Interface graphique et programmation orientée objet

Intérêt d'une interface graphique

Jusqu'à présent nous avons essentiellement utilisé des scripts Python ayant des structures linéaires. Dès de l'exécution une successions d'opérations se réalisent les une après les autres. Une fois toutes les opérations de l'algorithme réalisées le programme se termine.

Les interfaces graphiques permettent naturellement une interaction entre l'ordinateur et l'utilisateur. Elles sont un outil idéal pour mettre en œuvre des "programmes pilotés par événements".

Il existe de nombreuses interfaces graphiques développées pour Python.

L'interface graphique **tkinter** est un module complémentaire, pré-installé avec Python, richement documenté et assez simple à prendre en main.

Premières fenêtres avec tkinter

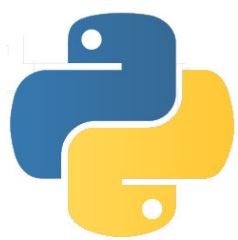

```
import tkinter as tk

fen1=tk.Tk()

texte1=tk.Label(fen1,text='Bonjour!',fg='red')

texte1.pack()

bouton1=tk.Button(fen1,text='Quitter',\n
command=fen1.destroy)

bouton1.pack()

fen1.mainloop()
```

Structure type d'un script textuel

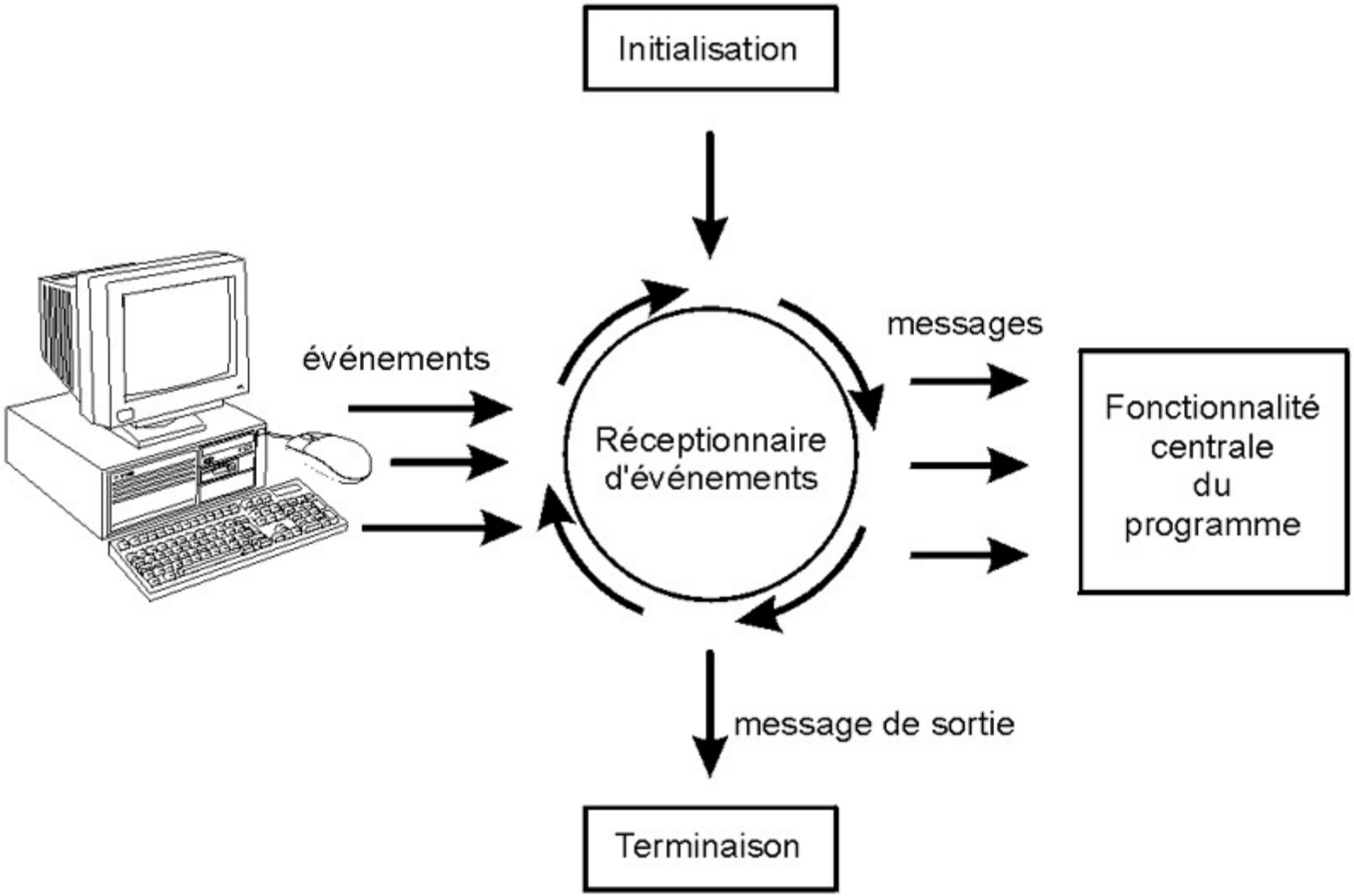
Trois phases principales :

- Initialisation : Importation des bibliothèques complémentaires, définitions des fonctions ouverture des fichiers utiles...
- Exécution la fonctionnalité principale du programme
- Terminaison : Affichage des résultats, fermeture des fichiers...

Ces trois étapes se font successivement de manière globalement linéaire et l'interaction utilisateur/machine est limitée (input)



Structure d'un programme piloté par événements



Les widgets de tkinter

Widget	Description
Button	Un bouton classique, à utiliser pour provoquer l'exécution d'une commande quelconque.
Canvas	Un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner, créer des éditeurs graphiques, et aussi pour implémenter des widgets personnalisés.
Checkbutton	Une case à cocher qui peut prendre deux états distincts (la case est cochée ou non). Un clic sur ce widget provoque le changement d'état.
Entry	Un champ d'entrée, dans lequel l'utilisateur du programme pourra insérer un texte quelconque à partir du clavier.
Frame	Une surface rectangulaire dans la fenêtre, où l'on peut disposer d'autres widgets. Cette surface peut être colorée. Elle peut aussi être décorée d'une bordure.
Label	Un texte (ou libellé) quelconque (éventuellement une image).
Listbox	Une liste de choix proposés à l'utilisateur, généralement présentés dans une sorte de boîte. On peut également configurer la Listbox de telle manière qu'elle se comporte comme une série de « boutons radio » ou de cases à cocher.
Menu	Un menu. Ce peut être un menu déroulant attaché à la barre de titre, ou bien un menu « *pop up* » apparaissant n'importe où à la suite d'un clic.
Menubutton	Un bouton-menu, à utiliser pour implémenter des menus déroulants.
Message	Permet d'afficher un texte. Ce widget est une variante du widget Label, qui permet d'adapter automatiquement le texte affiché à une certaine taille ou à un certain rapport largeur/hauteur.
Radiobutton	Représente (par un point noir dans un petit cercle) une des valeurs d'une variable qui peut en posséder plusieurs. Cliquer sur un bouton radio donne la valeur correspondante à la variable, et « vide » tous les autres boutons radio associés à la même variable.
Scale	Vous permet de faire varier de manière très visuelle la valeur d'une variable, en déplaçant un curseur le long d'une règle.
Scrollbar	Ascenseur ou barre de défilement que vous pouvez utiliser en association avec les autres widgets : Canvas, Entry, Listbox, Text.
Text	Affichage de texte formaté. Permet aussi à l'utilisateur d'éditer le texte affiché. Des images peuvent également être insérées.
Toplevel	Une fenêtre affichée séparément, au premier plan.

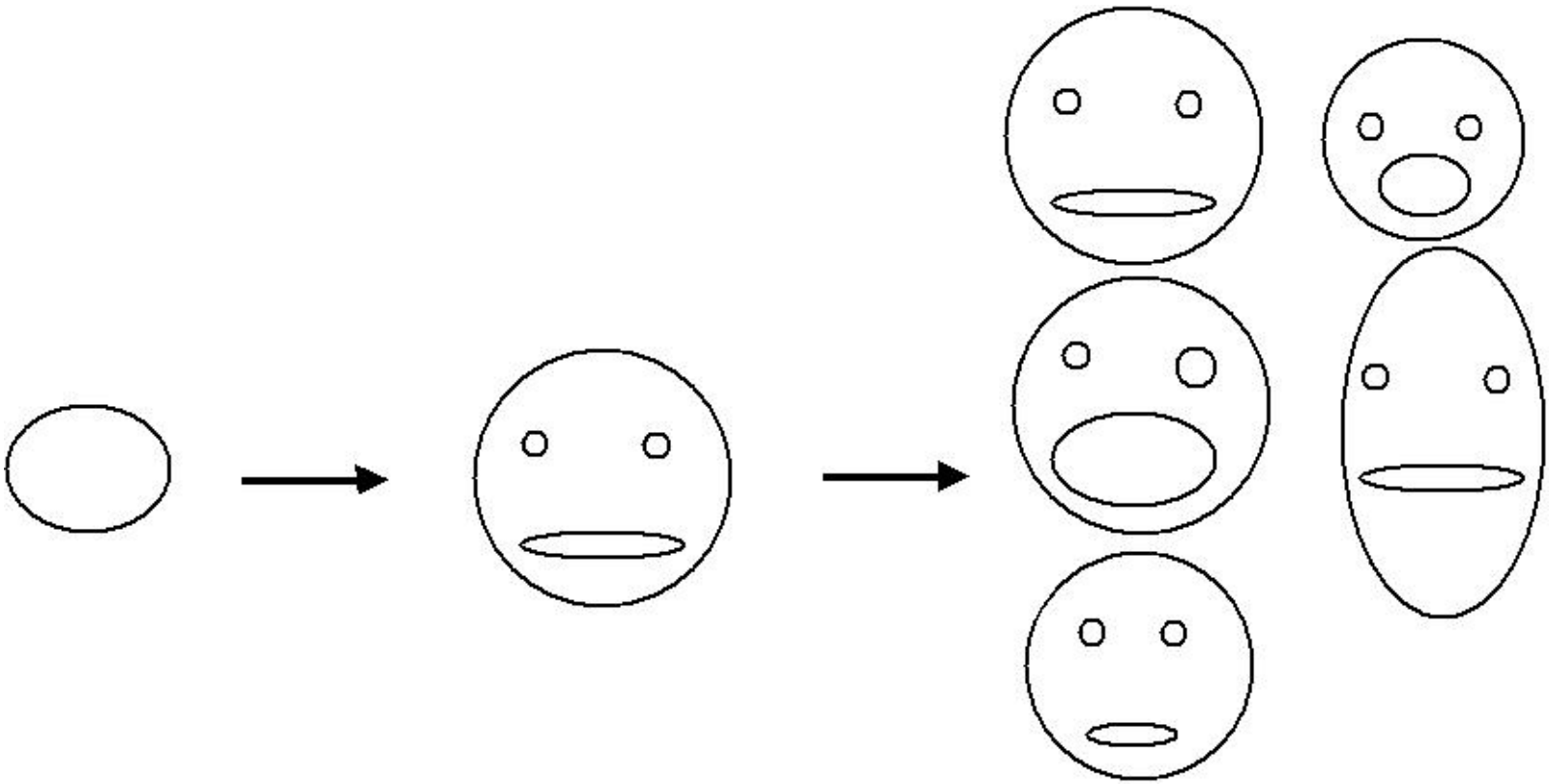
Exemple : Réaliser une calculatrice

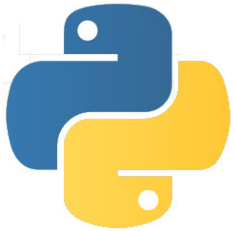

```
def evaluer(event):
    chaine.configure(text="Resultat = "   \n
    + str(eval(entree.get())))

fenetre=tk.Tk()
fenetre.title("Calculatrice")
entree=tk.Entry(fenetre)
entree.bind("<Return>",evaluer)
chaine=tk.Label(fenetre)
entree.pack()
chaine.pack()

fenetre.mainloop()
```

Intérêts de la POO

- Concevoir des objets informatiques qui vont pouvoir interagir entre eux et avec le monde extérieur (clavier, souris ...).
- Encapsuler des fonctionnalités internes de l'objet réalisé (création de méthodes).
- Concevoir des objets complexes à partir d'objets de base. Une ellipse... un visage.... un personnage ... des personnages ... des personnages qui bougent, parlent...

Exemple : générer une foule de visage





Classes d'objets

Les classes sont les principaux outils de la POO. Tels des moules, ils permettent de fabriquer (instancier) des objets complèxes capables d'intéragir avec leur environnement. Il existe de nombreuses classes d'objets prédéfinies dans Python.

Exemple : listes = objets de classe `list`. Des méthodes spécifiques aux objets de classe `list` peuvent leur être appliquées (`.append()`, `.pop()`, `.reverse()`...voir `help(list)`)

Les méthodes sont des fonctions définies au niveau de la classe d'objets.

Un objet est donc un ensemble de donnée contenant des attributs (ou paramètres), qui décrivent ses propriétés et de méthodes, qui lui permettent d'interagir avec son environnement. Les attributs et les méthodes sont définis dans la classe à partir de laquelle est instancié l'objet. Il sont donc encapsulés dans l'objet lui même.

Il est possible de créer de nouvelles classes d'objets avec Python.

Structure type d'une classe d'objet


```
class Machin():
        """Definition des objets de
        classe Machin"""

        def __init__(self,valeur1,valeur2,...):
                """Methode constructeur
                d'un objet de classe Machin"""
                self.attribut1=valeur1
                self.attribut2=valeur2
                ...

        def autre_methode(self,valeurA,...):
                """Descriptif de la methode"""
                self.attribut1=valeurA
                self.attribut2=valeurB
                ...
```