TD 2

Exercices d'application

Savoirs et compétences :

- *Alg – C15 : Récursivité : avantages et inconvénients.*

Exercice 1 – Fonction mystère

D'après ressources de C. Lambert. On donne la fonction suivante :

■ Python

```python
def mystere(L):
    """
    Ceci est la fonction mystère, saurez-vous
    trouver son but ?
    Entrée :
        * L(list) : liste de nombres entiers
    ou réels
    Sortie :
        * ???
    """
    n = len(L)
    if n==0 :
        return (None)
    elif n==1 :
        return (L[0])
    else :
        x = mystere(L[0:n-1])
        if x<=L[-1] :
            return (x)
        else :
            return (L[-1])
```

Question 1 *Sans coder la fonction, déterminer le résultat de l'instruction* ***print(mystere([14, 20, 3, 16]))*** *? Vous pourrez représenter de façon graphique l'empilement et le dépilement de la pile d'exécution.*

Question 2 *D'après vous quel est le but de cette fonction ?*

Question 3 *Programmer la fonction et tester l'instruction précédente. Sur plusieurs exemples, vérifiez la conjecture faite à la question précédente.*

Question 4 *Question subsidiaire. – Montrer que la propriété suivante est une propriété d'invariance :*
$\mathscr{P}(k)$ *: l'algorithme retourne le plus petit élément de toute liste de taille* k.

R

- Il faudra montrer que l'algorithme se termine au moyen d'un variant de boucle.
- Il faudra montrer $\mathscr{P}$ par récurrence.

Correction Soit L une liste de taille n. Alors n est un variant de boucle car :

- si $n=1$ ou $n=0$, l'algorithme se termine ;
- si $n>1$ chaque appel récursif est réalisé avec l'argument $L[0:n-1]$, qui est de longueur $n-1$. Ainsi n décrit une suite strictement décroissante, jusqu'à ce que $n=1$ (terminaison de l'algorithme).

Soit la propriété suivante : soit L une liste de taille k. L'appel à la fonction mystère retourne le plus petit élément de L. Montrons-le par récurrence.

Pour une liste de longueur 0 ou 1, le résultat est immédiat.

Soit une liste de taille $k+1$. Alors x reçoit le résultat de `mystere(L[0:k])`. D'après la propriété, x contient donc le plus petit élément de la liste `L[0:k]`. Ensuite x est comparé à l'élément `L[k]`. Si x est inférieur à cet élément, c'est donc le plus petit élément, et x est bien retourné. Sinon c'est que l'élément `L[k]` est le plus petit de la liste. C'est bien celui qui est retourné.

La propriété énoncée est donc bien héréditaire, et l'algorithme renvoie bien toujours le minimum de la liste entrée en argument.

Exercice 2 – Palindrome...

D'après ressources de C. Lambert. On souhaite réaliser une fonction `miroir` dont le but est de retourner le «miroir» d'une chaîne de caractères. Par exemple le résultat de `miroir("miroir")` serait *"riorim"*.

Question 1 *Programmer la fonction* `miroir_it` *permettant de répondre au problème de manière itérative.*

Question 2 *Programmer la fonction* `miroir_rec` *permettant de répondre au problème de manière récursive.*

Question 3 *Que renvoie la fonction si la chaîne de caractère est "Eh ! ça va la vache" ?*

Question 4 *Évaluer la complexité algorithmique de chacune des deux fonctions.*

Exercice 3 – Suite de Fibonacci

D'après ressources de C. Lambert.

On définit la suite de Fibonacci de la façon suivante :

$$\forall n \in \mathbb{N}, \left\{ \begin{array}{l} u_0 = 0, u_1 = 1 \\ u_{n+2} = u_n + u_{n+1} \end{array} \right.$$

Question 1 *Définir la fonction* `fibonacci_it` *permettant de calculer u_n par une méthode itérative. Évaluer la complexité algorithmique de l'algorithme.*

Question 2 *Définir la fonction* `fibonacci_rec` *permettant de calculer u_n par une méthode récursive « intuitive». Évaluer la complexité algorithmique de l'algorithme.*

Question 3 *Observer comment passer du couple (u_n, u_{n+1}) au couple (u_{n+1}, u_{n+2}). En déduire une autre méthode récursive pour calculer le n^e terme de la suite de Fibonacci. Évaluer la complexité algorithmique de l'algorithme.*

Exercice 4 – Faisons des Bulles

D'après les ressources de Mmes SEMBELY et VERDIER

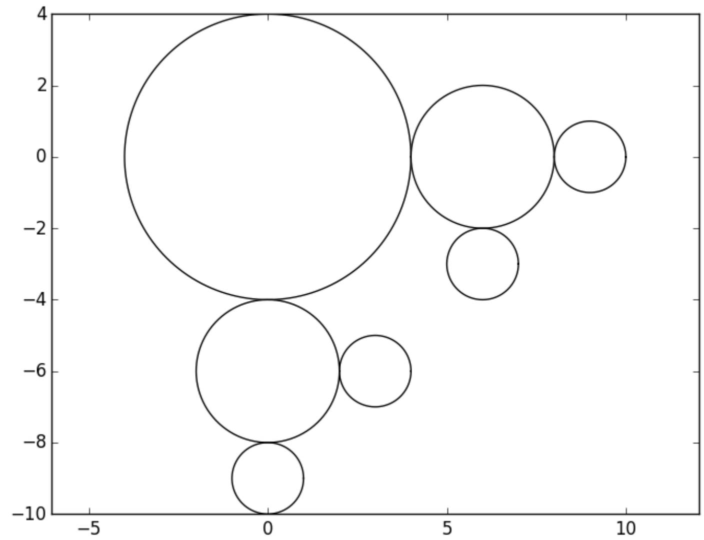
Les fractales sont des objets mathématiques fondés sur des figures géométriques se répétant à l'infini via un processus itératif.

"Une fractale est un objet irrégulier, dont l'irrégularité est la même à toutes les échelles et en tous les points"
Adrien DOUADY - mathématicien français

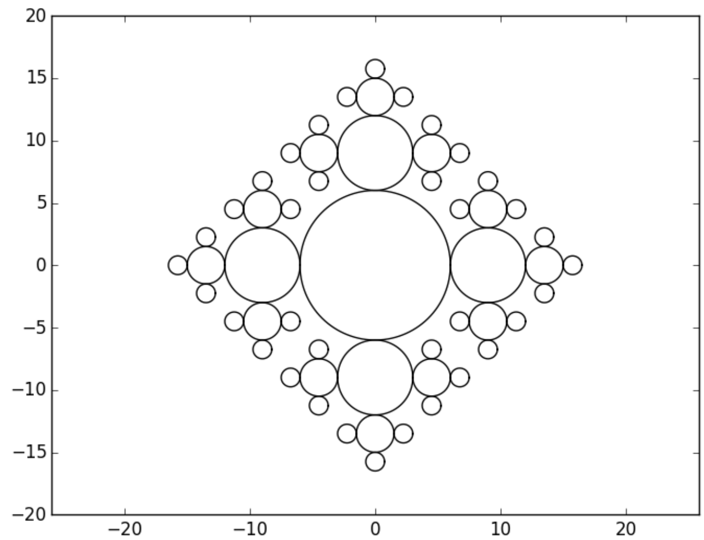
Ce type de structure se retrouve également dans la nature : architecture des côtes maritimes, ramifications nerveuses, nuages, galaxies ...
Etant donné leur structure, il est très intéressant d'utiliser la récursivité pour visualiser des ensembles fractals.

Question 1 *Ecrire une fonction* `cercle` *d'arguments* `x`, `y` *et* `r` *qui trace le cercle de centre le point $A(x, y)$ et de rayon r (supposé strictement positif).*

Question 2 *Ecrire une fonction récursive* `bubble` *d'arguments* `x`, `y`, `r` *et* `n`. *Elle effectuera la construction pour un nombre* `n` *d'étapes en ayant pour figure de base le cercle de centre* $A(x,y)$ *et de rayon* `r` *(supposé strictement positif). A chaque étape, le rayon du cercle est divisé par 2.*



Question 3 *Ecrire une fonction récursive* `bubbleComplet` *d'arguments* `x`, `y`, `r`, `n` *et une chaîne de caractères* `position`. *Elle effectuera la construction ci-dessous pour un nombre* `n` *d'étapes en ayant pour figure de base le cercle de centre* $A(x,y)$ *et de rayon* `r` *(supposé strictement positif).*



Correction
```
### tracé d'un cercle
import matplotlib.pyplot as plt
import numpy as np

def cercle(x,y,r):
    """tracé d'un cercle de centre A(x,y) et de rayon r"""
    angles=np.linspace(0,2*np.pi,500)
    les_X=[x+r*np.cos(angle) for angle in angles]
    les_Y=[y+r*np.sin(angle) for angle in angles]
    plt.plot(les_X,les_Y)
    plt.axis("equal")
    plt.show()
```

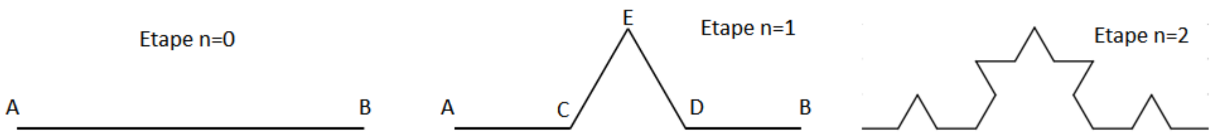
```
def cerclesRec(x,y,r,n):
    """tracé des cercles àdroite et en bas du cercle de centre A(x,y) et de rayon r"""
    cercle(x,y,r)
    if n>0:
        cerclesRec(x+1.5*r,y,r/2,n-1)
        cerclesRec(x,y-1.5*r,r/2,n-1)

def cerclesRec_2(x,y,r,n,position):
    """tracé des cercles autour du cercle de centre A(x,y) et de rayon r"""
    cercle(x,y,r)
    if n>0:
        if position=='centre':
            cerclesRec_2(x,y+1.5*r,r/2,n-1,'haut')
            cerclesRec_2(x-1.5*r,y,r/2,n-1,'gauche')
            cerclesRec_2(x+1.5*r,y,r/2,n-1,'droite')
            cerclesRec_2(x,y-1.5*r,r/2,n-1,'bas')
        if position=='droite':
            cerclesRec_2(x,y+1.5*r,r/2,n-1,'haut')
            cerclesRec_2(x+1.5*r,y,r/2,n-1,'droite')
            cerclesRec_2(x,y-1.5*r,r/2,n-1,'bas')
        if position=='bas':
            cerclesRec_2(x-1.5*r,y,r/2,n-1,'gauche')
            cerclesRec_2(x+1.5*r,y,r/2,n-1,'droite')
            cerclesRec_2(x,y-1.5*r,r/2,n-1,'bas')
        if position=='gauche':
            cerclesRec_2(x-1.5*r,y,r/2,n-1,'gauche')
            cerclesRec_2(x,y+1.5*r,r/2,n-1,'haut')
            cerclesRec_2(x,y-1.5*r,r/2,n-1,'bas')
        if position=='haut':
            cerclesRec_2(x-1.5*r,y,r/2,n-1,'gauche')
            cerclesRec_2(x+1.5*r,y,r/2,n-1,'droite')
            cerclesRec_2(x,y+1.5*r,r/2,n-1,'haut')
```

Exercice 5 – Flocon de Von Koch

Dans cet exercice, vous utiliserez des tableaux **numpy** pour représenter les points. C'est plus pratique que les listes python pour faire les calculs vectoriels.

- Si a et b représentent respectivement les points (x, y) et (x', y') alors $a+b$ représente le point $(x+x', y+y')$.
- Si r est un réel et a représente le point de coordonnées (x, y) alors $r*a$ représente le point (rx, ry).
- Si a et b sont des tableaux **numpy** alors $dot(a, b)$ représente le produit matriciel $a \times b$ (si ce produit est possible). La fonction **dot** est une fonction **numpy**.

Le mathématicien suédois Von Koch a défini la courbe du même nom dont voici les premières itérations.

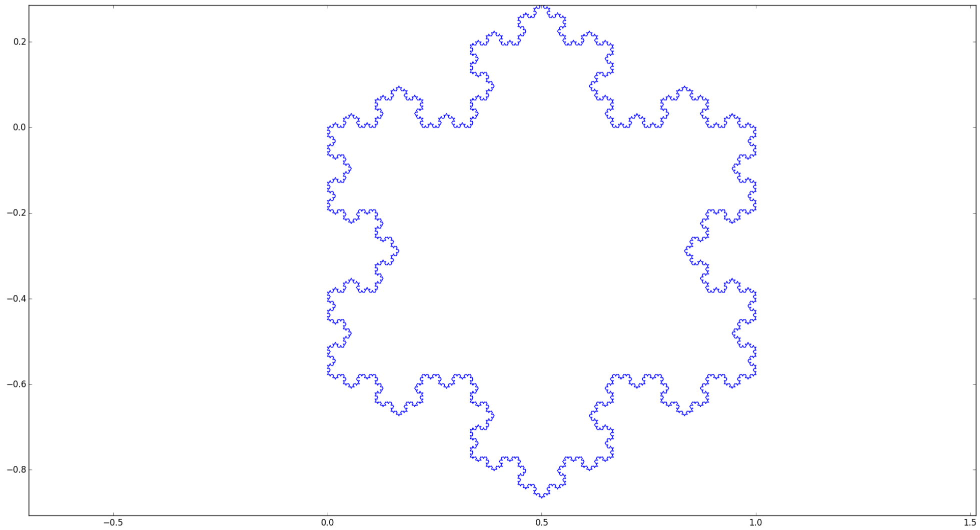


Question 1 *Ecrire une fonction* `rotation` *d'argument un réel* `alpha` *qui renvoie le tableau* ***numpy*** *correspondant à la matrice de rotation d'angle* `alpha`.

Question 2 *Pour l'étape $n = 1$, exprimer les points C et D en fonction de A et B.* En utilisant une matrice de rotation, exprimer E en fonction de C et D.

Question 3 *En déduire une fonction récursive* `koch` *d'arguments les points* `A` *et* `B` *et un entier* `n`. *Cette fonction tracera la courbe de Von Koch pour l'itération n à partir des points A et B.*

Question 4 *Ecrire une fonction* `flocon` *d'arguments les points A et B et un entier n. Cette fonction tracera le flocon de Von Koch pour l'itération n à partir des points A et B.*



Correction
```
#matrice rotation d'angle pi/3
def rotation(angle):
    return np.array([[np.cos(angle),-np.sin(angle)],[np.sin(angle),np.cos(angle)]])

# angle=np.pi/3
# print (rotation(angle))

def koch(a, b, n):
    R = rotation(np.pi/3)
    if n == 0:
        plt.plot([a[0], b[0]], [a[1], b[1]],'b')
    else:
        c=np.array([(b[0]-a[0])/3+a[0],(b[1]-a[1])/3+a[1]])
        d=np.array([2*(b[0]-a[0])/3+a[0],2*(b[1]-a[1])/3+a[1]])
        vecteur=d-c
        e=np.dot(rotation(np.pi/3),vecteur)+c
        koch(a, c, n - 1)
        koch(c, e, n - 1)
        koch(e, d, n - 1)
        koch(d, b, n - 1)

def flocon(a,b,n):
    koch(a,b,n)
    vecteur=b-a
    c=np.dot(rotation(-2*np.pi/3),vecteur)+b
    koch(b,c,n)
    koch(c,a,n)

n = 5
a = np.array([0,0])
b = np.array([1,0])
flocon(a, b, n)
plt.axis('equal')
plt.show()
```

Part. 5 – Ch. 1