

# Codes de la présentation

## Turtle

Cf. la [documentation officielle](#).

```
<sxh python; title : turtle-01.py> #!/usr/bin/python # -*- coding: UTF-8 -*-
```

```
# exemple de base turtle # from turtle import * import sys import time
```

```
reset() x=-100 y=-100 i=0 while i < 10:
```

```
    j=0
    while j <10:
        up()
        goto(x+i*20,y+j*20)
        down()
        fill(1)
        n=0
        while n <4 :
            forward(16)
            left(90)
            n=n+1
        color([i*0.1,j*0.1,0])
        fill(0)
        color(0,0,0)
        j=j+1
    i=i+1
```

```
# end
```

```
time.sleep(10) </sxh>
```

## Tkinter

En Python 2. Pour Python 3, utiliser “from tkinter import \*” et print() !

Cf. la [documentation officielle](#).

```
<sxh python; title : tkinter-simple-entry.py> #!/usr/bin/python # -*- coding: UTF-8 -*-
```

```
# lecture de 2 masses par une fenêtre tk
```

```
from Tkinter import *
```

```
fen01 = Tk() fen01.title("Lecture de deux masses") chaine1 = Label (fen01, text = "introduisez la première masse :") chaine2 = Label (fen01, text = "introduisez la deuxième masse :")
```

```
chaine1.grid(row =0) chaine2.grid(row =1) entr1= Entry(fen01) entr2= Entry(fen01) entr1.grid(row =0, column =1) entr2.grid(row =1, column =1)
bou1=Button(fen01,text='Continuer',command=fen01.quit) bou1.grid(row=2,column=1)

fen01.mainloop()

m1 = float(entr1.get()) m2 = float(entr2.get()) fen01.destroy()

print 'Masses lues : ', m1,' et ',m2

</sxh>
```

## Canvas Tkinter : rebond d'une balle

```
<sxh python; title : anima_auto_rebond.py> #!/usr/bin/env python # -*- coding: utf-8 -*-

# Petit exercice utilisant la librairie graphique Tkinter

from Tkinter import *

# définition des gestionnaires # d'événements :

def move():

    "déplacement de la balle"
    global x1, y1, vx, vy, dt, flag
    x1, y1 = x1 +vx*dt, y1 + vy*dt
    if x1 < 0 or x1 > 220:
        vx=-vx
    if y1 < 0 or y1 > 220:
        vy = -vy
    can1.coords(oval1,x1,y1,x1+30,y1+30)
    if flag >0:
        fen1.after(2,move)          # boucler après 50 millisecondes

def stop_it():

    "arret de l'animation"
    global flag
    flag =0

def start_it():

    "démarrage de l'animation"
    global flag
    if flag ==0: # pour éviter que le bouton ne puisse lancer plusieurs
    boucles
        flag =1
```

```
move()
```

#

## Programme principal

```
# les variables suivantes seront utilisées de manière globale : x1, y1 = 40, 115 # coordonnées initiales vx, vy = 10, 5 # vitesse du déplacement dt=0.1 # pas temporel flag =0 # commutateur
```

```
# Création du widget principal ("parent") : fen1 = Tk() fen1.title("Exercice d'animation avec Tkinter")  
# création des widgets "enfants" : can1 = Canvas(fen1,bg='dark grey',height=250, width=250)  
can1.pack(side=LEFT, padx =5, pady =5) oval1 = can1.create_oval(x1, y1, x1+30, y1+30, width=2, fill='red')  
bou1 = Button(fen1,text='Quitter', width =8, command=fen1.quit)  
bou1.pack(side=BOTTOM) bou2 = Button(fen1, text='Démarrer', width =8, command=start_it)  
bou2.pack() bou3 = Button(fen1, text='Arrêter', width =8, command=stop_it) bou3.pack() #  
démarrage du récepteur d'évènements (boucle principale) : fen1.mainloop() fen1.destroy()  
</sxh>
```

From:

<https://dvillers.umons.ac.be/wiki/> - Didier Villers, UMONS - wiki

Permanent link:

[https://dvillers.umons.ac.be/wiki/teaching:progappchim:codes\\_presentation?rev=1424191672](https://dvillers.umons.ac.be/wiki/teaching:progappchim:codes_presentation?rev=1424191672)Last update: **2015/02/17 17:47**