I Les nombres complexes en Python

Les nombres complexes sont pris en charge en *Python* (voir documentation à la fin).
Il est préférable cependant de savoir calculer en complexes *via* la manipulation de listes :
le complexe $z = a + ib$ ($a, b \in \mathbf{R}$) est modélisé par la liste `[a,b]`.

1. Voilà un exemple de codage pour une fonction `somme` :
```
def somme(z1,z2) :
    [a,b],[c,d] = z1,z2
    return [a+c,b+d]
```
2. Définir une fonction `produit(z1,z2)` renvoyant la liste représentant le complexe $z_1 \times z_2$.
3. Définir une fonction `module(z)` renvoyant le réel positif $|z|$.
4. Définir une fonction `quotient(z1,z2)` renvoyant la liste représentant le complexe $\dfrac{z_1}{z_2}$.
5. Utiliser la fonction `produit` et une boucle pour définir la fonction `puissance(z,n)` renvoyant z^n.
6. Définir une fonction `expi(a)` renvoyant le complexe e^{ia}.
7. La fonction `acos(x)` du module `math` renvoie l'angle $\alpha \in [0, \pi]$ tel que $\cos(\alpha) = x$.
   En utilisant la fonction `module`, écrire une fonction `arg(z)` renvoyant l'argument principal de z.
   Rappel : si $\mathrm{Im}(z) < 0$, alors $\mathrm{Arg}(z) \in]-\pi, 0[$.

II Ensembles de Julia

1 Définition

Soit une suite complexe $(z_n)_{n\in\mathbf{N}}$ définie par : $\begin{cases} z_0 \in \mathbf{C} \\ \forall n \in \mathbf{N},\ \ z_{n+1} = (z_n)^2 + c \end{cases}$
où c est un nombre complexe fixé. Deux cas peuvent alors se présenter :

- ou bien la suite des modules $(|z_n|)_n$ est bornée dans $\mathbf{R}_+$,
  c'est-à-dire qu'il existe une constante $M \geqslant 0$ telle que : $\forall n \in \mathbf{N},\ \ |z_n| \leqslant M$.
  Le complexe z_0 appartient alors à l'*ensemble de Julia complet* associé au complexe c, noté J_c.
- ou bien cette même suite n'est pas bornée, et $z_0 \notin J_c$.

2 Fonction complexe $z \longmapsto z^2 + c$

(a) Définir une fonction `f(z,c)` renvoyant la liste représentant le complexe $z^2 + c$.
On pourra utiliser les fonctions de la partie précédente.

(b) À l'aide d'une boucle, définir une fonction `suite(z0,c,n)` renvoyant le terme z_n de la suite $(z_n)_n$.

3 Seuil de calcul

Il est impossible de calculer une infinité de valeurs de la suite $(z_n)_n$ pour savoir si elle est bornée ou non. Par ailleurs, les flottants étant limités en taille, des erreurs de dépassement de capacité sont possibles lorsque $|z_n|$ devient trop grand.

On admet que, si $-1 \leqslant \mathrm{Re}(c) \leqslant 1$ et $-1 \leqslant \mathrm{Im}(c) \leqslant 1$, et s'il existe un indice n tel que $|z_n| > 1.8$, alors la suite $(z_n)_n$ n'est pas bornée.

Écrire une fonction `suiteSeuil(z0,c,n)` renvoyant le premier indice $k \leqslant n$ tel que $|z_k| > 1,8$ si un tel indice existe, et renvoie z_n sinon.

4 Visualisation d'un ensemble de Julia

Soit $c \in \mathbf{C}$ un complexe fixé, tel que $\mathrm{Re}(c), \mathrm{Im}(c) \in [-1, 1]$.

On veut créer une image indiquant pour des complexes z_0 tels que $\mathrm{Re}(z_0), \mathrm{Im}(z_0) \in \left[-\dfrac{3}{2}, \dfrac{3}{2}\right]$ si la suite $(|z_n|)$ correspondante est majorée ou non. Chaque pixel de l'image doit correspondre à un complexe z_0, et on utilisera un code couleur pour répondre à la question.

Soit $N \in \mathbf{N}$, on crée une image de taille $N \times N$.
Chaque pixel de cette image est donc repéré par un couple (i, j) avec $i, j \in [\![0, N-1]\!]$.
On utilise la fonction `np.linspace` pour définir une liste X de N flottants de $\left[-\frac{3}{2}, \frac{3}{2}\right]$.

À l'aide d'une double boucle pour $i, j \in X$, on définit le complexe z_0 par : `z0 = [X[i],X[j]]`

On utilise ensuite la fonction `suiteSeuil(z0,c,n)` pour définir le pixel de position (i, j) de l'image :

- noir si un entier est renvoyé,
- blanc si une liste est renvoyée.

L'entier N est la *résolution de l'image*.
L'entier n est la *profondeur de calcul* : pour chaque complexe z_0, on effectue au maximum n calculs pour déterminer si $z_0 \in J_c$.

On obtient le script :

```
def Julia(c,N,n) :
    A = im.new('RGB',(N,N))
    X = np.linspace(-1.5,1.5,N)
    for i in range(N) :
        for j in range(N) :
            z0 = [X[i],X[j]]
            z0 = suiteSeuil(z0,c,n)
            if type(z0) == int : pixel = (0,0,0)
            else : pixel = (255,255,255)
    A.putpixel((i,N-1-j),pixel)
    A.show()
```

III Ensemble de Mandelbrot

1 Définition

L'ensemble de Mandelbrot est l'ensemble des complexes c tels que la suite $(z_n)_{n\in\mathbf{N}}$ définie précédemment est bornée lorsque $z_0 = 0$.

2 Visualisation de l'ensemble de Mandelbrot

Modifier la fonction `Julia` ci-dessus pour coder une fonction `Mandelbrot(N,n)` permettant d'afficher l'ensemble de Mandelbrot, avec la résolution N et la profondeur de calcul n.
On testera les complexes c tels que : $\mathrm{Re}(c) \in [-2, 1]$ et $\mathrm{Im}(c) \in [-1, 1]$.

IV Améliorations possibles

- Modifier la fonction `Julia` est définissant la couleur du pixel selon le module maximal atteint au cours du calcul de z_n (par exemple par un dégradé de couleurs).
- Créer une image en juxtaposant plusieurs images données par la fonction `Julia`, pour le même complexe c et la même résolution N, mais avec des profondeurs de calcul variables.
- Créer une image en juxtaposant plusieurs images données par la fonction `Julia`, pour des complexes c variables.

V Documentation

Python permet d'effectuer des opérations sur les nombres complexes, utilisant les commandes suivantes :

`z = 2 + 1j` définit le nombre complexe $z = 2 + i$.

Le complexe i est noté `1j` en langage python.

Si la partie imaginaire est différente de 1 ou -1, l'appel est plus naturel :

`z2 = -2 + 3j` pour définir $z_2 = -2 + 3i$.

`z + z2`, `z * z2`, `z / z2`
sont des opérations prises en charge par *Python*. Les résultats sont donnés sous forme algébrique.

`z.real`, `z.imag` renvoient respectivement les parties réelle et imaginaire du complexe z.

`abs(z)` renvoie le module $\sqrt{a^2 + b^2}$ du complexe $z = a + ib$.

L'importation du module `cmath` permet d'utiliser d'autres fonctions :

`phase(z)` renvoie un argument du complexe z.

`polar(z)` renvoie un (`tuple`) : ($|z|$, θ), où θ est un argument de z.

`rect(r, θ)` renvoie sous forme algébrique le complexe $re^{i\theta}$.