TP 6 graphiques

I Pour s'initier aux graphiques avec matplotlib

Q1 Recopier et exécuter les commandes python suivantes:

Première exemple:

```
import matplotlib.pyplot as plt # pyplot: bibliotheque graphique
liste_x=[0,1,2]
liste_y=[0,1,0]
plt.figure() # nouvelle figure
plt.plot(liste_x,liste_y)
plt.show()
```

On constate que s'affiche dans une fenêtre la ligne brisée reliant les points $A = (1, 0)$, $B = (1, 1)$ et $C = (2, 0)$.

Deuxième exemple:

```
from math import sin
liste_x=[0,0.5,1,1.5,2,2.5,3]
liste_y=[sin(0),sin(0.5),sin(1),sin(1.5),sin(2),sin(2.5),sin(3)]
plt.figure() # nouvelle figure
plt.plot(liste_x,liste_y)
plt.show()
```

On constate que s'affiche dans une fenêtre la ligne brisée reliant les points $(0, \sin(0))$, $(0.5, \sin(0.5))$, $(1, \sin(1))$,$(1.5, \sin(1.5))$, $(2, \sin(2))$, $(2.5, \sin(2.5))$, $(3, \sin(3))$.

Troisième exemple:

```
from math import pi
liste_x=[]
liste_y=[]
for k in range(101):
    liste_x.append(pi*k/100)
    liste_y.append(sin(pi*k/100))
plt.figure() # nouvelle figure
plt.plot(liste_x,liste_y)
plt.show()
```

On constate que la ligne brisée représentée a l'allure d'une courbe. Cela s'explique par le grand nombre de points de cette ligne brisée.

II Affichage d'une courbe d'équation $y = f(x)$ sur un intervalle $[a, b]$

On souhaite utiliser une fonction affiche une courbe $y = f(x)$ sur un intervalle $[a, b]$ avec un nombre de points quelconques. Pour cela ,on dit que $(x_0, x_1, \ldots, x_n)$ est une division régulière de [a,b], si $x_0 = a$ et que $x_{i+1} = x_i + dx$ avec $dx = (b - a)/n$.

Q2a Sur une feuille de papier, dessiner un l'intervalle [a,b] et la division $(x_0, x_1, \ldots, x_n)$ lorsque $a = 1$, $b = 2$ et $n = 5$.

Q2b Exprimer x_i en fonction de a, b, i et n? Combien vaut x_n?

Q2c Ecrire une fonction **division(a,b,n)** qui renvoie la liste $[x_0, x_1, \ldots, x_n]$ définie comme indiqué précédemment.

Q2d Ecrire une fonction **liste_f(f,a,b,n)** qui renvoie la liste $[f(x_0), f(x_1), \ldots, f(x_n)]$. On utilisera la fonction précédente.

Q2e Ecrire une fonction **g(x)** qui renvoie, d'argument un nombre x et qui renvoie le nombre $\dfrac{x}{x^2+1}$.

Q2f Utiliser les instructions précédentes pour afficher la courbe d'équation $y = \dfrac{x}{x^2+1}$ sur l'intervalle $[a, b]$ sur l'intervalle $[-2, 2]$ avec une précision correcte.

III Cercles, polygones et polygones étoilés ($n \in \mathbb{N}$, $n \geq 2$)

Le cercle trigonométrique est l'ensemble des points $M(t)$ de coordonnées $(\cos(t), \sin(t))$ pour $t \in [0, 2\pi]]$. Les polygones auquels on s'intéresse dans la suite sont réguliers, inscrits dans le cercle trigonométrique et ont le point $M(0) = (1, 0)$ comme sommet. Par exemple, le polygone régulier à 3 cotés est le triangle équilatéral $M(0)$,$M\left(\frac{2\pi}{3}\right)$,$M\left(\frac{4\pi}{3}\right)$. Pour le tracer, il faut le refermer, c'est-à-dire qu'il faut tracer la ligne brisée reliant les points $M(0)$,$M\left(\frac{2\pi}{3}\right)$,$M\left(\frac{4\pi}{3}\right)$ et $M(2\pi) = M(0)$.

Q3a Sur une feuille de papier, tracer ce polygone à 3 cotés. Tracer aussi le polygone régulier à 5 cotés (pentagone). Pour tracer un polygone régulier à n cotés, quels sont les points de la ligne brisée?

Q3b Ecrire une fonction **polygone(n)** qui affiche le polygone régulier à n cotés inscrit dans le cercle trigonométrique et contenant le point (1,0). (on pourra utiliser la fonction **liste_f**).

Q3c Exécuter cette fonction avec $n = 300$.

Q3d Sur la figure du pentagone faite sur papier, relier les points en partant de $M(0)$ en "sautant" un point sur deux. On constate qu'on obtient un polygone "étoilé" à cinq cotés.

Dans un polygone régulier à n cotés, $\alpha = \dfrac{2\pi}{n}$ est appelé angle au centre et les points du polygone sont $M(0)$,$M(\alpha)$,$M(\alpha)$,...,$M((n-1)\alpha)$ et on referme ce polygone avec $M(n\alpha 2\pi) = M(2\pi) = M(0)$. Lorsque n et p sont deux entiers qui n'ont pas de diviseurs communs autre que 1 (on dit qu'ils sont premiers entre eux), le polygone p-étoilé à n cotés est le polygone de sommets $M(0)$,$M(p\alpha)$,$M(2p\alpha)$,...,$M((n-1)p\alpha)$ qui se referme avec $M(np\alpha) = M(2p\pi) = M(0)$
- Ecrire une fonction **polygone_etoile(n,p)** ou $2 \leq$ p $\leq$ n-2 et n et p qui affiche le polygone p-étoilé à n cotés.
- Tracer les différents polygones étoilés à 11 cotés obtenus avec les fonctions précédentes.
- Que se passe-t-il si n et p ne sont pas premiers entre eux.

On suppose maintenant que β est un réel quelconque et on souhaite tracer la ligne polygonale $M(0)$,$M(\beta)$,$M(2\beta)$,...,$M(N\beta)$ pour différentes valeurs de N (si $\dfrac{\pi}{\beta}$ est irrationnel, cette ligne polygonale ne se referme pas: voir la remarque)

Q3e Modifier légèrement ce que vous avez fait précédement de manière à tracer cette ligne polygonale lorsque $\beta = 2$. (on prendra des valeurs de N de plus en plus grandes (sans dépasser 1000)).

Remarque *Le polygone se referme si il existe* $(p, q) \in (\mathbb{N}^*)^2$ *tels que* $p\beta = q2\pi$*, c'est-à-dire si* $\pi = \dfrac{p\beta}{2q}$ *Si* $\beta = 2$, $\pi = \dfrac{p\beta}{2q} \Leftrightarrow \pi = \dfrac{p}{q}$*, ce qui n'est pas possible car* π *est irrationnel.*

IV Marche aléatoire dans le plan

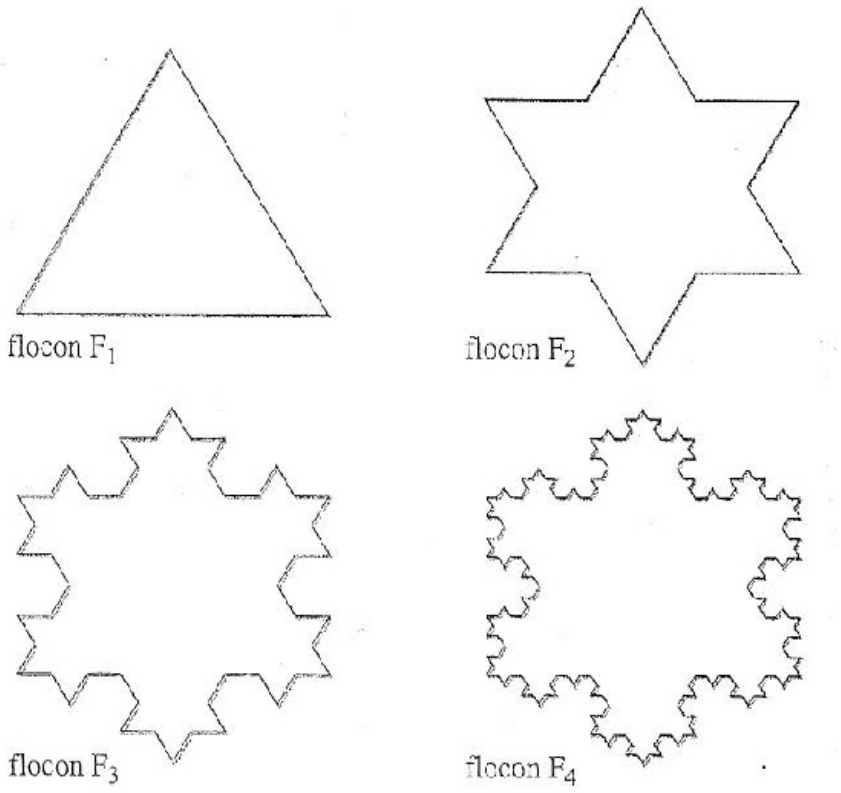
Q4 Un individu se déplace à partir de la position $(0, 0)$ à l'instant $t = 0$. S'il est en position (i, j) à l'instant t, il sera en position $(i + di, i + dj)$ à l'instant $t+1$ avec (di, dj) tiré au sort parmi les valeurs $(1, 0)$, $(-1, 0)$, $(0, 1)$, $(0, -1)$, c'est-à-dire que l'individu se déplace d'une case à droite, à gauche, vers le haut ou vers le bas. Ecrire une fonction **marche_aleatoire(n)** qui renvoie la liste [[i0,i1,...,in],[j0,j1,...,jn]] ou (ik,jk) est la position à l'instant k. Faire afficher une marche aléatoire. On utilisera la fonction randint du module random.

```
>>> import random
>>> x=random.randint(0,3) # renvoie un entier tire au sort entre 0 et 3 compris.
```

V Le flocon de Van Koch

Dans cette section, on considère des listes dont les éléments sont eux même des listes.

Le but de la section est de faire afficher plusieurs générations du flocon de Van-Koch.



Dans la suite, un point A du plan est assimilé à la liste [xA,yA] des coordonnées de A et une ligne polygonale est une liste de points.

Q5 La génération 1 du flocon de Van Koch est le triangle équilatéral ABC avec $z_A = e^{i\frac{\pi}{2}}$, $z_B = e^{i(\frac{\pi}{2}+\frac{2\pi}{3})}$ et $z_C = e^{i(\frac{\pi}{2}+\frac{4\pi}{3})}$.
Affecter à une variable F1 la ligne polygonale correspondante (on prendra garde à bien refermer le triangle).

Pour passer de la génération n à la génération $n+1$ du flocon de Van Koch, on remplace tout segment [A,B] de cette ligne polygonale par la ligne polygonale [A,C,E,D,B] où C et D sont obtenus en coupant en 3 le segment AB et E est tel que CDE est un triangle équilatéral direct.

Q6 Ecrire une fonction **coupe(A,B)** qui renvoie la liste [C,D] où C et D sont les points obtenus de A et B par les relations $\overrightarrow{AC} = \frac{1}{3}\overrightarrow{AB}$ et $\overrightarrow{AD} = \frac{2}{3}\overrightarrow{AB}$.
La relation $\overrightarrow{AM} = \lambda\overrightarrow{AB}$ traduite en coordonnées cartésiennes donne

$$\begin{cases} x_M = x_A + \lambda\,(x_B - x_A) \\ y_M = y_A + \lambda\,(y_B - y_A) \end{cases}$$

Q7 Ecrire une fonction **rotation(C,D)** qui renvoie le point E tel que CDE est un triangle équilatéral indirect, c'est-à-dire que E est image de D par la rotation de centre C et d'angle $-\pi/3$ (la relation $z_E - z_C = e^{i\frac{-\pi}{3}}\,(z_D - z_C)$ traduite en coordonnées cartésiennes donne

$$\begin{cases} x_E = x_C + \dfrac{1}{2}\,(x_D - x_C) + \dfrac{\sqrt{3}}{2}\,(y_D - y_C) \\ y_E = y_C - \dfrac{\sqrt{3}}{2}\,(x_D - x_C) + \dfrac{1}{2}\,(y_D - y_C) \end{cases}$$

Q9 Ecrire une fonction **suivant(F)** qui a pour argument la liste de points F correspondant à la génération n du flocon de Van Koch et qui renvoie la liste de points correspondant à la génération n+1 de ce flocon.

Q10 Ecrire une fonction **flocon(n)** qui renvoie la liste de points correspondant à la génération n du flocon.de Van Koch.

Q11: Ecrire une fonction **listes_coord(F)** ou F est une liste de points et qui renvoie la liste [lx,ly] avec lx, liste des abscisses des points de G et ly, liste des ordonnées des points de F.

Q12: Faire afficher différentes générations du flocon de Van Koch.

```
#Q2c
def division(a,b,n):
    h=(b-a)/n
    lx=[a]
    x=a
    for i in range(n):
        x=x+h
        lx.append(x)
    return lx
#Q2d
def liste_f(f,a,b,n):
    liste_x=division(a,b,n)
    liste_y=[]
    for i in range(n+1):
        liste_y.append(f(liste_x[i]))
    return liste_y
#Q2e
def g(x):
    return (x/(x**2+1))
#Q2f
liste_x=division(0,2,200)
liste_y=liste_f(g,0,2,200)
plt.figure()
plt.plot(liste_x,liste_y)
plt.show()
#Q2g
affichage(g,-2,2,300)
#Q3a
from math import pi
from math import sin
def polygone(n):
    lx=liste_f(cos,0,2*pi,n)
    ly=liste_f(sin,0,2*pi,n)
    plt.plot(lx,ly)
#Q3c
polygone(300) # polygone regulier a 300 cotes ressemble au cercle
#Q3d
def polygone_etoile(n,p):
    lx=liste_f(cos,0,2*p*pi,n)
    ly=liste_f(sin,0,2*p*pi,n)
    plt.plot(lx,ly)
#Q3e
def ligne_polygonale(n,beta):
    lx=liste_f(cos,0,n*beta,n)
    ly=liste_f(sin,0,n*beta,n)
    plt.plot(lx,ly)
```

```
#Q4import random
def marche_aleatoire(n):
    D=[(1,0),(-1,0),(0,1),(0,-1)]
    liste_x,liste_y=[0],[0]
    x,y=0,0
    for i in range(n):
        j=random.randint(0,3)
        dx,dy=D[j]
        x,y=x+dx,y+dy
        liste_x.append(x)
        liste_y.append(y)
    plt.plot(liste_x,liste_y)
# Flocon de Van Koch: travail supplementaire facultatif

#Q5
def coupe(A,B):
    xc=2/3*A[0]+1/3*B[0]
    yc=2/3*A[1]+1/3*B[1]
    xd=1/3*A[0]+2/3*B[0]
    yd=1/3*A[1]+2/3*B[1]
    return [[xc,yc],[xd,yd]]
#Q6
from math import sqrt
def rotation (C,D):
    xe=C[0]+1/2*(D[0]-C[0])+sqrt(3)/2*(D[1]-C[1])
    ye=C[1]-sqrt(3)/2*(D[0]-C[0])+1/2*(D[1]-C[1])
    return [xe,ye]
#Q7
def decoupe(A,B):
    C,D=coupe(A,B)[0],coupe(A,B)[1]
    return [A,C,rotation(C,D),D]

def suivant(G):
    NG=[]
    for i in range(len(G)-1):
        NG=NG+decoupe(G[i],G[i+1])
    NG=NG+decoupe(G[len(G)-1],G[0])
    NG.append(G[0])
    return NG
#Q8
def flocon(n):
    G=[[1,0],[-1/2,sqrt(3)/2],[-1/2,-sqrt(3)/2],[1,0]]
    for i in range(n):
        G=suivant(G)
    return G
#Q9
def liste_coord(G):
```

```
    lx,ly=[],[]
    for i in range(len(G)):
        lx.append(G[i][0])
        ly.append(G[i][1])
    return [lx,ly]

#Q10
import matplotlib.pyplot as plt
L=liste_coord(flocon(7))
plt.plot(L[0],L[1])
```