

**Ingénierie des  
systèmes interactifs**

L2 MIASHS Parcours informatique

UE MIB0405V

*Réalisation de systèmes interactifs  
à manipulation directe  
en Python  
avec tkinter*

Nouveautés par rapport à ce que  
vous avez fait en algo-prog  
jusqu'à présent

# Objets, objets d'interaction, événements, programmation événementielle

Tout ce que vous avez vu en algorithmique-programmation jusqu'à maintenant (variables, constantes, expressions, ... structures de contrôle, structures de données, ...) est toujours utilisable.

Mais :

- ❑ les interactions avec l'utilisateur vont se faire par l'intermédiaire d'objets d'interaction
- ❑ vos programmes vont donc manipuler des **objets d'interaction** qui sont basés sur le "**paradigme objet**"
- ❑ les objets d'interaction répondent à des **événements**
- ❑ il va falloir écrire des programmes qui répondent aux événements que reçoivent les objets. Il s'agit de "**programmation événementielle**".

# Le paradigme objet (en bref)

Très brièvement :

- ❑ un programme manipule des **objets**
- ❑ les objets sont des instances de **classes**
- ❑ une classe décrit les caractéristiques communes à tous ses objets :
  - ❑ un ensemble de **propriétés**
  - ❑ un ensemble de **méthodes** qui sont les opérations réalisables sur les objets
- ❑ une classe peut être construite à partir d'une classe existante
  - ❑ elle **hérite** de la description de la classe existante
  - ❑ l'étend ou la modifie

exemple de classe :

une droite

propriétés :

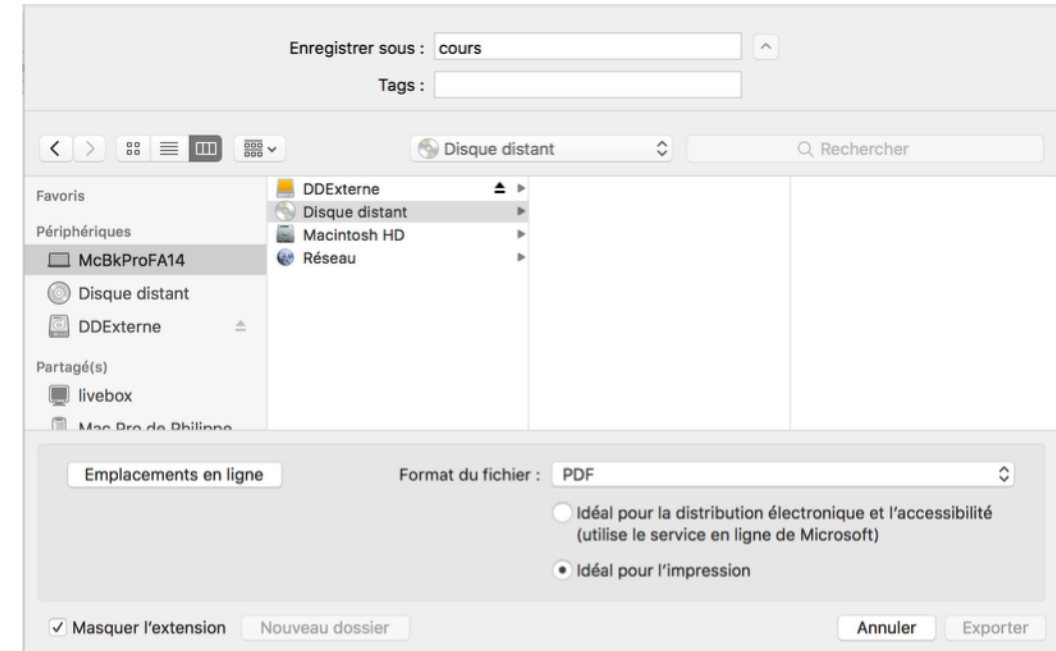
- coeff\_directeur
- ord\_a\_origine

méthode :

- dessiner

# Objets d'interaction et bibliothèques d'objets d'interaction

- ❑ un objet d'interaction est un objet (conforme au paradigme objet) avec lequel l'utilisateur peut interagir
- ❑ les classes d'objets d'interaction définissent de façon générale ces objets
  - ❑ par exemple, la classe "bouton" définit de façon générale la notion de bouton d'interaction (propriétés, méthodes et événements)

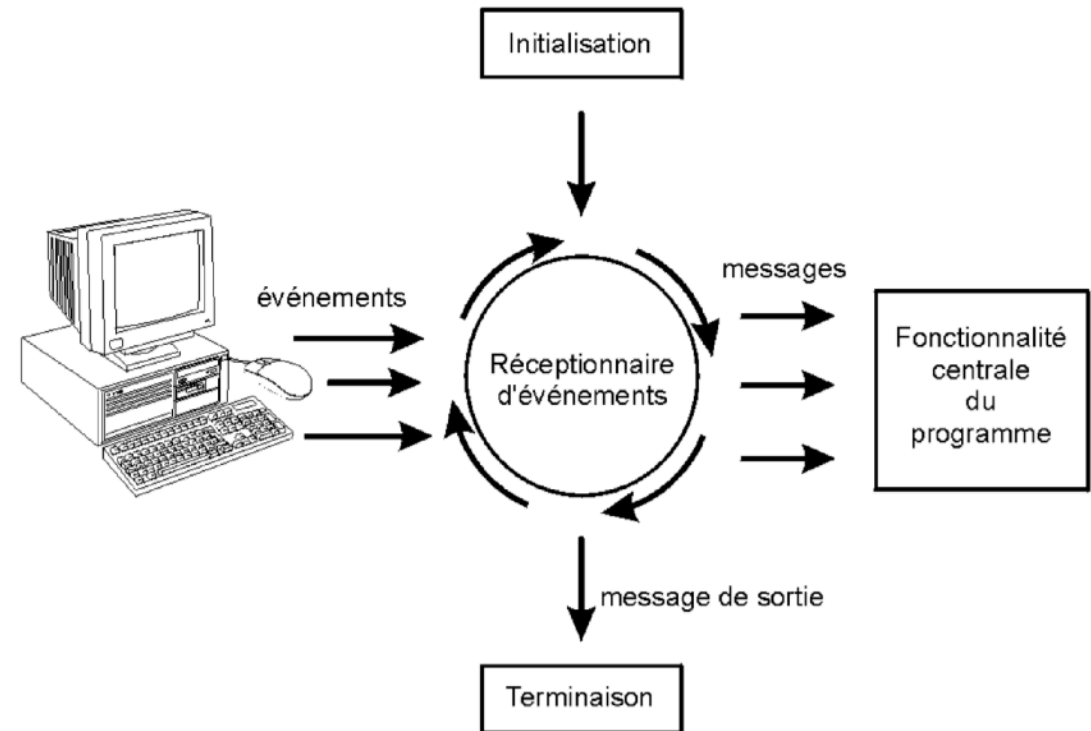


# Objets d'interaction et bibliothèques d'objets d'interaction

- ❑ les langages de programmation disposent de bibliothèques de classes d'objets d'interaction
- ❑ nous utiliserons la bibliothèque tkinter de Python
  - ❑ il en existe d'autres : wxPython, pyQT, pyGTK
  - ❑ intérêts de tkinter :
    - ❑ portabilité sur différents systèmes,
    - ❑ disponible par défaut (pas d'installation supplémentaire),
    - ❑ popularité,
    - ❑ basée sur tk du langage tcl.

# La programmation événementielle

- ❑ jusqu'à maintenant, vous avez écrit des programmes pour lesquels l'exécution se fait de manière séquentielle
- ❑ en programmation événementielle, l'exécution d'un programme est pilotée par les événements
  - ❑ les programmes sont construits sur une boucle qui réceptionne et gère les événements



# Schéma général d'un programme

```
from tkinter import * #importation de la bibliothèque tkinter

#éventuellement, définition des sous-programmes

#création de la fenêtre principale
ma_fenetre = Tk()

#création et positionnement des widgets
mon_bouton = Button(ma_fenetre, text = "quitter", command = ma_fenetre.quit)
mon_bouton.pack()

#lancement du gestionnaire d'événements
ma_fenetre.mainloop()
```



# Les classes de la bibliothèque tkinter (1)

Présentation, quelques propriétés et méthodes, code Python de création  
d'objets (widgets)

# La classe Tk

## fenêtre

- c'est dans la fenêtre que seront placés les autres widgets
- on créera d'abord un widget Tk puis les autres widgets

## Autres classes =>

- un fois un widget créé
- il faut le placer dans la fenêtre (ou dans un Frame) : méthode pack ou grid

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 « <i>pop up</i> » 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.

# Les propriétés des classes d'objets d'interaction

❑ On retrouve certaines propriétés dans beaucoup de classes ; par exemple :

❑ les dimensions : width, height (en pixels)

❑ la couleur de fond du widget : bg

❑ d'autres sont spécifiques ; par exemple :

❑ la propriété command du bouton

la valeur de la couleur  
s'exprime

- en rgb (ex : #000fff000)
- ou avec une constante prédéfinie (ex. 'red')

❑ La valeur d'une propriété d'un widget peut être donnée à la création du widget ;  
exemple :

❑ `mon_label = label(ma_fenetre, text = "bonjour !")`

❑ et peut être par la suite modifiée ; exemple :

❑ `mon_label["text"] = "hello!"` (1)

(1) on peut aussi utiliser la méthode config

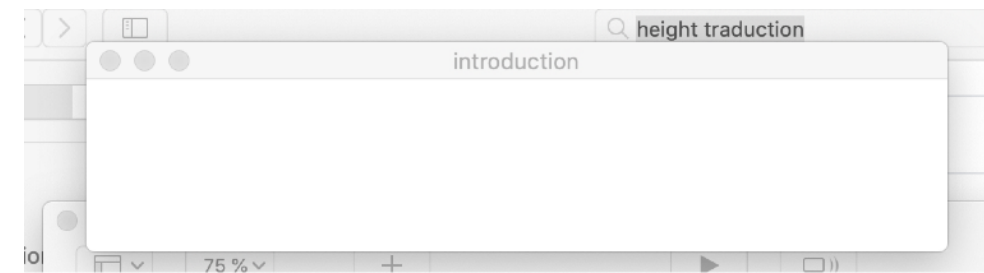
# La classe Tk

- ❑ permet de définir la fenêtre principale de l'application
- ❑ quelques méthodes :
  - `mainloop()` : affiche la fenêtre et lance le gestionnaire d'événements
  - `destroy()` : détruit la fenêtre et ses enfants
  - `geometry()` : précise les dimensions (en pixels)
  - `title()` : donne un titre à la fenêtre

- ❑ code Python de création

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

```
from tkinter import *  
  
ma_fenetre = Tk()  
ma_fenetre.title('introduction')  
ma_fenetre.geometry('500x100-150+50')  
  
ma_fenetre.mainloop()
```



# La classe Tk - exemple et commentaires

```
from tkinter import *
```

```
ma_fenetre = Tk()
```

```
ma_fenetre.mainloop()
```

```
from tkinter import *
```

```
ma_fenetre = Tk()
```

```
ma_fenetre.title('introduction')
```

```
ma_fenetre.geometry('500x100-150+50')
```

```
ma_fenetre.mainloop()
```

- ❑ création d'un widget de la classe Tk
- ❑ qui dispose de la définition générale d'une fenêtre que possède la classe Tk
  - exemples : redimensionnement, case de fermeture (démonstration)
- ❑ qui sera identifié dans le programme par `ma_fenetre`
- ❑ notation pointée pour invoquer une méthode
  - exemple : `ma_fenetre.mainloop()`

# La classe Button

## ❑ quelques propriétés :

- bg : couleur de fond
- bd : couleur du cadre
- height, width (en pixels)
- padx, pady : espace autour du texte (en pixels)
- text : texte du bouton
- font : police et taille des caractères
- command : commande à exécuter lors d'un clic (sans paramètre)

## ❑ code Python de création :

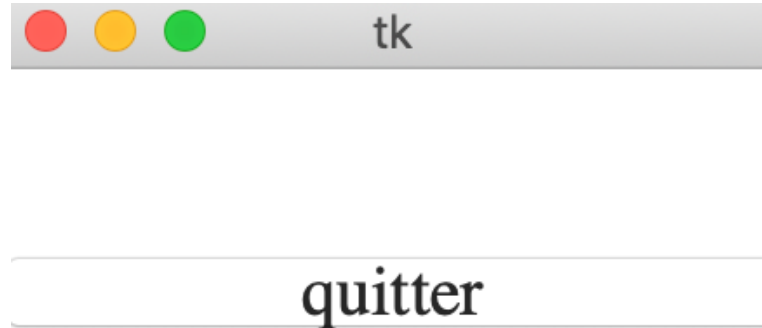
❑ `mon_bouton = Button(ma_fenetre, text = "quitter", command = ma_fenetre.quit)`

la valeur de la fonte  
s'exprime sous la forme d'un  
tuple ; exemples :

- ('Helvetica', '18')
- ('Times', '30', 'bold')

# La classe Button - Exemple

```
from tkinter import *  
ma_fenetre = Tk()  
mon_bouton = Button(ma_fenetre, text = "quitter", font = ("Times", 20), \  
height = 5, width = 20, command = ma_fenetre.quit)  
mon_bouton.pack()  
ma_fenetre.mainloop()
```



# La classe Button - Exemple

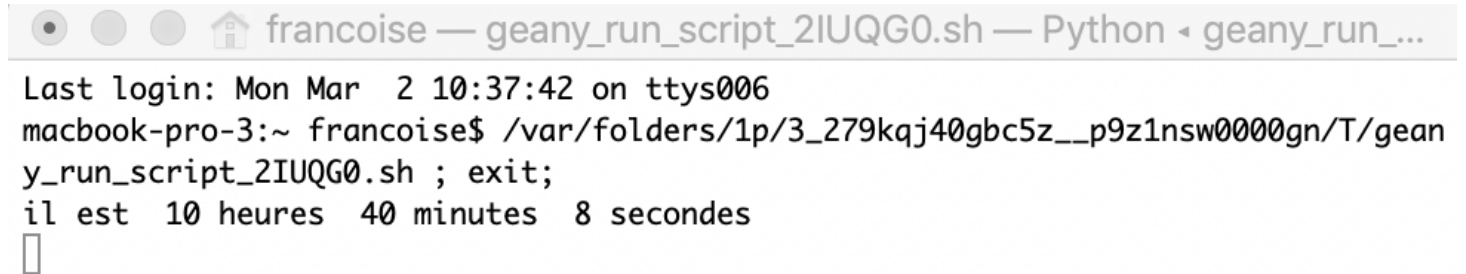
```
from tkinter import *
import time

def affiche_heure():
    heure = time.localtime()
    print('il est ', heure[3], 'heures ', heure[4], 'minutes ', \
    heure[5], 'secondes')

ma_fenetre = Tk()

mon_bouton = Button(ma_fenetre, text = "Quelle heure est-il ?", \
    command = affiche_heure, padx = 20, pady = 20, font = ('helvetica', 20))
mon_bouton.pack()

ma_fenetre.mainloop()
```

A screenshot of a terminal window. The title bar shows 'francoise — geany\_run\_script\_2IUQG0.sh — Python 3.8.10 geany\_run...'. The terminal output shows the execution of a Python script that prints the current time. The output is: 'Last login: Mon Mar 2 10:37:42 on ttys006', 'macbook-pro-3:~ francoise\$ /var/folders/1p/3\_279kqj40gbc5z\_\_p9z1nsw0000gn/T/geany\_run\_script\_2IUQG0.sh ; exit;', and 'il est 10 heures 40 minutes 8 secondes'. There is a cursor at the end of the last line.

```
francoise — geany_run_script_2IUQG0.sh — Python 3.8.10 geany_run...
Last login: Mon Mar 2 10:37:42 on ttys006
macbook-pro-3:~ francoise$ /var/folders/1p/3_279kqj40gbc5z__p9z1nsw0000gn/T/geany_run_script_2IUQG0.sh ; exit;
il est 10 heures 40 minutes 8 secondes
█
```



# La classe Label

## ❑ principales propriétés :

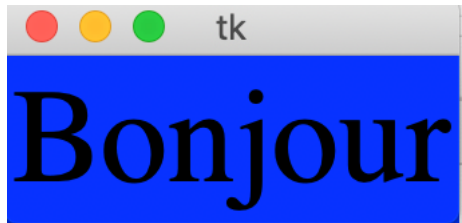
- bg, fg : couleur de fond, couleur du texte
- bd : couleur du cadre
- text ou textvariable: texte du label
- font
- width (en nombre de caractères)

## ❑ code Python de création :

❑ `mon_label = label(ma_fenetre, text = "bonjour !")`

## La classe Label - Exemple (avec un texte constant)

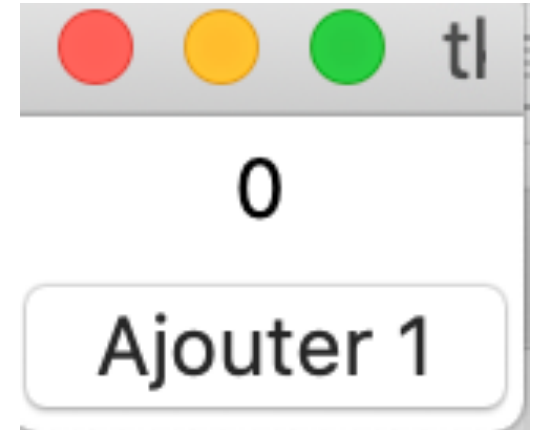
```
from tkinter import *  
  
ma_fenetre = Tk()  
  
mon_label = Label(ma_fenetre, text = "Bonjour", bg = 'blue', \  
font = ("Times", 50))  
mon_label.pack()  
  
ma_fenetre.mainloop()  
|
```



# La classe Label - Exemple (avec un texte variable)

- ❑ présentation

- ❑ on veut réaliser l'interface suivante :
  - ❑ chaque fois que l'utilisateur va cliquer sur le bouton "Ajouter 1", le label du dessus sera incrémenté de 1



# La classe Label - Exemple (avec un texte variable)

```
from tkinter import *

def compte():
    inter = int(nombre.get())+1
    nombre.set(str(inter))

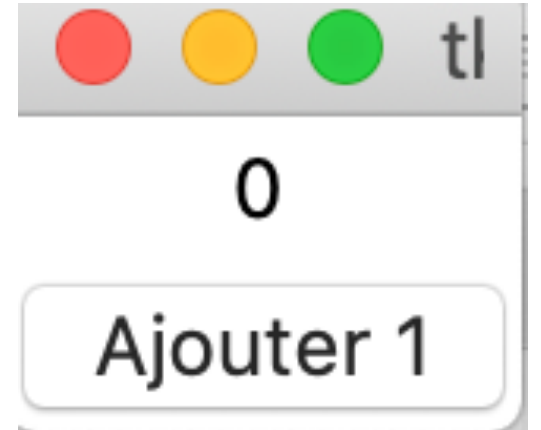
ma_fenetre = Tk()

nombre = StringVar()
nombre.set(str(0))

mon_label = Label(ma_fenetre, textvariable = nombre)
mon_label.pack()

mon_bouton = Button(ma_fenetre, text = "Ajouter 1", command = compte)
mon_bouton.pack()

ma_fenetre.mainloop()
```



- ❑ remarque : la classe StringVar
  - ❑ et les méthodes set et get

# La classe Entry

## ❑ quelques propriétés :

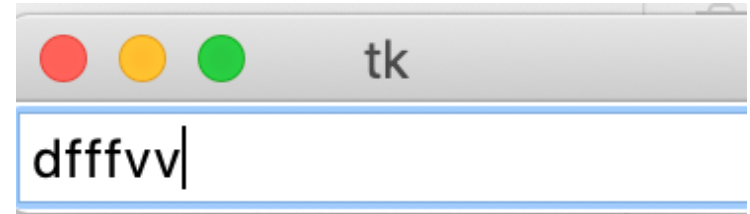
- bg, fg : couleur de fond, du texte
- width : taille (en nombre de caractères)
- textvariable: texte de la zone de saisie
- font : police et taille des caractères
- show ; exemple : show='\*'

## ❑ code Python de création :

❑ `mon_champ = entry(ma_fenetre)`

## La classe Entry - exemple

```
from tkinter import *  
ma_fenetre = Tk()  
mon_champ = Entry(ma_fenetre)  
mon_champ.pack()  
ma_fenetre.mainloop()
```



# La classe Entry - exemple

```
from tkinter import *  
  
def efface():  
    ma_var.set('')  
  
ma_fenetre = Tk()  
  
ma_var = StringVar ()  
mon_champ = Entry(ma_fenetre, textvariable = ma_var, width = 50)  
mon_champ.pack()  
  
mon_bouton = Button(ma_fenetre, text = "Effacer", command = efface)  
mon_bouton.pack()  
  
ma_fenetre.mainloop()
```



## Exercice

Ecrire et tester les programmes précédents :

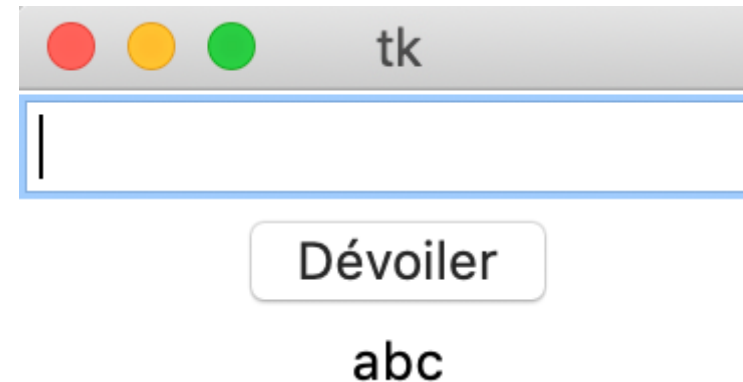
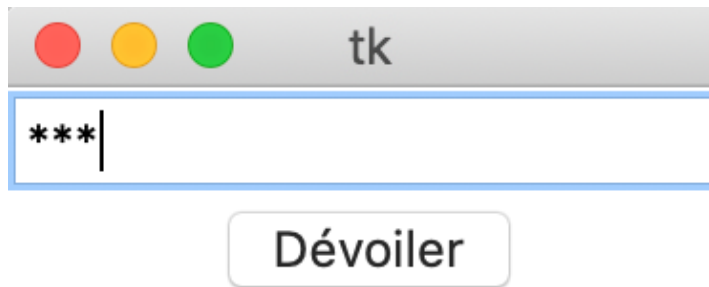
- celui du bouton avec "quelle heure est-il ?"
- celui du label avec texte variable



## Exercice

Ecrire un programme :

- qui demande à l'utilisateur son mot de passe
  - le mot de passe n'apparaît pas en clair : \*\*\*\*\*
- qui permet de l'afficher en clair



## Exercice

Ecrire un programme

- qui demande à l'utilisateur un nombre entier
- et affiche si ce nombre est pair ou impair

