Cours d'algorithmique et programmation 2

Anne Parrain

UFR des Sciences
Licence Sciences et Technologie
mentions Mathématiques et Informatique
Semestre 2

Section 1

Les interfaces graphiques avec tkinter

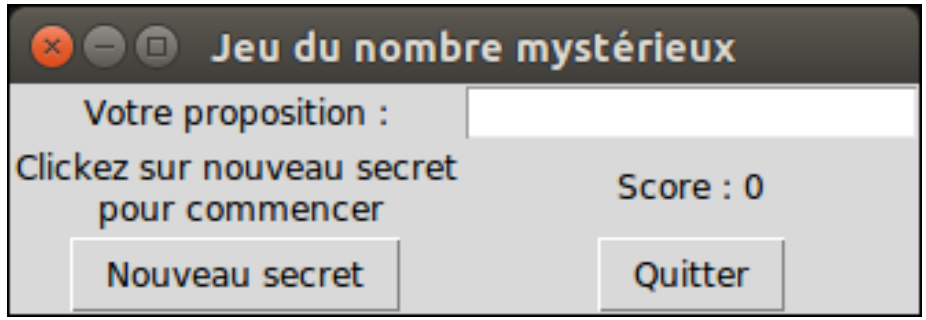
Les librairies graphiques

Nécessité d'utiliser des librairies spécialisées :

- Tkinter
- PyGtk
- PyQt
- pygame
- ...

Nous utiliserons **Tkinter** qui est la librairie graphique usuelle pour Python et qui est libre et multi-plateforme.

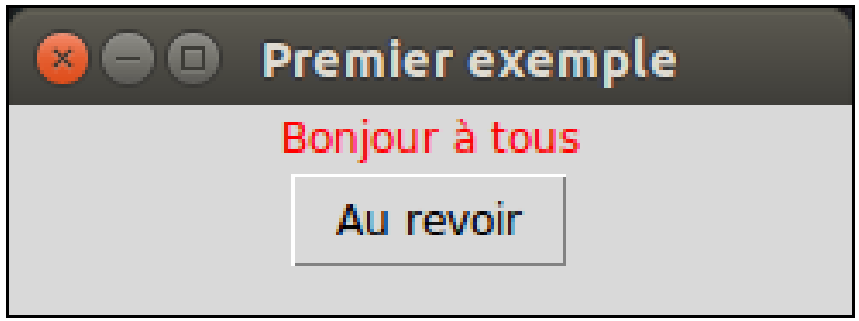
Les applications avec interface graphique



Dans une application graphique, on trouve au moins :

- la fenêtre principale : c'est le conteneur dans lequel se déroule l'application
- des composants graphiques élémentaires :
  - des étiquettes
  - des boutons
  - des zones de saisie . . .

Un premier exemple



Et voici le code en python :

`src/exemple1.py`

Un deuxième exemple

Et voici le code en python :

`src/exemple2.py`

Concevoir une application graphique

Concevoir une application graphique c'est :

- concevoir l'application, sans se préoccuper de ce qui concerne les interactions avec l'utilisateur (c'est ce qu'on appelera le modèle de l'application)
- concevoir la partie graphique indépendamment de l'application :
  - dessiner la maquette
  - créer les composants et les conteneurs
  - disposer les composants sur les conteneurs

  C'est ce qu'on appelera la partie vue de l'application.
- concevoir les interactions entre le modèle et la vue : c'est la partie contrôle de l'application qui va gérer les événements (les actions de l'utilisateur).

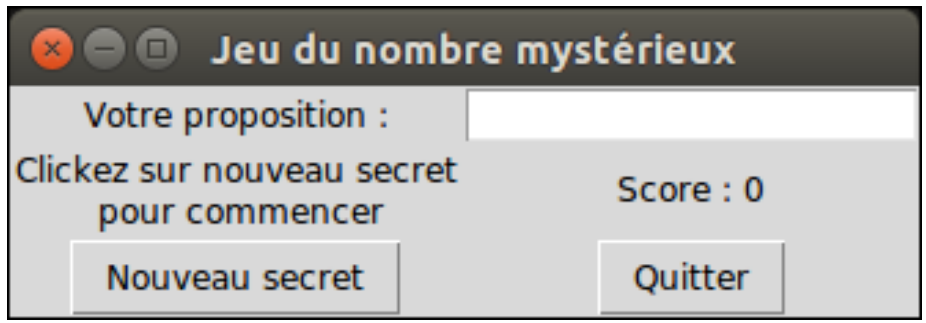
Cette conception séparée de l'interface graphique et du fonctionnement de l'application est le

modèle MVC (Modèle - Vue - Contrôleur)

Section 2

Les composants graphiques élémentaires

Concevoir la partie graphique d'une application



Besoins en terme de composants graphiques :

- la fenêtre principale
- une zone de saisie
- trois étiquettes
- deux boutons

Les conteneurs

- Un conteneur (ou container) est un composant graphique qui peut contenir d'autres conteneurs, ou des composants graphiques élémentaires. On trouve essentiellement dans cette catégorie des `Tk`, des `Toplevel` et des `Frame`.
- La fenêtre principale qui contient une application est toujours un composant `Tk`.
- Les `Toplevel` ouvrent des fenêtres indépendantes.

Exemples

- `fen = Tk()` : pour créer la fenêtre principale de l'application
- `fen.title("Mon application")` : pour ajouter un titre sur l'application
- `fen.mainloop()` : pour écouter les actions de l'utilisateur
- `fen.destroy()` : pour quitter l'application graphique.

Les composants graphiques élémentaires

- Chaque composant est construit en indiquant le conteneur auquel il appartient.
- Il possède un certain nombre de caractéristiques qui peuvent être modifiées.

Les étiquettes - Label

- Les étiquettes ou Label : des zones de texte (ou d'image) non modifiables par l'utilisateur
- Principales propriétés : le texte (text), ou l'image (image), la couleur de la police (fg), la couleur du fond (bg), ...

```
1 lbl_mess = Label(fen,text="Bonjour",fg="red",
2                   bg="light grey")
```

- Pour les modifier, deux possibilités :

```
1 lbl_mess.config(fg="black",text="Bye")
```

ou bien :

```
1 lbl_mess['text'] = "Salut !"
2 lbl_mess['fg'] = "white"
```

Les boutons - Button - et les zones de saisie - Entry

- Les boutons ou Button : destinés à être cliqués par l'utilisateur. On retrouve les mêmes propriétés élémentaires plus command

```
1 btn_quitter = Button(fen,text="Bye",bg="yellow",
2                       command=fen.destroy)
```

- Les zones de saisie ou Entry : pour que l'utilisateur fournisse des informations. Possède des méthodes spécifiques pour récupérer le texte (get) ou effacer le contenu de la zone (delete)

```
1 zn_saisie = Entry(fen,fg="green", bg="light grey")
2 zn_saisie.get()
3     # retourne le texte saisi par l'utilisateur
4 zn_saisie.delete(0,len(znSaisie.get()))
5     # efface le contenu de la zone de saisie
```

Et voici le code en python :

src/exemple21.py

Les boutons de choix - Radiobutton

- Les boutons de choix `Radiobutton` : pour un choix exclusif entre plusieurs éléments.
- Ils partagent une propriété commune `variable` qui contient la valeur de l'élément sélectionné.
- Cette propriété est d'un type `IntVar`, ou `StringVar`, ou `BooleanVar` suivant le type de la valeur souhaitée

```
1  radio_var = IntVar()
2  rad_bleu = Radiobutton(fen, text="Bleu",
3                         variable=radio_var, value=1)
4  rad_rouge = Radiobutton(fen, text="Rouge",
5                          variable=radio_var, value=2)
6  radio_var.get()
7    # retourne le bouton de choix selectionne (1 ou 2)
8  radio_var.set(2)
9    # selectionne le choix rouge
10 rad_bleu.pack()
11 rad_rouge.pack()
```

Et voici le code en python : `src/exemple22.py`

Les cases à cocher - Checkbutton

- les cases à cocher (check button) : pour des choix multiples
- chaque case a sa propre propriété `variable`

```
1  choix_entree = BooleanVar()
2  chk_entree = Checkbutton(fen,text="Entree",
3                           variable=choix_entree)
4  choix_plat = BooleanVar()
5  chk_plat = Checkbutton(fen,text="Plat",
6                         variable=choix_plat)
7  choix_dessert = BooleanVar()
8  chk_dessert = Checkbutton(fen,text="Dessert",
9                            variable=choix_dessert)
10 chk_entree.pack()
11 chk_plat.pack()
12 chk_dessert.pack()
```

Et voici le code en python : `src/exemple23.py`

Section 3

Positionner les composants graphiques

Poser des composants sur des conteneurs

Trois méthodes (on parle de *gestionnaires de positionnement*) co-existent :

- méthode `pack()` : pour poser les éléments les uns après les autres, avec des indications à gauche, à droite, en haut, en bas
- méthode `grid()` : pour positionner les éléments dans une grille
  - placement régulier
  - fusion de cellules possibles
  - ajustements pour les contenus plus petits
  - méthode très souple !
- méthode `place()` : pour positionner de manière absolue (en utilisant un système de coordonnées) les composants. À éviter le plus possible...

Attention !

- tous les composants d'un même conteneur doivent être posés avec la même méthode de positionnement
- mais on peut tout combiner en utilisant plusieurs conteneurs !

Poser des éléments avec `pack()` (1)

Le code en python : `src/exemplePack1.py`

Poser des éléments avec `pack()` (2)

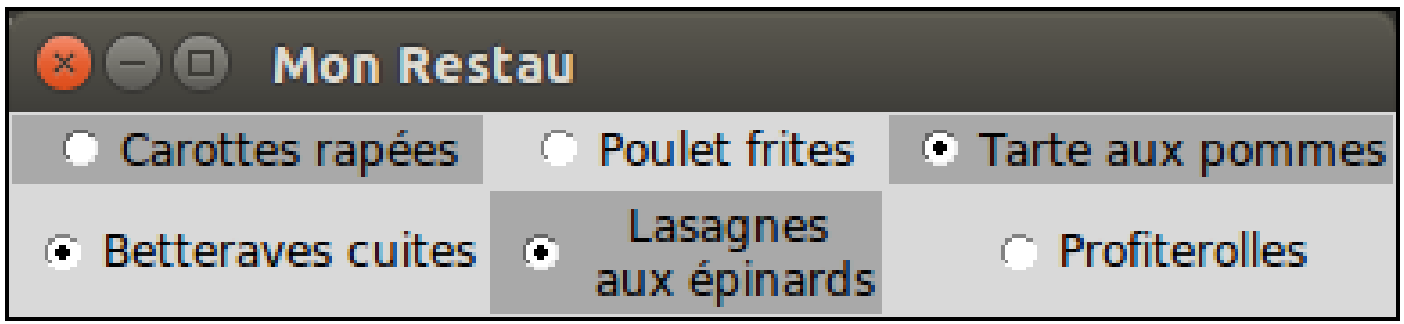
Le code en python : `src/exemplePack2.py`

Poser des éléments avec `pack()` et avec deux images (3)

Le code en python : `src/exemplePack3.py`

Poser des éléments avec `grid()`

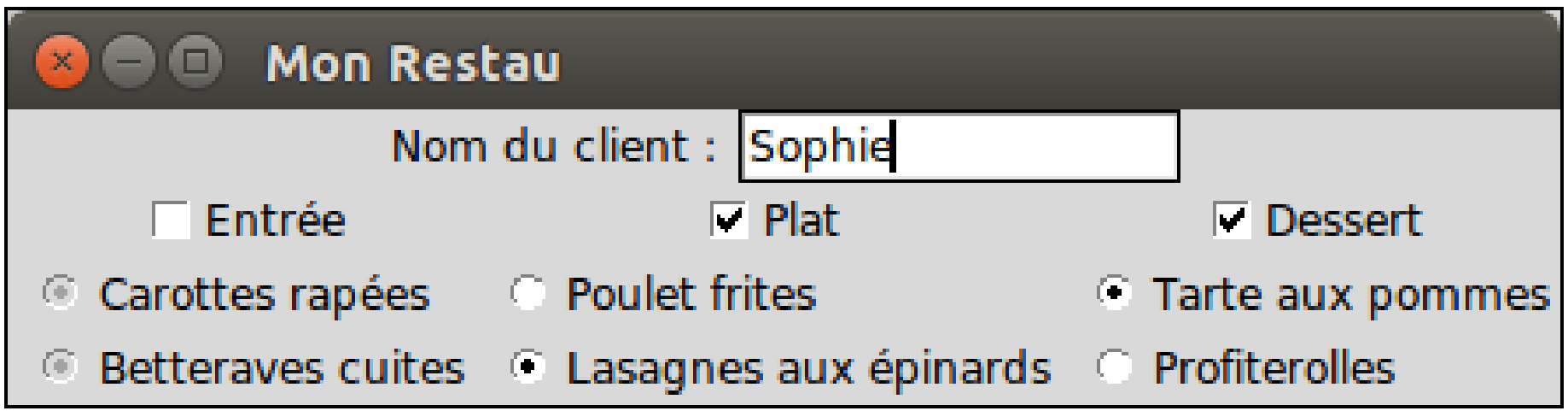
- le gestionnaire de positionnement en grille découpe le conteneur en cellules
- on indique dans quelle ligne (`row`) et quelle colonne (`column`) on pose l'élément
- les lignes sont numérotées de haut en bas, les colonnes de gauche à droite



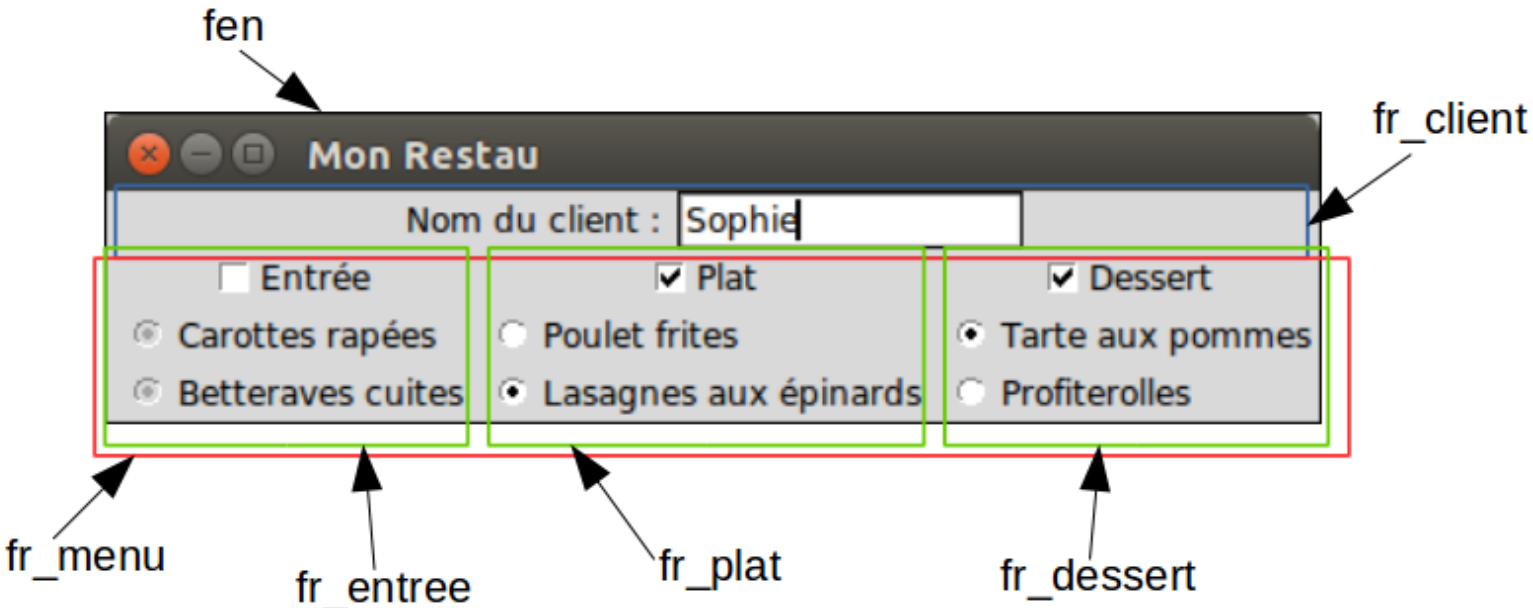
Le code en python : `src/exempleGrid1.py`

Utiliser plusieurs conteneurs

- Un gestionnaire de positionnement par conteneur
- Utiliser plusieurs conteneurs permet de diversifier et de maîtriser le positionnement des éléments
- Les conteneurs intermédiaires sont des éléments de type `Frame`



Utiliser plusieurs conteneurs



Le code en python : `src/restaurant.py`

Section 4

Gérer les événements

Gérer les événements

Les événements sont (souvent) les actions de l'utilisateur :

- cliquer sur un bouton
- choisir un élément dans un menu
- choisir un élément dans une liste déroulante
- cocher une case ou un bouton de choix
- appuyer sur une touche (`'a'`, `'b'`, `<Return>`, ...)
- déplacer la souris
- presser le bouton de la souris
- relâcher le bouton de la souris

Ces événements se produisent sur des composants. Pour chaque composant, on peut associer un événement particulier à une action particulière.

L'attribut `command` de certains composants (1)

Certains composants ont un événement privilégié.
Par exemple, ces composants attendent un click :

- un bouton (Button)
- un bouton de choix (Radiobutton)
- une case à cocher (Checkbutton)

Ils possèdent un attribut `command` qui attend le nom de la fonction à exécuter. Cette fonction ne doit attendre aucun paramètre (ou être une méthode sans autre paramètre que l'objet sur lequel elle s'applique).

```
1 btn_quitter = Button(fen,text="Bye",
2                       command = fenetre.destroy)
```

Exemples :

- `src/exemple21.py`
- `src/exemple3.py`

L'attribut command de certains composants (2)

```
1  # dans def __init__() de la classe VueCommande
2      ...
3      self.__choix_entree = tkinter.BooleanVar()
4      chk_entree = tkinter.Checkbutton(fr_entree,text="Entre
5                                       variable=self.__choix_entree,
6                                       command=self.choisit_entree)
7      ...
8      self.__les_entrees = [rad_carottes, rad_betteraves]
9      ...
10
11 def choisit_entree(self):
12     if not self.__choix_entree.get() :
13         for radio in self.__les_entrees :
14             radio.deselect()
15             radio['state'] = "disable"
16     else :
17         for radio in self.__les_entrees :
18             radio['state'] = "normal"
```

Le code en python : src/restaurantEvt.py