pas fermée (ou que `quit` n'est pas appelée), nous n'avons pas la main sur le prompteur (shell). Impossible donc d'afficher manuellement le contenu d'une variable du programme, par exemple. Si on veut pouvoir interroger Python pendant que la fenêtre principale est ouverte, il ne faut pas appeller `mainloop` : c'est ce que nous faisons pendant la phase de débogage de nos programmes. Une fois achevé le débogage, on restaure l'appel de `mainloop`, bien entendu.

Nous testons ces méthodes à la section suivante.

1.5 Widgets conteneurs

Un widget *conteneur* a pour vocation de recevoir des widgets. Les widgets conteneurs sont les fenêtres (principale ou secondaires), les cadres (`Frame` et `Labelframe`), les fenêtres à pans (`PanedWindow`) et d'une certaine manière, les canevas (`Canvas`), ainsi que les zones de texte (`Text`).

On utilise les widgets conteneurs pour organiser l'espace offert par les fenêtres.

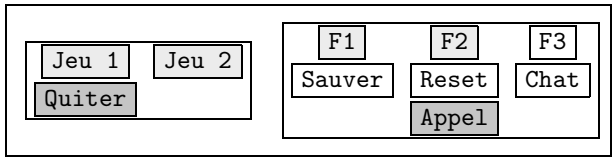


Figure 1.3. Deux groupes de widgets dans une fenêtre (chaque groupe est dans un cadre)

Nous reviendrons sur la notion de conteneur (*master*, *children*, etc.) à la section 1.12. En attendant, nous retiendrons que `Tk`, `Toplevel`, `Frame`, `Labelframe`, `PanedWindow`, `Canvas` et `Text` sont des types de widgets conteneurs.

1.6 Instancier un widget

Les widgets sont les objets graphiques que l'on insère dans la fenêtre principale ou les fenêtres secondaires : affichages, champs de saisie, menus, boutons, etc. L'instanciation se fait en appelant le type désiré.

La signature d'une classe quelconque de widgets est

```
(master=None, cnf={}, *, name=None, **kwargs)
```

où :

- `master` reçoit le nom du conteneur dans lequel est inséré le widget ;
- `cnf` reçoit un dictionnaire d'options (*configuration options*) ;
- `name` reçoit une chaîne de caractères (nom Tkinter, sous-section 1.9.1) ;
- `**kwargs` reçoit les options et leurs valeurs (chaque option est un paramètre avec nom obligatoire).

La plupart du temps, on utilise `**kwargs` pour faire passer des options et non pas `cnf` (dans ce livre nous n'utilisons jamais `cnf`).

Voici différentes manières d'instancier `Button`, par exemple. La manière la plus courante :

```
b = Button(parent, opt1=v1, opt2=v2, etc.)
```

(où `parent` désigne un conteneur).

Certains préfèrent la manière ci-dessous :

```
b = Button(master=parent, opt1=v1, opt2=v2, etc.)
```

ou alors carrément

```
b = Button(opt1=v1, opt2=v2, etc.)
```

ce qui sous-entend que le père est la fenêtre principale (le père par défaut est la fenêtre principale).

On peut reprendre ces 3 manières en y ajoutant le paramètre `name`.

Voici un exemple plus précis (et réel) :

```
b = Button(racine, text="Reeds", bg="red", fg="black", name="b")
```

Voici la liste des widgets offerts par Tkinter : `Tk`, `Toplevel`, `Label`, `Message`, `Entry`, `Spinbox`, `Scale`, `Button`, `Checkbutton`, `Radiobutton`, `Listbox`, `OptionMenu`, `Canvas`, `Text`, `Frame`, `LabelFrame`, `PanedWindow`, `Scrollbar`, `Menu` et `Menubutton`.

Note. Nous incluons `Tk` et `Toplevel` dans la liste des widgets, même si cela peut paraître abusif.

Attention. Il existe deux exceptions à ce que nous avons dit :

- l'instanciation de `Tk` se fait sans argument ;
- l'instanciation de `OptionMenu` se fait en passant obligatoirement le parent, une variable de contrôle et au moins une valeur.

Le cas de `OptionMenu` est un petit caillou dans la chaussure de l'introspection. En effet, tous les programmes où on écrit `w()` déclencheront une erreur pour `w` égal à `OptionMenu`. Nous verrons au chapitre 26 que la signature de cette classe est `(master, variable, value, *values, command=None)`.

1.7 Placer un widget

Une fois instancié, nous devons *placer* notre widget pour qu'il apparaisse à l'écran. Ceci peut être fait avec la méthode `grid`, par exemple :

```
from tkinter import *
racine = Tk()
b = Button(racine, text="Mon beau bouton", bg="red")
b.grid()
racine.mainloop()
```

Nous consacrons le chapitre 13 à cette méthode.

La méthode `grid` n'est pas le seul *gestionnaire de position*, il en existe d'autres comme `pack` et `place` (dans ce livre, c'est `grid` que nous préférons).

Note. Quand on oublie de placer un widget, celui-ci existe en tant qu'objet dans la mémoire mais Tkinter ne le représente pas graphiquement à l'écran. Pour l'utilisateur, c'est comme s'il n'existait pas. Il s'agit d'une erreur courante quand on débute.

1.8 Détruire un widget

Tout widget possède une méthode `destroy`.

`w.destroy()` :

- effet : destruction du widget `w` et ses descendants.

Le programme ci-dessous place trois boutons dans la fenêtre principale nommée `racine`. Le bouton QUIT déclenche `racine.quit()`, le bouton DESTROY RACINE déclenche `racine.destroy()` et le bouton DESTROY BOUTON déclenche `b2.destroy()` :

```
from tkinter import *
racine = Tk()
```

```
print("Affichage commandé avant l'appel de mainloop")
b1 = Button(racine, text="QUIT", command=racine.quit)
b1.grid()
b2 = Button(racine, text="DESTROY RACINE", command=racine.destroy)
b2.grid()
b3 = Button(racine, text="DESTROY BOUTON", command=b2.destroy)
b3.grid()
racine.mainloop()
print("Affichage commandé après l'appel de mainloop")
```

Une fois le programme lancé, il faut soit fermer la fenêtre principale (en cliquant sur ☒), soit cliquer sur `b1` ou `b2` pour pouvoir récupérer la main dans le shell. Le cas échéant, le programme affiche la chaîne `"Affichage commandé après l'appel de mainloop"`.

Tout widget possède une méthode `winfo_exists`.

`w.winfo_exists()` :

– effet : retourne `1` si le widget `w` existe toujours (ie n'a pas été détruit). Retourne `0` dans le cas contraire.

Ne pas perdre de vue que même si on a appelé `w.destroy()`, le nom `w` pointe vers un objet en mémoire. Si on lance le programme précédent, que l'on clique sur DESTROY BOUTON puis sur QUIT, on obtient les réponses suivantes dans le shell :

```
>>> b1.winfo_exists()
1
>>> b2.winfo_exists()
0
```

1.9 Nom Tkinter d'un widget

1.9.1 Nom

Tout widget (hormis la fenêtre principale) possède un attribut `_name` contenant une chaîne de caractères : il s'agit du *nom Tkinter* du widget.

```
>>> from tkinter import *
>>> racine = Tk()
>>> a = Button(racine, text="Pilar")
>>> a.grid()
>>> a._name
'!button'
```

Nous voyons que le nom Tkinter du bouton `a` est la chaîne `"!button"`.