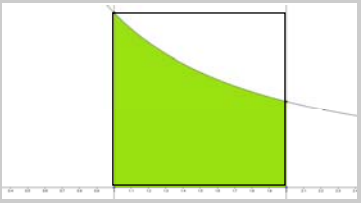
La méthode de Monte-Carlo pour calculer ln2

Le problème*

Le logarithme népérien de 2, que l'on note ln 2, est égal à l'aire comprise entre l'axe (Ox) et l'hyperbole d'équation $y = 1/x$ entre les abscisses 1 et 2.
Autrement dit, c'est l'aire de la partie grisée dans le graphe ci-contre.



Pour trouver une valeur approchée de ln 2, les programmes vont utiliser une méthode statistique, la méthode « de Monte-Carlo ». Elle consiste à « bombarder » aléatoirement un grand nombre de fois le carré $[1 ; 2] \times [0 ; 1]$ de côté 1 représenté sur le graphe et à calculer ensuite la fréquence des tirs qui ont atteint la partie grisée.

Complément culturel

C'est John Napier ou Neper (1550–1617) qui, en publiant en 1614 son traité *Mirifici Logarithmorum Canonis Descriptio,* introduit les logarithmes, qui possèdent la propriété remarquable suivante : **le logarithme d'un produit de deux nombres est la somme des logarithmes de ces deux nombres**.

Les programmes

Python

Le programme « approcheln2 » ci-dessous effectue dix millions de « bombardements » aléatoires sur le carré $[1;2] \times [0;1]$.
Le compteur *i* est un compteur de boucles, le compteur K enregistre le nombre de fois où le point est tombé sous l'hyperbole.
Il est nécessaire, avant de taper le programme, d'importer la fonction random() qui affiche un nombre aléatoire compris entre 0 et 1.

le programme

```
>>> from random import random
>>> def approcheln2():
        K=0
        i=0
        while i<10**7:
                x=1+random()
                y=random()
                if y<1/x:
                        K=K+1
                i=i+1
        return(K/10**7)
```

le résultat

```
>>> approcheln2()
0.69300600000000001
```

Scratch

Pour créer un point qui tombe aléatoirement à l'intérieur du carré, il suffit de prendre deux variables aléatoires x et y variant respectivement sur [1 ; 2] et sur [0 ; 1]. La variable K comptabilise le nombre de fois où le point tombe dans la zone grisée, c'est-à-dire tel que $y < 1 / x$. À la fin, le programme renvoie la fréquence de ces points, c'est-à-dire le résultat de la division de K par 1 000.

AlgoBox

Les variables k et i sont des compteurs. Le compteur k comptabilise le nombre de fois où le point de coordonnées $(x ; y)$ tombe dans la zone grisée. La variable i est un compteur de boucles que l'on fait varier de 1 à 10 000. random() permet d'obtenir un nombre (pseudo)aléatoire compris entre 0 et 1 et donc 1 + random() un nombre aléatoire compris entre 1 et 2.

```
DEBUT_ALGORITHME
    k PREND_LA_VALEUR 0
    POUR i ALLANT_DE 1 A 10000
        DEBUT_POUR
        x PREND_LA_VALEUR 1+random()
        y PREND_LA_VALEUR random()
        SI (y<1/x) ALORS
            DEBUT_SI
            k PREND_LA_VALEUR k+1
            FIN_SI
        FIN_POUR
    f PREND_LA_VALEUR k/10000
    AFFICHER f
FIN_ALGORITHME
```

Prolongement*

Le mathématicien allemand Gottfried Wilhelm Leibniz (1646–1716) a montré au XVII^{e} siècle que ln 2 peut être obtenu comme somme de la « série alternée » $1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + \ldots$ et que si l'on arrête la somme à un certain rang l'erreur commise est inférieure à la valeur absolue du terme de rang suivant. Cette formule est considérée « inexploitable » : il faut par exemple calculer plus de mille termes pour obtenir une approche du résultat inférieure à 0,001. Mais la nouveauté à prendre en compte est la rapidité de calcul des ordinateurs ; le programme « formuleln2 » calcule la somme des dix millions premiers termes, et un ordinateur personnel répond en une vingtaine de secondes.

le programme

```
>>> def formuleln2():
        i=1
        s=1
        while i<=10**7:
            s=s+(-1)**i/(i+1)
            i=i+1
        return s
```

le résultat

```
>>> formuleln2()
0.69314723056009631
```