Interfaces graphiques

Développeur Python

Sommaire

1. Généralités.
2. Les différents types de widgets
3. Positionnement des widgets.
4. Les évènements.

1. Généralités.

1. Généralités.

CLI : Command Line Interface

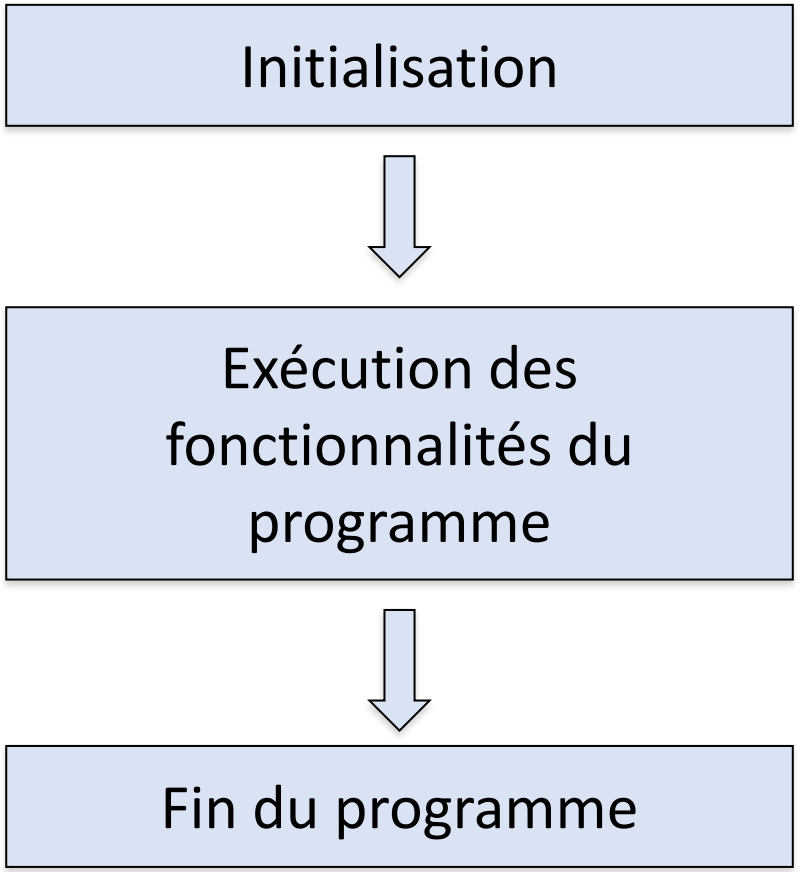
- En mode CLI, on dit que le programme pilote les événements.

- Même si de l'interactivité est possible, elle est à l'initiative du programme.

- C'est ce dernier qui impose en particulier l'ordre des saisies de l'utilisateur.

1. Généralités.

CLI : représentation imagée



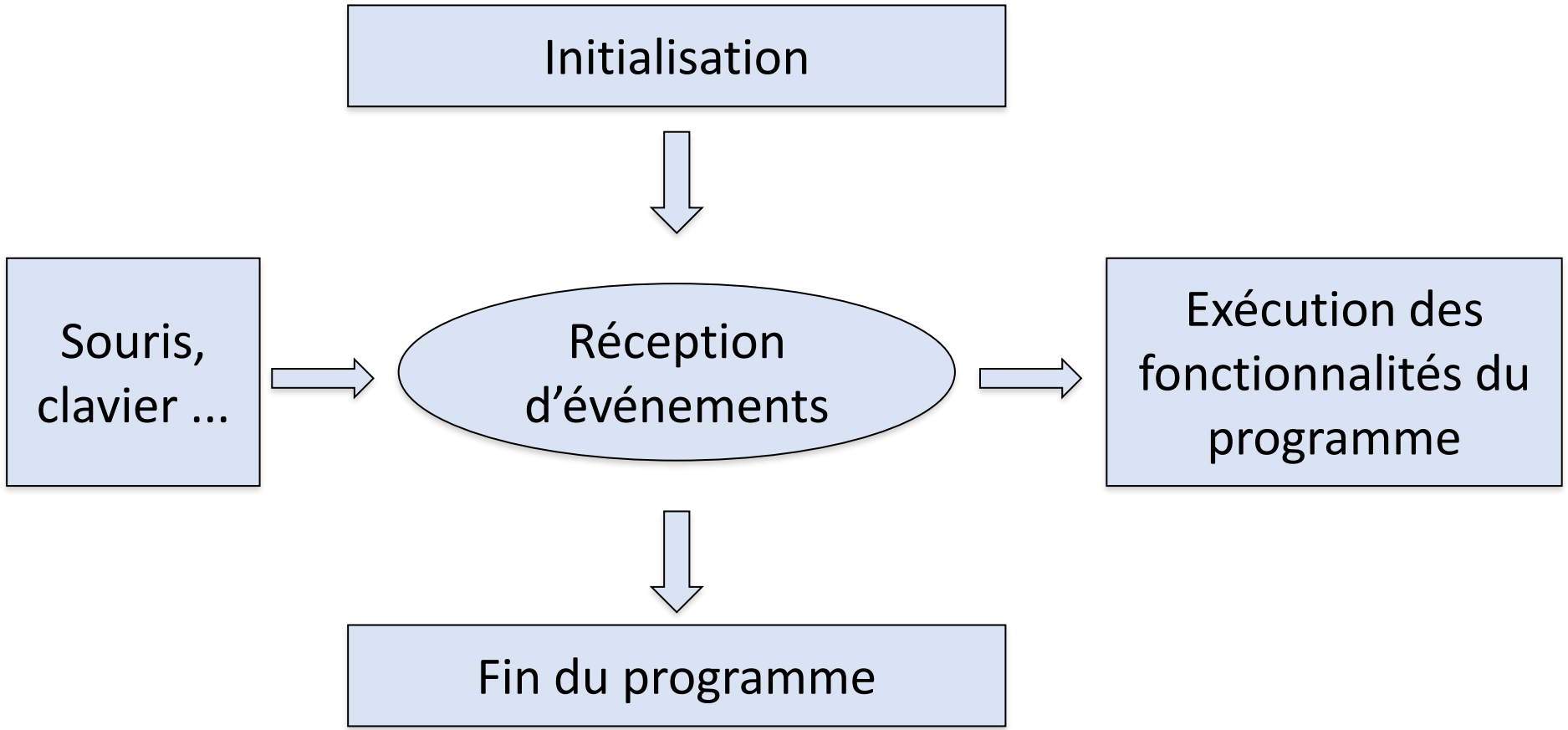
1. Généralités.

GUI : Graphic User Interface

- En mode GUI, on dit que le programme est piloté par les événements.
- La phase de réception des événements est permanente, et traite ceux-ci selon leurs ordres d'apparition.
- Selon la nature de ces événements, telle ou telle fonctionnalité du programme est exécutée.
- Cette fois-ci l'ordre des exécutions des fonctionnalités est laissée à l'utilisateur.

1. Généralités.

GUI : représentation imagée

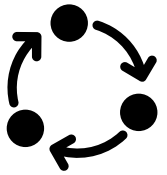


1. Généralités.

GUI : boucle “de jeu“

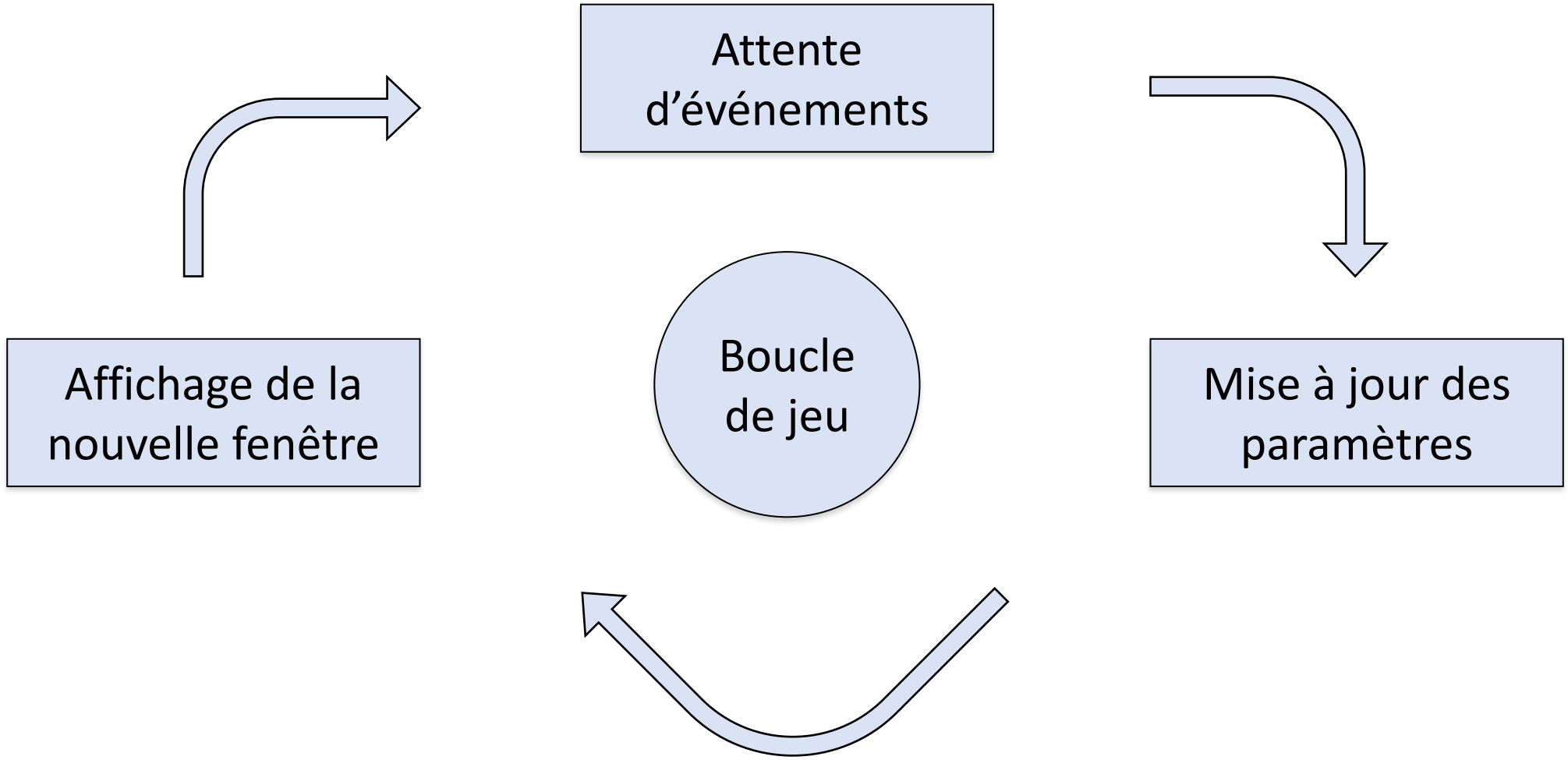
- La phase de réception des événements est implémentée avec une boucle qui “tourne” tant qu’aucun événement ne se produit.

- Il va sans dire que la réception d’événements conduira à une mise à jour de la fenêtre graphique.



1. Généralités.

GUI : autre représentation

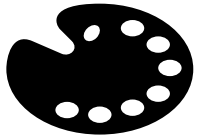


1. Généralités.

Librairie Tkinter

- Permet de manipuler en Python les widgets de la librairie Tk.

- Permet de manipuler ces widgets sans avoir recours (généralement) au langage Tcl pour lequel ils étaient initialement destinés.



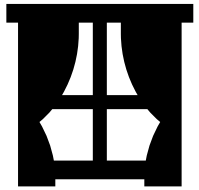
1. Généralités.

Tkinter : notion de fenêtre

- Pour rappel une fenêtre est une zone rectangulaire de l'écran dédiée à l'affichage d'une partie ou de la totalité des fonctionnalités d'une application.

- Chaque application comportera une fenêtre mère appelée aussi racine.

- Elle pourra posséder également d'autres fenêtres qui seront dépendantes (au niveau cycle de vie) de la fenêtre mère mais indépendantes entre elles.

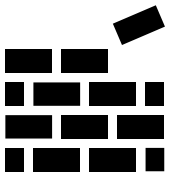


1. Généralités.

Tkinter : notion de widget

- Terme générique utilisé pour désigner les composants d'une interface graphique.

- Il peut s'agir de boutons, zones d'affichage, zones de saisie, menus déroulants, barre de défilement, etc.



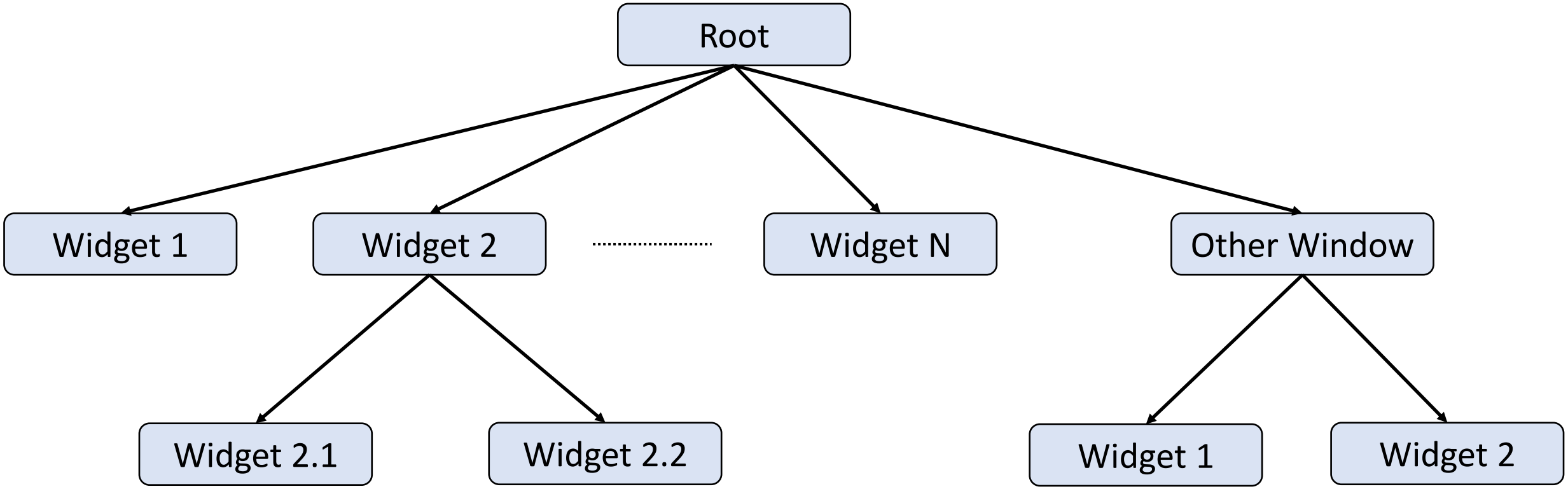
1. Généralités.

Tkinter : relation de parenté

- Une application est d'abord composée d'une racine unique, la fenêtre mère.
- Les widgets et éventuelles autres fenêtres sont ensuite définis comme enfants de la racine ou d'un composant déjà existant.
- Cette relation permettra (entre autres) de réaliser le positionnement des différents éléments.

1. Généralités.

Tkinter : relation de parenté



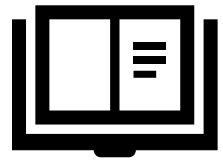
1. Généralités.

2. Les différents types de widgets.

2. Les différents types de widgets.

Webographie

- Trois excellentes ressources à consulter pour le rôle et la syntaxe précise des différents widgets :
  - Tkinter pour ISN
  - Interface graphique Tkinter
  - http://pascal.ortiz.free.fr/contents/tkinter/tkinter/



2. Les différents types de widgets.

Widgets basiques

Widget	***Fonction***
Label	Affichage de textes et/ou images
Button	Permet de déclencher une action
Checkbutton	Sélection/désélection d'une option
Radiobutton	Sélection d'une option dans une liste
Entry	Saisie d'une valeur
Combobox	Sélection d'une option dans un menu déroulant
Progressbar	Permet de visualiser l'avancée d'une tâche
Scrollbar	Bar de défilement associée à un autre widget

2. Les différents types de widgets.

Widgets plus avancés

Widget	***Fonction***
Spinbox	Sélection d'une valeur dans une liste par incrémentation/décrémentation
Scale	Sélection d'une valeur dans un intervalle à l'aide d'un curseur
Text	Affichage et saisie de textes "longs" Permet aussi l'affichage d'images
Treeview	Affichage sous forme d'arbre d'options sélectionnables
Menu	Affichage sous forme de barre de menus, contenant chacun une liste déroulante d'options sélectionnables

2. Les différents types de widgets.

Widgets message box : fenêtres de dialogues (pop up)

<table>
<tr><th>Widget</th><th colspan="2">Fonction</th></tr>
<tr><td>showinfo</td><td colspan="2" rowspan="3">Afficher un message d’information
Une seule réponse possible “OK“</td></tr>
<tr><td>showwarning</td></tr>
<tr><td>showerror</td></tr>
<tr><td>askyesno</td><td rowspan="5">Pose une question simple, réponses possibles :</td><td>Oui / Non</td></tr>
<tr><td>askquestion</td><td>Oui / Non</td></tr>
<tr><td>askyesnocancel</td><td>Oui / Non / Annuler</td></tr>
<tr><td>askokcancel</td><td>OK / Annuler</td></tr>
<tr><td>askretrycancel</td><td>Réessayer / Annuler</td></tr>
</table>

2. Les différents types de widgets.

Widgets positionnels : conteneurs d'autres widgets

Widget	***Fonction***
Frame	Plusieurs Frames dans la même fenêtre Gestion indépendante des positionnements dans chaque Frame
Panedwindow	Plusieurs panneaux dans la même fenêtre agencés horizontalement ou verticalement Gestion facilitée de l'espace dédié à chaque panneau
Notebook	Plusieurs onglets "superposés" dans la même fenêtre Sélection de l'onglet à afficher par une barre

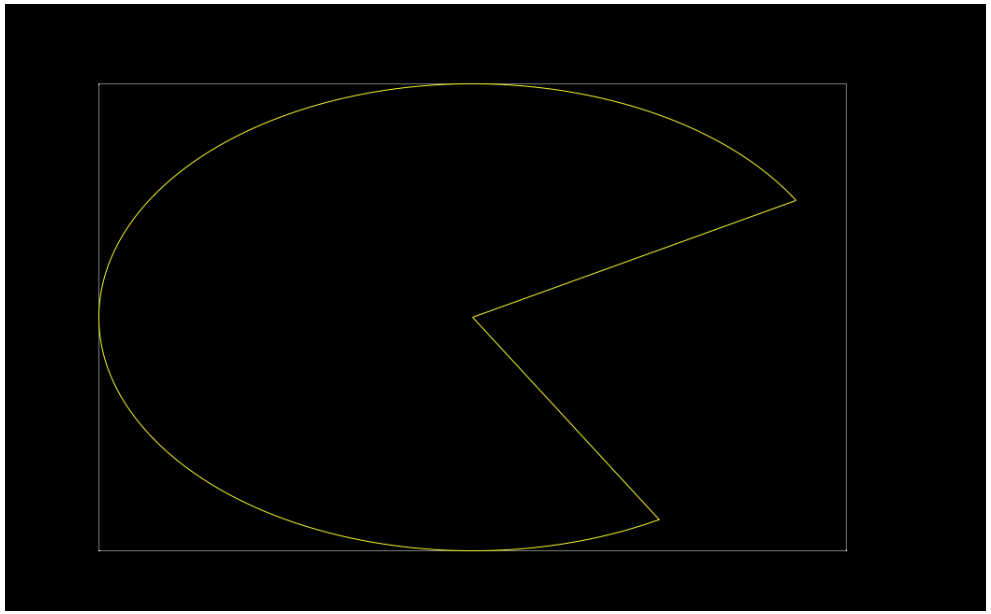
2. Les différents types de widgets.

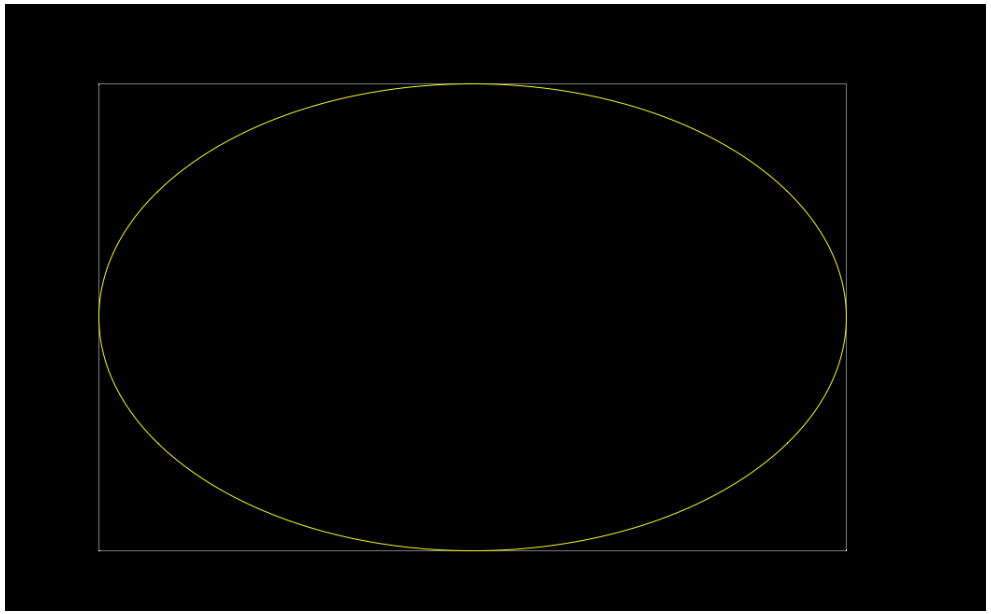
Un widget particulier : le Canvas

- Permet de dessiner des formes géométriques (segments, polygones, cercles, ellipses, arcs, etc.) en spécifiant les coordonnées de leurs extrémités.
- Possibilité également d'insérer du texte et des images.
- On peut réserver une partie du Canvas pour insérer d'autres widgets regroupés dans un "frame".

2. Les différents types de widgets.

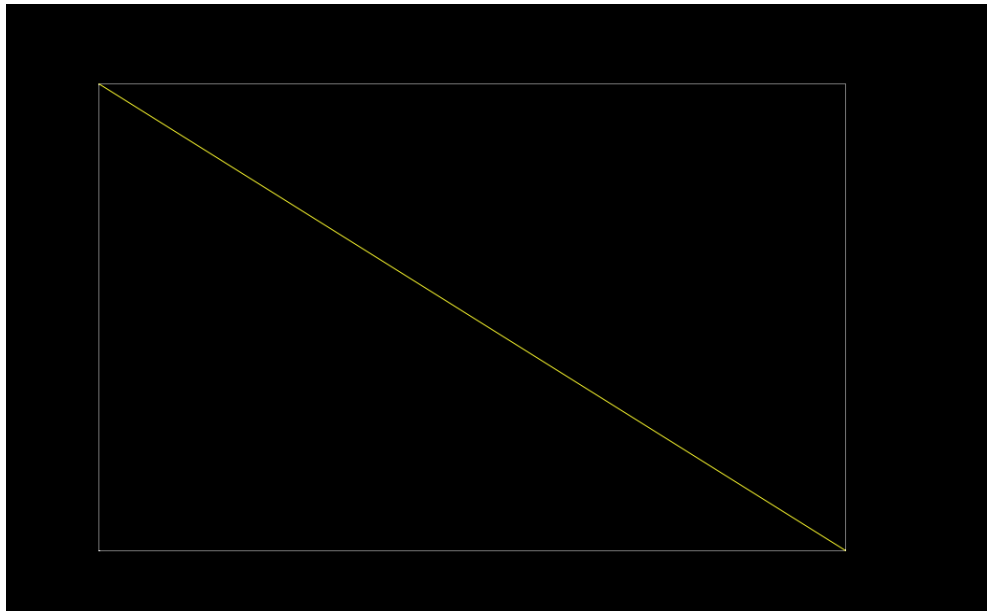
Canvas : notion de “bounding box“





2. Les différents types de widgets.

Canvas : notion de “bounding box“



2. Les différents types de widgets.

3. Positionnement des widgets.

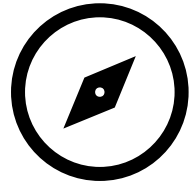
3. Positionnement des widgets.

Placement des widgets

- Pour chaque élément, à commencer par la racine, on doit préciser la façon dont on va positionner les widgets qui le composent.

- Trois modes de positionnement :

    - pack
    - grid
    - place



3. Positionnement des widgets.

Positionnement “pack“

- On positionne un widget par rapport à l’un des bords de la fenêtre ou par rapport au côté d’un widget déjà positionné.
- Mode de positionnement le plus simple de Tkinter.
- Permet facilement d’empiler des widgets horizontalement ou verticalement.
- Insuffisant pour des situations complexes.

3. Positionnement des widgets.

Positionnement “pack“

- Bien adapté à l’empilement vertical ou horizontal des widgets.

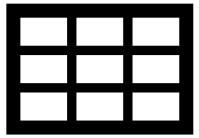
3. Positionnement des widgets.

Positionnement “grid“

- Utilisation d’une grille “virtuelle“ pour positionner les widgets en utilisant des coordonnées (numéro de ligne, numéro de colonne).

- Mode plus élaboré que “pack“, il permet de mettre au point des dispositions complexes.

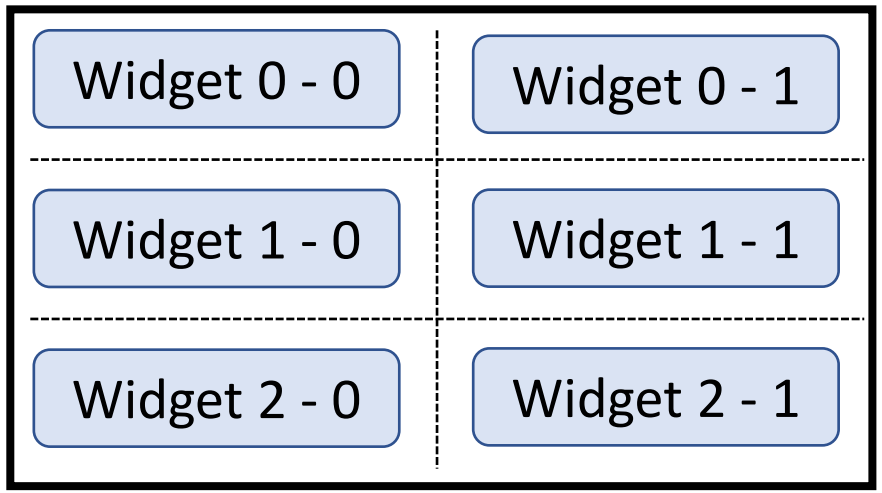
- Possibilité de positionner un widget en fusionnant deux (ou plus) cellules de la grille (horizontalement ou verticalement).

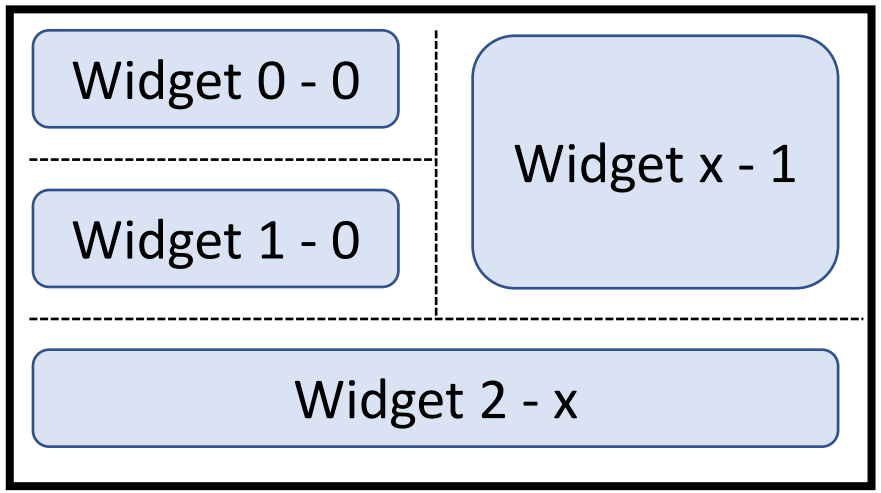


3. Positionnement des widgets.

Positionnement “grid“

- On repère un widget par ses coordonnées “fictives“. Possibilité de fusionner des cellules.





3. Positionnement des widgets.

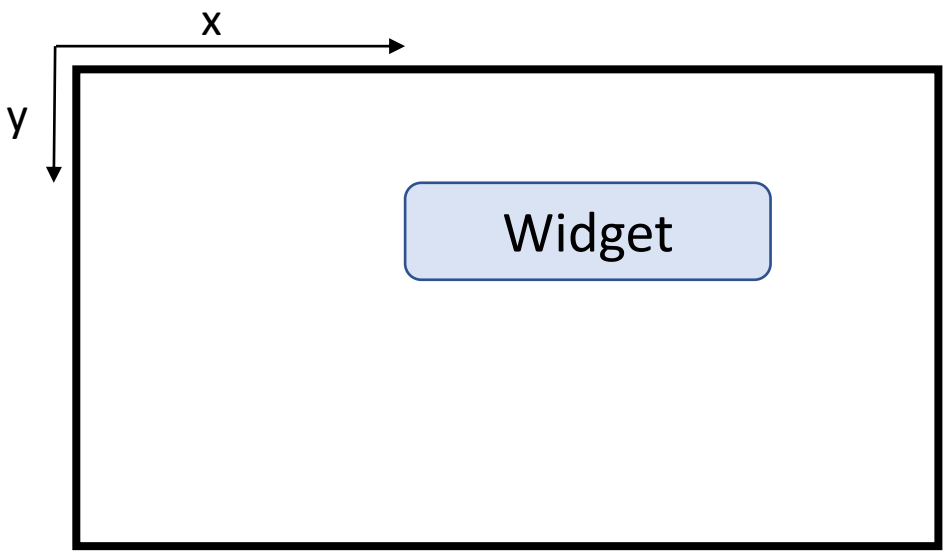
Positionnement “place“

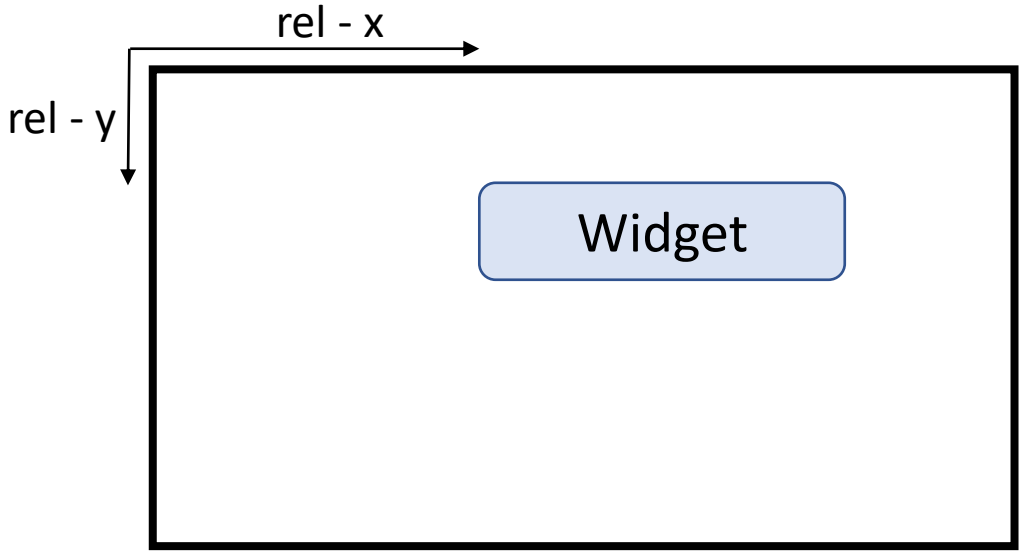
- On positionne un widget en indiquant les coordonnées de l’un de ses points d’ancrage (coin nord-est ou centre par exemple).
- On peut utiliser des coordonnées absolues, en pixels, ou des coordonnées relatives, entre 0 et 1.
- Mode plus riche que les précédents, éventuellement délicat à gérer en présence d’un grand nombre de widgets.

3. Positionnement des widgets.

Positionnement “place“

- On repère un widget par les coordonnées absolues ou relatives de l’un de ses points d’ancrage.

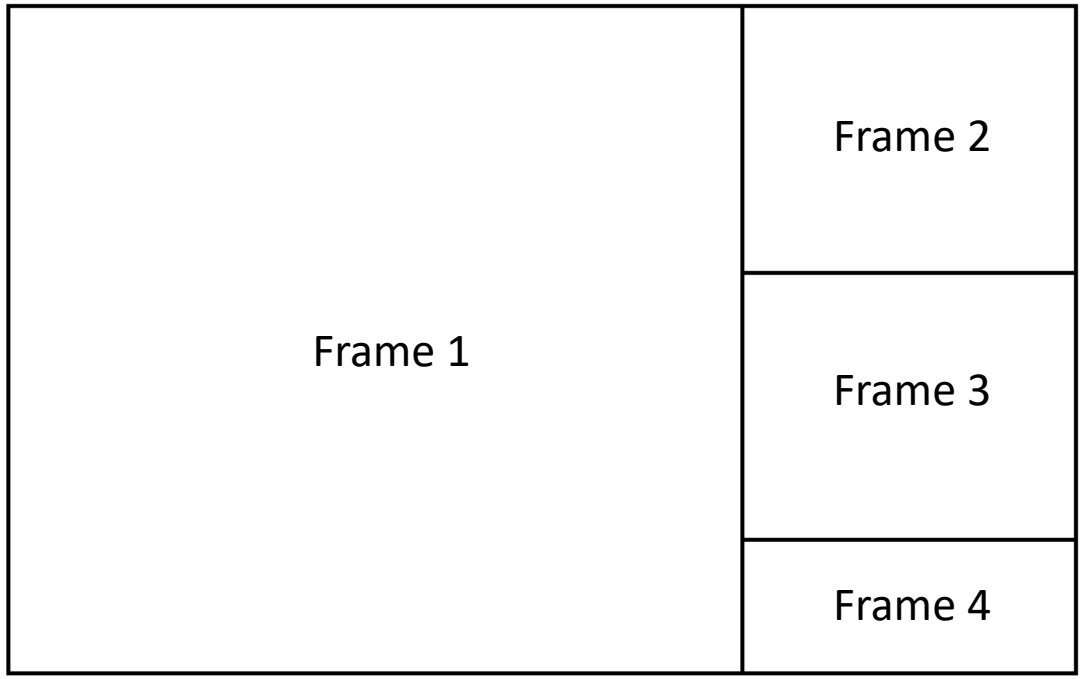




3. Positionnement des widgets.

Remarque

- Découper la fenêtre à l'aide de widgets "frame" permet d'utiliser des modes de positionnement différents dans chacun d'eux.



3. Positionnement des widgets.

4. Les évènements.

4. Les évènements.

Différents types d’évènements

- Réponse à une interaction avec un bouton (Button, Checkbutton, Radiobutton, Spinbox, Scale, Scrollbar).

- Liaison d’une action, clavier ou souris, avec un widget, en particulier un Canvas.

4. Les évènements.

Interaction avec un bouton

- Paramètre “command“ du bouton (et certains autres widgets) en question.
- Renvoi vers une méthode personnalisée.
- Exemple de syntaxe :

```
Button(self.__root, text='My Button', command=self.myMethod)
```

4. Les évènements.

Liaison d’une action avec un widget

- Méthode “bind“ du widget en question, souvent un Canvas.
- Renvoi vers une méthode personnalisée.
- Exemples de syntaxe :

```
self.__canvas.bind('<Button-1>', self.myMethod1)
self.__canvas.bind('<Key>', self.myMethod2)
```

4. Les évènements.

Liaison d'une action avec plusieurs widgets

- On peut lier d'un seul coup tous les widgets d'une certaine classe (*e.g.* Canvas) à une même méthode :

```
bind_class
```

- On peut procéder de même avec tous les widgets de l'application :

```
bind_all
```

4. Les évènements.

Liaison d'une action avec un widget : remarques

- Le premier paramètre de la méthode "bind" caractérise l'évènement à lier.
- Principaux types d'évènements : <Button-1>, <Button-2>, <Key>, <Motion>.
- La méthode liée prend en paramètre l'évènement en question.

```
def keyMove(self, event):
    ...
```

4. Les évènements.

Récupération des caractéristiques des évènements

- Se fait avec les attributs du paramètre “event“.
- Coordonnées d’un clic de souris :

```
event.x, event.y
```

- Nature d’une touche :

```
keysym
keysym_num
```

4. Les évènements.

Quelques exemples de touches et leur “keysym“ correspondant

- Chiffres du pavé numérique : KP_0, KP_1, KP_2, etc.
- Flèches : Up, Down, Right, Left.
- Touches spéciales : space, Delete, Escape, Tab.
- Touches fonction : F1, F2, F3, etc.

4. Les évènements.