TP2 : Calcul du premier entier tel qu'une condition est vérifiée

(Révisions sur la structure itérative `while`)

I. Introduction du problème

Le problème qui nous intéresse ici est le suivant.

Problème.

Données :

- Une suite (u_n) et une suite (d_n) telle que $d_n \underset{n\to+\infty}{\longrightarrow} 0$.
- L'existence d'un réel α tel que : $\forall n \in \mathbb{N},\ |u_n - \alpha| \leqslant d_n$.

But :

1) Déterminer un indice N tel que le terme u_N vérifie : $|u_N - \alpha| \leqslant 10^{-4}$.

2) En déduire une valeur approchée de α à 10^{-4} près.

Remarque

- Les inégalités du type $|u_n - \alpha| \leqslant d_n$ sont fréquentes dans les exercices. On pense notamment à l'utilisation de l'inégalité des accroissements finis pour l'étude des suites du type $u_{n+1} = f(u_n)$.
- La valeur de α n'est pas forcément connue précisément. C'est par exemple le cas lorsque α est fourni par le théorème de la bijection : on sait alors dans quel intervalle se situe α mais on ne connaît pas sa valeur exacte.
- Comme $|u_n - \alpha| \leqslant d_n$ et $d_n \underset{n\to+\infty}{\longrightarrow} 0$, on en conclut, à l'aide du théorème d'encadrement, que $|u_n - \alpha| \underset{n\to+\infty}{\longrightarrow} 0$ et donc que (u_n) est convergente, de limite α. Ceci démontre que l'élément N du point *1)* existe bien et que l'inégalité $|u_N - \alpha| \leqslant 10^{-4}$ est vérifiée à partir d'un certain rang.
- L'idée de base pour déterminer N est de calculer successivement les termes de (u_n) jusqu'à celui qui vérifie $|u_n - \alpha| \leqslant 10^{-4}$. Cependant, on ne peut procéder de la sorte si la valeur de α n'est pas connue (calcul de $u_n - \alpha$ impossible). On se sert alors de la suite (d_n).
  Il suffit en effet de déterminer un entier N tel que $d_N \leqslant 10^{-4}$. On obtient alors, par transitivité :
  $$|u_N - \alpha| \ \leqslant\ d_n \ \leqslant\ 10^{-4}$$
  ce qui permet de résoudre le problème.
- Pour l'entier N précédent déterminé, u_N est une valeur approchée de α à 10^{-4} près.

II. Un exemple classique

On commence par illustrer le problème et sa résolution par l'étude d'une suite de type $u_{n+1} = f(u_n)$ dans le cadre de l'utilisation de l'inégalité des accroissements finis.

On considère la fonction $f : x \mapsto \mathrm{e}^{-\frac{x^2}{2}}$ et on définit la suite (u_n) par : $\left\{ \begin{array}{l} u_0 = \frac{1}{2} \\ \forall n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{array} \right.$

Rappelons les différentes étapes de ce type d'étude. Les démonstrations sont laissées en exo.

1) En appliquant le théorème de la bijection à la fonction $g : x \mapsto f(x) - x$, on démontre que l'équation $f(x) = x$ admet une unique solution dans $[0, 1]$, que l'on note α.

2) Après avoir démontré que l'intervalle $[0, 1]$ est stable par f, on en déduit, par récurrence, que : $\forall n \in \mathbb{N},\ u_n \in [0, 1]$.

3) ***a)*** Par étude de la fonction f', on démontre : $\forall x \in [0, 1],\ |f'(x)| \leqslant \dfrac{1}{\sqrt{\mathrm{e}}}$.

b) On est alors dans le cadre de l'application de l'IAF, qui permet de démontrer que :

$$\forall n \in \mathbb{N},\ |u_{n+1} - \alpha| \leqslant \frac{1}{\sqrt{\mathrm{e}}}\, |u_n - \alpha|$$

(on démontre en fait que $|f(u_n) - f(\alpha)| \leqslant \dfrac{1}{\sqrt{e}}\, |u_n - \alpha|$*)*

c) On en déduit que : $\forall n \in \mathbb{N},\ |u_n - \alpha| \leqslant \left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n$.

d) Comme $\left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \underset{n \to +\infty}{\longrightarrow} 0$, on en déduit que (u_n) est convergente, de limite α.

Le but est alors de calculer une valeur approchée de α à 10^{-4} près.

- Quelle condition permet d'assurer que $|u_n - \alpha| \leqslant 10^{-4}$?

Si $\left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$, on en déduit par transitivité que $|u_n - \alpha| \leqslant \left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$.

- Écrire un script utilisant une boucle while et permettant d'afficher le premier entier n tel que $\left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$ en les testant tous à partir de 0.

```
import numpy as np
n = 0
while (1/(np.sqrt(np.exp(1))))**n > 10**(-4):
    n = n+1
print(f'la valeur de n recherchée est {n}')
```

► Compléter le programme précédent afin qu'il affiche également une valeur approchée à 10^{-4} près de α.

```
import numpy as np
n = 0
u = 1/2
while (1/(np.sqrt(np.exp(1))))**n > 10**(-4):
    u = np.exp(-u**2/2)
    n = n+1
print(f'la valeur de n recherchée est {n}')
print(f'une valeur approchée de alpha à 10^(-4) près est {u}')
```

► Déterminer une formule mathématique donnant le premier entier n tel que $\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$.

$$\begin{aligned}
\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4} \quad &\Leftrightarrow \quad n \ln\left(\frac{1}{\sqrt{\mathrm{e}}}\right) \leqslant -4\ln(10) \quad \textit{(par stricte croissance de la fonction } \ln\textit{)} \\
&\Leftrightarrow \quad -n \ln\left(\sqrt{\mathrm{e}}\right) \leqslant -4\ln(10) \\
&\Leftrightarrow \quad n \geqslant \frac{\cancel{-}4\ln(10)}{\cancel{-}\ln\left(\sqrt{\mathrm{e}}\right)} = \frac{4\ln(10)}{\ln(\mathrm{e}^{\frac{1}{2}})} = \frac{4\ln(10)}{\frac{1}{2}\ln(\mathrm{e})} = 8\ln(10)
\end{aligned}$$

Ainsi, le premier entier tel que $\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$ est le premier entier tel que : $n \geqslant 8\ \ln(10)$.

L'entier cherché est donc $\lceil 8\ \ln(10) \rceil$.

► Comparer la valeur obtenue dans la question précédente et celle affichée par le programme.

L'instruction `print(np.ceil(8*np.log(10)))` fournit bien le résultat 19 qui correspond à la valeur affichée par le programme.

► Soit $\varepsilon > 0$. Donner la formule permettant d'obtenir le premier entier n tel que $\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant \varepsilon$.

Par une étude similaire, on trouve : $\lceil -2\ \ln(\varepsilon) \rceil$.

► En déduire une fonction `calcApproch` qui prend en paramètre un réel strictement positif `eps` et qui calcule une valeur approchée de α à `eps` près à l'aide d'une boucle `for`.

```
import numpy as np
def calcApproch(eps):
    u = 1/2
    n = int(np.ceil(-2*np.log(eps)))
    for i in range(n):
        u = np.exp(-u**2/2)
    return u
```

III. Les exemples aux concours

Il est fréquent de devoir coder des programmes permettant de calculer un entier n / le premier entier n tel qu'une condition est vérifiée. On retrouve ce type d'exercice sous de nombreuses variantes.

III.1. EDHEC 2016

Dans l'épreuve `EDHEC 2016`, on considérait une suite (u_n) définie implicitement (u_n unique élément de l'intervalle $[n, +\infty[$ tel que $f_n(u_n) = 1$). Avant la question **Python**, il était demandé de démontrer :

$$\forall n \in \mathbb{N}, \quad \mathrm{e}^{-\sqrt{u_n}} \leqslant u_n - n \leqslant \mathrm{e}^{-\sqrt{n}}$$

► *Utiliser la question précédente pour compléter les commandes **Python** suivantes afin qu'elles permettent d'afficher un entier n pour lequel $u_n - n$ est inférieur ou égal à 10^{-4}.*

```
import numpy as np
n = 0
while ----
    n = ----
print(n)
```

```
import numpy as np
n = 0
while np.exp(-np.sqrt(n)) > 10**(-4):
    n = n+1
print(n)
```

► *Le script ci-dessus affiche l'une des trois valeurs $n = 55$, $n = 70$ et $n = 85$. Préciser laquelle en prenant 2,3 comme valeur approchée de* $\ln(10)$.

Par stricte croissance de la fonction ln, on a :

$$\mathrm{e}^{-\sqrt{n}} \leqslant 10^{-4} \quad \Leftrightarrow \quad -\sqrt{n} \leqslant -4\ln(10) \quad \Leftrightarrow \quad \sqrt{n} \geqslant 4\ln(10) \quad \Leftrightarrow \quad n \leqslant 16\,(\ln(10))^2$$

Or : $16\,(\ln(10))^2 \simeq 16 \times (2,3)^2 = 16 \times 5,29 \simeq 16 \times 5,3 = 80 + 4,8 = 84,8$.
Le script précédent affiche 85.

Remarque

- À première vue, on s'écarte un peu du cadre annoncé en introduction. Ce n'est pas le cas. Il suffit pour s'en convaincre de poser $v_n = u_n - n$, $\alpha = 0$ et $d_n = \mathrm{e}^{-\sqrt{n}}$.
  On retombe alors (comme $v_n \geqslant 0$) sur l'inégalité : $|v_n - \alpha| \leqslant d_n$.
- En pratique, cette question n'a pas d'intérêt algorithmique fort puisque, comme on le démontre dans la question qui suit, il est simple d'obtenir la valeur n recherchée par une étude mathématique.

On répondra **toujours** aux questions **Python** de l'**EDHEC**. Deux bonnes raisons à cela :

$\times$ elles sont abordables.

$\times$ les instructions **Python** à utiliser sont généralement rappelées dans l'énoncé.

Ce sont donc des points qu'il faut s'obliger à prendre.

III.2. EML 2016

Une partie de l'épreuve `EML 2016` consistait en l'étude d'une suite (u_n) récurrente définie par :

$$\left\{ \begin{array}{l} u_0 = \frac{1}{2} \\ \forall n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{array} \right. \qquad \text{où} \qquad f : t \mapsto \left\{ \begin{array}{cl} t^2 - t\,\ln(t) & \text{si } t \neq 0 \\ 0 & \text{si } t = 0 \end{array} \right.$$

On devait démontrer les propriétés suivantes.

- $\forall n \in \mathbb{N},\ u_n \in [\frac{1}{2}, 1]$ *(par récurrence !)*.
- La suite (u_n) est croissante *(par récurrence on démontre : $\forall n \in \mathbb{N},\ u_{n+1} \geqslant u_n$)*.
- La suite (u_n) est convergente de limite 1 *(1 est l'unique solution de l'équation $f(\ell) = \ell$)*.

▶ *Écrire un script **Python** qui calcule et affiche un entier naturel N tel que $|1 - u_N| < 10^{-4}$.*

```
import numpy as np
u = 1/2
n = 0
while abs(1-u) >= 10**(-4):
    u = u**2 - u * np.log(u)
    n = n+1
print(n)
```

Remarque

- On peut s'interroger sur l'intérêt pratique de cette question. En effet, la suite (u_n) est définie de manière explicite. On connaît aussi la valeur de α ($\alpha = 1$).
- Concédons que le N fournit permet d'avoir une idée sur la vitesse de convergence de u_n vers 1.
- Au-delà de l'intérêt pratique, c'est surtout la manipulation de **Python** qui est testée : savoir écrire une boucle `while`, savoir calculer les termes d'une suite du type $u_{n+1} = f(u_n)$.
- C'est un type de questions qui est facile à caser dans un sujet et qui peut donc permettre d'assurer la présence d'au moins une question **Python** dans un sujet (en l'occurrence, l'épreuve **EML 2016** ne comportait que cette question **Python** ...).

III.3. EML 2017

L'épreuve `EML 2017` comportait une étude de suite récurrente (u_n) définie par :

$$\left\{\begin{array}{l} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{array}\right. \qquad \text{où} \qquad f : x \mapsto \mathrm{e}^x - \mathrm{e}\ \ln(x)$$

On devait démontrer les propriétés suivantes.

- Pour tout $n \in \mathbb{N}$, u_n existe et $u_n \geqslant 2$ *(par récurrence !)*.
- Étudier les variations puis le signe de la fonction $g : \left| \begin{array}{ccc} [2, +\infty[& \to & \mathbb{R} \\ x & \mapsto & f(x) - x \end{array} \right.$.
- En déduire que la suite (u_n) est croissante *(de manière directe puisque $f(x) \geqslant x$ pour tout $x \geqslant 2$ et donc notamment pour $x = u_n \geqslant 2$)*.
- Démontrer que la suite (u_n) admet $+\infty$ pour limite *(on démontre par l'absurde que (u_n) n'est pas majorée : si elle était majorée, elle serait convergente vers l'un des zéros de la fonction g ...)*.

La question **Python** de cet exercice consistait (une nouvelle fois, quelle surprise !) à déterminer un entier N telle qu'une condition est vérifiée.

► Écrire une fonction en **Python** qui, étant donné un réel A, renvoie un entier naturel N tel que $u_N \geqslant A$.

```
import numpy as np
def calcEntier(A):
    u = 2
    n = 0
    while u < A:
        u = np.exp(u) - np.exp(1)*np.log(u)
        n = n+1
    return u
```

III.4. EML 2018

L'épreuve `EML 2018` comportait une étude de suite récurrente (u_n) définie par :

$$\begin{cases} u_0 = 4 \\ \forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2 \end{cases}$$

On devait démontrer les propriétés suivantes.

- $\forall n \in \mathbb{N}$, $u_n \geqslant b$.
- $\forall n \in \mathbb{N}$, $u_{n+1} - b \leqslant \dfrac{1}{2}\ (u_n - b)$.
- $\forall n \in \mathbb{N}$, $0 \leqslant u_n - b \leqslant \dfrac{1}{2^{n-1}}$.

► 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.

On rappelle la fonction écrite lors du TP précédent.

```
import numpy as np
def suite(n):
    u = 4
    for k in range(n):
        u = np.log(u) + 2
    return u
```

► Recopier et compléter la ligne 3 de la fonction **Python** suivante afin que, prenant en argument un réel `epsilon` strictement positif, elle renvoie une valeur approchée de b à `epsilon` près.

```
1 def valeur_approchee(epsilon):
2     n = 0
3     while ...........
4         n = n + 1
5     return suite(n)
```

```
1 def valeur_approchee(epsilon):
2     n = 0
3     while 1/2**(n-1) > epsilon:
4         n = n + 1
5     return suite(n)
```

- On cherche ici à trouver un entier N tel que u_N est une valeur approchée de b à une précision ε près (fournie par l'utilisateur). Autrement dit, on souhaite exhiber $N \in \mathbb{N}$ tel que :
$$|u_N - b| \quad \leqslant \quad 10^{-3}$$
- Or, d'après la question ***6.b)*** : $\forall n \in \mathbb{N},\ 0 \leqslant u_n - b \leqslant \dfrac{1}{2^{n-1}}$.
- Il suffit alors de trouver $N \in \mathbb{N}$ tel que : $\dfrac{1}{2^{N-1}} \leqslant \varepsilon$.
Si c'est le cas, on obtient alors, par transitivité :
$$0 \leqslant u_N - b \leqslant \frac{1}{2^{N-1}} \leqslant \varepsilon$$
- On complète alors le programme **Python** de la façon suivante :

```
3       while 1 / 2**(n-1) > epsilon
```

À la fin de la boucle, on est assuré que : $\dfrac{1}{2^{n-1}} \leqslant \varepsilon$ (on itère tant que ce n'est pas le cas).
Il reste alors à calculer la valeur approchée de b : on l'obtient par le calcul de u_n où n est la valeur obtenue à l'issue de cette boucle.

```
5       return suite(n)
```

Commentaire

- Lorsqu'on écrit une boucle `while` il est préférable de s'assurer en amont de sa terminaison. C'est bien le cas ici. En effet, la suite $(\frac{1}{2^{n-1}})_{n \geqslant 1}$ est convergente de limite 0. Ce qui signifie :
$$\forall \varepsilon > 0,\ \exists n_0 \in \mathbb{N},\ \forall n \geqslant n_0,\ \left|\frac{1}{2^{n-1}} - 0\right| < \varepsilon$$
Ainsi, quelle que soit la précision $\varepsilon > 0$ choisie au départ, on est toujours en mesure de trouver un rang n_0 à partir duquel on aura : $\dfrac{1}{2^{n-1}} < \varepsilon$.
- On pouvait déterminer, sans utiliser de boucle, un entier N tel que u_N est une valeur approchée à ε près de b. Pour ce faire, on remarque :
$$\frac{1}{2^{n-1}} \leqslant \varepsilon \quad \Leftrightarrow \quad 2^{n-1} \geqslant \frac{1}{\varepsilon} \quad \Leftrightarrow \quad (n-1)\ \ln(2) \geqslant \ln\left(\frac{1}{\varepsilon}\right) \quad \Leftrightarrow \quad (n-1) \ \geqslant \ \frac{-\ln(\varepsilon)}{\ln(2)}$$
L'entier $N = \left\lceil \dfrac{-\ln(\varepsilon)}{\ln(2)} \right\rceil$ convient.

III.5. ECRICOME 2018

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

On démontrait dans cet exercice que la suite (u_n) était convergente, vers une limite notée $\gamma \in \mathbb{R}$ puis :

$$\forall n \in \mathbb{N}^*,\ |u_n - \gamma| \leqslant \frac{1}{n}$$

- On rappelle que l'instruction `np.floor(x)` renvoie la partie entière d'un réel x et on suppose que la fonction **Python**, nommée `u`, de la question ***1.e)*** a été correctement programmée. Expliquer l'intérêt et le fonctionnement du script ci-dessous :

```python
import numpy as np
eps = float(input('Entrer un réel strictement positif : '))
n = int(np.floor(1/eps)) + 1
print(u(n))
```

- Ce script a pour but d'afficher une valeur approchée de γ à ε près (où ε est un réel strictement positif fourni par l'utilisateur et stocké dans la variable `eps`).
  Pour ce faire, il faut commencer par trouver un entier $N \in \mathbb{N}^*$ tel que :

$$\left|u_N - \gamma\right| \ \leqslant\ \varepsilon$$

- Or, d'après ce qui précède : $\forall n \in \mathbb{N}^*,\ \left|u_n - \gamma\right| \ \leqslant\ \frac{1}{n}$.
- Afin de trouver l'entier N recherché, il suffit de trouver un entier $N \in \mathbb{N}^*$ tel que :

$$\frac{1}{N} \ \leqslant\ \varepsilon$$

  Si c'est le cas, on obtient alors, par transitivité :

$$\left|u_N - \gamma\right| \ \leqslant\ \frac{1}{N} \ \leqslant\ \varepsilon$$

- Raisonnons par équivalence pour trouver N :

$$\frac{1}{n} \ \leqslant\ \varepsilon \ \Leftrightarrow\ n \ \geqslant\ \frac{1}{\varepsilon} \qquad \textit{(par décroissance de la fonction inverse sur }]0,+\infty[\textit{)}$$

  Ainsi, tout entier plus grand que $\frac{1}{\varepsilon}$ convient. En particulier, l'entier $N = \lfloor \frac{1}{\varepsilon} \rfloor + 1$ convient.

Ce script affiche la valeur u_N où $N = \lfloor \frac{1}{\varepsilon} \rfloor + 1$. C'est une valeur approchée de γ à ε près.