

# T.P. IV - Fonctions & Suites

Code Capytale : 62c9-794137

## I - Fonctions

**Exercice 1. (Fonction définie par morceaux)** On considère la fonction  $g$  définie sur  $\mathbb{R}$  par

$$\forall x \in \mathbb{R}, g(x) = \begin{cases} 0 & \text{si } x < 0 \\ e^{-2x} & \text{si } x \geq 0 \end{cases}.$$

Compléter le script Python suivant pour que la fonction  $g$  prenne en entrée un réel  $x$  et calcule  $g(x)$ .

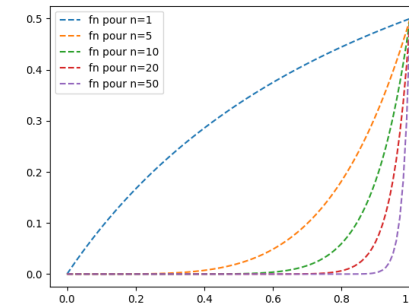
```
import numpy as np

def g(x):
    if x < 0:
        ....
    else:
        ....
```

1.

```
print (...)
print (...)
```

**Exercice 2. (Une suite d'intégrales)** Pour tout  $n$  entier naturel, on note  $f_n(x) = \frac{x^n}{1+x}$  et  $I_n = \int_0^1 f_n(x) dx$ . Le graphique ci-dessous contient des représentations graphiques des fonctions  $f_1, f_5, f_{10}, f_{20}$  et  $f_{50}$  sur l'intervalle  $[0, 1]$ .



1. Compléter le code ci-dessous pour qu'il trace les courbes des fonctions ci-dessus.

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

def f(x):
    return ...

X = np.linspace(0, 1, 100)

for n in [...]:
    plt.plot(..., ..., "--", label=r"fn pour n="+str(n))

plt.legend()
...
```

2. Pour tout  $n \in \mathbb{N}$ , interpréter géométriquement l'intégrale  $I_n$ .

3. En utilisant le graphique ci-dessus, conjecturer la limite de la suite  $(I_n)$  lorsque  $n$  tend vers  $+\infty$ .

## II - Suites

**Exercice 3. (Étude de suite)** Pour tout  $n$  entier naturel, on pose  $c_n = 2 - \frac{3^n + 25}{4^{n-1}}$ .

1. Déterminer  $\lim_{n \rightarrow +\infty} c_n$ .

2. On considère le code Python suivant :

```
n = 1
c = 2 - (3**n + 25)/4**(n-1)
while c < 1.95 :
    n = n + 1
    c = 2 - (3**n + 25)/4**(n-1)

print(n)
```

On obtient l'affichage suivant : 16.

Interpréter le résultat dans le contexte de l'énoncé.

**Exercice 4. (Suite récurrente double)** On considère la suite définie par  $u_0 = 0$ ,  $u_1 = 1$  et  $\forall n \in \mathbb{N}^*$ ,  $u_{n+1} = 4u_n + 2u_{n-1}$ .

Compléter les 3 lignes du script Python ci-dessous pour qu'il calcule et affiche la valeur de  $u_{10}$ .

```
v = 0
u = 1
for i in range(..., ...):
    a = u
    ...
    v = a

print(u)
```

## III - Suites et fonctions : la dichotomie

**Exercice 5. (Exemple de dichotomie)** On pose  $h(x) = x^3 + \frac{1}{x^3} - 3$ .

1. Montrer que  $h$  est strictement croissante sur  $]0, 1]$  et strictement décroissante sur  $[1, +\infty[$ .

2. Montrer que l'équation  $h(x) = 0$  possède une unique solution  $\alpha$  dans l'intervalle  $[1, +\infty[$ .

3. Compléter le code Python suivant pour qu'il renvoie une valeur approchée à  $10^{-5}$  près de  $\alpha$  par la méthode de dichotomie.

```
def h(x):
    return ...

a = ...
b = ...
while (b - a) ...:
    m = ...
    if h(m) * h(a) <= 0:
        b = ...
    else:
        ...

print(...)
```

=====

## IV - Introduction au produit matriciel

Le module `numpy`, importé via la ligne de commande `import numpy as np` permet de manipuler les matrices avec Python.

- \* `A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])` permet de définir une matrice ligne par ligne et d'obtenir ainsi la matrice  $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$  qui sera ici stockée dans la variable `A`.
- \* `3 * A` permet de multiplier `A` par le nombre 3.
- \* Si `A` et `B` sont des matrices de mêmes tailles, `A + B` permet d'en calculer la somme.
- \* Si `A` et `B` sont des matrices de tailles compatibles, `np.dot(A, B)` permet de multiplier les matrices `A` et `B`.

**Exercice 6. (Produit matriciel)** Soit  $A = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$ . On définit les suites  $(u_n)_{n \in \mathbb{N}}$  et  $(v_n)_{n \in \mathbb{N}}$  par

$$u_0 = 0, v_0 = 1 \text{ et } \forall n \in \mathbb{N}, \begin{cases} u_{n+1} &= u_n + v_n \\ v_{n+1} &= 2u_n \end{cases}.$$

Pour tout  $n$  entier naturel, on note  $C_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$ .

1. Calculer  $C_0$ .
2. Montrer que, pour tout  $n$  entier naturel,  $C_{n+1} = AC_n$ .
3. Montrer par récurrence que, pour tout  $n$  entier naturel,  $C_n = A^n C_0$ .
4. Compléter le script suivant pour qu'il calcule et affiche les termes  $u_{12}$  et  $v_{12}$ .

```
import numpy as np

n = ...

A = np.array([[...], [...]])

C = np.array([1, 0])

for k in range(..., ...):
    C = ...

print(C[0], C[1])
```