APPLICATIONS AUX SUITES

Exercice 1 –

1. a) Écrire une fonction `Python` qui calcule et affiche le terme u_n de la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 10$ et $u_{n+1} = \dfrac{u_n}{2} + \dfrac{1}{u_n}$.

Solution : Tous les termes u_i seront contenus successivement dans une variable `u`.

```
1.  def calculu(n):
2.      u=10
3.      for i in range(n):
4.          u=u/2+1/u
5.      return(u)
```

je déclare la fonction
j'initialise la variable u avec le premier terme u_0
n répétitions, pour i allant de 0 à $n-1$
je calcule u_{i+1} en fonction de u_i
je renvoie u qui contient u_n en sortie

b) Afficher les dix premiers termes de la suite. Que remarquer quant aux valeurs prises par cette suite ? Sont-ce ces valeurs identifiables ?

Solution : Très vite, je m'aperçois que les termes de la suite deviennent égaux.
Il s'agit là d'une suite convergente dont la limite semble être $\sqrt{2}$.
Je remarque aussi que la convergence est extrêmement rapide puisque avant le dixième terme, la valeur approchée affichée est celle de l'approximation de $\sqrt{2}$.

c) On souhaite tracer le nuage de points représentant cette suite $(u_n)_{n\in\mathbb{N}}$. Pour cela, il faut modifier la fonction de calcul d'**un terme** pour qu'elle stocke et renvoie **l'ensemble des premiers termes** de la suite. Alors on peut effectuer le tracé comme pour une fonction :

- Plutôt que `np.linspace(a,b,n)`, on utilise `np.arange(n+1)` qui construit la liste des entiers allant de 0 à n.
- Plutôt que `plt.plot(X,Y)`, on utilise `plt.scatter(X,U)` qui affiche un nuage de points à la place d'une ligne brisée.

Solution : Je modifie la fonction `calculu` pour qu'elle retourne la liste des n premiers termes de la suite puis je demande le tracé.

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  def calculU(n):
4.      U=np.zeros(n)
5.      u=10
6.      U[0]=u
7.      for i in range(n-1):
8.          u=u/2+1/u
9.          U[i+1]=u
10.     return(U)
11. X=np.arange(10)
12. plt.scatter(X,calculU(10))
13. plt.show()
```

j'importe numpy pour les tableaux
j'importe pyplot pour le tracé
je déclare la fonction
je crée un tableau de zéros
j'initialise la variable u
je stocke le premier terme
$n-1$ termes restants à calculer
je calcule le terme suivant
je stocke le terme calculé
je renvoie le tableau des valeurs
je crée le tableau des abscisses
je crée le nuage de points
j'affiche le tracé

2. a) Soit $(v_n)_{n\in\mathbb{N}^*}$ la suite réelle définie pour tout entier $n \geqslant 1$ par $v_n = \left(1+\dfrac{1}{n}\right)^n$.
Afficher les premiers termes de la suite et estimer la limite de cette suite.

Solution : Cette fois, la suite est définie explicitement donc je n'ai plus besoin d'initialiser la variable `v`. Je fais en revanche bien attention à commencer à $n = 1$.

```
1. for n in range(1,11):      Pour tout n entre 1 et 10,
2.     v=(1+1/n)**n           je calcule v_n
3.     print(v)               et je l'affiche.
```

La convergence est cette fois beaucoup plus lente et il me faut afficher plus d'un millier de termes pour remarquer que la limite de cette suite semble être e.

b) Écrire une fonction `Python` qui prend en entrée une distance $d > 0$ et renvoie en sortie le rang du premier terme de la suite se trouvant à une distance inférieure à d de la limite. Quel est le rang du premier terme à moins de 0.001 de la limite ?

Solution : J'utilise une boucle `while` pour que celle-ci s'arrête dès que $|e - v_n| < d$.

```
1. import numpy as np                  j'importe numpy pour e et abs
2. def distance(d):                    je déclare la fonction
3.     n=1                             j'initialise n à 1
4.     v=2                             j'initialise v à v_1 = 2
5.     while np.abs(np.e-v)>d:         tant que v n'est pas suffisamment proche
6.         n=n+1                       j'incrémente n de 1
7.         v=(1+1/n)**n                je calcule le terme suivant
8.     return(n)                       je renvoie la valeur qui fait sortir de la boucle
9. print(distance(0.001))              j'exécute pour d = 0.001
```

Il s'agit donc du 1359-ième terme.

3. 32 000 euros sont placés sur un compte rémunéré à un taux annuel de 1%.
Écrire un programme `Python` permettant de savoir au bout de combien d'années le montant placé sur ce compte dépassera 40 000 euros.

Solution : Je raisonne avec une boucle `while`.

```
1. n=0                         j'initialise n à 0
2. u=32000                     j'initialise u à 32 000
3. while u<40000:              tant que u ne dépasse pas 40 000
4.     n=n+1                   j'incrémente n de 1
5.     u=u*1.01                je calcule le montant l'année suivante
6. print(n)                    je renvoie la valeur pour laquelle je sors de la boucle
```

Au bout de 23 années, le capital aura dépassé 40 000 euros.