Les graphiques - Utilisation de Matplotlib

(A partir du document Edupython1.3)

Sommaire

Si la tortue permet de revisiter la géométrie de collège, vous remarquerez rapidement que les possibilités graphiques sont très limitées pour un usage en classe.
Nous vous présentons ici la bibliothèque matplotlib.pyplot, qui va nous permettre de tracer des graphes de fonctions et autres diagrammes.

I. Placer des points, Afficher le repère, Les couleurs

plot(*x,y,options*) *matplotlib.pyplot*

Place dans la fenêtre graphique un point de coordonnées (x,y). Si aucune échelle n'est spécifiée, une échelle adaptée est proposée.
Le point est représenté avec la couleur et le style défini par la chaîne options. Cette chaîne comprend usuellement deux caractères : le premier étant une couleur, le deuxième le style, comme l'indique le tableau ci-dessous :

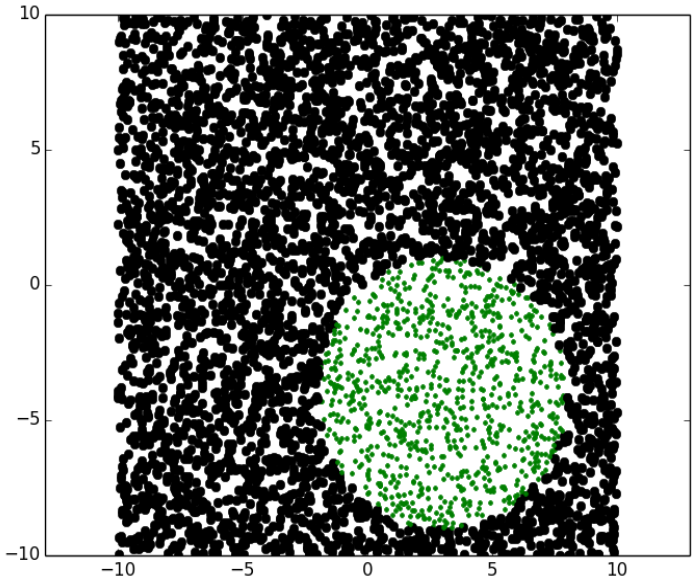
Couleur	Style
b bleu	- ligne continue
g vert	- - tirets
r rouge	: pointillés
c cyan	. des points
m magenta	o des billes
y jaune	x des croix
k noir	v des triangles
w blanc	-. points-tirets
...	...

Exemple:
la commande **plot(2,3,'gx')** place un point sous forme de croix verte de coordonnées (2,3).

Remarque:
Les options '-' ou '–' n'ont pas d'effet sur les points, mais seront utiles pour tracer des courbes, comme nous le verrons pas la suite.

Exemple:
On cherche, dans un repère orthonormé, tous les points de coordonnées (x,y) tels que :

$$x * (6 - x) < y * (8 + y).$$



Code : Ligne de niveau.

```
from random import *
from matplotlib.pyplot import *
for i in range( 10000 ):
     x = uniform(-10 ,10)
     y = uniform(-10 ,10)
     if x*(6-x)<y*(8+y):
          plot(x,y,'ko')
          axis('equal')
     else :
          plot(x,y,'g.')
show ()
```

Remarque:
Le programme se termine par l'instruction show() qui a pour effet, vous l'aurez déjà deviné, d'afficher la fenêtre.

II. Nuage de points ou diagrammeXY

plot(*X,Y*) *matplotlib.pyplot*

Dessine la fonction affine par morceaux passant par les points de coordonnées (x_i,y_i) où les nombres x_i et y_i sont respectivement les éléments de la liste X et la liste Y .
Pour dessiner le segment [AB], l'instruction plot([xA,xB],[yA,yB]) suffit.

Il est facile de tracer la représentation graphique d'une fonction affine :

Code : Tracer d'une droite.

```
from matplotlib.pyplot import *
a = float(input("Entrez le coefficient directeur"))
b = float(input("Entrez l'ordonnée à l'origine"))
x = np.array([-10,10])
plot(x,a*x+b)
show()
```

Code : Tracer de la fonction carré.

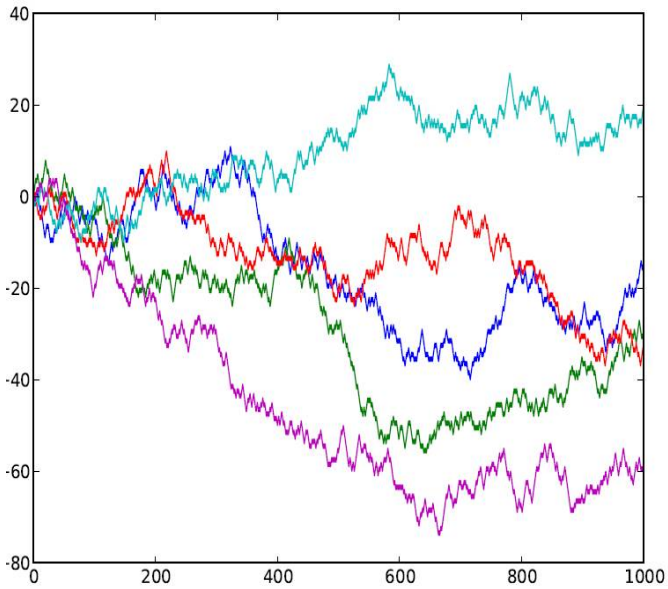
```
from matplotlib.pyplot import *
x = np.arange(-10,10,0.1)
plot(x,x**2)
show()
```

Remarque :
Précision technique : ici, on a recours à l'instruction **np.arange(debut,fin,pas)**, qui crée une liste de valeurs entre debut et fin avec un pas pouvant être décimal (contrairement au **range**). Le gros avantage de cet objet, c'est que l'on peut alors effectuer des opérations comme x * x ou 3 * x + 4... (Consultez l'aide sur les objets de type NDArray de la bibliothèque numpy pour plus d'informations.)

Autre exemple : une puce située à l'origine d'un axe gradué effectue 1 000 sauts successifs. A chaque saut, elle avance ou recule aléatoirement d'une unité sans préférence pour un sens ou l'autre. Représentez le chemin parcouru par la puce.

Code : Marche aléatoire.

```
from random import *
from matplotlib.pyplot import *
p = 0
x = []
y = []
for i in range(1000):
     x = x+[i]
     y = y+[p]
     if randint(0,1)==0 :
          p = p+1
     else :
          p = p-1
plot(x,y)
show()
```



Remarque:
Ici le programme donné ne trace qu'une seule courbe, mais si vous décidez de ne pas fermer la fenêtre graphique et de relancer le programme, les graphiques se superposent et une échelle adaptée est proposée ce qui représente un apport pédagogique. Le paragraphe suivant vous explique néanmoins comment forcer le nettoyage de la fenêtre en début de programme.

III. Les axes et la grille

clf() *matplotlib.pyplot*

Efface et ré-initialise le contenu de la fenêtre graphique.

Par défaut, si aucune échelle n'est imposée, celle-ci est calculée automatiquement. Quelquefois, il peut s'avérer nécessaire de la modifier manuellement.

axis(*[xmin,xmax,ymin,ymax]*) *matplotlib.pyplot*

Impose une nouvelle échelle au repère.

Vous pouvez aussi choisir d'afficher une grille ou non en plus :

grid(*mode*) *matplotlib.pyplot*
si le booléen *mode* vaut True, la grille est affichée, si mode vaut False, elle est masquée, comme c'est le cas par défaut.

IV. Les titres et légendes

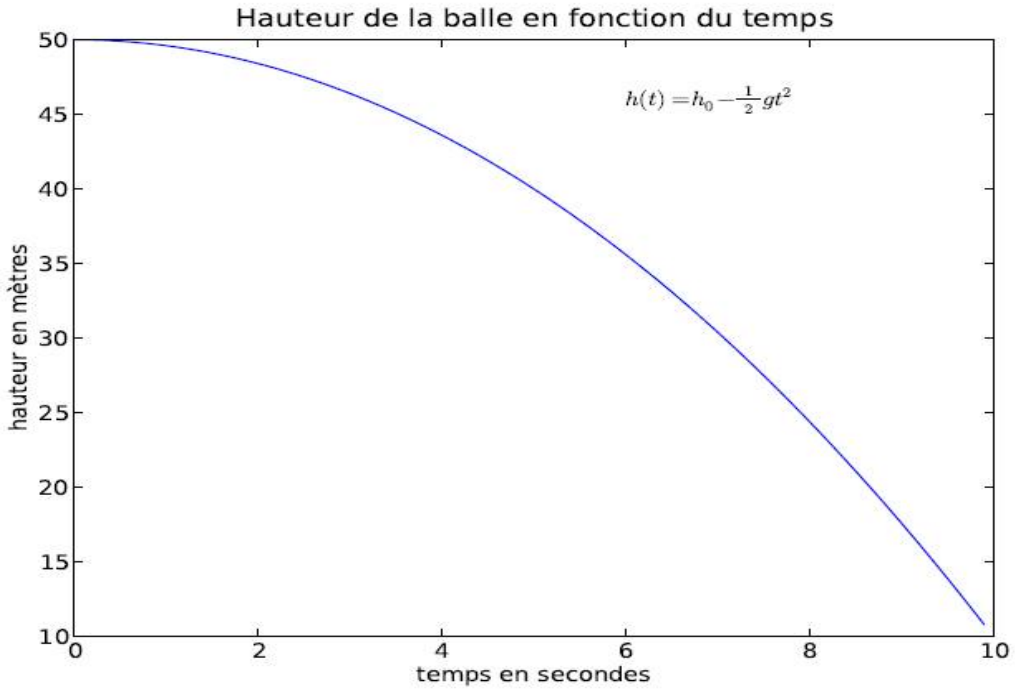
title(*texte*) *matplotlib.pyplot*
Ajoute le titre *texte* au graphique.

xlabel(*texte*) et ylabel(*texte*) *matplotlib.pyplot*
Affiche le texte *texte* sur l'axe des abscisses ou l'axe des ordonnées.

text(x,y,texte) *matplotlib.pyplot*
Affiche le texte *texte* sur le graphique à la position (x,y).

Code : Chute d'une balle.

```
from matplotlib.pyplot import *
t = np.arange (0, 10 , 0.1)
plot(t,50-0.5*0.8*t**2)
title('Hauteur de la balle en fonction du temps')
ylabel("hauteur en mètres")
xlabel(" temps en secondes")
text(6,45,r'$h=h_0-\frac{1}{2}gt^2$')
show()
```



Remarque:
Vous aurez remarqué qu'en ajoutant un r à la chaine de texte, on peut faire afficher des formules mathématiques pour peu que l'on connaisse un minimum la syntaxe LATEX... Enfin, on est davantage dans la décoration qu'autre chose !

V. Repères multiples

Il existe une instruction permettant de fractionner la fenêtre graphique en plusieurs sous graphiques. C'est peut-être un peu complexe pour des élèves, nous vous laissons donc juger de l'utilité de leur présenter cette fonction.

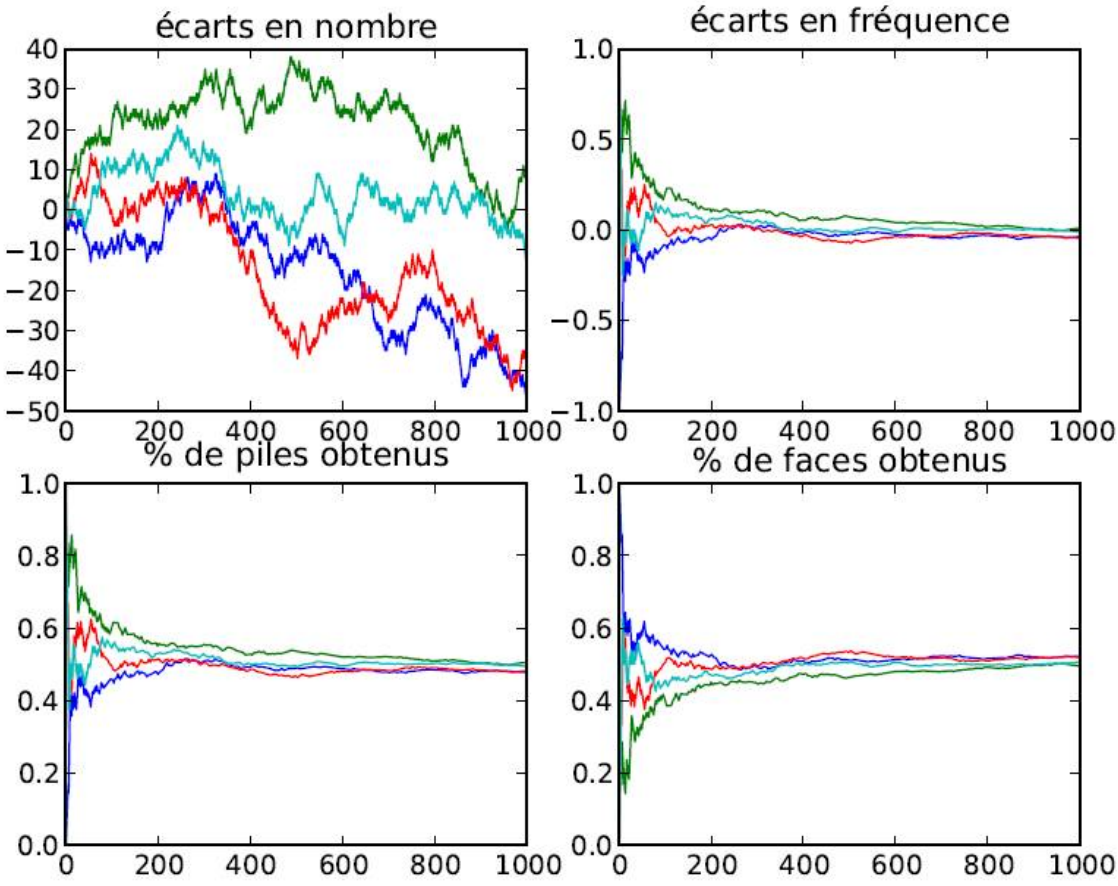
subplot(*n*) *matplotlib.pyplot*
Fragmente la fenêtre graphique en plusieurs repères. *n* est un nombre entier de 3 chiffres construit ainsi :

- Le chiffre des centaines indique le nombre de lignes à créer.
- Le chiffre des dizaines indique le nombre de colonnes à créer.
- Le chiffre des unités indique quel est le repère actif.

Exemple :
L'instruction **subplot(234)** découpe la fenêtre graphique en 6 parties numérotées ainsi :

1	2	3
4	5	6

Pour l'enseignant cela peut être pertinent avec le vidéoprojecteur pour donner du sens à la notion de probabilité comme dans l'exemple ci-dessous : faire apparaître ce qui est convergent et ce qui est chaotique...



Code : Pile ou face.

```
from matplotlib.pyplot import *
from random import *
couleur = ['r','b','g','c']
for c in couleur:
     p = 0
     f = 0
     e = 0
     n = []
     ecart = []
     ecartm = []
     freqP = []
     freqF = []
     for i in range(1,1000 ):
          if randint(0,1)==0:
               p = p+1
          else :
               f = f+1
               e = p-f
          n = n+[i]
               ecart = ecart+[e]
               ecartm  = ecartm+[e/i]
               freqP = freqP+[p/i]
               freqF = freqF+[f/i]
     subplot(221)
     plot(n,ecart)
     title("écarts en nombre")
     subplot(222)
     plot(n,ecartm )
     title("écarts en fréquence")
     subplot(223)
     plot(n,freqP)
     title("% de piles obtenus")
     subplot(224)
     plot(n,freqF)
     title("% de faces obtenus")
show()
```