T.P. IV - Suites. . .

Code Capytale : `7db3-1087927`

I - Ce qu'il faut savoir

II - . . . définies par leur terme général

Solution de l'exercice 1.

1. En factorisant le numérateur, on obtient

$$\begin{aligned} c_n &= 2 - \frac{3^n}{4^{n-1}}\left(1 + \frac{25}{3^n}\right) \\ &= 2 - \left(\frac{3}{4}\right)^n \times 4 \times \left(1 + \frac{25}{3^n}\right). \end{aligned}$$

Comme $3^n \to +\infty$, alors $1 + \frac{25}{3^n} \to 1$.
Comme $\frac{3}{4} \in]0, 1[$, alors $\left(\frac{3}{4}\right)^n \to 0$.
D'après les théorèmes d'addition des limites,

$$\lim_{n \to +\infty} c_n = 2.$$

2. Avant la boucle conditionnelle, `n` contient la valeur 1 et `c` la valeur c_1. Ensuite, on incrémente `n` et on calcule les valeurs de c_n. La boucle s'arrête dès que $c_n \geqslant 1.95$. Ainsi, la valeur renvoyée est le plus petit rang n pour lequel $c_n \geqslant 1.95$. Ce plus petit rang vaut donc 16. □

Solution de l'exercice 2. Le plus petit entier naturel n pour lequel $c_n \geqslant 0{,}95$ est égal à 11. □

Solution de l'exercice 3. [D'après Ecricome - 2022 - Exercice 1]

```python
n = 0
a = 1
b = -1/3
while a > 0.334 and b < 0.333:
    n = n + 1
    a = 1/3 * (1 + 2/4**n)
    b = 1/3 * (1 - 2/4**n)

print(n)
```

□

III - . . . récurrentes simples

Solution de l'exercice 4.

1. Ne pas oublier que `range(a, b)` est la liste des entiers compris entre `a` et `b-1`.

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

def u(n):
    u = 1
    for i in range(1, n+1):
        u = u / 2 + 3
    return u

n = 20
X = np.arange(0, n+1, 1)
Y = [u(n) for n in X]

plt.figure()
plt.plot(X, Y, 'o')
plt.show()
```

2.

```python
n = 0
u = 1
```

```
while u <= 5.5:
    u = u/2 + 3
    n = n + 1

print("n", n)
```

□

Solution de l'exercice 5.

1. a)

```
def suite_geom(n, q):
    v = 1
    for i in range(1, n+1):
        v = q * v
    return v
```

b)

```
print("n = 10, q = 0.1", suite_geom(10, 0.1))
print("n = 100, q = 2", suite_geom(100, 2))
print("n = 110, q = 0.5", suite_geom(110, 0.5))
```

2. a) On utilise ici la fonction précédente (même si on cela induit de nombreux calculs inutiles).

```
def serie_geom(n, q):
    s = 0
    for i in range(0, n+1):
        s = s + suite_geom(i, q)
    return s
```

b)

```
import matplotlib.pyplot as plt
n = 101
s = [serie_geom(n, 0.01) for n in range(0, n+1)]

plt.plot(range(0, n+1), s, 'o')
plt.show()
```

c)

```
import matplotlib.pyplot as plt
n = 101
s = [serie_geom(n, 1.2) for n in range(0, n+1)]

plt.plot(range(0, n+1), s, 'o')
plt.show()
```

□

IV - . . . récurrentes du type $u_{n+1} = f(u_n)$

Solution de l'exercice 6.

1.

```
import numpy as np
n = 5
u = 1
for k in range(1, n+1):
    u = np.log(1 + u**2)

print(u)
```

2. Le plus petit entier naturel n pour lequel $u_n < 0.0001$ est égal à 6. □

Solution de l'exercice 7. [D'après ESCP BSB - 2016 - Exercice 2]

1.

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

U = np.ones((21, 1))
for i in range(1, 21):
    U[i] = np.log(U[i-1] + 2)

X = np.arange(0, 21, 1)
plt.figure()
plt.plot(X, U, '+')
plt.show()
```

2. On peut conjecturer que la suite (u_n) est croissante. □

Solution de l'exercice 8. [D'après Ecricome - 2020 - Exercice 2]

```
def f(x):
    return x/(1 + x + x**2)

u = 1
n = 0
while u > 1/1000:
    u = f(u)
    n = n + 1

print(n)
```

□

V - . . . récurrentes dépendant du rang

Solution de l'exercice 9.

```
import numpy as np
n = 20
I = 1/2 - 1/(2 * np.exp(1))
for k in range(1, n+1):
    I = k * I - 1/(2 * np.exp(1))

print(I)
```

□

Solution de l'exercice 10. [D'après Ecricome - 2016 - Exercice 2]

```
import numpy as np

n = 10
I = 1

for k in range(n+1):
    I = 1/np.exp(1) + (k + 1) * I

print(I)
```

□

VI - . . . imbriquées

Solution de l'exercice 11.

```
n = 50
u = 0
v = 1
for k in range(1, n+1):
    u = (u + v)/2
    v = (u + v)/2

print("u50", u)
print("v50", v)
```

□

Solution de l'exercice 12. [D'après BCE ESCP - 2017 - Exercice 2]

1.

```
import numpy as np
n = 10
```

```
u = 1
v = 2
for k in range(2, n+1):
    a = u
    u = u**2/(u + v)
    v = v**2/(u + v)

print("u10", u)
print("v10", v)
```

2.

```
import numpy as np
import matplotlib.pyplot as plt
n = 10
u = 1
v = 2
s = np.zeros((n+1, 1))
s[1] = u
for k in range(2, n+1):
    a = u
    u = u**2/(u + v)
    v = v**2/(u + v)
    s[k] = u

X = np.arange(0, n+1)
Y = np.cumsum(s)
plt.figure()
plt.plot(X, Y)
plt.show()
```

a) La variable `s` contient la liste des termes 0, u_1, $u_2, \dots, u_{10}$.
La variable `y` contient la liste des termes 0, u_1, $u_1+u_2, \dots, u_1+\cdots+u_{10}$.

3. On peut conjecturer que la série $\sum u_n$ converge vers 2,4. ☐

VII - ... récurrentes doubles

Solution de l'exercice 13. La variable sert à stocker la valeur de `u` avant qu'on ne la modifie. Ainsi, à l'issue du `i`^e passage dans la boucle, `u` contient la valeur de u_i et `v` contient la valeur de u_{i-1}.

```
v = 0
u = 1
for i in range(2, 11):
    a = u
    u = 4 * u + 2 * v
    v = a

print(u)
```

☐

VIII - ... & fonctions : la dichotomie

Solution de l'exercice 14.

1. La fonction h est dérivable sur $\mathbb{R}_+^*$ et

$$\begin{aligned} h'(x) &= 3x^2 - \frac{3}{x^4} = \frac{3}{x^4}(x^6-1) \\ &= \frac{3}{x^4}(x^3-1)(x^3+1). \end{aligned}$$

D'une part, $\frac{3}{x^4} > 0$.
D'autre part, comme $x > 0$, alors $x^3 + 1 > 0$.
Enfin, $x^3 - 1 \geqslant 0$ si et seulement si $x^3 \geqslant 1$ si et seulement si $x \geqslant 1$.
On obtient ainsi le tableau de variations suivant :

x	0		1		$+\infty$
$h'(x)$	$\Vert$	$-$	0	$+$	
$h(x)$	$\Vert$	$\searrow$	-1	$\nearrow$	

2. Comme $\lim\limits_{x\to+\infty} x^3 = +\infty$, alors $\lim\limits_{x\to+\infty} \frac{1}{x^3} = 0$. D'après les théorèmes d'addition des limites, $\lim\limits_{x\to+\infty} h(x) = +\infty$.
D'après la définition de h, $h(1) = -1$.

La fonction h est continue et strictement croissante de $[1, +\infty[$ dans $[-1, +\infty[$. Comme $0 \in [-1, +\infty[$, il existe un unique réel α tel que $h(\alpha) = 0$.

3. Comme $h(2) = 2^3 + \frac{1}{2^3} - 3 = 5 + \frac{1}{8}$, alors $\alpha \leqslant 2$. On cherche donc α entre 1 et 2.

```
def h(x):
    return x**3 + 1/x**3 - 3

a = 1
b = 2
while (b - a) > 10**(-5):
    m = (a + b)/2
    if h(m) * h(a) <= 0:
        b = m
    else:
        a = m

print(a)
```

$\square$