BTS ABM 1
corrigé du TP n°2
Sommes des inverses des entiers

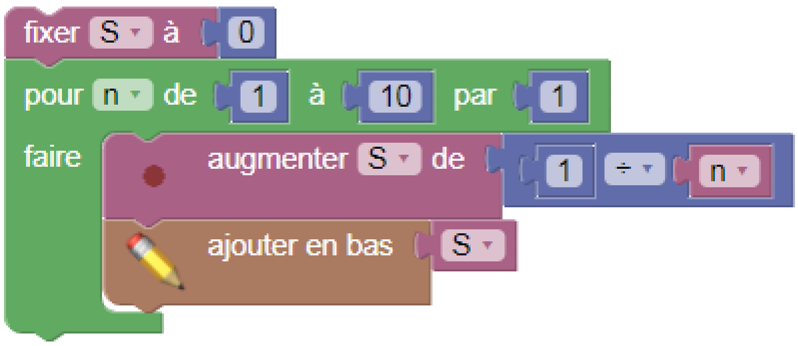
<u>Sujet</u> : Le but du TP était d'étudier comment varient les sommes des inverses des entiers consécutifs, en comparaison avec la fonction ln.

<u>I/ Somme des inverses</u>

Pour additionner les 1/n où n est entier, on utilise le même algorithme que dans le TP n°1 :

- Initialiser à 0 une variable S (comme « somme »)
- ajouter à S, au fur et à mesure, les 1/n

Avec Sofus cela donne



On obtient ces nombres :

```
1
1.5
1.8333333333333333
2.083333333333333
2.283333333333333
2.4499999999999997
2.5928571428571425
2.7178571428571425
2.8289682539682537
2.9289682539682538
```

Avec SofusPy on récupère le code Python:

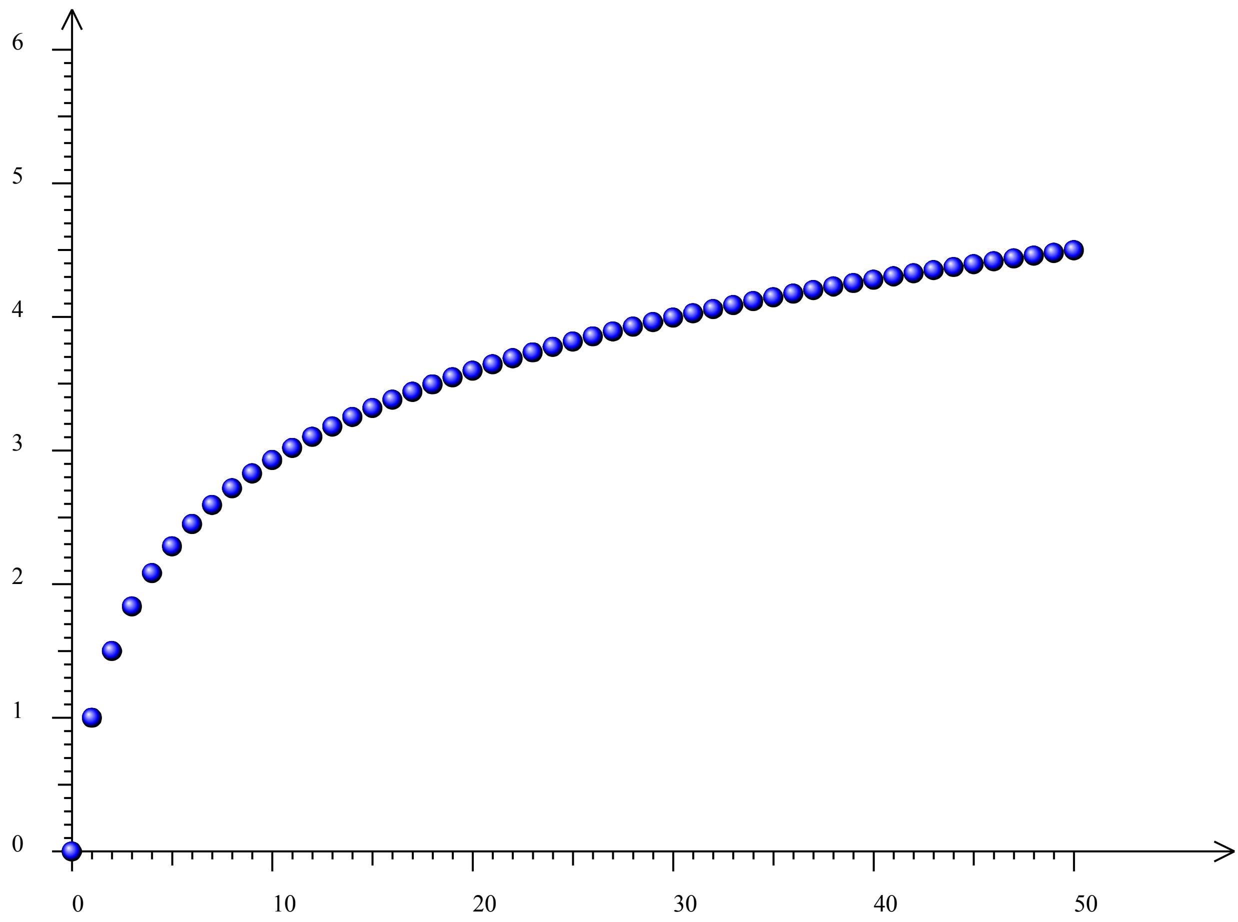
```
S = 0
for n in range(1, 11):
     S = S + 1 / n
     print(S)
```

Avec alcoffeethmique on peut aussi représenter graphiquement la suite :

```
S = 0
suite = [0]
pour n dans [1..50]
    S = S+1/n
    suite.empile S
dessineSuite suite, 50, 0, 6
```

Au passage on récupère le pseudocode :

S ← 0
pour n∈[1..50]
  S ← S+1/n



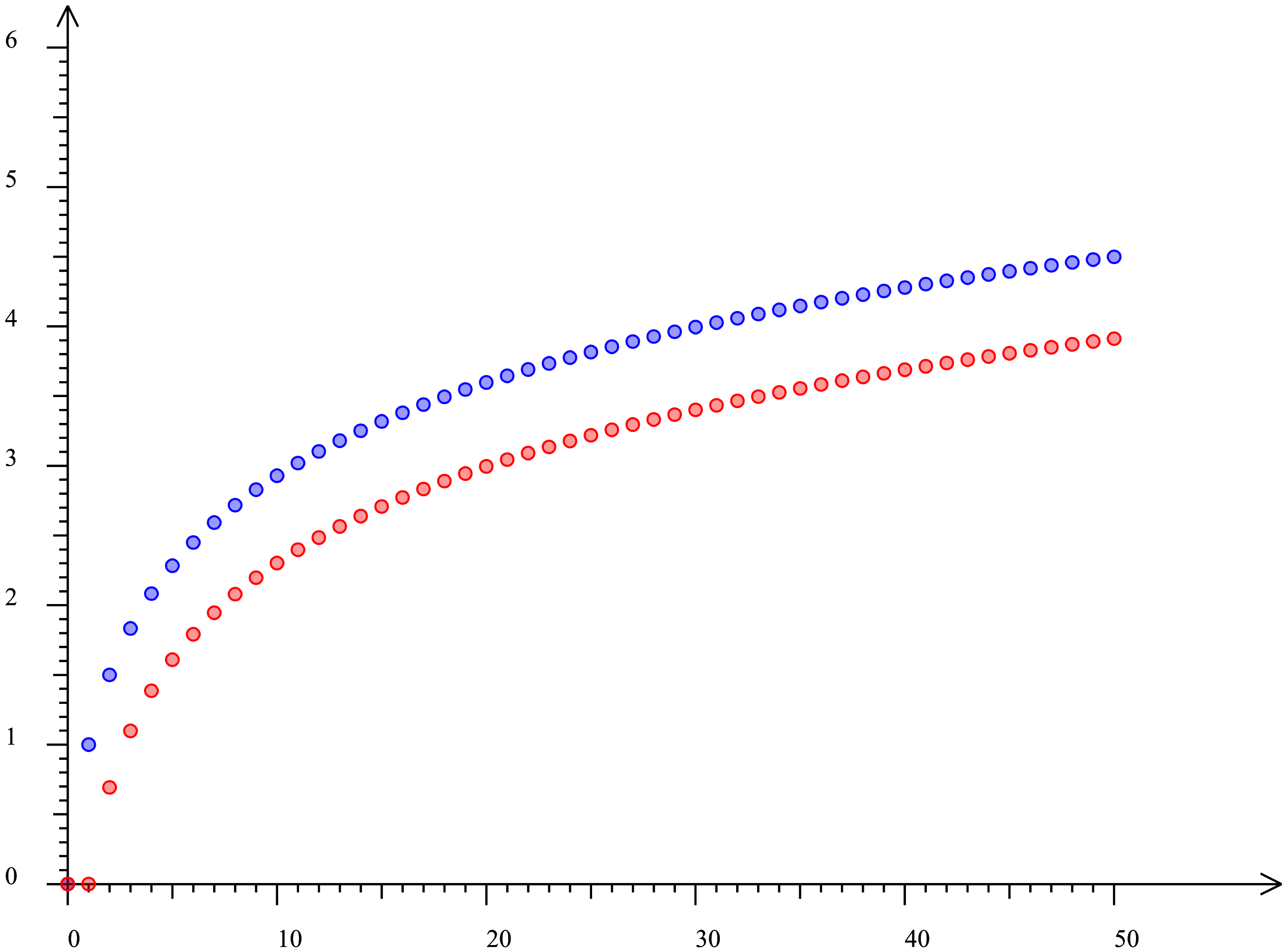
Le graphique suggère

- que la suite tend vers l'infini
- mais aussi qu’elle ressemble à celle des logarithmes népériens des entiers.

Cela se confirme avec la superposition des deux graphiques, le second étant obtenu avec ce script :

```
suite = [0]
pour n dans [1..50]
    suite.empile ln(n)
dessineSuite suite, 50, 0, 6, 3
```

La superposition des graphiques suggère que la suite des sommes des inverses est simplement une version décalée de celle des logarithmes :



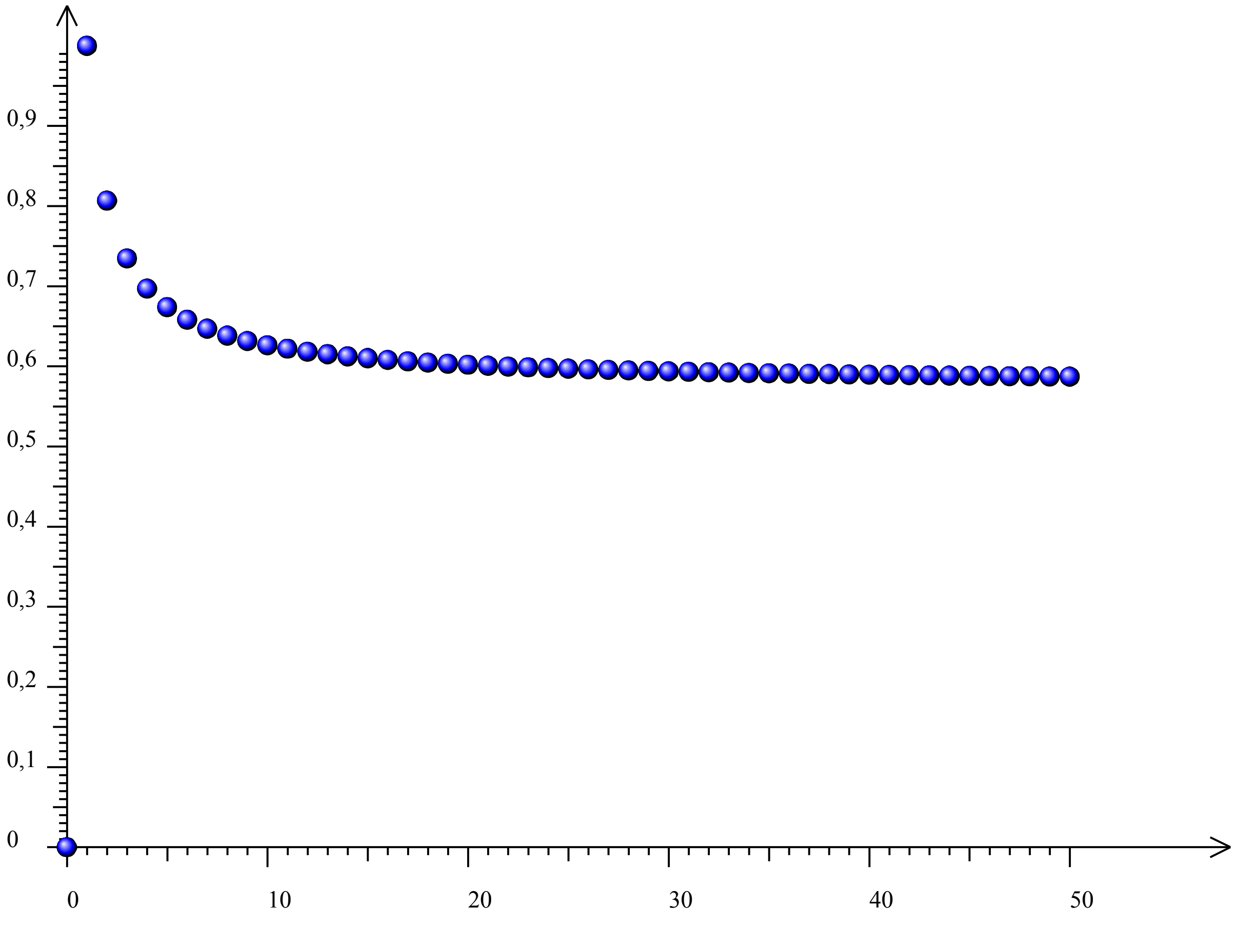
On va donc , dans la seconde partie, calculer et afficher les différences pour voir si celles-ci semblent converger vers une limite à estimer.

II/ Comparaison avec le logarithme népérien

Il suffit pour étudier la différence, de stocker dans le tableau, non pas les valeurs de S elles-mêmes, mais celles de S-ln(n) :

```
S = 0
suite = [0]
pour n dans [1..50]
    S = S+1/n
    suite.empile S-ln(n)
dessineSuite suite, 50, 0, 1, 3, 'blue'
```

Le graphique montre que la suite des différences converge :



Vers quoi ?

On peut calculer un tableau :

```
S = 0
suite = [0]
pour n dans [1..1000000]
    S = S+1/n
    si n in [10,100,1000,10000,100000,1000000]
        affiche "#{n}\t\t#{S-ln(n)}"
```

La limite semble être proche de 0,577 :

```
10          0.6263831609742079
100         0.5822073316515288
1000        0.5777155815682065
10000       0.5772656640681646
100000      0.5772206648931064
1000000     0.5772161649007153
```

Le nombre calculé ainsi s'appelle constante d'Euler. Elle est difficile à calculer parce que la suite converge lentement.