Python - TP1
Calculer les termes d'une suite

Commandes utiles

La bibliothèque `numpy`, à importer via la commande `import numpy as np`, fournit les commandes pour les fonctions usuelles.

Objectif

Calculer les termes d'une suite, pour différents modes de génération d'une suite...

1. **Suites récurrentes**

   Pour chacune des suites définies ci-dessous, écrire une fonction prenant en argument d'entrée un entier naturel n ($n \in \mathbb{N}$ ou $n \in \mathbb{N}^*$) et renvoyant u_n en sortie.

   1.a. $\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = 3u_n^2 - 1 \end{cases}$

```python
import numpy as np

def suite_u(n):
    u=2 #correspond à u_0
    for k in range(1,n+1): #on commence à calculer u_1
        u=0.5*u**2-1
    return u
```

   On peut aussi en donner une écriture récursive :

```python
import numpy as np

def suite_u(n):
    if n==0:
        return 2
    else:
        return 0.5*suite_u(n-1)**2-1
```

   1.b. $\begin{cases} u_1 = 0 \\ \forall n \in \mathbb{N}^*,\ u_{n+1} = e^{-u_n} \end{cases}$

```python
import numpy as np

def suite_u(n):
    u=0 #correspond à u_1
    for k in range(2,n+1): #on commence à calculer u_2
        u=np.exp(-u)
    return u
```

   1.c. $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = 1 + \dfrac{u_n}{n+1} \end{cases}$

```python
import numpy as np

def suite_u(n):
    u=1
    for k in range(1,n+1):
        u=1+u/k #ATTENTION, k et pas k+1 ou n+1 ou n
    return u
```

   1.d. $\begin{cases} u_0 = 1,\ u_1 = 1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$

```python
def suite_u(n):
    if n==0:
        return 1
    elif n==1:
        return 1
    else :
        u,v=1,1
        for k in range (2,n+1):
            u,v=v,u+v
        return v
```

Objectif secondaire

En admettant que la suite (u_n) n'est pas majorée, écrire une fonction renvoyant le premier rang n_0 à partir duquel u_n dépasse 10^{10}.

Petite remarque

On veut $n \in \mathbb{N}^*$ ici...

Important !

En écrivant `for k in range(1,n+1)`, la lettre k permet en fait d'indiquer le rang du terme calculé. Par conséquent, la ligne 6 revient à faire $u_k = 1 + \frac{u_{k-1}}{k}$, ce qui correspond bien à la relation de récurrence sur (u_n)... En particulier, ce n'est ni `u=1+u/(k+1)` (ce qui donnerait par exemple $u_1 = 1 + \frac{u_0}{2}$), ni `u=1+u/(n)` (ce qui reviendrait à prendre toujours la même valeur, puisque n est fixé).

✘ Attention !

On veut que le programme fonctionne également pour $n = 0$ et $n = 1$...

On peut aussi en donner une écriture récursive :

```
def suite_u(n):
    if n==0:
        return 1
    elif n==1:
        return 1
    else:
        return suite_u(n-1)+suite_u(n-2)
```

1.e. $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = \displaystyle\sum_{k=0}^{n} \frac{u_k}{k^2+1} \end{cases}$

On remarque que $u_1 = 1$, puis :

$$\forall n \in \mathbb{N}^*,\ u_{n+1} = \sum_{k=0}^{n-1} \frac{u_k}{k^2+1} + \frac{u_n}{n^2+1}$$

C'est à dire :

$$\forall n \in \mathbb{N}^*,\ u_{n+1} = u_n + \frac{u_n}{n^2+1} = u_n \frac{n^2+2}{n^2+1}$$

```
def suite_u(n):
    if n==0:
        return 1
    elif n==1:
        return 1
    else:
        u=1
        for k in range(2,n+1):
            u=u*((k-1)**2+2)/((k-1)**2+1)
        return u
```

2. Suites imbriquées

Pour chacune des suites définies ci-dessous, écrire une fonction prenant en argument d'entrée un entier naturel n ($n \in \mathbb{N}$) et renvoyant les termes de rang n des suites en sortie.

2.a. $\begin{cases} u_0 = 1\ ;\ v_0 = 0 \\ \forall n \in \mathbb{N},\ u_{n+1} = u_n + v_n \text{ et } v_{n+1} = u_n + 2v_n \end{cases}$

```
def suites_uv(n):
    u,v=1,0
    for k in range(1,n+1):
        u,v=u+v,u+2*v
    return u,v
```

Petite remarque

Penser aux affectations simultanées...

2.b. $\begin{cases} u_0 = 1\ ;\ v_0 = 0\ ;\ w_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = 3u_n - v_n + w_n \text{ et } v_{n+1} = u_n + v_n + w_n \text{ et } w_{n+1} = -u_n + 2v_n + 3w_n \end{cases}$

```
def suites_uvw(n):
    u,v,w=1,0,2
    for k in range(1,n+1):
        u,v,w=3*u-v+w,u+v+w,-u+2*v+3*w
    return u,v,w
```

3. Suites définies par une somme ou un produit

Pour chacune des suites définies ci-dessous, écrire une fonction prenant en argument d'entrée un entier naturel n ($n \in \mathbb{N}$ ou $n \in \mathbb{N}^*$) et renvoyant u_n en sortie.

3.a. $\forall n \in \mathbb{N},\ u_n = \displaystyle\sum_{k=0}^{n} \frac{k}{2^k}$

Objectif secondaire

Comparer, pour tout $n \in [\![0;20]\!]$, u_n avec $2-(n+2)\left(\frac{1}{2}\right)^n$.

```
def suite_u(n):
    S=0
    for k in range(0,n+1):
        S=S+k/2**k
    return S

N=0
for k in range(0,21):
    x=2-(k+2)*(1/2)**k
    if suite_u(k)==x:
        N=N+1 #N compte le nombre de fois où les deux expressions sont égales
print(N)
```

3.b. $\forall n \in \mathbb{N}^*,\ u_n = \sum_{k=1}^{n} \frac{1}{k^2}$

```
def suite_u(n):
    S=0
    for k in range(1,n+1):
        S=S+1/k**2
    return S
```

3.c. $\forall n \in [\![2;+\infty[\![,\ u_n = \prod_{k=2}^{n}\left(1-\frac{1}{k^3}\right)$

```
def suite_u(n):
    P=1
    for k in range(2,n+1):
        P=P*(1-1/k**3)
    return P
```

Petite remarque

Pour le calcul des sommes, deux façons de faire :
- on commence par calculer le premier terme, et on initialise sur cette valeur ; puis on ajoute à partir du second terme...
- on commence par initialiser à 0, puis on ajoute à partir du premier terme...

Analogue pour les produits, il faut juste initialiser à 1 si on ne calcule pas le premier facteur !