TP3 : Calcul de sommes finies en **Python**

I. Avant propos

On considère dans ce TP les suites $(u_n)_{n\in\mathbb{N}}$ et (v_n) suivantes :

$$\forall n \in \mathbb{N},\ u_n = \frac{n^2}{3^n} \qquad \text{et} \qquad \begin{cases} v_0 = 1 \\ \forall n \in \mathbb{N},\ v_{n+1} = \dfrac{2\,v_n}{\mathrm{e}^{v_n} + \mathrm{e}^{-v_n}} \end{cases}$$

Objectif du TP : il s'agit d'explorer les différentes méthodes permettant le calcul des sommes partielles d'ordre n : $S_n = \sum_{k=0}^{n} u_k$ et $T_n = \sum_{k=0}^{n} v_k$.

II. Calcul des sommes partielles d'ordre n

II.1. Calcul de S_n

Il s'agit ici d'illustrer le cas où la suite (u_n) est donnée sous forme explicite.

II.1.a) Méthode itérative

- Écrire une fonction `calculSn` qui :
  - × prend en paramètre un entier `n`,
  - × renvoie la valeur de S_n.

  On pourra créer une variable `S`, initialement égale à 0, et, à l'aide d'une structure itérative, mettre à jour la variable `S` pour calculer la valeur de S_n.

- Que vaut S_0 ? S_5 ? S_{10} ? S_{100} ? S_{1000} ?

II.1.b) Utilisation des fonctionnalités de la bibliothèque numpy

- Écrire une fonction `premSuiteU` qui :
  - × prend en paramètre un entier `n`,
  - × renvoie un tableau contenant les `n` + 1 premiers termes de la suite (u_n).

  On pourra créer un tableau `U` initialement rempli de $n+1$ zéros et le remplir à l'aide d'une structure itérative.

- Que réalise l'appel `np.sum(np.arange(5))` ? Détailler le rôle de la fonction `np.sum`.

- En déduire un appel permettant de calculer S_{10}.

On souhaite maintenant utiliser les opérations algébriques sur les tableaux pour construire le tableau contenant les $n+1$ premiers termes de la suite (u_n), sans passer par une structure itérative.

- Que réalisent les appels `np.arange(11)`, `np.arange(11)**2` et `3**np.arange(11)` ?

- En déduire un appel renvoyant un tableau contenant $u_0, \ldots, u_{10}$.

- A l'aide de ce qui précède, écrire une nouvelle fonction `premSuiteUopal` qui :
  - × prend en paramètre un entier `n`,
  - × renvoie un tableau contenant les `n` + 1 premiers termes de la suite (u_n).

II.2. Calcul de T_n

Il s'agit ici d'illustrer le cas où la suite (v_n) est donnée sous forme récurrente.

II.2.a) Méthode itérative

- Écrire une fonction `calculTn` qui :
  - × prend en paramètre un entier `n`,
  - × renvoie la valeur de T_n,

  On pourra utiliser une variable auxiliaire `v` afin de calculer les différents termes de la suite (v_n).

- Que vaut T_0 ? T_{100} ? T_{10000} ?

II.2.b) Utilisation des fonctionnalités de la bibliothèque numpy

- Écrire une fonction `premSuiteV` qui :
  - × prend en paramètre un entier `n`,
  - × renvoie un tableau contenant les `n` + 1 premières valeurs de la suite (v_n).

  On utilisera la bibliothèque numpy.

- En déduire un appel permettant de calculer T_{10}.

III. Calcul des n premières sommes partielles

III.1. Calcul des n premières sommes partielles de la série $\sum u_n$

▶ Soit $n \in \mathbb{N}$. Quel lien y a-t-il entre S_n et S_{n+1} ?

▶ En tirant profit de l'égalité précédente, écrire une fonction `premSn` qui :

× prend en paramètre une variable `n`,

× renvoie un tableau contenant les `n` + 1 premières sommes partielles de la série $\sum u_n$, *i.e.* les sommes $S_0, S_1, \ldots, S_{\texttt{n}}$.

On ne devra pas effectuer d'appel à `calculSn` mais on pourra s'inspirer de son code.

▶ Comme précédemment, on aurait aussi pu tirer parti des fonctionnalités de la bibliothèque numpy. Que réalise l'appel `np.cumsum(np.arange(6))` ? Détailler le rôle de la fonction `np.cumsum`.

▶ Définir une fonction nommée `premSnbis`, utilisant la fonction `np.cumsum` et la fonction `premSuiteU`, qui

× prend en paramètre une variable `n`,

× renvoie un tableau contenant les `n` + 1 premières sommes partielles de la série $\sum u_n$, *i.e.* les sommes $S_0, S_1, \ldots, S_{\texttt{n}}$.

III.2. Calcul des n premières sommes partielles de la série $\sum v_n$

▶ Soit $n \in \mathbb{N}$. Quel lien y a-t-il entre T_n et T_{n+1} ?

▶ En tirant profit de l'égalité précédente, écrire une fonction `premTn` qui :

× prend en paramètre une variable `n`,

× renvoie un tableau contenant les `n` + 1 premières sommes partielles de la série $\sum v_n$, *i.e.* les sommes $T_0, T_1, \ldots, T_{\texttt{n}}$.

On ne devra pas effectuer d'appel à `calculTn` mais on pourra s'inspirer de son code.

À retenir : on a étudié deux méthodes en **Python** pour obtenir les éléments de (S_n).

1) La suite des sommes partielles étant une suite (grande nouvelle), on peut se servir des procédés vus en TP1.

2) On peut aussi préférer créer le vecteur des premiers éléments de la suite (u_n) et utiliser les fonctions `np.sum` ou `np.cumsum` suivant ce que l'on cherche à obtenir.

IV. Tracé des sommes partielles de $\sum u_n$

▶ Compléter le programme suivant afin qu'il permette d'effectuer le tracé des 51 premiers éléments de (S_n). On exécutera ce programme.

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 N=51
4 U = np.arange(N)**2 / 3**np.arange(N)
5 tabS = ________________
6 plt.plot(tabS,'+')
```

▶ Quelle conjecture peut-on émettre sur la nature de la série $\sum u_n$?

V. Suite des sommes partielles aux concours

V.1. EML 2015

On considère l'application $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto f(x) = x^3\, e^x$
et la suite réelle $(u_n)_{n\in\mathbb{N}}$ *définie par :* $u_0 = 1$ *et* $\forall n \in \mathbb{N}, u_{n+1} = f(u_n)$.
Il était demandé de démontrer que la série $\displaystyle\sum_{n\geqslant 1} \frac{1}{f(n)}$ converge (de somme S) et que :

$$\forall n \in \mathbb{N}^*,\ \left| S - \sum_{k=1}^{n} \frac{1}{f(k)} \right| \leqslant \frac{1}{(\mathrm{e}-1)\,\mathrm{e}^n}$$

► *En déduire une fonction* ***Python*** *qui calcule une valeur approchée de* S *à* 10^{-4} *près.* (*cf* TP2)

V.2. ECRICOME 2018

Pour tout entier naturel n non nul, on pose : $u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n)$.

- *Écrire une fonction d'en-tête* `def u(n):` *qui prend en argument un entier naturel n non nul et qui renvoie la valeur de u_n.*

Commentaire

V.3. ECRICOME 2015

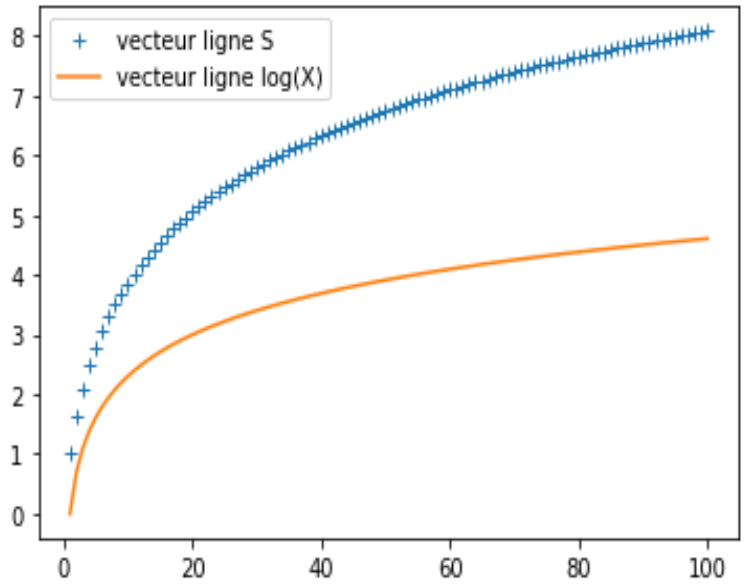
Au TP1, on a déjà introduit l'épreuve `ECRICOME 2015` qui commençait par l'étude d'une suite récurrente $(u_n)_{n\in\mathbb{N}^*}$, définie par une relation de récurrence de type $u_{n+1} = F(u_n)$.
La première question, consistait à compléter le programme permettant le calcul et le tracé des 100 premiers éléments de (u_n). On rappelle ce programme ci-dessous.

```
1  import numpy as np
2  import matplotlib as plt
3  U = np.zeros(100)
4  U[0] = 1
5  for n in range(99):
6      U[n+1] = 1 - np.exp(-U[n])
7  X = [n for n in range(1,101)]
8  plt.plot(X,U,'+')
```

► *On modifie le programme précédent en remplaçant la dernière ligne par :*

```
8  S = np.cumsum(U)
9  plt.plot(X,S,'+', label='vecteur ligne S')
10 plt.plot(X,np.log(X), label='vecteur ligne log(X)')
11 plt.legend(loc='best')
```

Le programme ci-dessus permet d'obtenir la représentation graphique suivante :



► *Que représente le vecteur-ligne* `S` *?*
Quelle conjecture pouvez-vous émettre sur la nature de la série de terme général u_n ?

▶ *À l'aide de la question* ***2.h)*** (consistait à démontrer : $\forall n \in \mathbb{N}^*,\ u_n \geqslant \frac{1}{n}$) *établir la nature de la série de terme général* u_n.