TP1 : Calcul du $n^{\text{ème}}$ terme et tracé des n premiers termes d'une suite

(Révisions sur la structure itérative `for`*)*

Commencer par importer les bibliothèques suivantes dans chaque fichier **Python** utilisé :

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

Objectifs du TP : manipuler la boucle for, calculer les termes d'une suite récurrente, faire le tour des questions **Python** classiques aux concours sur les suites.

I. Calcul du $n^{\text{ème}}$ terme d'une suite récurrente (où $n \in \mathbb{N}^*$)

Dans cette partie, on souhaite uniquement calculer le $n^{\text{ème}}$ terme de la suite (u_n), sans garder en mémoire les termes précédents.

I.1. Relation de récurrence de la forme $u_{n+1} = f(u_n)$

I.1.a) Un exemple où le premier terme est u_1

On considère la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par : $\left\{ \begin{array}{l} u_1 \ = \ 1 \\ \forall n \in \mathbb{N}^*, \ u_{n+1} \ = \ u_n + \mathrm{e}^{-u_n} \end{array} \right.$.

- Renvoyer le $n^{\text{ème}}$ terme de la suite (u_n), c'est renvoyer u_k pour quelle valeur de k ?

- Compléter le script **Python** suivant pour qu'il affiche le $n^{\text{ème}}$ terme de la suite (u_n) lorsque l'utilisateur rentre n dans la console :

```
1 n = int(input('Rentrez la valeur de n :'))
2 u = 1
3 for i in _____________:
4     u = ___________________
5 print(u)
```

- Pourquoi a-t-on utiliser `int` à la première ligne ?

► En reprenant le fonctionnement du script précédent, écrire une fonction en **Python**, nommée `suiteU1(n)`, qui prend en argument l'entier n et renvoie le $n^{\text{ème}}$ terme de la suite (u_n).

► Selon vous, quels sont les avantages de la représentation sous forme de script avec dialogue utilisateur ? Sous forme de fonction ?

I.1.b) Un exemple où le premier terme est u_0

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par : $\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = u_n^2 + \ln(1+u_n) \end{cases}$.

► Renvoyer le $n^{\text{ème}}$ terme de la suite (u_n), c'est renvoyer u_k pour quelle valeur de k ?

► Compléter la fonction **Python** `suiteU0(n)`, qui prend en argument l'entier n et renvoie le $n^{\text{ème}}$ terme de la suite (u_n).

```
1  def suiteU0(n):
2      u = 2
3      for i in _____________:
4          u = _____________________
5      return u
```

Dans la plupart des énoncés de concours, il n'est pas demandé de renvoyer « le $n^{\text{ème}}$ terme » mais plutôt u_n, qui peut être le $n^{\text{ème}}$ terme (si la suite commence par u_1) ou le $(n+1)^{\text{ème}}$ terme (si la suite commence par u_0).

▶ Compléter la fonction **Python** `suiteU0bis(n)`, qui prend en argument l'entier n et renvoie u_n.

```
def suiteU0bis(n):
    u = 2
    for i in _____________:
        u = _________________________
    return u
```

I.2. Relation de récurrence de la forme $u_{n+1} = f_n(u_n)$

On considère la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par : $\begin{cases} u_1 = 1 \\ \forall n \in \mathbb{N}^*,\ u_{n+1} = 2\,u_n + n + 1 \end{cases}$.

▶ Calculer à la main les termes u_2, u_3 et u_4.

Vous vous servirez de ces valeurs pour vérifier vos programmes.

▶ Quelle taille doit faire la boucle `for` pour calculer u_n à partir de u_1 ?

▶ Compléter les fonctions **Python** suivantes, qui prennent en argument l'entier n et renvoient u_n.

```
def suiteU(n):
    u = 1
    for i in range(n-1):
        u = ___________________
    return u
```

```
def suiteUbis(n):
    u = 1
    for i in range(1,n):
        u = _________________
    return u
```

On remarque que n'importe quelle boucle `for` de taille $n-1$ peut convenir, à condition d'écrire une formule de récurrence adaptée.
Au contraire, la formule de récurrence ne dépend pas du choix des bornes du `range` dans le cas où la fonction f ne dépend pas de n.

▶ Calculer u_4, u_7 et u_{15} à l'aide des fonctions précédentes.

II. Calcul des n premiers termes d'une suite récurrente

Dans cette partie, nous allons réviser l'utilisation des tableaux `Numpy` ainsi que des listes, pour stocker les n premiers termes d'une suite récurrente.
Pour éviter beaucoup de calculs inutiles, il ne faut pas utiliser les fonctions précédemment écrites, mais plutôt utiliser conjointement la structure de données avec une boucle `for`.

II.1. Révision sur les tableaux

On considère à nouveau la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par : $\left\{ \begin{array}{l} u_1 \ = \ 1 \\ \forall n \in \mathbb{N}^*, \ u_{n+1} \ = \ u_n + \mathrm{e}^{-u_n} \end{array} \right.$.

► Rappeler à quoi sert la commande `import numpy as np`.

► Représenter schématiquement un tableau `Numpy` contenant les n premiers termes de la suite (u_n). On fera également apparaître sous le tableau les indices des cases du tableau.

► Compléter la fonction **Python** `premSuiteU1(n)`, qui prend en argument l'entier n et renvoie un tableau `Numpy` contenant les n premiers termes de la suite (u_n).

```
1  def premSuiteU1(n):
2      T = ________________
3      T[0] = _____
4      for i in _______________:
5          ____________________________________
6      return T
```

II.2. Révisions sur les listes

On considère à nouveau la suite $(u_n)_{n\in\mathbb{N}}$ définie par : $\left\{ \begin{array}{l} u_0 \ = \ 2 \\ \forall n \in \mathbb{N}, \ u_{n+1} \ = \ u_n^2 + \ln(1+u_n) \end{array} \right.$.

► Etant donné une liste `L`, quelle commande permet de rajouter à la fin de la liste `L` un élément `a` ?

► Représenter schématiquement une liste contenant les n premiers termes de la suite (u_n). On fera également apparaître sous la liste les indices des éléments de la liste.

- Compléter la fonction **Python** `premSuiteU0(n)`, qui prend en argument l'entier n et renvoie une liste contenant les n premiers termes de la suite (u_n). Cette fonction utilise une variable auxiliaire.

```
def premSuiteU0(n):
    u = _____
    L = _______
    for i in _______________:
        ____________________________
        ________________
    return L
```

- Compléter la fonction **Python** `premSuiteU0bis(n)`, qui prend en argument l'entier n et renvoie une liste contenant les n premiers termes de la suite (u_n). Cette fonction n'utilise pas de variable auxiliaire.

```
def premSuiteU0bis(n):
    L = _______
    for i in _______________:
        ____________________________________________
    return L
```

- Selon vous, quels sont les avantages de l'utilisation de tableaux par rapport aux listes et inversement ?

III. Tracé des n premiers termes d'une suite récurrente

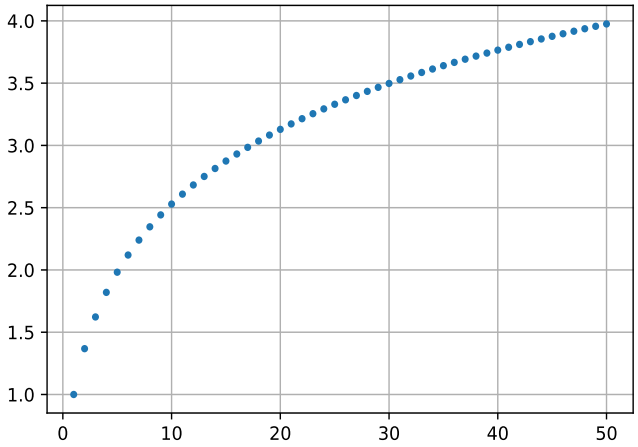
- Rappeler à quoi sert la commande `import matplotlib.pyplot as plt`.

III.1. Un exemple où le premier terme est u_1

On considère à nouveau la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par : $\begin{cases} u_1 = 1 \\ \forall n \in \mathbb{N}^*,\ u_{n+1} = u_n + \mathrm{e}^{-u_n} \end{cases}$.

► En utilisant la fonction `premSuiteU1(n)`, compléter le script **Python** suivant pour qu'il trace les 50 premiers termes de la suite (u_n). On affiche le tracé obtenu à sa droite.

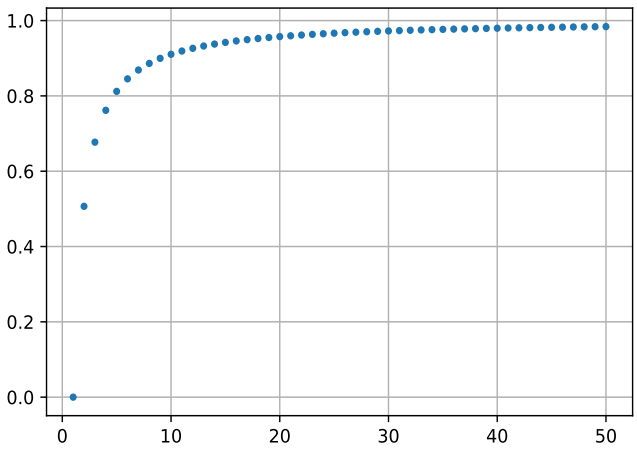
```
1  n = _______
2  Xabs = ____________________
3  Yord = ____________________
4  plt.plot(________, ________, '.')
5  plt.grid()
6  plt.show()
```



► On change la ligne 3 du script précédent et on la remplace par la ligne suivante :

```
4  Yord = np.log(np.arange(1,n+1)) / premSuiteU1(n)
```

On observe alors ce graphique :



Laquelle des quatre conjectures suivantes êtes vous amené à faire ?

❶ $u_n \underset{n\to+\infty}{\longrightarrow} \ln(n)$ ❷ $u_n \underset{n\to+\infty}{\sim} 1$ ❸ $u_n \underset{n\to+\infty}{\sim} \ln(n)$ ❹ $u_n \underset{n\to+\infty}{\sim} \mathrm{e}^n$

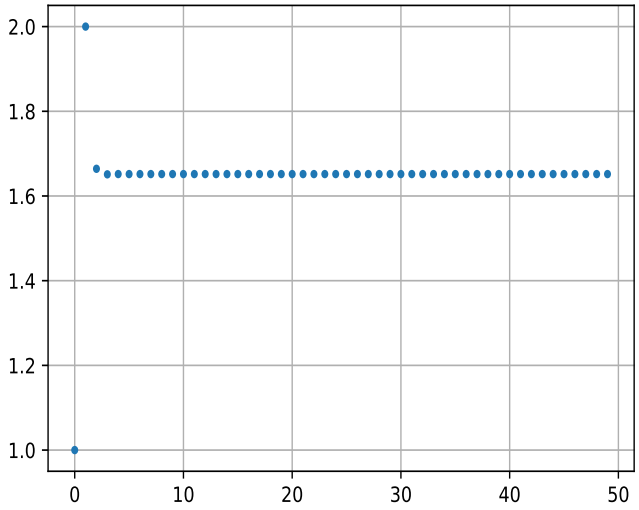
III.2. Un exemple où le premier terme est u_0

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par : $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = \dfrac{1}{u_n^2} + \sqrt{u_n} \end{cases}$.

▶ Écrire une fonction **Python**, nommée `premSuiteV(n)`, qui prend en argument un entier naturel non nul n et qui renvoie une liste ou un tableau contenant les n premiers termes de la suite (v_n).

▶ En utilisant la fonction `premSuiteV(n)`, compléter le script **Python** suivant pour qu'il trace les 50 premiers termes de la suite (u_n). On affiche le tracé obtenu à sa droite.

```
1 n = _______
2 Yord = ____________________
3 plt.plot(_________, '.')
4 plt.grid()
5 plt.show()
```



▶ Que pouvez-vous conjecturer à l'aide du tracé précédent ?

▶ Pourquoi était-il important de construire la liste `Xabs` pour effectuer le tracé dans la partie précédente et pas dans cette partie ? Quelle fonctionnalité de Python est mise en valeur ici ?

IV. Suites $u_{n+1} = f(u_n)$ aux concours

IV.1. ECRICOME - 2015

L'épreuve **ECRICOME - 2015** commençait par l'étude d'une suite récurrente de type $u_{n+1} = F(u_n)$. La fonction F est définie comme suit :

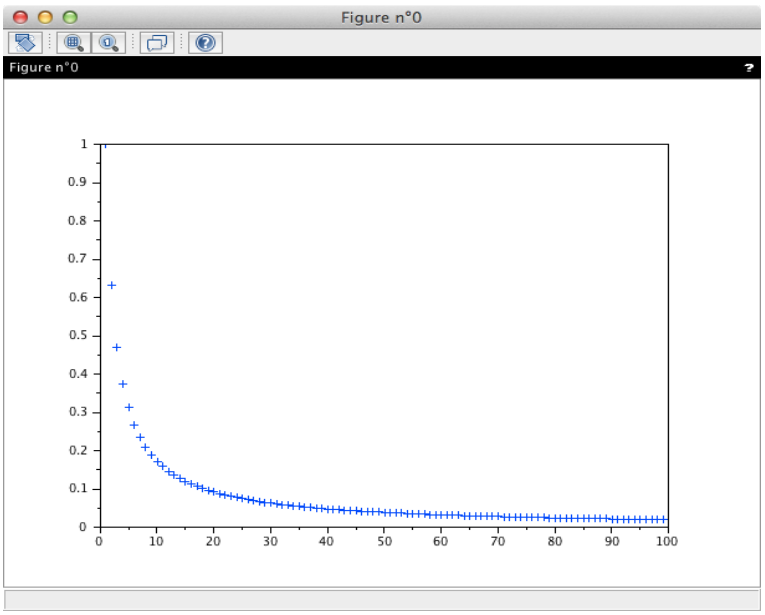
$$F(x) = \begin{cases} 0 & \text{si } x < 0, \\ 1 - \mathrm{e}^{-x} & \text{si } x \geqslant 0 \end{cases}$$

On considère la suite $(u_n)_{n\geqslant 1}$ définie par $u_1 = 1$ et pour tout $n \in \mathbb{N}^*$ par : $u_{n+1} = F(u_n)$.
On admet que : $\forall n \geqslant 1,\ u_n > 0$.

- Compléter le programme **Python** suivant qui permet de représenter les cent premiers termes de la suite $(u_n)_{n\geqslant 1}$:

```
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] = _____________________
7  X = [n for n in range(1,101)]
8  plt.plot(X,U,'+')
```

- *Le programme précédent complété permet d'obtenir la représentation graphique suivante.*



Quelle conjecture pouvez-vous émettre sur la monotonie et la limite de la suite $(u_n)_{n\geqslant 1}$?

IV.2. EML - 2018

On pose : $u_0 = 4$ et $\forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2$.

- Écrire une fonction **Python** d'en-tête `def suite(n):` qui, prenant en argument un entier n de $\mathbb{N}$, renvoie la valeur de u_n.

Commentaire

IV.3. EDHEC - 2023

On considère la fonction f définie par :

$$\forall t \in \mathbb{R},\ f(t) = \frac{1}{1+\mathrm{e}^t}$$

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par la donnée de $u_0 = 0$ et par la relation de récurrence $u_{n+1} = f(u_n)$, valable pour tout entier naturel n.

- Compléter la fonction **Python** ci-dessous afin qu'elle renvoie, pour une valeur donnée de n, la valeur de u_n à l'appel de `suite(n)` :

```
1  def suite(n):
2      u = ------
3      for k in range(1, n+1):
4          u = -----
5      return u
```

IV.4. EDHEC - 2026

On considère un nombre réel a strictement supérieur à 1, ainsi que la suite $(u_n)_{n\in\mathbb{N}}$ définie par la donnée de $u_0 = a$ et par la relation de récurrence, valable pour tout entier naturel n :

$$u_{n+1} = u_n^2 - u_n + 1$$

▶ Compléter la fonction **Python** suivante afin qu'elle retourne la valeur de u_n pour des valeurs données de n et de a :

```
1  def suite_u(a,n):
2      u=------
3      for k in range(1,n+1):
4          u=------
5      return u
```

IV.5. ECRICOME - 2018 (suite récurrente linéaire d'ordre 2 matricielle)

On considère les matrices : $A = \begin{pmatrix} 2 & 1 & -2 \\ 0 & 3 & 0 \\ 1 & -1 & 5 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & -1 & -1 \\ -3 & 3 & -3 \\ -1 & 1 & 1 \end{pmatrix}$.

On pose $X_0 = \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}$, $X_1 = \begin{pmatrix} 3 \\ 0 \\ -2 \end{pmatrix}$, et pour tout entier naturel n : $X_{n+2} = \frac{1}{6}\, A\, X_{n+1} + \frac{1}{6}\, B\, X_n$.

a) Compléter la fonction ci-dessous qui prend en argument un entier n supérieur ou égal à 2 et qui renvoie la matrice X_n :

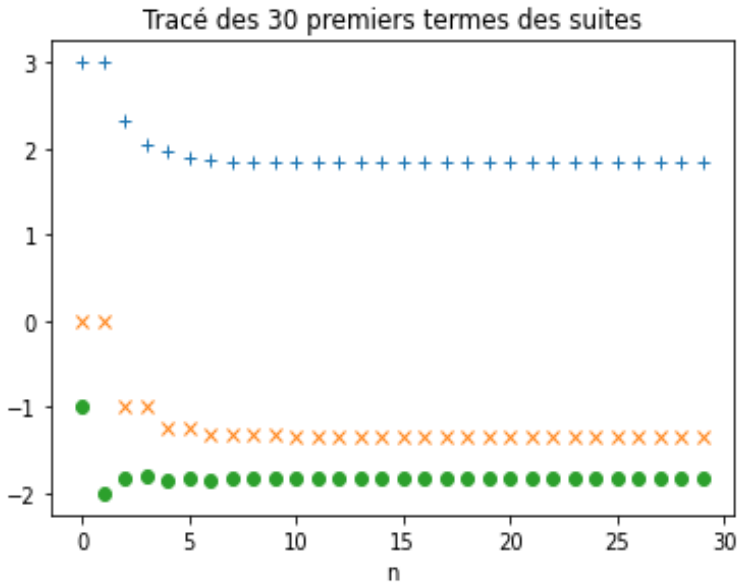
```
1  import numpy as np
2  def CalcVecteur(n):
3      A = np.array([[2,1,-2], [0,3,0], [1,-1,5]])
4      B = np.array([[1,-1,-1], [-3,3,-3], [-1,1,1]])
5      Xold = [3,0,-1]
6      Xnew = [3,0,-2]
7      for i in range(2,n+1):
8          Aux = ________
9          Xold = ________
10         Xnew = ____________________________________________________________
11     return ________
```

- Dans l'exercice, il était noté $X_n = \begin{pmatrix} \alpha_n \\ \beta_n \\ \gamma_n \end{pmatrix}$ et on démontrait, pour tout $n \in \mathbb{N}$:

$$\alpha_n = \frac{11}{6} + \frac{2}{3}\left(-\frac{1}{2}\right)^n + \left(\frac{1}{2}\right)^{n-1} - \frac{3}{2}\left(-\frac{1}{3}\right)^n$$

$$\beta_n = \left(\frac{1}{2}\right)^{n-1} - \frac{2}{3}\left(-\frac{1}{2}\right)^n - \frac{4}{3} \quad \text{et} \quad \gamma_n = -\frac{11}{6} - \frac{2}{3}\left(-\frac{1}{2}\right)^n + \frac{3}{2}\left(-\frac{1}{3}\right)^n$$

b) La fonction précédente a été utilisée dans un script permettant d'obtenir graphiquement les valeurs de α_n, β_n et γ_n en fonction de n.



Associer chacune des trois représentations graphiques à chacune des suites $(\alpha_n)_{n\in\mathbb{N}}$, $(\beta_n)_{n\in\mathbb{N}}$, $(\gamma_n)_{n\in\mathbb{N}}$ en justifiant votre réponse.

IV.6. ESSEC II - 2016 (suite récurrente dépendant de tous les termes précédents)

L'épreuve **ESSEC II - 2016** comportait une unique question d'informatique [1] qui consistait au codage d'une suite récurrente définie comme suit :

$$\begin{cases} u_0 = 1 \\ u_n = u_{n-1}\, p_1 + \ldots + u_0\, p_n \end{cases}$$

▶ *En **Python**, soit $P = [p_1, p_2, \ldots, p_n]$ la liste telle que $P(j) = p_{j+1}$ pour j dans $[\![0, n-1]\!]$.*
*Écrire une fonction en **Python**, nommée* `calcSuiteU`*, qui calcule u_n à partir de P.*

Remarque

Cette question est bien plus compliquée que ce que laisse entrevoir son énoncé. Détaillons pourquoi.

1) En terme de structures de données :

La suite (u_n) est une suite récurrente dont le $n^{\text{ème}}$ terme dépend de `TOUS` les précédents.
Ce type de suite est bien plus délicate à traiter que les suites récurrentes d'ordre 1 (dont la relation de récurrence s'écrit $u_{n+1} = f(u_n)$) ou d'ordre 2 ($u_{n+1} = g(u_n, u_{n-1})$).

Pour calculer le terme d'indice n, il faut avoir accès aux termes d'indice $0, \ldots, n-1$ de la suite.
↪ il faut donc se servir d'un vecteur (ou d'une liste) pour stocker au fur et à mesure ces valeurs.

2) En terme d'indices :

La suite (u_n) est définie à partir du rang 0 et la numérotation des listes (ou des vecteurs) commence à 0 en **Python**. Il n'y a donc pas de danger ici. Par contre, il n'en est pas de même pour le vecteur P. De plus, la formule de récurrence mélange ordre croissant pour les termes de (u_n) et décroissant pour ceux de (p_n).

3) En terme de paramètre :

La fonction consiste à calculer le $n^{\text{ème}}$ terme de la suite (u_n) mais n n'est pas annoncé comme paramètre de la fonction. En effet, le seul paramètre annoncé pour la fonction est le vecteur P. C'est la taille de ce vecteur qui nous fournit la valeur de n.

Pour ce type de questions, il est conseillé de commencer par écrire au brouillon les premières étapes de l'algorithme. Autrement dit, d'effectuer le calcul de u_0, u_1, u_2, u_3, afin de comprendre le mécanisme de calcul.

[1] Le nombre de questions d'informatique a beaucoup augmenté depuis la réforme de 2022.