TP 04 – Révisions

I Fonctions

Exercice 1 Soit $f : \mathbb{R} \to \mathbb{R}$ la fonction définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = \begin{cases} x^2 - 2x + 1 x > 0 \\ \exp(x) & \text{sinon.} \end{cases}$$

Écrire une fonction `f` qui prend en argument un réel `x` et qui renvoie la valeur de `f(x)`. Que doit renvoyer le programme pour `x=1` ? Vérifier le.

Entrée [1]:
```
import numpy as np
def f(x):
    if x>0:
        return(x**2-2*x+1)
    else:
        return(np.exp(x))
```

Entrée [2]:
```
#Test pour x=1
f(1)
```

Out [2]:
```
0
```

Exercice 2 Ecricome 2019, Maths E

Pour tout entier naturel n non nul, on note h_n la fonction définie sur $\mathbb{R}_+^*$ par :

$$\forall x > 0, \qquad h_n(x) = x^n + 1 + \frac{1}{x^n}$$

Écrire une fonction `Python`, appelée `h`, qui renvoie la valeur de $h_n(x)$ lorsqu'on lui fournit un entier naturel n non nul et un réel $x \in \mathbb{R}_+^*$ en entrée.

> « *L'écriture de la ligne calculant explicitement $h_n(x)$ par l'ordinateur ne posant pas vraiment de difficulté, les correcteurs examinent avec attention si les candidats comprennent ce que signifie le fait d'écrire une fonction.* »

Entrée [3]:
```
def h(n,x):
    y=x**n+1+1/(x**n)
    return(y)
```

Vérifier que l'évaluation de la suite en $n = 2$ et $x = 1$ renvoie 3.

Entrée [4]:
```
h(2,1)
```

Out [4]:
```
3.0
```

II Suites

Exercice 3 On considère la suite $(u_k)_{k\in\mathbb{N}}$ définie par

$$\forall k \in \mathbb{N}, \qquad u_k = \frac{1}{k^2+1}$$

1. Écrire une fonction, appelée `suite`, qui prend en argument un entier `n` et qui renvoie u_n. *On vérifiera que l'évaluation de* `suite` *en* `5` *donne environ* `0.038`.

Entrée [5]:
```
def suite(n):
    u = 1/(n**2+1)
    return(u)
```

Entrée [6]:
```
suite(5)
```

Out [6]:
```
0.038461538461538464
```

2. Écrire une fonction, appelée `listesuite`, qui prend en argument un entier `n` et qui renvoie la liste de tous les termes de la suite (depuis u_0) jusqu'au `n`-ième. *On vérifiera que l'évaluation de* `listesuite` *en* `5` *donne une liste dont le premier terme est* `1` *et le dernier terme est environ* `0.038`.

Entrée [7]:
```
def listesuite(n):
    L=[]
    for k in range(0, n+1):
        u = 1/(k**2+1)
        L.append(u)
    return(L)
```

Entrée [8]:
```
listesuite(5)
```

Out [8]:
```
[1.0, 0.5, 0.2, 0.1, 0.058823529411764705, 0.038461538461538464]
```

Exercice 4 ECRICOME 2022, Maths E

Pour tout réel $x > 0$, on pose :

$$g(x) = \exp\left(\left(2 - \frac{1}{x}\right)\ln(x)\right)$$

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par son premier terme $u_0 > 0$ et la relation de récurrence :

$$\forall n \in \mathbb{N}, \qquad u_{n+1} = g(u_n)$$

Écrire une fonction `Python` qui prend en argument un réel `u0` et un entier `n` et renvoie sous la forme d'une liste les $n+1$ premières valeurs de la suite $(u_n)_{n\in\mathbb{N}}$ de premier terme u_0 =`u0`.

Entrée [9]:
```
import numpy as np

def listesuite(u0,n):
    u=u0
    L=[u]
    for k in range(1,n+1):
        u=np.exp( (2-1/u) * np.log(u) )
        L.append(u)
    return L
```

Vérifier que l'évaluation de la suite en `(1,5)` renvoie une liste de six termes ne contenant que des un. Comment expliquez-vous cela ?

Entrée [10]:
```
listesuite(1,5)
```

Out [10]:
```
[1, 1.0, 1.0, 1.0, 1.0, 1.0]
```

Exercice 5 EMLyon 2023, Maths E

Pour $x \in]0, +\infty[$, on pose : $f(x) = \frac{e^{-x}}{x}$.
On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et par la relation de récurrence $u_{n+1} = f(u_n)$ valable pour tout entier naturel n.

1. Recopier et compléter la fonction Python suivante afin que l'appel `fonc_1(a)` renvoie le plus petit entier n tel que $u_n > a$.

Entrée [11]:
```
def fonc_1(a):
    from numpy import exp
    u=1
    n=0
    while u<=a :
        u=exp(-u)/u
        n=n+1
    return n
```

2. On considère maintenant la fonction `Python` :

```
def fonc_2(a):
    from numpy import exp
    u=1
    n=0
    while u>a:
        u=exp(-u)/u
        n=n+1
    return n
```

Les appels `fonc_1(10**6)` et `fonc_2(10**(-6))` donnent respectivement 6 et 5. Qu'en déduire pour u_5 et u_6 ? Commenter ce résultat en une ligne.

Cela signifie que $u_6 > 10^6$ (et que c'est le premier terme vérifiant cette inégalité) et que $u_5 \leqslant 10^{-6}$ (et que c'est le premier terme vérifiant cette inégalité). La suite semble donc osciller fort (puisque le cinquième terme est très petit tandis que le suivant, le sixième, est très grand) et donc diverger.

3. Écrire une fonction Python qui a pour argument un entier n et qui renvoie la valeur de u_n.
On vérifiera que l'évaluation de cette fonction en `6` *renvoie environ* `1007430`.

Entrée [12]:
```
import numpy as np

def suite(n):
    u=1
    for k in range(1, n+1):
        u=np.exp(-u)/u
    return (u)
```

Entrée [13]:
```
suite(6)
```

Out [13]:
```
1007430.5938296199
```

Exercice 6 On considère la suite $(u_n)_{n\in\mathbb{N}}$ telle que $u_0 = 1$ et

$$\forall n \in \mathbb{N}, \quad u_{n+1} = 2(\ln(2))^{n+1} - (n+1)u_n.$$

1. Écrire des instructions qui permettent de créer la liste $L = [u_0, u_1, \dots, u_{20}]$. La suite (u_n) semble t-elle convergente ? *On vérifiera que le dernier terme de cette liste vaut environ* `50.40`.

Entrée [14]:
```
u=1
L = [u]
for k in range(20):
    u = 2*np.log(2)**(k+1) - (k+1)*u
    L.append(u)
print(L)
```

2. Écrire une fonction qui, étant donné n, renvoie la valeur de u_n. *On vérifiera que l'évaluation de cette fonction en* `20` *donne environ* `50.40`.

Entrée [15]:
```
import numpy as np
def SuiteU(n):
    u=1
    for k in range(n):
        u = 2*np.log(2)**(k+1) - (k+1)*u
    return u
```

3. Déterminer le premier entier n tel que $u_n < -10^4$. *Le programme doit renvoyer* `23`.

Entrée [16]:
```
# Solution 1
n=0
while SuiteU(n) >= -10**(4):
    n = n+1
print(n)
# Solution 2
n=0
u=1
while u >= -10**(4):
    u = 2*np.log(2)**(n+1) - (n+1)*u
    n = n+1
print(n)
```

4. Déterminer le premier entier n tel que $u_n > 10^4$. *Le programme doit renvoyer* `22`.

Entrée [17]:
```
# Solution 1
n=0
while SuiteU(n) <= 10**(4):
    n = n+1
print(n)
# Solution 2
n=0
u=1
while u <= 10**(4):
    u = 2*np.log(2)**(n+1) - (n+1)*u
    n = n+1
print(n)
```

III Matrices

Exercice 7 ECRICOME 2018, Maths E

On pose

$$X_0 = \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix} \qquad \text{et} \qquad X_1 = \begin{pmatrix} 3 \\ 0 \\ -2 \end{pmatrix}$$

et pour tout entier naturel n :

$$X_{n+2} = \frac{1}{6}AX_{n+1} + \frac{1}{6}BX_n$$

où

$$A = \begin{pmatrix} 2 & 1 & -2 \\ 0 & 3 & 0 \\ 1 & -1 & 5 \end{pmatrix} \qquad \text{et} \qquad B = \begin{pmatrix} 1 & -1 & -1 \\ -3 & 3 & -3 \\ -1 & 1 & 1 \end{pmatrix}$$

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

> « *Les questions d'informatique, à leur habitude sont fortement valorisées dans le barème de l'épreuve. Il est dommage que seule une moitié des candidats n'aborde cette question. Nécessitant uniquement des notions élémentaires d'algorithmique, elle devrait être un objectif raisonnable pour tout candidat. La principale difficulté était ici qu'on imposait l'ordre dans lequel on devait affecter les variables.* »

Entrée [18]:
```
import numpy as np

def X(n):
    Xold = np.array([[3],[0],[-1]])
    Xnew = np.array([[3],[0],[-2]])
    A = np.array([[2,1,-2],[0,3,0],[1,-1,5]])
    B = np.array([[1,-1,-1],[-3,3,-3],[-1,1,1]])
    for k in range(2, n+1):
        Aux = Xold
        Xold = Xnew
        Xnew = 1/6*np.dot(A,Xnew) + 1/6*np.dot(B,Aux)
    return(Xnew)
```

On vérifiera que `X(3)` renvoie une matrice colonne de trois lignes contenant les coefficients `2.05` (environ), `-1` et `-1.80` (environ).

Entrée [19]:
```
X(3)
```

Out [19]:
```
array([[ 2.05555556],
       [-1.        ],
       [-1.80555556]])
```

Exercice 8 Écrire une fonction, appelée `somme`, qui prend en argument une matrice `A` et qui renvoie la somme de tous ces coefficients. *On testera son programme sur une matrice quelconque.)*

Entrée [20]:
```
def somme(A):
    (n,p) = np.shape(A)
    S=0
    for i in range(n):
        for j in range(p):
            S = S + A[i,j]
    return S
```

Entrée [21]:
```
A=np.array([[1,1],[1,1]])
somme(A)
```

Out [21]:
```
4
```

IV Listes

Exercice 9 EMLyon 2024, Maths E

Le programme en langage Python ci-dessous définit une fonction "ajout" qui prend en argument une liste `L` et un entier `x`.

```
def ajout(L,x):
    if (x in L) == False:
        L.append(x)
```

Expliquer succinctement comment et à quelle condition l'exécution de la commande `ajout(L,x)` modifie la liste `L`.

> Cette fonction commence par tester si l'élément `x`, donné en argument, est déjà dans la liste `L`, également donnée en argument. Si ce n'est pas le cas, c'est-à-dire si l'élément `x` n'appartient pas à la liste `L`, alors l'élément `x` est ajouté à la liste `L` (en dernière position).

Exercice 10

1. Créer une liste L de 20 entiers aléatoires entre 1 et 100. On pourra utiliser la commande `rd.randint(a,b)` qui renvoie un entier aléatoire entre a et $b-1$.

Entrée [22]:
```
import numpy.random as rd
L = [rd.randint(1,101) for k in range(20)]
print(L)
```

2. Déterminer la somme des éléments de L.

Entrée [23]:
```
# Version 1
S=0
for k in range(len(L)):
    S = S + L[k]
print(S)
# Version 2
S=0
for x in L:
    S=S+x
print(S)
```

3. Déterminer la valeur maximale de L.

Entrée [24]:
```
# Version 1
M = L[0]
for k in range(1, len(L)):
    if L[k] > M:
        M = L[k]
print(M)
# Version 2
M = L[0]
for x in L:
    if x > M:
        M=x
print(M)
```