Python td 3 - Fonctions et tracé de courbes

I. Définir une fonction

I.1) Fonctions usuelles définies dans le package numpy

Nom	Fonction
np.log	logarithme népérien
np.exp	exponentielle
np.sqrt	racine carrée
np.abs	valeur absolue
np.floor	partie entière
np.sin	sinus
np.cos	cosinus
np.tan	tangente

Ces fonctions peuvent s'appliquer à une variable numérique ou directement à une matrice.

I.2) Définir une nouvelle fonction

La syntaxe est simple !

```
def f(x):
    return valeur en fonction de x
```

On peut appliquer la fonction ainsi définie à une valeur ou directement à une matrice.

Exemple

```
def f(x):
    return x^2

L=np.arange(1,11,1)
print(f(L))
```

II. Tracé de la courbe d'une fonction

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

Puis si x est la matrice des abscisses et si y est la matrice des ordonnées, la commande

```
plt.plot(x,y)
plt.show() #pour afficher la figure
```

relie les points de coordonnées x et y

Exemple
Courbe de la fonction sinus sur $[-2\pi, 2\pi]$:

```
x=np.arange(-2*np.pi,2*np.pi,0.01)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```


On peut rajouter des options pour que ce soit plus joli :

```
x=np.arange(-2*np.pi,2*np.pi,0.01)
y=np.sin(x)
plt.plot(x,y,label="y=sin(x)")
plt.legend() # pour afficher la légende
plt.title("Fonction sinus") # pour le titre
plt.xlabel("x") # nom de l'axe des abscisses
plt.ylabel("y") # nom de l'axe des ordonnées
plt.ylim(-1,1) # on trace y entre -1 et 1 uniquement
plt.show() # pour afficher tout ca
```

III. Suites

III.1) Suites explicites du types $u_n = g(n)$

Si la suite est définie à partir d'une fonction $\forall n \in \mathbb{N}$, $u_n = g(n)$, alors on définit g comme précédemment et en posant `x=np.arange(0,n+1,1)` et `u=f(x)`, `u` est la matrice ligne dont les coordonnées sont les $n+1$ premières valeurs u_0, ..., u_n. On peut alors utiliser `plt.plot(x,u,"o")` pour représenter les termes de la suite (le "o" sert à indiquer que l'on veut mettre un point pour les termes de la suite et ne pas les relier entre eux).

Exemple
On considère la suite définie par : $\forall n \in \mathbb{N}$, $u_n = \dfrac{n^2}{2^n}$. Le programme suivant permet de tracer les premiers termes de la suite :

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

def f(x):
    return (x**2)/(2**x)

x=np.arange(0,12,1)
y=f(x)
plt.plot(x,y,"o")
plt.show()
```

Exécuter ce programme et émettre une conjecture quand à la monotonie de cette suite au vu du graphique obtenu.

III.2) Suites récurrentes du types $u_{n+1} = g(u_n)$

Si la suite est définie par récurrence, on utilisera en général une boucle `for`.

On considère une suite définie par $u_0 = a$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = g(u_n)$.

Première méthode : nous calculons les valeurs de la suite en les effaçant au fur et à mesure (pas de tracé possible), et on affiche le terme u_n.

```
n=int(input("Entrer n"))
u=a
for k in range(1,n+1):
    u=g(u)
print(u)
```

Deuxième méthode : stocker les valeurs dans une matrice au fur et à mesure
Nous pouvons ensuite représenter graphiquement les premiers termes de la suite.

```
n=int(input('Entrer n: ')
u=np.zeros([n+1])  # on définit une matrice unidimensionnelle nulle à n+1 cases
u[0]=a
for k in range(1,n+1):
    u[k]=g(u[k-1])
print(u)
plt.plot(np.arange(0,n+1,1),u,"o")
```

Exercice 1
On considère la suite définie par $u_0 = 0$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \sqrt{2+u_n}$.
A l'aide de la première méthode, calculer le n-ième terme de cette suite. Puis à l'aide de la deuxième méthode, calculer et afficher les termes u_0, ..., u_n.
Que peut-on conjecturer pour cette suite ? Comment pourrait-on le démontrer ?

IV. Exercices

Exercice 2
Tracés de courbes

1. On considère la fonction g définie par : $\forall x \in]0,+\infty[\quad g(x) = \dfrac{\ln(x)}{x}$.
   (a) Tracer la courbe de g sur $[1,10]$ en utilisant Python.
   (b) Montrer que la courbe de g admet un point d'inflexion sur $[1,10]$ et trouver les coordonnées de ce point. Vérifier la cohérence avec le tracé de la courbe.
2. On considère la fonction h définie par : $\forall x \in \mathbb{R} \quad h(x) = x + 1 + e^{-x}$
   (a) Tracer la courbe de h sur $[-2,3]$ en utilisant Python.
   (b) Montrer que la courbe de h admet une asymptote au voisinage de $+\infty$ et préciser une équation de cette droite.
   (c) Illustrer ce résultat en traçant sur le même graphe la courbe de h et cette droite sur $[-2,3]$.

Exercice 3
Tracé d'une courbe d'une application réciproque d'une bijection

1. **La fonction arctangente :**
   (a) Tracer la courbe de la fonction tangente sur l'intervalle $\left]-\dfrac{\pi}{2}+0.1; \dfrac{\pi}{2}-0.1\right[$.
   (b) Tracer la courbe de la fonction arctangente ainsi que la droite d'équation $y = x$ sur le même graphe.
2. **Un autre exemple**
   On note $\forall x \in \mathbb{R} \quad f(x) = \dfrac{e^x - e^{-x}}{2}$.
   (a) Montrer que f est une bijection.
   (b) Tracer la courbe de f sur $[-2,2]$ et la courbe de son application réciproque en Python.

Exercice 4
Illustration d'un développement limité

1. **Le développement limité de $x \to \ln(1+x)$ au voisinage de 0**
   (a) Tracer la courbe de $x \to \ln(1+x)$ sur $[-0.999,4]$ en utilisant Python.
   (b) Rappeler le développement limité à l'ordre de 2 de $x \to \ln(1+x)$ au voisinage de 0.
   (c) Illustrer ce résultat grâce à scilab sur $[-0.999,4]$.
2. **Le développement limité de $x \to e^x$ au voisinage de 0**
   (a) Tracer la courbe de $x \to e^x$ sur $[-1,1]$ en utilisant Python.
   (b) Rappeler le développement limité à l'ordre de 2 de $x \to e^x$ au voisinage de 0.
   (c) Illustrer ce résultat grâce à scilab sur $[-1,1]$.
   (d) Rappeler l'inégalité fondamentale de l'exponentielle, puis illustrer ce résultat sur $[-2,2]$
   (e) Rappeler l'inégalité fondamentale du logarithme, puis illustrer ce résultat sur $[0.001,3]$