Questions faisant intervenir un algorithme

Exercice 1

On considère la suite (u_n) définie par $u_0 = 80$ et pour tout entier naturel n, $u_{n+1} = 0{,}8u_n + 2$.
Écrire une fonction Python `seuil()` qui renvoie le plus petit entier n tel que $u_n < 40$.
Quel est le nombre renvoyé par cette fonction ?

Exercice 2

On considère la suite (v_n) définie par $v_0 = \frac{1}{2}$ et pour tout entier naturel n :

$$v_{n+1} = \frac{2v_n}{1 + v_n}$$

a. Calculer v_1 en détaillant.
b. Écrire une fonction Python `suite(n)` qui renvoie v_n pour tout entier naturel n.

Exercice 3

Dans une usine, un four cuit des céramiques à la température de 1 000°C. Pour un entier naturel n, on note T_n la température en degré Celsius du four au bout de n heures écoulées à partir de l'instant où il a été éteint. On a ainsi $T_0 = 1\,000$.
L'algorithme ci-contre permet de calculer la température du four au bout de n heures.

```python
def temperature(n):
    T = 1000
    for k in range(n)
        T = 0.82*T + 3.6
    return T
```

a. Au vu de l'algorithme, exprimer T_{n+1} en fonction de T_n.
b. Calculer T_4 et donner une valeur approchée à l'unité.

```python
def temps():
    T = 1000
    n = 0
    while ..........
        T = 0.82*T + 3.6
        n = ................
    return ...
```

c. La porte du four peut être ouverte sans risque pour les céramiques dès que la température est inférieure à 70°C.
Compléter l'algorithme ci-contre, qui doit renvoyer le temps en heures à attendre avant d'ouvrir la porte.
d. Quelle est la valeur renvoyée par l'algorithme ?

Exercice 4

Le but de cet exercice est d'étudier la suite (u_n) définie par la donnée de son premier terme u_1 et pour tout entier naturel n supérieur ou égal à 1, par la relation : $u_{n+1} = (n+1)u_n - 1$.
1. Vérifier, en détaillant le calcul, que si $u_1 = 0$, alors $u_3 = -4$ et $u_4 = -17$.

```python
def suite(x,n):
    u = ...
    for k in range(1,n):
        u = ................
    return u
```

2. Compléter l'algorithme ci-contre pour qu'il renvoie la valeur de u_n , pour $u_1 = x$.
Par exemple, au vu de la question précédente,
`suite(0,3)` doit renvoyer -4 ,
et `suite(0,4)` doit renvoyer -17.

3. On a exécuté cet algorithme pour $u_1 = 0{,}7$ puis pour $u_1 = 0{,}8$, pour n allant de 2 à 12.
Voici ci-contre les valeurs obtenues.
Quelle semble être la limite de cette suite si $u_1 = 0{,}7$?
Et si $u_1 = 0{,}8$?

Pour $u_1 = 0{,}7$	Pour $u_1 = 0{,}8$
0,4	0,6
0,2	0,8
−0,2	2,2
−2	10
−13	59
−92	412
−737	3 295
−6 634	29 654
−66 341	296 539
−729 752	3 261 928
−8 757 025	39 143 135
−113 841 326	508 860 754

Exercice 5

Dans une usine, on se propose de tester un prototype de hotte aspirante pour un local industriel.
Avant de lancer la fabrication en série, on réalise l'expérience suivante : dans un local clos équipé du prototype de hotte aspirante, on diffuse du dioxyde de carbone (CO2) à débit constant.
Dans ce qui suit, t est le temps exprimé en minutes.
À l'instant $t = 0$, la hotte est mise en marche et on la laisse fonctionner pendant 20 minutes. Les mesures réalisées permettent de modéliser le taux (en pourcentage) de CO2 contenu dans le local au bout de t minutes de fonctionnement de la hotte par l'expression $f(t)$ où f est la fonction définie pour tout réel t de l'intervalle $[0; 20]$ par $f(t) = (0{,}8t + 0{,}2)e^{-0{,}5t} + 0{,}03$.
On donne ci-contre son tableau de variations.

t	0		1,75		20
$f'(t)$		+	0	−	
f	0,23	↗		↘	

1. Dans cette question, on arrondira les deux résultats au millième.

a. Calculer $f(20)$. Arrondir à 10^{-4}.

b. Déterminer le taux maximal de CO2 présent dans le local.

2. On souhaite que le taux de CO2 dans le local retrouve une valeur V inférieure à 3,5 %.

a. Justifier qu'il existe un unique instant T satisfaisant cette condition.

b. On considère l'algorithme ci-contre.
Quelle est la valeur affichée à la fin de l'algorithme ? Que représente cette valeur dans le contexte de l'exercice ?

```
from math import exp
t = 1.75
p = 0.1
V = 0.7
while V > 0.035:
    t = t + p
    V = (0.8*t + 0.2)*exp(-0.5*t) + 0.03
print(t)
```

Exercice 6

On considère la suite (u_n) définie par $u_0 = 5$ et pour tout entier naturel n, $u_{n+1} = \sqrt{u_n + 1}$.

1. Démontrer par récurrence que pour tout entier naturel n, on a $1 \leq u_{n+1} \leq u_n$.

2. En déduire que la suite (u_n) converge.

3. Démontrer que la suite (u_n) converge vers $\ell = \frac{1+\sqrt{5}}{2}$

4. On considère le script Python ci-contre :
On rappelle que la fonction `abs` en Python correspond à la valeur absolue, et `sqrt` correspond à la racine carrée.

```
from math import *
def seuil(n):
    u = 5
    i = 0
    L = (1 + sqrt(5))/2
    while abs(u - L) >= 10**(-n) :
        u = sqrt(u + 1)
        i = i + 1
    return  i
```

a. Donner la valeur renvoyée par `seuil(2)`.

b. La valeur renvoyée par `seuil(4)` est 9.
Interpréter cette valeur dans le contexte de l'exercice.

Les exercices suivants sont plus difficiles.

Exercice 7

On définit la suite de réels (a_n) par :

$$\begin{cases} a_0 = 0 \\ a_1 = 1 \\ a_{n+1} = a_n + a_{n-1} \text{ pour } n \geq 1 \end{cases}$$

On appelle cette suite la suite de Fibonacci.

a. Vérifier que $a_6 = 8$.

b. Compléter l'algorithme pour qu'il renvoie a_n pour tout entier naturel n.

```
def fibonacci(n):
    a = 0
    b = 1
    for i in range(n):
        c = a + b
        a = ...
        b = ...
    return a
```

Exercice 8

On considère la suite (u_n) définie sur $\mathbb{N}$ par :

$$u_n = 1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots + \left(\frac{3}{4}\right)^n$$

1. Donner l'écriture décimale de u_0, u_1 et u_2.

2. On considère l'algorithme Python ci-contre.

Permet-il de calculer S_n ? Justifier.

3. Déterminer la limite de la suite (u_n).

```
def somme(n):
    S = 0
    for i in range(n+1):
        S = S + (3/4)**n
    return S
```

Exercice 9

On considère une fonction f définie, continue et strictement croissante sur un intervalle $[a;b]$ telle que $f(a) = -1$ et $f(b) = 1$.

1. Justifier que la fonction f s'annule en un réel α sur l'intervalle $[a;b]$.

2. Déterminer parmi ces 4 propositions, la fonction en langage Python qui permet de donner une valeur approchée de α à 0,001 près. On rappelle que la fonction abs en Python correspond à la valeur absolue.

a.
```
def racine(a, b):
    while abs(b - a) >= 0.001:
        m = (a + b)/2
        if f(m) < 0:
            b = m
        else:
            a = m
    return m
```

b.
```
def racine(a, b):
    m = (a + b)/2
    while abs(b - a) >= 0.001:
        if f(m) < 0:
            a = m
        else:
            b = m
    return m
```

c.
```
def racine(a, b):
    m = (a + b)/2
    while abs(b - a) <= 0.001:
        if f(m) < 0:
            a = m
        else:
            b = m
    return m
```

d.
```
def racine (a, b):
    while abs (b - a) >= 0.001:
        m = (a + b)/2
        if f(m) < 0:
            a = m
        else:
            b = m
    return m
```