# TP14 - Résolution numérique d'une équation

## 1 . Dichotomie

**Exercice 1**
Écrire une fonction d'en-tête `def resolution(f,a,b,epsilon)` qui renvoie déterminer une valeur approchée à epsilon près de la solution de $f(x) = 0$. On suppose ici que $f$ est strictement monotone et s'annule sur $[a, b]$.

In [1]:
```
def resolution(f,a,b,epsilon):
    while b-a > epsilon:
        m = (a+b)/2
        if f(a)*f(m)<0 :
            (a,b) = (a,m)
        else:
            (a,b) = (m,b)
    return a
```

Tester la resolution avec l'équation $x - 3 = 0$ sur $[1, 6]$ avec $\varepsilon = 10^{-2}$.

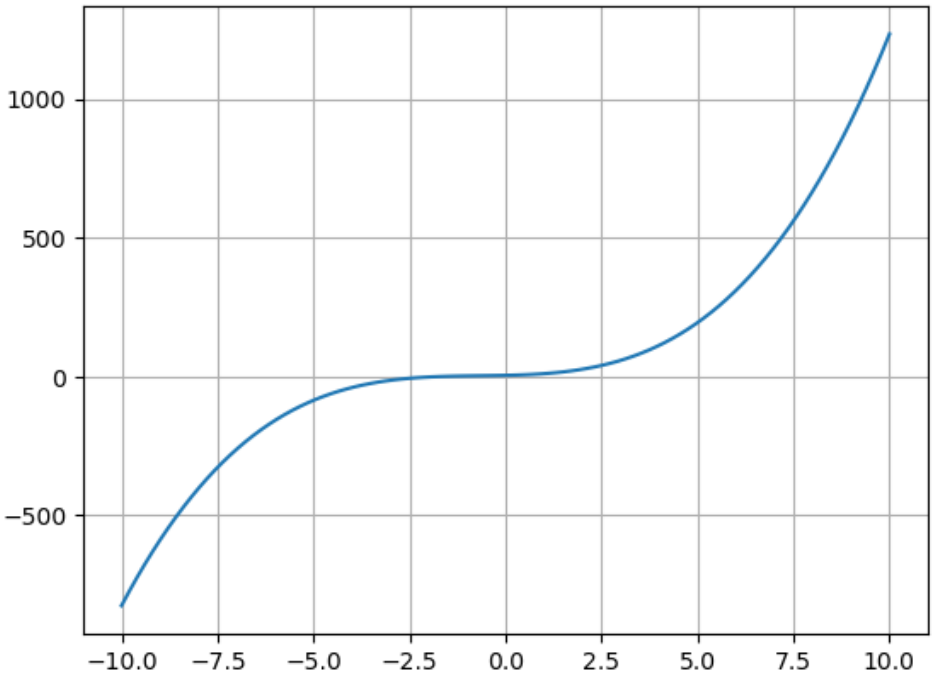
In [2]:
```
def f(x):
    return x-3

resolution(f,1,6,0.01)
```

Out[2]:
```
2.9921875
```

**Exercice 2**

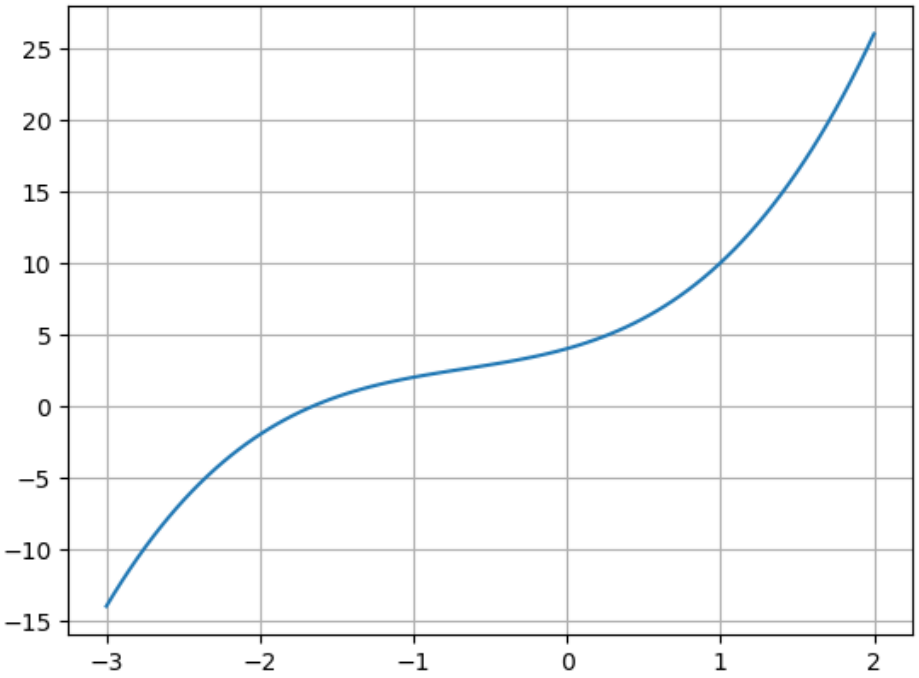
1. Définir la fonction $f : x \mapsto x^3 + 2x^2 + 3x + 4$ puis tracer sa courbe sur $[-10, 10]$.

In [6]:
```
def f(x):
    return x**3+2*x**2+3*x+4

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10,10,100)
y = f(x)
plt.plot(x,y)
plt.grid() # pour afficher une grille, cela facilite la lecture
plt.show()
```




2. À l'aide du graphique, trouver un intervalle $[a, b]$ avec $f(a)$ et $f(b)$ de signes contraires.

In [9]:
```
# on affine le tracé
x = np.linspace(-3,2,100)
y = f(x)
plt.plot(x,y)
plt.grid() # pour afficher une grille, cela facilite la lecture
plt.show()
```




Il semble n'y avoir qu'une seule solution. L'intervalle $[-3, 2]$ convient mais on peut prendre plus petit ou plus grand sans problème.

2. Utiliser la fonction resolution pour trouver une valeur approchée de la solution de l'équation $f(x) = 0$ avec epsilon = 0.001 .

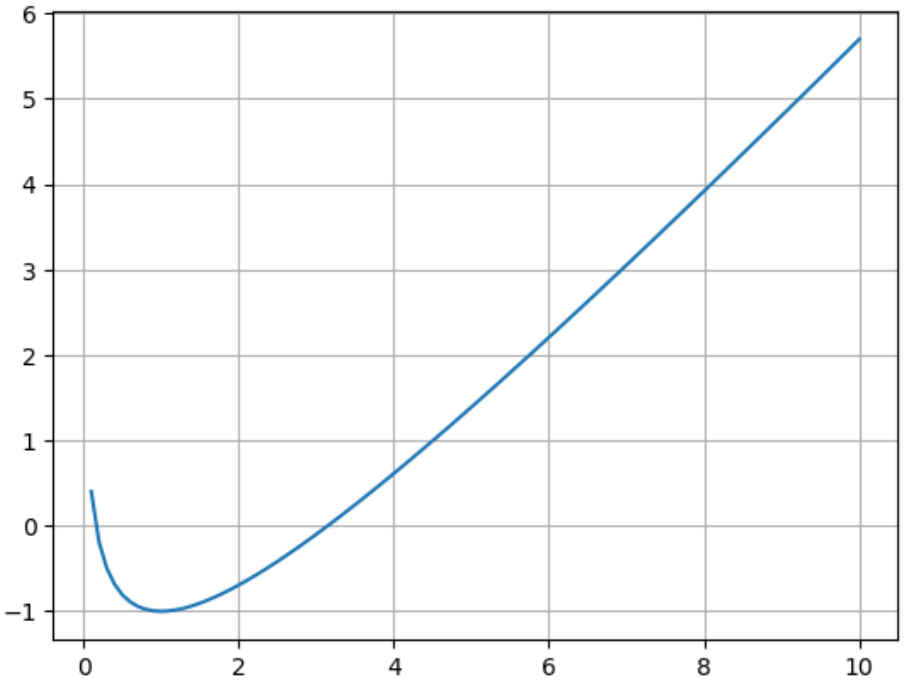
In [10]:
```
resolution(f,-3,2,0.001)
```

Out[10]:
```
-1.651123046875
```

**Exercice 3**

Résoudre numériquement l'équation $x - \ln(x) = 2$.

In [14]:
```
def f(x):
    return x-np.log(x)-2

x = np.linspace(0.1,10,100)
y = f(x)
plt.plot(x,y)
plt.grid() # pour afficher une grille, cela facilite la lecture
plt.show()
```




In [15]:
```
resolution(f,0.1,1,0.001)
```

Out[15]:
```
0.1580078125
```

In [16]:
```
resolution(f,1,4,0.001)
```

Out[16]:
```
3.14599609375
```

## 2. Utilisation d'une suite récurrente

**Exercice 4**

On souhaite trouver une valeur approchée de $\sqrt{2}$, c'est-à-dire résoudre $x^2 = 2$ sur $\mathbb{R}_+$. On note $h : x \mapsto \frac{1}{3}\left(2x + \frac{2}{x}\right)$ et $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 2$ et $u_{n+1} = h(u_n)$ pour tout $n \in \mathbb{N}$.

1. Écrire un programme qui affiche les $11$ premiers termes de la suite $(u_n)$ (jusqu'à $u_{10}$) et observer que $(u_n)$ semble tendre vers $\sqrt{2}$.

In [18]:
```
u = 2
for k in range(10):
    u = 1/3*(2*u+2/u)
    print(u)
```

```
1.6666666666666665
1.511111111111111
1.4485838779956426
1.4259421672224422
1.4181552538798716
1.4155311114534013
1.4146531541829614
1.4143601385095614
1.414262426148686
1.4142298508610516
```

2. On peut montrer que, pour tout $n \in \mathbb{N}$, $|u_n - \sqrt{2}| \leqslant \left(\frac{2}{3}\right)^n$ (vu en DM9). Écrire un programme qui affiche un entier $n \in \mathbb{N}$ tel que $u_n$ soit une valeur approchée de $\sqrt{2}$ à $\varepsilon = 10^{-4}$ près. On affichera également la valeur de $u_n$ correspondante.

**Attention : interdiction d'utiliser np.sqrt(2) ou $2^{1/2}$ ici.**

In [20]:
```
n = 0
u = 2
while (2/3)**n > 10**(-4):
    n = n+1
    u = 1/3*(2*u+2/u)
print('valeur approchée =', u)
```

```
valeur approchée = 1.4142135623833116
```

In [ ]: