Les suites

1 Calcul du $n^{ème}$ terme

1.1 Suite explicite

Chaque terme de la suite est exprimé en fonction de n et indépendamment des termes précédents :

$$U_n = f(n)$$

Pour calculer le $n^{ème}$ terme de la suite, il suffit de définir une fonction qui prend l'indice n comme paramètre et calcule le terme correspondant.

Exemple

Soit $(U_n)_{n\in\mathbb{N}}$ une suite définie par :

$$U_n = \frac{1}{1+n}$$

[1]:
```
def U(n):
    return 1/(1+n)
```

[2]:
```
U(0)
```

[2]: 1.0

[3]:
```
U(9)
```

[3]: 0.1

On peut calculer et stocker les termes d'indices 0 à n en utilisant une liste :

Exemple

[4]:
```
n = 3

termes = []                # initialisation de la liste des termes
for i in range(n+1):       # i va de 0 à n, donc n+1 termes vont être calculés
    termes.append(U(i))    # calcul et ajout du ième terme à la liste

print(termes)
```

```
[1.0, 0.5, 0.3333333333333333, 0.25]
```

[5]:
```
# ou bien en utilisant les listes en compréhension

n = 3

termes = [U(i) for i in range(n+1)]

termes
```

[5]:
```
[1.0, 0.5, 0.3333333333333333, 0.25]
```

1.2 Suite récurrente

1.2.1 Simple

Dans le cas d'une suite récurrente simple chaque terme de la suite s'obtient à partir du terme précédent :

$$U_{n+1} = f(U_n)$$

Pour calculer le $n^{ème}$ terme on initialise en affectant à une variable la valeur initiale de la suite, et, en utilisant une boucle qui se répète n fois, on calcule le terme actuel en fonction du terme précédent, et la variable prend le terme qu'on vient de calculer.

Exemple

Soit la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} U_0 = 1 \\ U_{n+1} = 3U_n + 1 \end{cases}$$

[6]:
```
def U(n):
    x = 0                   # initialisation
    for _ in range(n+1):    # répétition n fois
        x = 3*x + 1         # mise à jour du terme
    return x
```

[7]:
```
U(1)
```

[7]:
```
4
```

[8]:
```
U(7)
```

[8]:
```
3280
```

En cas de besoin, on peut garder les termes précédents en utilisant une liste qui permettra de stocker progressivement chaque terme calculé.

Exemple

[9]:
```
def U(n):
    x = 1
```

```
    termes = [1]                # initialisation de la liste des termes
    for _ in range(n):          # pour calculer n termes
        x = 3*x + 1             # calcul et mise à jour
        termes.append(x)        # ajout du terme à la liste
    return termes

U(3)
```

[9]: `[1, 4, 13, 40]`

1.2.2 Double

Dans ce cas chaque terme est obtenu en combinant les deux termes précédents :

$$U_{n+2} = f(U_n, U_{n+1})$$

En utilisant la même technique vue dans le cas simple, on aura besoin d'initialiser en affectant à deux variables les deux valeurs initiales de la suite. Ensuite, en utilisant une boucle qui se répète n fois, on calcule le terme actuel en fonction des termes précédents, et on met à jour les deux variables.

Exemple

Soit la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} U_0 = 0, U_1 = 1 \\ U_{n+2} = 3U_{n+1} + 2U_n + 1 \end{cases}$$

[10]:
```
def U(n):
    x1 = 0
    x2 = 1
    for _ in range(2, n+1):   # on saute les deux termes initiaux
        x = 3*x2 + 2*x1 + 1   # calcul du terme suivant
        x1 = x2               # mise à jour
        x2 = x
    return x

U(2)
```

[10]: `4`

Pareillement, on peut garder les termes précédents en utilisant une liste :

[11]:
```
def U(n):
    x1 = 0
    x2 = 1
    termes = [x1, x2]         # initialisation de la liste des termes
    for _ in range(2, n+1):
        x = 3*x2 + 2*x1 + 1   # calcul du terme suivant
```

```
        termes.append(x)      # ajout du terme à la liste
        x1 = x2               # mise à jour
        x2 = x
    return termes

U(3)
```

[11]: [0, 1, 4, 15]

2 Représentation graphique

Après avoir créé la liste des abscisses (indices des termes) et la liste des ordonnées (les termes), on peut représenter graphiquement la suite en utilisant la fonction $plot(indices, termes, marker)$ du module *matplotlib.pyplot* en utilisant le paramètre optionnel *marker* pour marquer les points de différents signes comme ‘.’, ‘o’, ‘x’, …

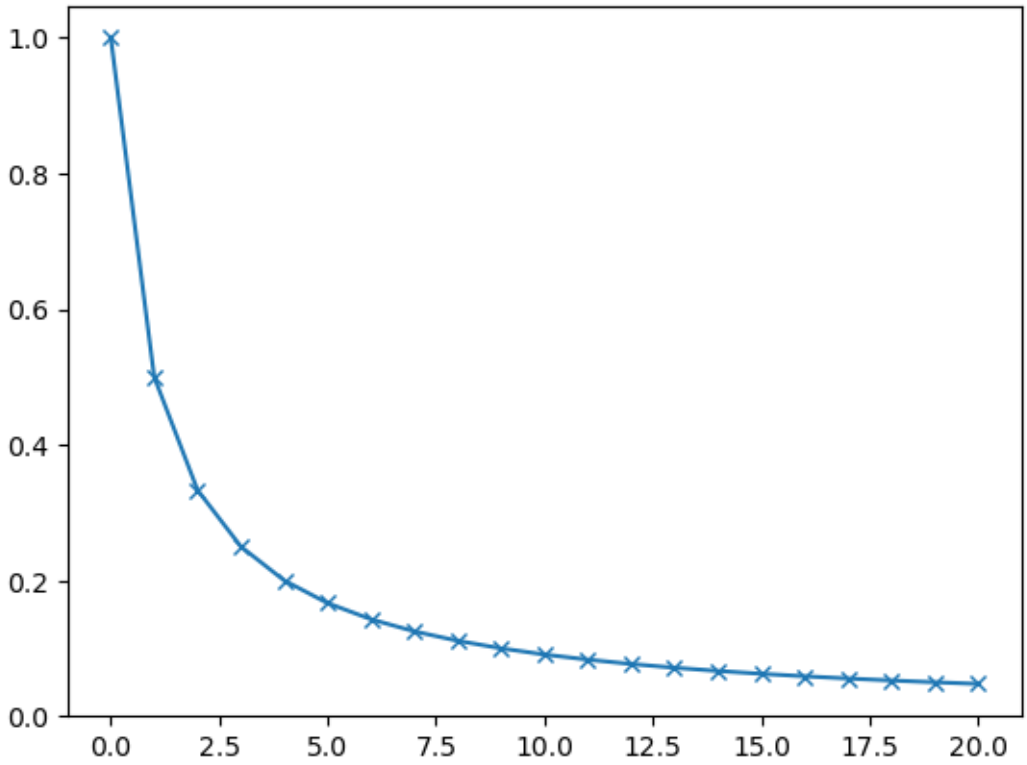
Exemple

On reprend la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$U_n = \frac{1}{1+n}$$

[12]:
```
import numpy as np
import matplotlib.pyplot as mpl

def U(n):
    return 1/(1+n)

n = 20

# Création des indices
indices = np.arange(0, n+1, 1)

# Calcul des termes
termes = U(indices)

mpl.plot(indices, termes, marker='x')
```

[12]: [<matplotlib.lines.Line2D at 0x105de7750>]


1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

3 TP

Exercice 1

Soit $(U_n)_{n\in\mathbb{N}}$ une suite explicite définie par : $U_n = \frac{1}{2^n}$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Calculer la limite de la suite et vérifier le résultat graphiquement.

Exercice 2

Soit $(U_n)_{n\in\mathbb{N}}$ une suite définie par : $U_n = (n-3)^2$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Étudier la monotonie de la suite et vérifier le résultat graphiquement.

Exercice 3

Une usine assure, en 2025, une production de 100 000 articles. Elle s'engage à augmenter sa production de 3% chaque année.

1. Quelle est la nature de cette suite production?
2. Créer une fonction qui renvoie la production en une année donnée.
3. Créer une liste de nombres d'articles produits entre 2025 et 2035.
4. Représenter graphiquement la production entre 2025 et 2035.
5. Combien d'articles auront été fabriqués dans cette période?

Exercice 4

Soit la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \end{cases}$$

1. Créer une fonction pour calculer le $n^{ème}$ terme de la suite.
2. Stocker dans une liste les $\frac{F_n}{F_{n-1}}$ pour $n \in [2, 20]$.
3. Tracer graphiquement cette suites de rapports.
4. Tracer la ligne d'équation $f(x) = \varphi = \frac{1+\sqrt{5}}{2}$, pour $x \in [2, 20]$ (Le nombre d'or).
5. Que peut-on en déduire?

[13]:
```
# Exercice 1

```

```
# 1

def U(n):
    return 1/2**n
```

[14]:
```
# 2

n = int(input('Une borne :'))

termes = [U(i) for i in range(n+1)]
```

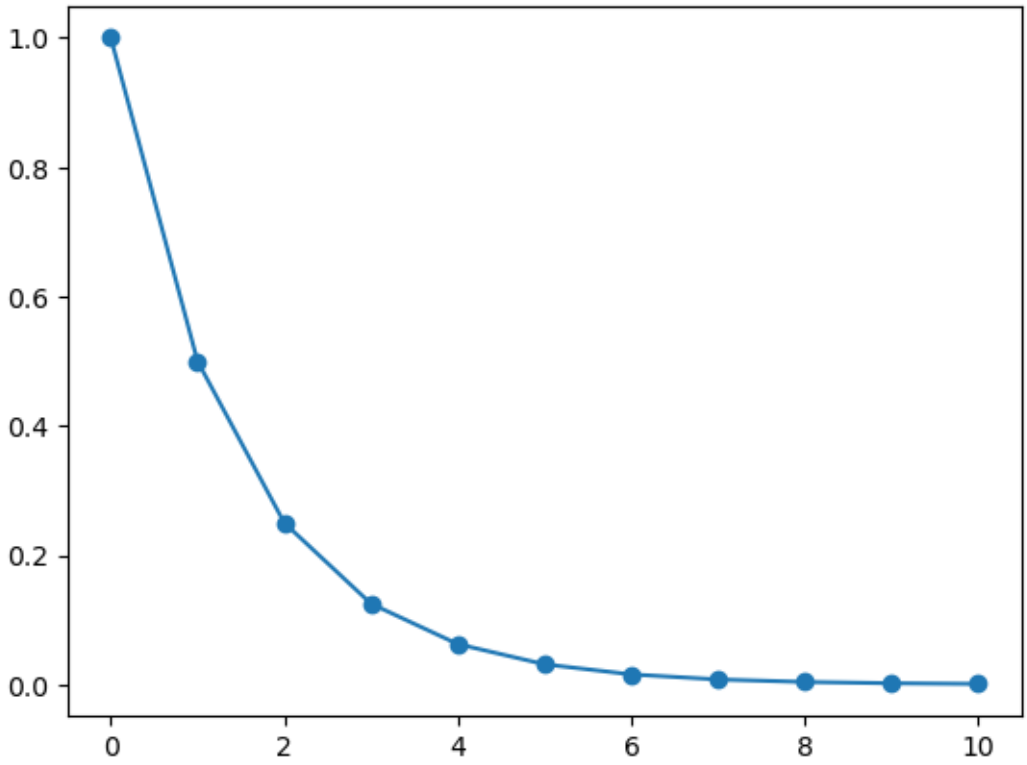
```
Une borne :10
```

[15]:
```
termes
```

[15]:
```
[1.0,
 0.5,
 0.25,
 0.125,
 0.0625,
 0.03125,
 0.015625,
 0.0078125,
 0.00390625,
 0.001953125,
 0.0009765625]
```

[16]:
```
# 3

indices = range(n+1)

mpl.plot(indices, termes, marker='o')
```

[16]:
```
[<matplotlib.lines.Line2D at 0x1068d29d0>]
```



4. On a $\lim\limits_{n\to\infty} U_n = 0$, ce qui est confirmé graphiquement.

[17]:
```
# Exercice 2

# 1

def U(n):
    return (n-3)**2
```

[18]:
```
# 2

n = int(input('Une borne :'))

termes = [U(i) for i in range(n+1)]
```

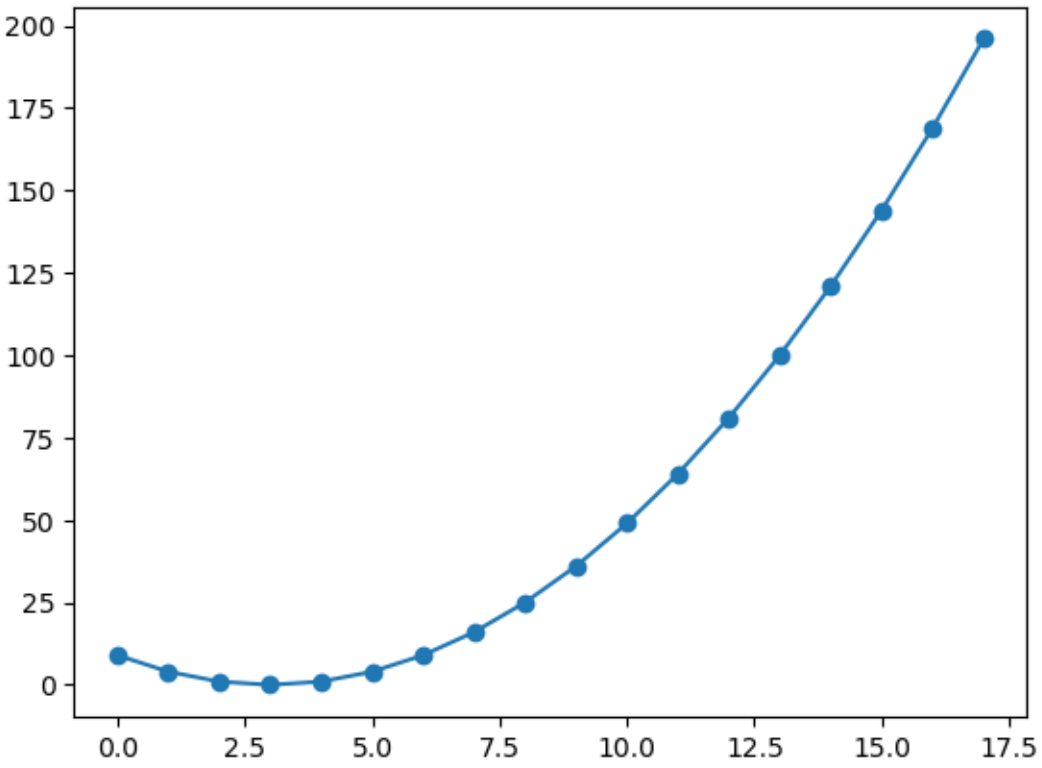
```
Une borne :17
```

[19]:
```
termes
```

[19]:
```
[9, 4, 1, 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
```

[20]:
```
# 3

indices = range(n+1)

mpl.plot(indices, termes, marker='o')
```

[20]:
```
[<matplotlib.lines.Line2D at 0x106977890>]
```

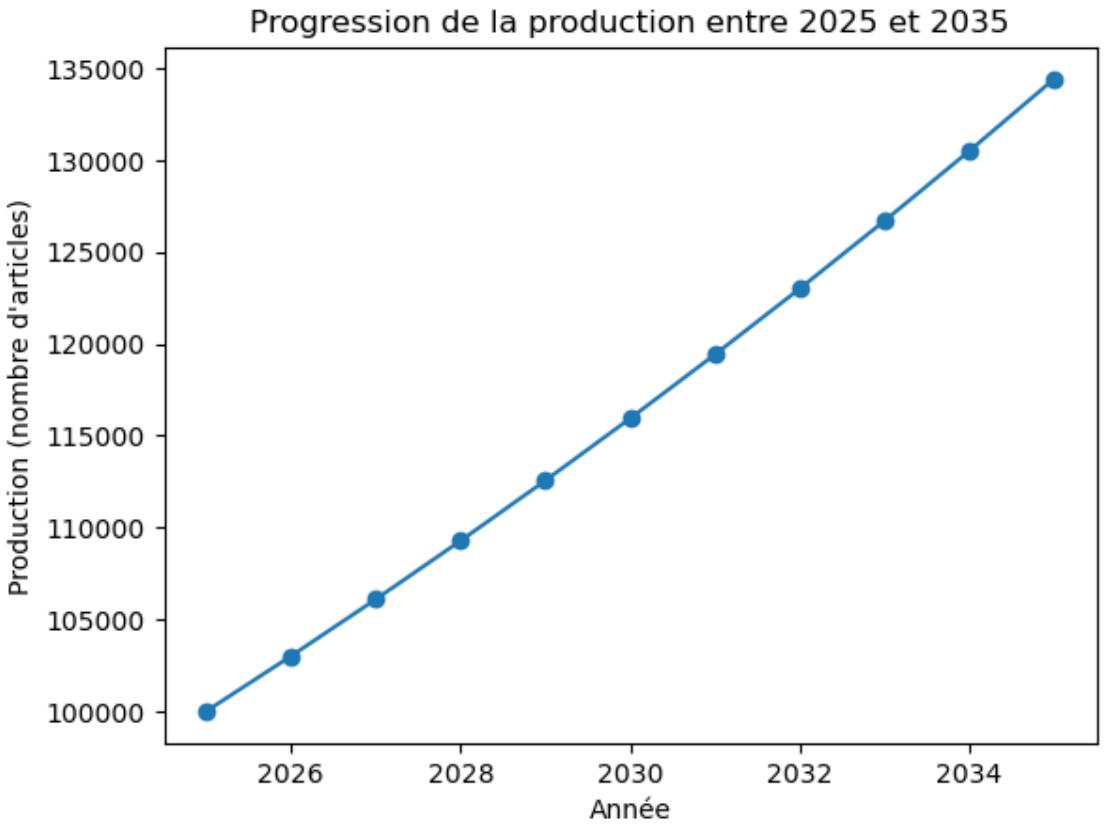


4. La suite est décroissante pour $n < 3$ et croissante sinon, ce qui est confirmé graphiquement.

[21]:
```
# Exercice 3

# 2

def production(an):
    n = an-2025    # nombre d'années à partir de 2025
    p0 = 100000
    return p0*1.03**n    # la production est une suite géométrique
```

[22]:
```
# 3

productions = [production(i) for i in range(2025, 2036)]
```

[23]:
```
# 4

années = range(2025, 2036)

mpl.plot(années, productions, marker='o')
mpl.xlabel('Année')
mpl.ylabel('Production (nombre d\'articles)')
mpl.title('Progression de la production entre 2025 et 2035')

mpl.show()
```



[24]:
```
print('Total des articles produits :', int(sum(productions)))
```

```
Total des articles produits : 1280779
```

[25]:
```
# Exercice 4

# 1

def F(n):
    assert isinstance(n, int)
```

```
    if n<=1:
        return n
    else:
        x0 = 0
        x1 = 1
        for _ in range(2, n+1):
            x = x0 + x1
            x0 = x1
            x1 = x
        return x

F(2)
```

[25]: 1

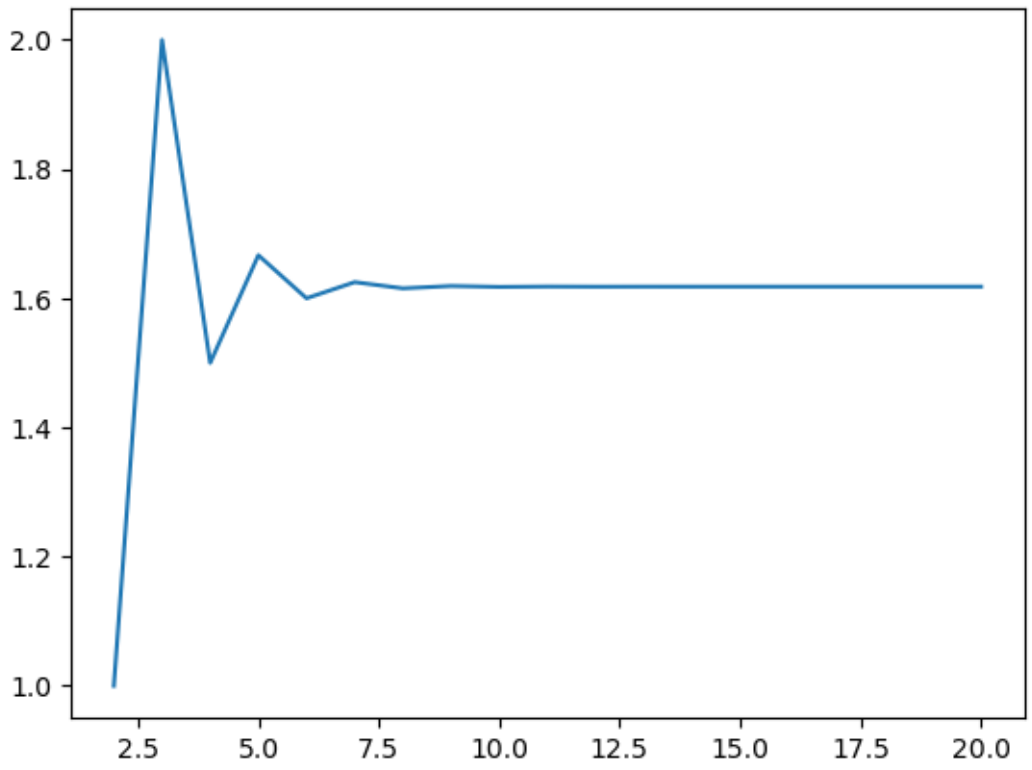
[26]:
```
F(10)
```

[26]: 55

[27]:
```
# 2

rapports = [F(n)/F(n-1) for n in range(2, 21)]
```

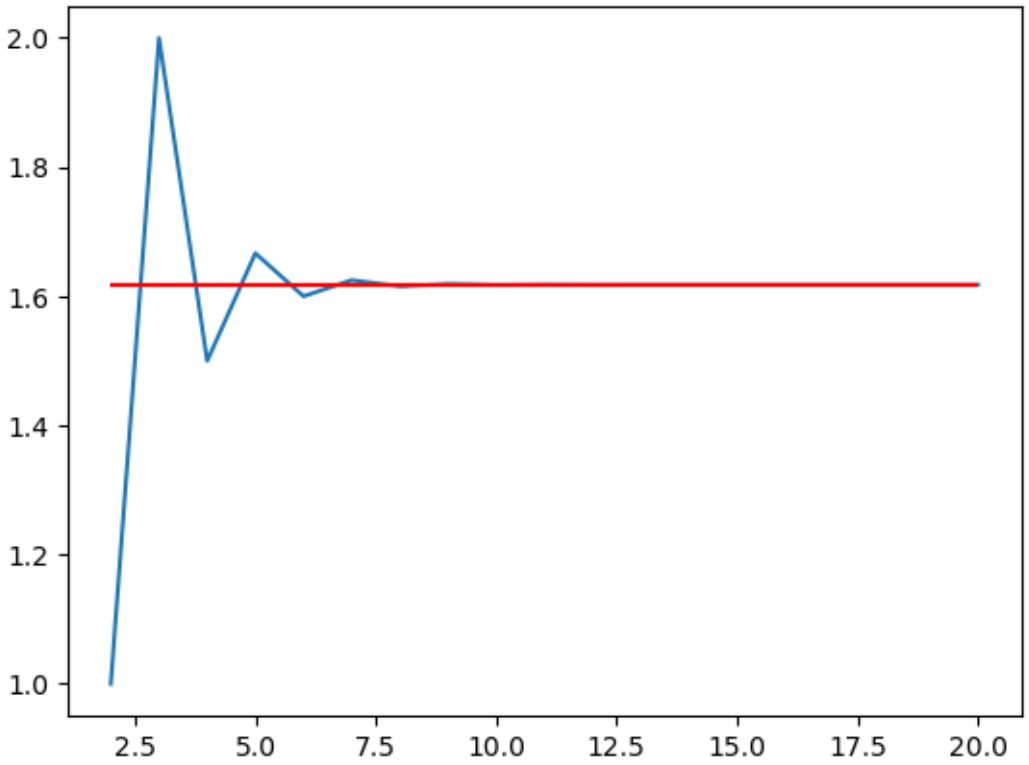
[28]:
```
# 3

indices = range(2, 21)

mpl.plot(indices, rapports)
```

[28]: [<matplotlib.lines.Line2D at 0x1068d0710>]



[29]:
```
# 4

mpl.plot(indices, rapports)

# tracer une ligne horizontale en nombre d'or entre 2 et 20
mpl.hlines((1+np.sqrt(5))/2, 2, 20, color='red')
```

[29]: <matplotlib.collections.LineCollection at 0x106a7b4d0>



5. On peut observer graphiquement que $\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \varphi$