

T.P. III - Fonctions

Code Capytale : a656-2028011

I - Définition & Tracé

Exercice 1. On considère la fonction f définie sur \mathbb{R}_+^* par

$$\forall x \in \mathbb{R}_+^*, f(x) = x^x.$$

1. Compléter le script Python suivant pour que la fonction `f` prenne en entrée un réel x et renvoie la valeur de $f(x)$.

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

2. Compléter le code suivant pour qu'il affiche le graphe de f sur l'intervalle $[0,01;5]$.

```
import matplotlib.pyplot as plt  
import numpy as np  
  
X = np.arange(..., ..., 0.01)  
Y = [ ... for x in X]  
  
plt.figure()  
plt.plot(..., ..., 'r')  
  
....
```

3. On rappelle que $x^x = e^{x \ln(x)}$. Définir une fonction g telle que l'appel $g(x)$ retourne la valeur de $g(x) = e^{x \ln(x)}$.

```
import numpy as np  
  
def g(x):  
    return ...
```

4. Compléter le script suivant pour qu'on puisse constater la superposition entre les courbes représentatives de f et de g .

```
import matplotlib.pyplot as plt  
  
X = np.arange(..., ..., 0.01)  
Yf = [ ... for x in X]  
Yg = [ ... for x in X]  
  
plt.figure()  
plt.plot(..., ..., 'r')  
plt.plot(..., ..., 'g--')  
....
```

Exercice 2. Pour tout n entier naturel, on pose

$$u_n = \frac{e^n}{2e^n + n^2}.$$

1. Compléter le script suivant pour que l'appel `u(n)` renvoie la valeur de u_n .

```
def u(n):  
    y = ...  
    return y
```

2. Compléter le code suivant pour qu'il affiche les valeurs de u_{10} et u_{50} :

```
print("La valeur de u_10 est :", ...)  
print("La valeur de u_50 est :", ...)
```

3. Compléter le script suivant pour qu'il trace la suite de points de coordonnées $((n, u_n))_{n \in \llbracket 0, 100 \rrbracket}$.

```
import matplotlib.pyplot as plt
```

```
X = range(... , ...)
Y = [... for n in X]

plt.figure()
plt.plot(... , ...)

plt.show()
```

4. Que conjecturez-vous sur le comportement de la suite (u_n) ?

II - Fonction définie par morceaux

Exercice 3. On considère la fonction g définie sur \mathbb{R} par

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

1. 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:
        ....
```

2. Utilisez le script précédent pour afficher les valeurs de $g(-1)$ et $g(\ln(2))$.

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

Exercice 4. On considère la fonction f définie sur \mathbb{R} par

$$\forall x \in \mathbb{R}, f(x) = \begin{cases} \frac{1}{2} & \text{si } x \in [1, 3] \\ 0 & \text{sinon} \end{cases}.$$

1. Compléter le script Python suivant pour que la fonction **f** prenne en entrée un réel x et renvoie la valeur de $f(x)$.

```
def f(x):
    if x <= 3 and ... :
        ....
    else:
        ....
```

2. Compléter le code suivant pour qu'il affiche le graphe de f sur l'intervalle $[-5, 5]$.

```
import matplotlib.pyplot as plt

X = np.arange(... , ... , 0.01)
Y = [... for x in X]

plt.figure()
plt.plot(... , ... , 'r')

....
```

III - Fonction avec paramètres

Exercice 5. Pour tout n entier naturel, on définit la fonction

$$f_n(x) = \begin{cases} x^n & \text{si } x \in [0, 1] \\ 0 & \text{sinon} \end{cases}.$$

1. Compléter le code suivant pour que l'appel **f(n, x)** renvoie la valeur de $f_n(x)$:

```
def f(n, x):
    if x >= 0 and ...:
        return ...
    else:
        return ...
```

2. Compléter le code suivant pour qu'il affiche la superposition, sur un même graphique, des courbes représentatives de f_n pour $n \in \{0, 2, 5, 10\}$.

```

import matplotlib.pyplot as plt
X = np.arange(0, 1.001, 0.001)

plt.figure()
for n in [...]:
    Yn = [...] for x in X
    plt.plot(X, Yn, label="Graphe de f_"+str(n))

plt.legend()
plt.show()

```

Exercice 6. (Une suite d'intégrales) Pour tout n entier naturel, on note $f_n(x) = \frac{x^n}{1+x}$ et $I_n = \int_0^1 f_n(x) dx$.

1. Compléter le code suivant pour qu'il affiche les représentations graphiques des fonctions f_1, f_5, f_{10}, f_{20} et f_{50} sur l'intervalle $[0, 1]$.

```

import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return ...

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

plt.figure()
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 $+\infty$.