

DS Informatique

Exercice 1. 1. Ecrire une fonction Python `somme` qui prend en argument un entier n et renvoie la valeur de

$$\sum_{k=1}^n \frac{(-1)^k}{k^2}$$

2. Soit f la fonction définie sur \mathbb{R} par $f(x) = \begin{cases} \frac{e^x - 1}{x} & \text{si } x \neq 0 \\ 1 & \text{sinon} \end{cases}$ Ecrire une fonction Python `f` qui prend en argument un flottant x et renvoie la valeur de $f(x)$

Exercice 2. On se propose d'écrire un programme Python qui permet de trier une liste d'entier du plus petit au plus grand : par exemple `tri([11, 4, 30, 1])` va renvoyer `[1, 4, 11, 30]`

La fonction suivante permet de retourner la valeur du plus grand élément :

```
def val_maxi(L):
    M=L[0]
    n=len(L)
    for i in range(1,n):
        if L[i]>M:
            M=L[i]
    return M
```

1. En modifiant la fonction `val_maxi` écrire une fonction `ind_maxi` qui renvoie l'indice du maximum d'une liste d'entiers. Par exemple `ind_maxi([11, 4, 30, 1])` renverra 2.
2. Ecrire une fonction `enleve_element` qui prend en argument une liste d'entier `L` et un entier `i` et renvoie la liste `L` où l'élément d'indice `i` a été enlevé. Par exemple `enleve_element([11, 4, 30, 1], 2)` renverra `[11, 4, 1]`.
3. Rappeler ce que fait l'opération `+` entre deux listes. On explicitera en particulier ce que vaut `[4]+[11, 30]`.
4. Compléter la fonction `tri` qui permet de réaliser l'opération de tri.

```
def tri(L):
    Ltrie=[]
    n=len(L)
    while ....:
        i=ind_maxi(L)
        Ltrie=.....+Ltrie
        L=enleve_element(L,i)
    return(Ltrie)
```

Exercice 3 (Ensemble de Mandelbrot). Soit $(z_n)_{n \in \mathbb{N}}$ la suite définie par $z_0 = 0$ et

$$z_{n+1} = z_n^2 + c$$

où $c \in \mathbb{C}$ est un complexe.

Selon la valeur de c , il y a deux possibilités : soit $(|z_n|)_{n \in \mathbb{N}}$ reste bornée, soit son module tends vers l'infini. Le but de ce problème est d'écrire un algorithme qui permet de tracer l'ensemble des c pour lesquels la suite $(z_n)_{n \in \mathbb{N}}$ reste bornée. Cette ensemble s'appelle l'ensemble de Mandelbrot.

Par exemple pour $c = i$ on a :

$$z_0 = 0, \quad z_1 = 0^2 + i = i, \quad z_2 = i^2 + i = i - 1, \quad z_3 = (i - 1)^2 + i = -i, \quad z_4 = (-i)^2 + i = i - 1 \dots$$

La suite semble périodique à partir de $n = 2$, car on retrouve $z_2 = z_4$. Ainsi (on peut le prouver rigoureusement), la suite $(|z_n|)_{n \in \mathbb{N}}$ reste bornée, et donc $c = i$ appartient à l'ensemble de Mandelbrot.

- Que vaut la suite $(z_n)_{n \in \mathbb{N}}$ pour $c = 0$? Est ce que $c = 0$ appartient à l'ensemble de Mandelbrot?
- Que valent les premières valeurs ($n = 0, 1, 2, 3$) de la suite $(z_n)_{n \in \mathbb{N}}$ pour $c = 1 + i$? A votre avis est-ce que $c = 1 + i$ appartient à l'ensemble de Mandelbrot?

Le nombre imaginaire i se code en Python par la syntaxe `1j`.

Ainsi le nombre complexe $2 + 3i$ se code en Python par la syntaxe `2+3*1j` (et non pas `2+3j`).

Si `z` est un nombre complexe en Python, on dispose des commandes suivantes

```
z.real #donne la valeur de la partie réelle de z
z.imag #donne la valeur de la partie imaginaire de z
```

Ainsi le script

```
z=2+3*1j
print(z.real)
print(z.imag)
```

affichera 2 puis 3.

La syntaxe des calculs algébriques sur les complexes en Python sont les mêmes que sur les réels, par exemple :

```
z=2+3*1j
z2=1+1j
z3=z*z
print(z-z2)
```

Affichera $1+2*1j$, puis $4+6*1j$

- Ecrire une fonction python qui prend en argument un complexe z et renvoie la valeur de son module.
- Ecrire une fonction Python `suite_z` qui prend en argument un entier $n \in \mathbb{N}$ et un complexe $c \in \mathbb{C}$ et qui renvoie la valeur de z_n .
- On peut montrer que c appartient à l'ensemble de Mandelbrot si et seulement pour tout $n \in \mathbb{N}$, $|z_n| < 2$. On suppose pour simplifier qu'un nombre c appartient à l'ensemble de Mandelbrot si et seulement si pour tout $n \in \llbracket 0, 100 \rrbracket$, $|z_n| < 2$. Ecrire une fonction `verif` qui prend un nombre complexe c et renvoie `True` si c appartient à l'ensemble de Mandelbrot et `False` sinon.

Pour information : voici un script qui permet de tracer l'ensemble de Mandelbrot

```
import matplotlib.pyplot as plt
def tracer(x,y):
    c=x+y*1j
    if verif(c)==True:
        plt.plot(x,y,'kx')

for x in range(-100,101):
    for y in range(-100,101):
        tracer(x/100,y/100)
plt.show()
```

