T.P. V - Fonctions

Code Capytale : `a656-2028011`

I - Définition & Tracé

Solution de l'exercice 1.

1. On teste la fonction sur des valeurs pour lesquelles on connaît le résultat, puis sur une valeur inconnue.

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

print("f(2) vaut 4 :", f(2))
print("f(3) vaut 3*3*3=27", f(3))
print("f(3.14) vaut", f(3.14))
```

2. On calcule la liste des abscisses et la liste des ordonnées des points à tracer.

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

# liste des abscisses
X = np.arange(0.01, 5.01, 0.01)
# liste des ordonnees :
# liste des images de x par f lorsque x parcourt X
Y = [ f(x) for x in X]

plt.figure()
# liste des abscisses,
# liste des ordonnees, 'r' signifie en rouge
plt.plot(X, Y, 'r')
plt.show()
```

3. On utilise les fonctions exponentielle et logarithme du module `numpy`. Comme précédemment, on teste la fonction pour s'assurer de son bon fonctionnement sur des valeurs connues.

```
import numpy as np

def g(x):
    return np.exp(x * np.log(x))

print("g(2) vaut 4 :", g(2))
print("g(3) vaut 3*3*3=27", g(3))
print("g(3.14) vaut", g(3.14))
```

4. On superpose le graphe des fonctions f et g pour vérifier qu'ils coïncident.

```
import matplotlib.pyplot as plt

plt.figure()
 # liste des abscisses des points traces
X = np.arange(0.01, 5.01, 0.01)
# liste des ordonnees du graphe de f :
Yf = [ f(x) for x in X]
# liste des ordonnees du graphe de g :
Yg = [ g(x) for x in X]

# trace de f en rouge trait plein
plt.plot(X, Yf, 'r')
# trace de g en vert pointilles
plt.plot(X, Yg, 'g--')
# affichage du graphique
plt.show()
```

□

Solution de l'exercice 2.

1. On utilise la fonction exponentielle du module `numpy` et on s'attache à bien parenthéser l'expression conformément aux règles de priorité des opérations. On teste ensuite la fonction sur quelques valeurs de n.

```
def u(n):
    y = np.exp(n) / (2 * np.exp(n) + n**2)
    return y

print("valeur de u_0 :", u(0))
print("valeur de u_1 :", u(1))
print("valeur de u_2 :", u(2))
print("valeur de u_100 :", u(100))
```

2. On utilise la fonction précédente qu'on appelle pour les valeurs demandées.

```
print("La valeur de u_10 est :", u(10))
print("La valeur de u_50 est :", u(50))
```

3. Dans le graphe suivant on a également ajouté la droite dont tous les points sont d'ordonnée 0,5 afin de constater la convergence de la suite vers cette valeur.

```
import matplotlib.pyplot as plt

# liste des abscisses
X = range(0, 101)
# liste des u_n lorsque n parcourt les valeurs de X
Y = [u(n) for n in X]
Z = [0.5 for n in X]

plt.figure()
# liste des abscisses, liste des ordonnees, trace ponctuel
plt.plot(X, Y, '.')
# trace la courbe reliant les points de coordonnees
# (n, 0.5) pour n = 0...100
plt.plot(X, Z, 'r')

# impose xmin, xmax, ymin, ymax (HP ?)
plt.axis([0, 100, 0, 0.55])
# trace la grille (HP ?)
plt.grid()
plt.show()
```

4. On conjecture que la suite (u_n) converge et que sa limite vaut 0,5. □

II - Fonction définie par morceaux

Solution de l'exercice 3.

1. La première version ci-dessous retourne directement les valeurs. La seconde version stocke les valeurs à renvoyer dans la variable `y` puis renvoie le contenu de cette variable. Ces deux méthodes sont équivalentes.

```
import numpy as np

def g(x):
    if x < 0:
        return 0
    else:
        return np.exp(-2 * x)

def g(x):
    if x < 0:
        y = 0
    else:
        y = np.exp(-2 * x)
    return y
```

2. On peut anticiper les valeurs que prend la fonction g et on vérifie donc que ces valeurs sont correctes.

```
print("g(-1) vaut 0", g(-1))
print("g(ln(2)) vaut 1/4 = 0.25", g(np.log(2)))
```

3. Pour créer la liste des abscisses, on utilise ici `linspace` du module `numpy`.
En utilisant le code suivant, on constate la présence d'un trait vertical à l'abscisse 0 qui est uniquement un artefact dû à cette méthode.

```
import matplotlib.pyplot as plt

# liste 100 points entre -5 et 5, repartis uniformement
X = np.linspace(-5, 5, 100)
Y = [g(x) for x in X]

plt.figure()
# Trace la courbe en noir
```

```
plt.plot(X, Y, 'k')
plt.show()
```

On peut supprimer cet artefact en effectuant deux tracés de courbe : le premier entre les abscisses -5 et 0, le second entre les abscisses 0 et 5.

```
import matplotlib.pyplot as plt

X1 = np.arange(-5, 0, 0.01)
Y1 = [g(x) for x in X1]
X2 = np.arange(0, 5, 0.01)
Y2 = [g(x) for x in X2]

plt.figure()
plt.plot(X1, Y1, color='orange')
plt.plot(X2, Y2, color='orange')
plt.show()
```

□

Solution de l'exercice 4.

1. On propose deux définitions, la première utilisant seulement `if` / `else`, la seconde utilisant `elif`. On teste notre fonction sur des valeurs pour lesquelles on connaît le résultat.

```
def f(x):
    if x <= 3 and x >= 1 :
        return 1/2
    else:
        return 0

def f(x):
    # si x est strictement inférieur à 1 :
    if x < 1:
        return 0
    # si x n'est pas strictement inférieur à 1
    # mais si x est inférieur ou égal à 3 :
    elif x <= 3:
        return 1/2
    # sinon
    else:
        return 0
```

```
print("f(2) vaut 1/2", f(2))
print("f(0.5) vaut 0", f(0.5))
print("f(45) vaut 0", f(45))
```

2. On utilise les mêmes stratégies que précédemment :

```
import matplotlib.pyplot as plt

plt.figure()
X = np.arange(-5, 5.01, 0.01)
Y = [ f(x) for x in X]

plt.plot(X, Y, 'r')
plt.show()
```

□

III - Fonction avec paramètres

Solution de l'exercice 5.

1. La fonction utilise ici deux paramètres. On teste ensuite cette fonction sur des valeurs simples (attention à bien prendre en compte l'ordre des paramètres).

```
def f(n, x):
    if x >= 0 and x <= 1:
        return x**n
    else:
        return 0

print("f(2, 3) vaut 0 car 3 > 1 :", f(2, 3))
print("f(2, 0.5) vaut 0.5**2 = 0.25 :", f(2, 0.5))
```

2. Pour chaque valeur de `n` de la liste `[0, 2, 5, 10, 20, 30]` on calcule la liste des ordonnées des points à tracer pour obtenir l'allure du graphe de f_n.

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

plt.figure()
```

```
for n in [0, 2, 5, 10, 20, 30]:
    Yn = [f(n, x) for x in X]
    # label : etiquette les courbes
    plt.plot(X, Yn, label="Graphe de f_"+str(n))

# affiche la legende
plt.legend()
plt.show()
```

□

Solution de l'exercice 6.

1. On remarque que, dans le code suivant, la valeur de `n` n'est pas considérée comme un paramètre de la fonction. Cet usage est dangereux et il est préférable d'utiliser la méthode de l'exercice précédent. Cependant, ce cas de figure peut être rencontré dans les sujets de concours.

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

def f(x):
    return x**n/(1 + x)

# liste de 0 à 1 avec 100 points uniformément répartis
X = np.linspace(0, 1, 100)

plt.figure()
for n in [1, 5, 10, 20, 50]:
    Y = [f(x) for x in X]
    plt.plot(X, Y, "--", label="f_n pour n = "+str(n))

plt.legend()
plt.show()
```

2. Comme la fonction f_n est positive, l'intégrale I_n est égale à l'aire comprise entre l'axe des abscisses, la courbe représentative de f et les droites d'abscisse $x = 0$ et $x = 1$.

3. En utilisant le graphique, on conjecture que l'aire tend vers 0, soit $\lim\limits_{n\to+\infty} I_n = 0$. □