

Pour les graphiques la bibliothèque à importer est `matplotlib.pyplot` que l'on importera ainsi :

```
import matplotlib.pyplot as plt
```

Pour représenter une liste `y` en fonction d'une liste `x` :

- `x` = liste des abscisses (ou tableau)
- `y` = liste des ordonnées (ou tableau)
- `plt.plot(x,y)`
- `plt.show()`

Rappel...

Par défaut, les points ainsi représentés sont reliés. Pour obtenir des points isolés, on ajoute un style en option, exemples :

- `plt.plot(x,y,'+')` : marque les points avec des +
- `plt.plot(x,y,'o')` : marque les points avec des •

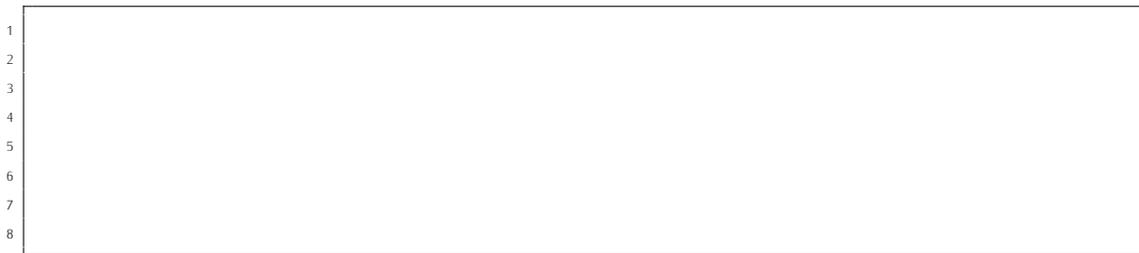
Dans cette fiche, nous allons reprendre des questions classiques sur des suites et sommes : calculs de termes, représentations graphiques, calculs de sommes, création de listes de termes...

I PETIT MÉLANGE...

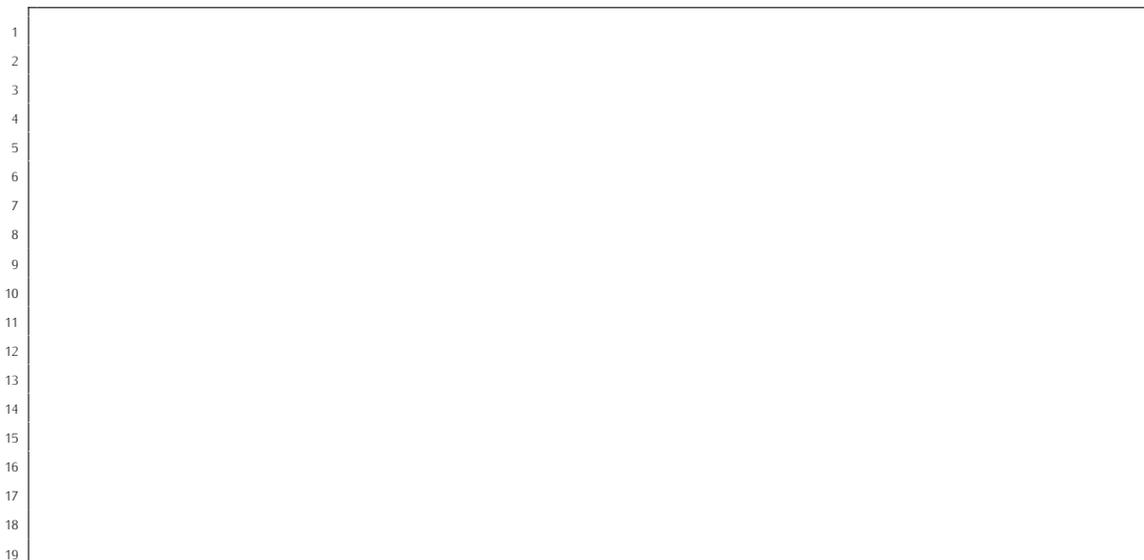
On considère les suites $(s_n)_{n \in \mathbb{N}}$, $(t_n)_{n \in \mathbb{N}}$, $(u_n)_{n \in \mathbb{N}}$, $(v_n)_{n \in \mathbb{N}}$ et $(w_n)_{n \in \mathbb{N}}$ définies par :

$$\forall n \in \mathbb{N}, s_n = \frac{n^2}{2^n} ; \begin{cases} t_0 = 1 \\ \forall n \in \mathbb{N}, t_{n+1} = e^{-t_n} \end{cases} ; \begin{cases} u_1 = 1 \\ \forall n \in \mathbb{N}^*, u_{n+1} = u_n + \frac{1}{n^2 u_n} \end{cases} ; \begin{cases} v_0 = v_1 = 1 \\ \forall n \in \mathbb{N}, v_{n+2} = v_{n+1} + 3v_n \end{cases} ; \forall n \in \mathbb{N}^*, w_n = \frac{(2n)!}{n^n}$$

1. 1.a. Écrire une fonction `liste_suite_s` prenant en argument d'entrée un entier naturel `n` et renvoyant la liste composée des valeurs `s0` à `sn`.
- 1.b. Représenter alors les termes `s0` à `s10` sur un graphique.



2. 2.a. Écrire une fonction `suite_t` prenant en argument d'entrée un entier naturel `n` et renvoyant la valeur de `tn`.
- 2.b. En utilisant la fonction `suite_t`, écrire une fonction `somme_t` prenant en argument d'entrée un entier naturel `n` et renvoyant la valeur de $\sum_{k=0}^n t_k$. Quel est l'inconvénient de cette méthode ? En proposer une autre.
- 2.c. Sans utiliser la fonction `suite_t`, écrire une fonction `liste_suite_t` prenant en argument d'entrée un entier naturel `n` et renvoyant la liste composée des valeurs `t0` à `tn`.



20
21
22
23
24
25
26
27
28
29
30
31
32

3. Écrire une fonction `suite_u` prenant en argument d'entrée un entier naturel non nul n et renvoyant la valeur de u_n .

1
2
3
4
5

4. Écrire une fonction `suite_v` prenant en argument d'entrée un entier naturel n et renvoyant la valeur de v_n .

1
2
3
4
5
6
7
8

5. A l'aide d'une liste définie en compréhension, écrire une fonction `suite_w` prenant en argument d'entrée un entier naturel non nul n et renvoyant la valeur de w_n .

1
2
3
4
5
6

Aide
La commande `np.prod(L)` renvoie le produit des nombres de la liste `L`.

6. Représenter la fonction $x \mapsto e^{-x}$, la première bissectrice ainsi que les termes de la suite $(t_n)_{n \in \mathbb{N}}$ sur un même graphique, en faisant apparaître les termes de $(t_n)_{n \in \mathbb{N}}$ en une spirale.

1
2
3
4
5
6
7
8
9
10
11
12
13
14

II SÉRIES HARMONIQUES

7. Recopier et exécuter le programme suivant :

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 L=[1/k for k in range(1,101)]
5 S=np.cumsum(L)
6 x=np.linspace(1,100,10000)
7 plt.plot(range(1,101),S, 'r+')
8 plt.plot(x,np.log(x))
9 plt.show()

```

Que permet la commande `np.cumsum(L)` ? Que met en évidence le graphique obtenu ?

8. Adapter le programme précédent afin qu'il affiche les 100 premiers termes de la suite des sommes partielles de la série $\sum_{n \geq 1} \frac{(-1)^n}{n}$. Que peut-on conjecturer ? Comment pourrait-on démontrer cette conjecture ?

```

1
2
3
4
5
6

```