TP # Révisions

Sommes, produits, suites...

Exercice 1

1. Compléter la fonction `somme` (resp. `produit`) pour qu'elle renvoie la valeur de la somme $\sum_{k=0}^{n} \frac{1}{(2k+1)^2}$ (resp. du produit $\prod_{i=2}^{n}\left(1-\frac{1}{i^2}\right)$)

```
def somme(n) :

  s = 0

  for k in range(n+1) :

    s = s + 1 / (2*k+1) **2

  return(s)
```

```
def produit(n) :

  p = 1

  for i in range(2,n+1) :

    p = p * ( 1 - 1/i**2 )

  return(p)
```

2. Proposer une autre définition de ces fonctions, sans boucle `for`.

```
import numpy as np
def somme(n) :

    L = [1/(2*k+1)**2 for k
            in range(n+1)]

    S = np.sum(L)

    return S
```

```
import numpy as np
def produit(n) :

    L = [1-1/i**2 for i
            in range(2,n+1)]

    P = np.prod(L)

    return P
```

Exercice 2

1. Pour $0 \leqslant p \leqslant n$, écrire comme un produit : $\binom{n}{p} = \prod_{k=0}^{p-1} \frac{n-k}{p-k}$

2. Compléter la fonction suivante pour qu'elle renvoie la valeur de $\binom{n}{p}$.

```
def binomial(p,n) :

  x = 1

  for k in range(p) :

    x = x * (n-k) / (p-k)

  return(x)
```

Exercice 3

On définit les suites récurrentes $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}^*}$ suivantes :

$$u_0 = 1 \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = \frac{e^{u_n}}{u_n}, \quad v_1 = 2 \text{ et } \forall n \in \mathbb{N}^*,\ v_{n+1} = \ln(v_n + 1).$$

1. Compléter `suiteU` (resp. `suiteV`) pour qu'elle renvoie la valeur de u_n (resp. v_n).

```
import numpy as np

def suiteU(n) :

  u = 1

  for k in range(n) :

    u = np.exp(u)/u

  return(u)
```

```
import numpy as np

def suiteV(n) :

  v = 2

  for k in range(1,n) :

    v = np.log(v+1)

  return(v)
```

2. Compléter la fonction `vectU` (resp. `vectV`) pour qu'elle renvoie un vecteur contenant les n premiers termes de la suite u (resp. v)

```
import numpy as np

def vectU(n) :

  U = np.zeros(n)

  U[0] = 1

  for k in range(1,n) :

    U[k] = np.exp(U[k-1])/U[k-1]

  return(U)
```

```
import numpy as np

def vectV(n) :

  V = np.zeros(n)

  V[0] = 2

  for k in range(n-1) :

    V[k+1] = np.log(V[k]+1)

  return(V)
```

Exercice 4

Pour tout $n \in \mathbb{N}^*$, soit $u_n = \left(1+\frac{1}{n}\right)^n$.

Construire (sans `for`) le vecteur contenant les 100 premiers termes de la suite u.

```
A = np.arange(1,101) # vecteur [1,2,3,...,100]

U = ( 1 + 1/A ) ** A

print(U)
```

Exercice 5

Proposer une fonction qui prend en entrée un réel $a > 0$ et renvoie le plus petit indice $n \geqslant 2$ tel que $\dfrac{\ln(n)}{\sqrt{n}} \leqslant a$. (Pourquoi un tel indice existe bien ?)

```
import numpy as np

def premierIndice(a) :

    n = 2 ; u = np.log(2)/np.sqrt(2)

    while u > a :

        n = n+1
        u = np.log(n)/np.sqrt(n)

    return n
```

Exercice 6

On pose $a_0 = 1$, $b_0 = 2$ et $\forall n \in \mathbb{N}$, $a_{n+1} = \sqrt{a_n b_n}$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$.

On admet que (a_n) est croissante, (b_n) décroissante et $\lim\limits_{n\to+\infty}(a_n - b_n) = 0$.

Ces deux suites sont ainsi adjacentes, donc convergent vers un même réel ℓ.

1. Pour tout $n \in \mathbb{N}^*$, quel encadrement de ℓ peut-on donner ?

$$a_n \leqslant \ell \leqslant b_n$$

2. Compléter la fonction suivante, qui prend en entrée un réel $\varepsilon > 0$, pour qu'elle renvoie un encadrement de la valeur de ℓ à ε près.

```
import numpy as np

def approx(eps) :

  a = 1 ; b = 2

  while b-a > eps :

        a1 = np.sqrt(a*b) ; b1 = (a+b)/2

        a = a1 ; b = b1

  return ( a , b )
```

3. Et si l'on veut juste une approximation de ℓ à ε près par valeur inférieure ?

```
return a
```

Exercice 7

On pose $w_0 = 1$, $w_1 = 2$ et $\forall n \in \mathbb{N}$, $w_{n+2} = \dfrac{w_{n+1}}{w_n}$.

1. Compléter la fonction `suiteW` pour qu'elle renvoie la valeur de w_n.
(Attention, elle devra fonctionner correctement pour $n = 0$ et $n = 1$!)

```
def suiteW(n) :

    w0 = 1 ; w1 = 2

    for k in range(n) :

        a = w1/w0

        w0 = w1

        w1 = a


    return( w0 )
```

2. Compléter la fonction `vectW` pour qu'elle renvoie le vecteur $[\,w_0, w_1, \ldots, w_n\,]$.

```
import numpy as np

def vectW(n) :

  W = np.zeros(n+1) ; W[0] = 1 ; W[1] = 2

  for k in range(2,n+1) :

    W[k] = W[k-1]/W[k-2]


  return( W )
```