L3 E - S5 2018-19 --- TD ordinateurs --- séance 4

1 Autour des nombres complexes

```
In [ ]: import os
        os.chdir('.........')
```

```
In [ ]: from pylab import *
```

Le nombre imaginaire pur i s'écrit `j` avec `python` :

```
In [ ]: z = 3 - 4.j
        print ('z=',z,'est de', type (z))
```

Attention à la syntaxe : `python` comprend `3-4.j` comme un nombre complexe mais pas `3-4*j` (où `j` est une variable quelconque)

```
In [ ]: z = 3 - 4.j
        print ('z=',z,'est de', type (z))
        z = 3 - 4*j
        print ('z=',z,'est de', type (z))
```

De la même façon :

```
In [ ]: w=-1j
        print ('w=',w,'est de', type (w))
        w = -j
        print ('w=',w,'est de', type (w))
```

Voici une autre manière de définir des nombres complexes :

```
In [ ]: u=-2.5
        v=3
        z=complex(u,v)
        print ('u=',u,'est de', type (u))
        print ('v=',v,'est de', type (v))
        print ('z=',z,'est de', type (z))
```

On peut utiliser les fonctions `real`, `imag` et `conjugate` qui font bien ce que leur nom dit :

```
In [ ]: print ('la partie réelle de z est',z.real)
        print ('la partie imaginaire de z est',z.imag)
        print('le conjugué de z est',z.conjugate())
        print( 'le module de 1+i est', abs(1+1.j))
        print ('l\'argument de 1+i est',angle(1+1.j))
```

et effectuer les opérations algébriques usuelles :

```
In [ ]: z1=-2+5.j
        print (z+z1, z*z1, z/z1, z *z1)
```

Ceci dit, certaines fonctions classiques ne fonctionneront pa si elles font appel à des calculs dans le corps des nombres complexes :

```
In [ ]: sqrt(-1)
```

La bibliothèque `cmath` permet de traiter des fonctions de la variable complexe ; par exemple, on peut importer la fonction `sqrt` de `cmath` :

```
In [ ]: from cmath import sqrt
        print (sqrt(-1))
        print (sqrt(9))
```

On trouve ainsi facilement les racines de $x^2 + x + 1 = 0$:

```
In [ ]: a,b,c=1,1,1
        x1=(-b-sqrt(b**2-4*a*c))/(2*a)
        x2=(-b+sqrt(b**2-4*a*c))/(2*a)
        print (x1)
        print (x2)
        print (x1+x2)
        print (x1*x2)
```

Voici une application graphique avec le dessin d'un pentagone avec la rotation d'angle $2\pi/5$:

```
In [ ]: z=1 +0.j
        x=[1]
        y=[0]
        for n in range (1,6):
            z=z*complex(cos(2*pi/5),sin(2*pi/5))
            x.append(z.real)
            y.append(z.imag)

        plot(x, y, "yo-")
        xlabel("abscisse x")
        ylabel("ordonnee y")
        axis([-1.2, 1.2, -1.2, 1.2])
        #legend(["un pentagone"])
        title("un pentagone")
        savefig("un pentagone.pdf")
        show()
```

Peut-être que vous préfèrerez ce dessin : (qu'est-ce qui a changé ?)

```
In [ ]: z=1 +0.j
        x=[1]
        y=[0]
        for n in range (1,6):
            z=z*complex(cos(2*pi/5),sin(2*pi/5))
            x.append(z.real)
```

```
        y.append(z.imag)
    plot(x, y, "yo-")
    xlabel("abscisse x")
    ylabel("ordonnee y")
    axis([-1.2, 1.2, -1.2, 1.2])
    axis("equal")
    #legend(["un pentagone"])
    title("un pentagone")
    savefig("un pentagone.pdf")
    show()
```

Et on peut aussi vouloir un titre plus précis :

```
In [ ]: z=1 +0.j
        x=[1]
        y=[0]
        k=5
        for n in range (1,k+1):
            z=z*complex(cos(2*pi/5),sin(2*pi/5))
            x.append(z.real)
            y.append(z.imag)
        plot(x, y, "yo-")
        xlabel("abscisse x")
        ylabel("ordonnee y")
        axis([-1.2, 1.2, -1.2, 1.2])
        axis("equal")
        #legend(["un pentagone"])
        title("un polygone régulier à %g côtés "%(k))
        savefig("un pentagone.pdf")
        show()
```

2 Exercice 1

La commande interactive `input()` permet de demander de rentrer une valeur au clavier. Si l'on écrit `n=input()`, cette valeur sera affectée à la variable `n` et on peut être plus précis en indiquant ce qu'on attend `n=input('donnez-moi un nombre entier')` ; par exemple :

```
In [ ]: n=input('entrez un entier :')
        print("n =",n)
```

Ecrire un programme qui trace un polygone régulier à n côtés (n étant la valeur demandée par l'utilisateur du programme) dans un repère orthonormé.

Puis afficher le polygone et l'enregistrer dans un fichier pdf dont le titre indique le nombre de côtés du polygone.

3 Exercice 2

La fonction `random` permet de générer des valeurs aléatoires. Par exemple, `x=random()` génèrera un nombre au hasard compris entre 0 et 1 que l'on affectera à la variable `x`.

Dans cet exercice, on va chercher à caractériser l'ensemble `E` des points `M` du plan d'affixe `z` tels que

$$|z+1|+|z-1| \leq 4$$

On peut bien entendu étudier cette inégalité en coordonnées cartésiennes. Mais ici, on va simplement prendre 2000 points au hasard (dans un rectangle censé contenir l'ensemble `E`) en marquant d'une croix chaque point `M` qui appartient `E`.

1.a. Voici un script incomplet (à vous de compléter les points de suspension !) :

```
In [ ]: x=[]
        y=[]
        for k in range(2000):
            u=4*random()-2
            v=4*random()-2
            z=...                ##### compléter !!
            if ...               ##### compléter !!
                x.append(u)
                y.append(v)

        plot(x, y, "+")
        xlabel("abscisse x")
        ylabel("ordonnee y")
        axis([...,...,...,...])    #### compléter !!
        title("TD05 - exercice 2")
        savefig("exercice2.pdf")
        grid()                  ### permet d'afficher un quadrillage du plan
        show()
```

1.b. Quel est le rectangle qu'on a choisi (comme rectangle contenant `E`) ?
Pourquoi ?

2. On considère la transformation du plan

$$f : (x,y) \mapsto \left(\frac{x}{2}, \frac{y}{\sqrt{3}}\right)$$

Tracer sur la même figure les points correspondants à E (comme dans la question 1.) et ceux correspondants à $f(E)$.

4 Exercice 3

Pour tout nombre complexe `c`, on définit *l'ensemble de Julia* J_c à partir de l'étude des suites récurrentes:

$$\begin{cases} z_0 \in \mathbf{C} \\ z_{n+1} = z_n^2 + c \end{cases}$$

Suivant la valeur z_0 de départ, la suite sera bornée ou ne sera pas bornée. Les valeurs z_0 pour lesquelles la suite est bornée sont les éléments de J_c.

Voici un programme pour tracer l'ensemble de Julia :

```
In [ ]: c=-0.8+0.2j

        a,N,Nb_iterations=1.5,1000,20
        x=linspace(-a,a,N)
        y=linspace(-a,a,N)
        U=[]
        V=[]
        for Re in x:
            for Im in y:
                z0=complex(Re,Im)
                z=z0
                p=0
                while abs(z)<2 and p<Nb_iterations:
                    p=p+1
                    z=z**2+c
                    if p==Nb_iterations:
                        U.append(z0.real)
                        V.append(z0.imag)

        plot(U,V, ".")

        axis([-2, 2, -2, 2])
        title("ensemble de Julia pour Nb_iterations=%g et depart %g + %g i"% (Nb_iterations,c.re
        savefig("ex3_ensemble-de-%g.pdf"% (Nb_iterations))
        grid()
        show()
```

1. Executer ce programme en faisant varier la variables `Nb_iterations` parmi les valeurs 10, 20, 30, 40, 50...
2. Comprendre ligne à ligne ce programme.
3. Quels sont les nombres complexes que l'on retient comme étant dans dans l'ensemble de Julia ?
4. Sur le même principe, on prend $z_0 = 0$ et l'ensemble de Mandelbrot est obtenu en ne retenant que les valeurs de la constante c pour lesquelles la suite obtenue est bornée. Dessiner l'ensemble de Mandelbrot.