TP Informatique commune 1A : Courbes

Exécuter et commenter le programme suivant :

```
import numpy as np
import matplotlib.pyplot as plt

plt.axis([0,5,-2,6])
plt.grid()
plt.xlabel('axe des abscisses')
plt.title('Kandinsky')

x1=np.array([1,2,4])
y1=np.array([1,4,3])
plt.plot(x1,y1,linestyle='None',marker='o',color='r')

x2=np.array([1,3])
y2=np.array([5,1])
plt.plot(x2,y2,linestyle='-',marker=' ',color='b', linewidth=3)

x3=np.array([2,4,3,2])
y3=np.array([-1,-1,0,-1])
plt.plot(x3,y3,'g-*')

plt.show()
```

On peut se contenter de spécifier la plage des abscisses par `plt.xlim(0,5)`, la plage des ordonnées par `plt.ylim(-2,6)`. Un repère orthonormé est obtenu grâce à `plt.axis('equal')`.

D'autres options possibles pour la couleur, le style du trait ou le *marker* :

`g`	vert
`r`	rouge
`k`	noir
`b`	bleu
`c`	cyan
`m`	magenta
`y`	jaune

`-`	trait continu
`--`	tirets
`-.`	points-tirets
`:`	pointillés
`None`	pas de trait

`.`	point
`,`	pixel
`+`	+
`x`	x
`*`	étoile
`None`	pas de marque

Exercice 1 *Les racines énièmes*

Faire un programme demandant à l'utilisateur un entier $n \geq 1$ et affichant les images dans le plan d'Argand-Cauchy des racines n-ièmes de l'unité.

Exercice 2 *La suite de Syracuse*

Faire un programme demandant à l'utilisateur un entier de départ u_0 et affichant l'ensemble du vol de la suite de Syracuse jusqu'au premier retour à 1.

Rappel: la suite est définie par $\forall n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$.

Exercice 3 *Les courbes de fonctions*

Exécuter et commenter le programme suivant qui est la manière classique de tracer des courbes avec Python.

En particulier, quelle est la nature de `X,Y1,Y2` ?

```
X=np.linspace(0,2*np.pi,200)

Y1=np.cos(X)
plt.plot(X,Y1,color='b',label='cos(x)')

Y2=np.sin(X)
plt.plot(X,Y2,color='g',label='sin(x)')

plt.legend()
plt.show()
```

Exercice 4 *Les courbes de fonctions*

1. Représenter sur un même dessin les courbes des fonctions ch et sh sur $[-5, 5]$.
2. Représenter sur un même dessin les courbes des fonctions $x \mapsto x^n$ pour $n \in [\![1, 10]\!]$ sur $[-1, 1]$.

Exercice 5 *Le mouvement Brownien*

On cherche à modéliser le mouvement aléatoire d'une particule en suspension dans un fluide, appelé *mouvement brownien*. La modélisation proposée est la suivante :

- Étape 0 : On suppose que la particule a pour coordonnées $(0, 0)$ au départ.
- De l'étape n à l'étape $n + 1$, la particule parcourt un segment de longueur 1, dans une direction formant un angle aléatoire θ_n (compris entre 0 et 2π) avec l'axe (Ox).
- Le nombre d'étapes (donc de segments parcourus par la particule) sera un entier N saisi au clavier par l'utilisateur.

Dans un deuxième temps, on pourra également modéliser la longueur du parcours par un réel au hasard choisi entre 0 et 1.

NB: on pourra utiliser la fonction `random` fournissant un réel au hasard entre 0 et 1. Cette fonction se trouve dans le module `random`.

Exercice 6 *Sommes de Weyl*

Étant donnée une suite réelle u, on considère la ligne polygonale $L(u)$ dont les sommets successifs sont les points $z_N = \sum_{n=0}^{N} e^{2i\pi u_n}$. On remarque que chaque côté de $L(u)$ est un segment de longueur 1.

D'après le critère de Weyl, si la suite u est équirépartie modulo 1, on a $z_N = o(N)$, donc la ligne $L(u)$ ne s'éloigne pas trop vite de l'origine.

Faire un programme dessinant $L(u)$.

On le testera pour :

- $u_n = n\alpha$ dans les cas $\alpha \in \{\frac{3}{11}, \sqrt{2}, \sqrt{3}, \sqrt{5}\}$;
- $u_n = \alpha n\sqrt{n}$ dans les cas $\alpha \in \{1, \sqrt{2}, \sqrt{7}\}$.

Si on note p_n le n-ième nombre premier, on a $p_n \sim n \ln n$ (Hadamard-La Vallée Poussin, 1896). Représenter les lignes $L(\sqrt{2}n \ln n)$, $L(\sqrt{2}\lfloor n \ln n \rfloor)$, $L(\sqrt{2}p_n)$, $L(\sqrt{2}\ \mathrm{randint}(1, n))$.

NB: la fonction ln est obtenue par `log` dans `numpy`. La partie entière est obtenue par `floor` (dans `numpy` également).

On peut obtenir un entier au hasard entre a et b (compris) par `randint(a,b)`. Cette fonction se trouve dans le module `random`.