Corrigé du Devoir Surveillé d'Informatique n°4

L'utilisation de toute calculatrice et de tout matériel électronique est **interdite**. Les candidats ne peuvent utiliser aucun document. Dans l'écriture de vos programmes en Python, **respectez la ponctuation et l'indentation.**

Exercice 1 — Un petit pot pourri...

Aucune cohérence dans cet exercice, ce sont juste des extraits de vos TP !

1. Faire une fonction `Dichotomie` prenant en entrée un entier naturel non nul n et renvoyant le $n^{ième}$ terme de $(u_n)_{n\in\mathbb{N}}$, la suite de dichotomie vue dans le cours sur l'intervalle $[1,3]$ pour la fonction $f : x \mapsto \sin(x^2) - \frac{1}{2}$.
   Réponse : On propose ce programme pour répondre à la question :

```
import math as m

def f(x):
    return m.sin(x**2)-1/2

def Dichotomie(n):
    a,b=1,3
    u=(a+b)/2
    for k in range(n):
        u=(a+b)/2
        if f(u)*f(a) <=0:
            b=u
        else:
            a=u
    return u
```

2. Reprendre la question précédente avec un raisonnement récursif.
   Réponse : On propose ce programme pour répondre à la question :

```
def Dichotomierecursif(n):
    a,b=1,3
    u=(a+b)/2
    if n==0:
        return u
    if f(u)*f(a) <=0:
        b=u
    else:
        a=u
    return Dichotomierecursif(n-1)
```

3. Écrire une fonction `rectangles` prenant en entrée un entier naturel n non nul et renvoyant une approximation de $\int_0^1 \cos(1+t^2)dt$ en utilisant la méthode des rectangles à n pas.

   Réponse : On propose ce programme pour répondre à la question :

```
def rectangles(n,f):
    r=0
    for k in range(n):
        r=r+f(k/n)
    return r/n

import math as m
def fonction(t):
    return m.cos(1+t**2)
```

4. Écrire une fonction `lignecolonne` qui prend en entrée un tableau carré et qui renvoie ce même tableau dans lequel la première ligne a été remplacée par la première colonne.

   Réponse : On propose ce programme pour répondre à la question :

```
def lignecolonne(T):
    for k in range(np.size(T,1)):
        T[0,k]=T[k,0]
    return T
```

5. Écrire une fonction `tranposetableau` qui prend en entrée un tableau carré et qui renvoie la transposée de ce tableau, c'est-à-dire le même tableau dans lequel on a échangé le rôle joué par les lignes et les colonnes (la première colonne devient la première ligne, la seconde colonne devient la seconde ligne, ...).

   Réponse : On propose ce programme pour répondre à la question :

```
def tranposetableau(T):
    for k in range(np.size(T,1)):
        for i in range(np.size(T,0)):
            T[i,k]=T[k,i]
    return T
```

Exercice 2 — Des graphiques

Soit $(x_n)_{n\in\mathbb{N}}$ la suite définie par $x_0 = 2$ et, pour tout entier naturel n, on pose :

$$x_{n+1} = \begin{cases} n\sqrt{x_n} & \text{si } n \text{ est impair} \\ x_n & \text{sinon.} \end{cases}$$

1. Créer un programme prenant en entrée un entier naturel n et renvoyant x_n.
   Réponse : On propose ce programme suivant :

```
import math as m

def suite(n):
    x=2
    for k in range (n):
        if k%2==1:
            x=k*m.sqrt(x)
    return x
```

2. Créer un programme prenant en entrée un entier naturel n et renvoyant la liste $[x_0, \cdots, x_n]$.
   Réponse : On propose ce programme suivant :

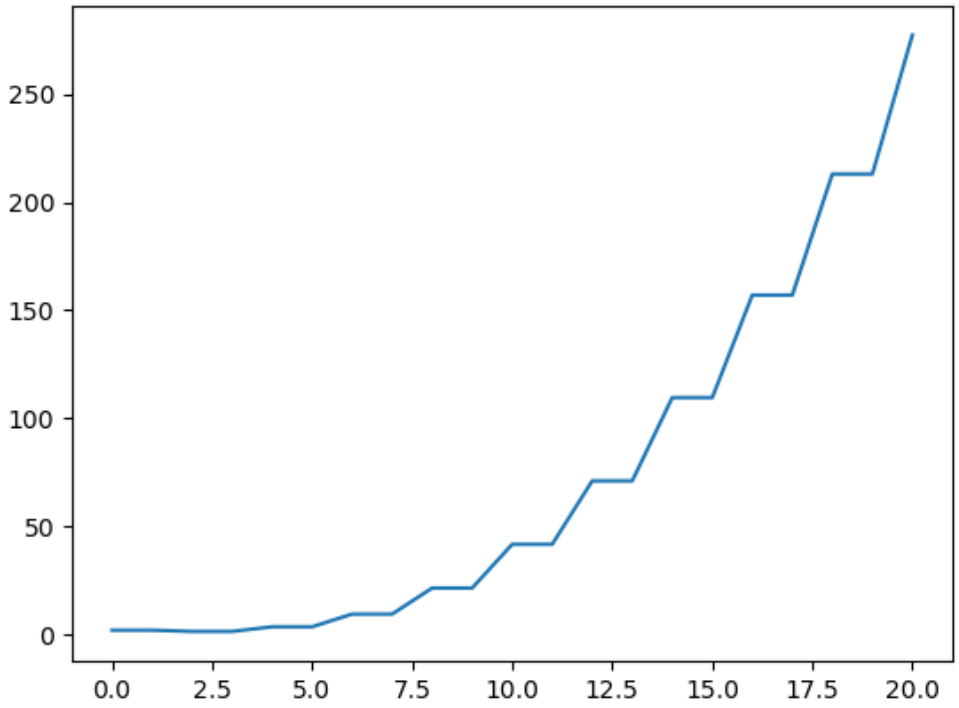
```
def listesuite(n):
    l=[2]
    x=2
    for k in range (n):
        if k%2==1:
            x=k*m.sqrt(x)
        l.append(x)
    return l
```

3. Faire un programme permettant d'obtenir un graphique de cette suite sur $[\![0, 20]\!]$
   Réponse : On propose ce programme suivant :

```
import numpy as np
import matplotlib.pyplot as plt

def graph():
    abs=[k for k in range (21)]
    ord=listesuite(20)
    plt.plot(abs, ord)
    plt.show()
```

   On a obtenu :



4. Créer un programme prenant en entrée un entier naturel p et renvoyant le maximum de $(x_n)_{n\in[\![0,p]\!]}$.
   Réponse : On propose ce programme suivant :

```
def maxx(p):
    m=2
    x=2
    for k in range (p+1):
        if k%2==1:
            x=k*m.sqrt(x)
        if x>m:
            m=x
    return m
```

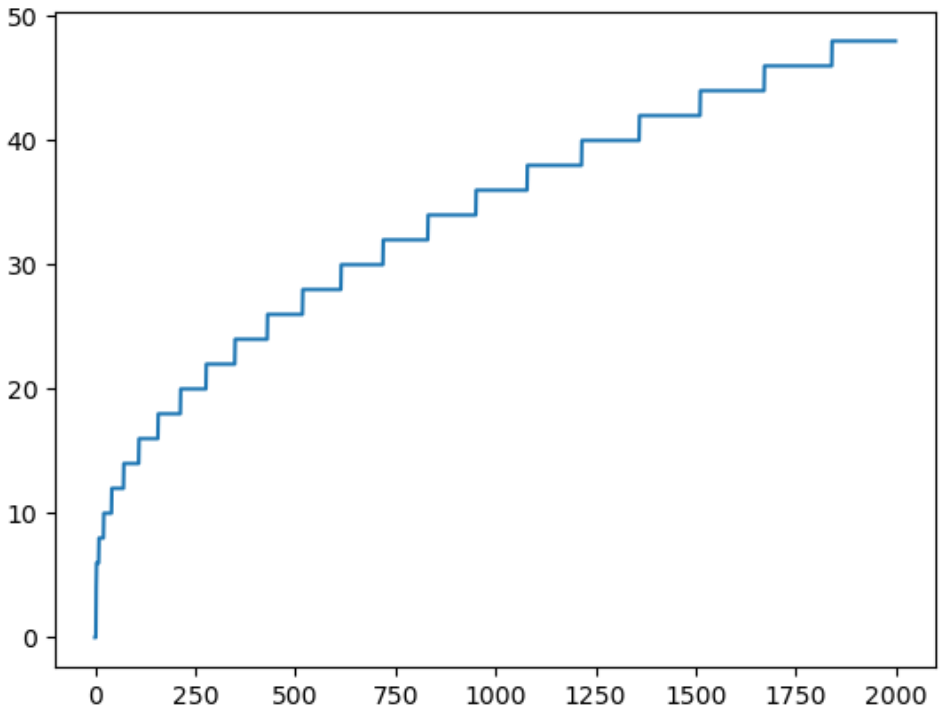
5. Créer un programme prenant en entrée un réel a et renvoyant $f(a)$ qui est défini comme le plus petit entier naturel n tel que $x_n \geqslant a$.
   Réponse : On propose ce programme suivant :

```
def f(a):
    x=2
    n=0
    while x<a:
        if n%2==1:
            x=n*m.sqrt(x)
        n+=1
    return n
```

6. Faire un programme permettant d'obtenir le graphique de f.
   Réponse : On propose ce programme suivant :

```
def graph2(n):
    abs=np.linspace(0,n,n**2+100)
    ord=[f(a) for a in abs]
    plt.plot(abs, ord)
    plt.show()
```

On a obtenu :



Exercice 3 — Un petit jeu

Soit n un entier naturel non nul. On considère un plateau de n cases, indicé de 0 à $n-1$. Sur chaque case, on écrit un numéro de case (le même numéro peut être inscrit sur plusieurs cases différentes). Par exemple, sur le plateau suivant, on a écrit 2 sur la case d'indice 0, et on a écrit 4 sur la case d'indice 1 et on a écrit 3 sur la case d'indice 3.

2	4	1	3	0

Informatiquement, on représente un plateau de taille n par une liste de taille n. La k-ième case de la liste contient le numéro inscrit sur la k-ième case. Ainsi, le plateau de l'exemple est représenté par la liste $[2,4,1,3,0]$.

On place un pion sur la case zéro. Ensuite, on fait avancer le pion comme suit : on regarde le numéro inscrit sur la case sur laquelle est le pion, puis on déplace le pion sur la case indicé par ce numéro. Sur l'exemple, on va déplacer le pion sur la case d'indice 2, puis la case d'indice 1, puis la case d'indice 4, puis la case d'indice 0, puis la case d'indice 1, etc.

1. On appelle **point fixe** une case sur laquelle est inscrit son propre indice. Ainsi, sur l'exemple il y a un unique point fixe, la case 3 (car il est écrit 3 sur la case d'indice 3). Écrire une fonction `compter` qui prend en entrée une liste représentant un plateau et qui renvoie le nombre de points fixes (la fonction renvoie 0 s'il n'y a aucun point fixe).

   Réponse : On propose ce programme suivant :

```
def compter(L):
    nb = 0
    for i in range(len(L)):
        if L[i] == i:
            nb += 1
    return nb
```

2. En déduire une fonction qui prend en entrée une liste représentant un plateau et qui renvoie True en cas de présence d'au moins un point fixe, False sinon. **On répondra avec le moins de ligne possible.**

   Réponse : On propose ce programme suivant :

```
def yaunpointfixe(L):
    return compter(L)!=0
```

3. Écrire une fonction avancer qui prend en entrée la liste représentant le plateau de jeu et un entier naturel p et qui donne le numéro de la case sur laquelle arrive le pion au bout de p étapes. Par exemple,`avancer([2,4,1,3,0], 1)` renvoie 2 et `avancer([2,4,1,3,0], 3)` renvoie 4.

   Réponse : On propose ce programme suivant :

```
def avancer(L, p):
    case = 0
    for _ in range(p):
        case = L[case]
    return case
```

4. On décide de s'arrêter quand le pion repasse à nouveau par une case déjà explorée. Écrire une fonction nb_de_coup qui prend en entrée la liste représentant le plateau de jeu et qui renvoie le nombre de tour de jeu effectué avant de terminer le jeu, la phrase "pas de fin" si on fait au moins 1000 déplacement.

   Réponse : On propose ce programme suivant :

```
def nb_de_coup(L):
    case = 0
    nc=0
    listeexplor=[]
    while nc<1001 and case not in listeexplor:
        case = L[case]
        listeexplor.append(case)
        nc=nc+1
    if nc<1001:
        return nc
    else:
        return "pas de fin"
```