Le calcul sous toutes ses formes ?
Journées de la CII Lycée - Université
Des programmes pour calculer et compter

Guillaume François - Philippe Lac - Luc Rinaudo

CII Lycée

18 et 19 janvier 2019

Introduction

Calculs approchés et représentation des nombres

On veut calculer u_{100} avec:

$$\begin{cases} u_0 = \frac{1}{3} \\ u_{n+1} = 4 \times u_n - 1 \end{cases}$$

Calculs approchés et représentation des nombres

Un exemple de script :

```
def recurrence(n):
        u = 1.0/3
        for i in range(n):
                u = 4*u - 1
        return u
print(recurrence(100))
```

Calculs approchés et représentation des nombres

Un exemple de script :

```
def recurrence(n):
        u = 1.0/3
        for i in range(n):
                u = 4*u - 1
        return u
print(recurrence(100))
```

On trouve -2,9734326931374163e+43. **Pourquoi?**

Calculs approchés et représentation des nombres

0	0	24	0.328125	48	-1.4660155037e+12
1	0.333333333333	25	0.3125	49	-5.86406201480e+12
2	0.333333333333	26	0.25	50	-2.34562480592e+13
3	0.333333333333	27	0.0	51	-9.38249922369e+13
4	0.333333333333	28	-1.0	52	-3.75299968948e+14
5	0.333333333333	29	-5.0	53	-1.50119987579e+15
6	0.333333333333	30	-21.0	54	-6.00479950316e+15
7	0.333333333333	31	-85.0	55	-2.40191980126e+16
8	0.333333333332	32	-341.0	56	-9.60767920506e+16
9	0.333333333328	33	-1365.0	57	-3.84307168202e+17
10	0.333333333314	34	-5461.0	58	-1.53722867281e+18
11	0.333333333256	35	-21845.0	59	-6.14891469124e+18
12	0.333333333023	36	-87381.0	60	-2.45956587649e+19
13	0.333333332092	37	-349525.0	61	-9.83826350598e+19
14	0.333333328366	38	-1398101.0	62	-3.93530540239e+20
15	0.333333313465	39	-5592405.0	63	-1.57412216096e+21
16	0.33333325386	40	-22369621.0	64	-6.29648864383e+21
17	0.333333015442	41	-89478485.0	65	-2.51859545753e+22
18	0.333332061768	42	-357913941.0	66	-1.00743818301e+23
19	0.33332824707	43	-1431655765.0	67	-4.02975273205e+23
20	0.333312988281	44	-5726623061.0	68	-1.61190109282e+24
21	0.333251953125	45	-22906492245.0	69	-6.44760437128e+24
22	0.3330078125	46	-91625968981.0	...	...
23	0.33203125	47	-3.66503875925e+11	100	-2.97343269314e+43

Calculs approchés et représentation des nombres

Faire effectuer le calcul suivant :

$$0,1+0,2$$

Calculs approchés et représentation des nombres

Faire effectuer le calcul suivant :

$$0,1+0,2$$

```
>>> 0.1+0.2
0.30000000000000004
>>>
```

Calculs approchés et représentation des nombres

Faire effectuer le calcul suivant :

$$0,1+0,2$$

```
>>> 0.1+0.2
0.30000000000000004
>>>
```

Remarque :

```
>>> 0.05+0.25
0.3
>>>
```

Écriture binaire des nombres

Notre numération est décimale, c'est-à-dire que nous utilisons une écriture en base 10, dont l'alphabet est constitué des dix chiffres que tout le monde connaît:

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Écriture binaire des nombres

Notre numération est décimale, c'est-à-dire que nous utilisons une écriture en base 10, dont l'alphabet est constitué des dix chiffres que tout le monde connaît:

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Dans les ordinateurs, les nombres sont représentés en binaire, c'est-à-dire en base 2, parce que c'est le plus simple à mettre en œuvre techniquement. Tout ce qui change, c'est que l'on a maintenant un alphabet réduit à deux chiffres:

$$\{0, 1\}$$

Écriture binaire des nombres

Dans le jargon des informaticiens on parle souvent de « bits » et non de « chiffres », c'est équivalent (à ceci près que le mot « bit », contraction de l'anglais « *binary digit* », n'est utilisé que pour la base 2).

Écriture binaire des nombres

Dans le jargon des informaticiens on parle souvent de « bits » et non de « chiffres », c'est équivalent (à ceci près que le mot « bit », contraction de l'anglais « *binary digit* », n'est utilisé que pour la base 2).

Les premiers nombres entiers s'écrivent

0, 1, 10, 11, 100, 101, 110, 111, ...

Écriture binaire des nombres

Par exemple, on a

$$171 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

donc le nombre 171 s'écrit « 10101011 » en base 2; il s'écrit sur 8 bits

Écriture binaire des nombres

la décomposition du nombre 87 en base 2 donne:

$$\begin{aligned} 87 &= 2 \times 43 + 1 \\ 43 &= 2 \times 21 + 1 \\ 21 &= 2 \times 10 + 1 \\ 10 &= 2 \times \ 5 + 0 \\ 5 &= 2 \times \ 2 + 1 \\ 2 &= 2 \times \ 1 + 0 \\ 1 &= 2 \times \ 0 + 1 \end{aligned}$$

$$87 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Ainsi 87 s'écrit 1010111 en base 2.

Écriture binaire des nombres

Inversement, considérons une écriture en binaire, par exemple : 10011.
Pour calculer sa valeur en base 10, il suffit de multiplier chaque chiffre par la puissance de 2 correspondant à sa position et d'additionner les résultats. On obtient:

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19$$

donc 10011 en base 2 s'écrit 19 en base 10.

Écriture binaire des nombres

Le nombre décimal $45,75$, si on le décompose en somme de puissances de 2, on a: $45,75 = 2^5 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2}$
Ainsi $45,75$ s'écrit $101101,11$ en base 2.

Écriture binaire des nombres

Le nombre décimal $45,75$, si on le décompose en somme de puissances de 2, on a: $45,75 = 2^5 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2}$
Ainsi $45,75$ s'écrit $101101,11$ en base 2.

$0,75 \times 2 = 1,5 \rightarrow 1$

Écriture binaire des nombres

Le nombre décimal $45,75$, si on le décompose en somme de puissances de 2, on a: $45,75 = 2^5 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2}$
Ainsi $45,75$ s'écrit $101101,11$ en base 2.

$0,75 \times 2 = 1,5 \rightarrow 1$

$0,5 \times 2 = 1 \rightarrow 1$

Écriture binaire des nombres

Le nombre décimal $45,75$, si on le décompose en somme de puissances de 2, on a: $45,75 = 2^5 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2}$
Ainsi $45,75$ s'écrit $101101,11$ en base 2.

$0,75 \times 2 = 1,5 \rightarrow 1$

$0,5 \times 2 = 1 \rightarrow 1$

D'où les 11 dans la partie décimale.

Écriture binaire des nombres

Écrire $0,1$, en base 2 :

Écriture binaire des nombres

Écrire $0,1$, en base 2 :

$\overline{0,1}^{10} \approx \overline{0,0001100110011001100110...}^{2}$

.

Écriture binaire des nombres

Il y a un problème d'approximation.
Il faut se méfier quand on utilise les flottants en programmation.

On se propose de résoudre, dans $\mathbb{N}^3$, l'équation $a^2+b^2+c^2=2386$ avec $a \leq b \leq c$.

1. Montrer que $a^2+b^2+c^2 \geq 3a^2$ et en déduire que $a<29$.
2. Justifier que $c^2<2386$ et en déduire que $c<49$.
3. Concevoir un algorithme qui recherche de façon exhaustive tous les triplets d'entiers $(a;b;c)$ solutions de l'équation $a^2+b^2+c^2=2386$ avec $a \leq b \leq c$.

Cet exercice repose sur le principe suivant :
On ne peut écrire un algorithme de recherche des solutions dans $\mathbb{N}^3$ tout entier. On peut par contre se débarrasser du problème d'infini, en montrant que les solutions de l'équation appartiennent à un ensemble borné.

- À partir de $a \leq b \leq c$, on montre facilement que $3a^2 \leq a^2 + b^2 + c^2$
  Par conséquent, pour $3a^2 > 2386$, c'est-à-dire $a \geq 29$, aucun triplet $(a; b; c)$ n'est solution.
- De même, $a^2 + b^2 + c^2 \geq c^2$ donc pour $c^2 > 2386$, c'est-à-dire $c \geq 49$, aucun triplet $(a; b; c)$ n'est solution.

On a ainsi $0 \leq a \leq 28$ et $0 \leq c \leq 48$ sachant que $a \leq b \leq c$.
Il ne reste plus qu'un nombre fini de valeurs à explorer, ce qui peut se traiter à l'aide d'un algorithme exhaustif :

```
pour a de 0 à 28 faire
    pour b de a à 48 faire
        pour c de b à 48 faire
            si a² + b² + c² = 2386 alors afficher (a; b; c);
        fin
    fin
fin
```

Résoudre dans $\mathbb{N}^2$ l'inéquation :$5x^2 - 4xy + y^2 \leq 100$.

Indication : $5x^2 - 4xy + y^2 = x^2 + (2x - y)^2$

Pour tous les réels x et y, on a :

$$\begin{aligned} 5x^2-4xy+y^2 &= x^2+4x^2-4xy+y^2 \\ &= x^2+(2x-y)^2 \end{aligned}$$

On constate que :
si $x^2>100$ ou $(2x-y)^2>100$ alors l'inéquation ne peut être vérifiée.
La première inéquation impose d'avoir

$$-10\leq x\leq 10.$$

La seconde inéquation impose d'avoir

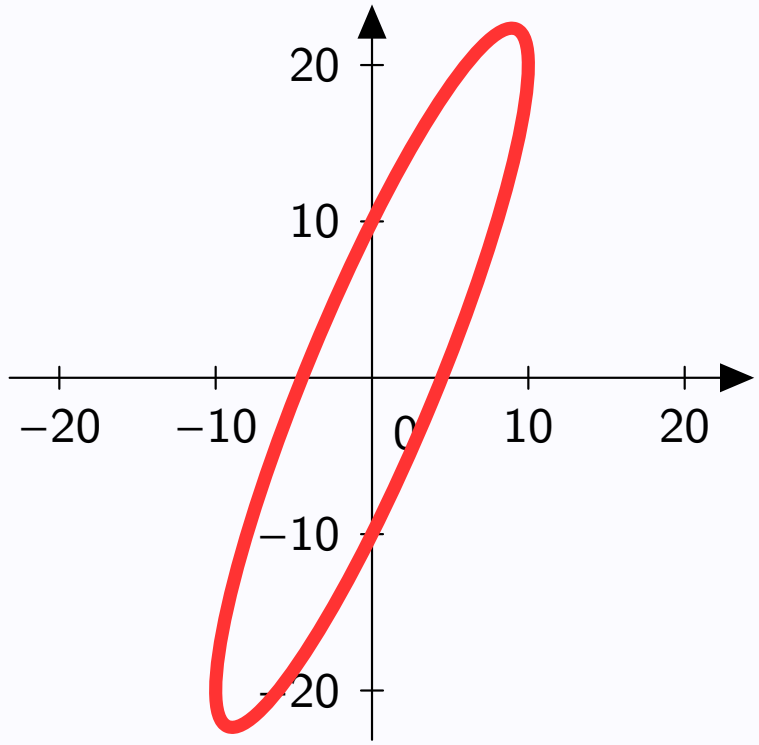
$$-10\leq 2x-y\leq 10.$$

ce dernier résultat donne $y\leq 2x+10$, puis $y\leq 30$.

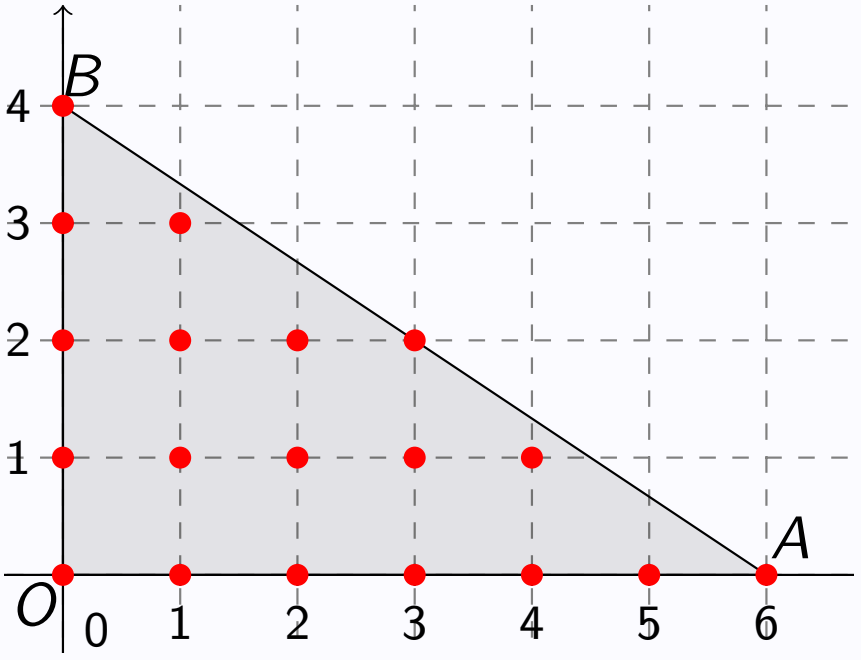
- Nous avons ainsi trouvé un majorant des solutions.
- La liste des solutions peut être obtenue à partir d'un algorithme exhaustif :

```
pour x de 0 à 10 faire
    pour y de 0 à 30 faire
        si 5x^2 − 4xy + y^2 ≤ 100 alors afficher (x; y);
    fin
fin
```

Remarque : ce problème revient à rechercher les points de coordonnées entières se trouvant à l'intérieur de l'ellipse tracée ci-dessous :



Une variante de ce problème[1] :



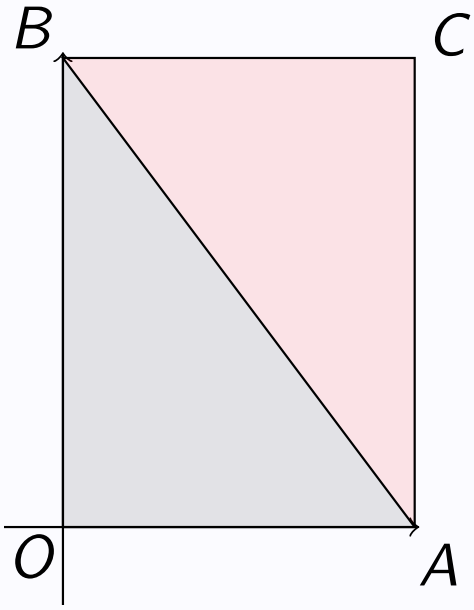
Le plan est muni d'un repère orthonormé.
Dans l'exemple ci-contre, on considère les points $A(6;0)$ et $B(0;4)$.
Le nombre N de points de coordonnées entières se trouvant à l'intérieur du triangle OAB, ou sur le triangle OAB, est égale à 19.

Que vaut N lorsque les points A et B ont pour coordonnées respectives $(300;0)$ et $(0;400)$?

[1] pour aller plus loin, voir le théorème de Pick

Une solution

On considère le rectangle $OACB$:



On note :

- n_1 le nombre de points du triangle OAB
- n_2 le nombre de points du triangle ACB
- n le nombre de points du segment $[AB]$

On a $n_1 + n_2 - n = 301 \times 401$ or $N = n_1 = n_2$ (par symétrie)
et donc $2N - n = 120701$

Une solution

Il nous reste à déterminer n :

Une équation de la droite (AB) est : $4x + 3y = 1200$.
On doit dénombrer les solutions de l'équation diophantienne :

$$4x + 3y = 1200$$

avec x et y entiers tels que $0 \leq x \leq 300$ et $0 \leq y \leq 400$

Une solution

On arrive[2] à :

$$\begin{cases} x = 1200 - 3k \\ y = -1200 + 4k \end{cases}, \quad k \in \mathbb{Z}$$

d'autre part les conditions
$0 \leq x \leq 300$ et $0 \leq y \leq 400$
entraîne $300 \leq k \leq 400$.

On en déduit alors que $n = 400 - 300 + 1 = 101$
puis que

$$N = \frac{120701 + 101}{2} = 60401$$

[2] Voir cours de TS spé par exemple

Un programme

```
def N(a,b):
    S=0
    for x in range(a+1):
        for y in range(b+1):
            if b*x+a*y<=a*b:
                S=S+1
    return S

print(N(6,4))
print(N(300,400))
```

Résultat :

```
19
60401
```

triplets $a < b < c$

Soit N un entier supérieur ou égal à 2.
On s'intéresse au nombre de triplets (a, b, c) appartenant à $[\![0; N-1]\!]^3$ vérifiant $a < b < c$?

1. Réaliser un programme itératif qui dénombre ces triplets pour un entier N donné.
2. Programmer la méthode de Monte-Carlo afin d'estimer ce nombre de triplets(nombre de tirages recommandé=10^6).
3. Tester ces programmes pour $N = 100, 1000, 10000$.

triplets $a < b < c$ Version itérative

```
def compte1(n):
    S=0
    for a in range(n):
        for b in range(n):
            for c in range(n):
                if a<b and b<c:
                    S=S+1
    return S
```

triplets $a < b < c$ Version itérative2

```
def compte1(n):
    S=0
    for a in range(n):
        for b in range(a+1,n):
            for c in range(b+1,n):
                S=S+1
    return S
```

triplets $a < b < c$ Monte-Carlo

```
def compteMC(n):
    S=0
    for k in range(10**6):
        a=randrange(n)
        b=randrange(n)
        c=randrange(n)
        if a<b and b<c:
            S=S+1
    return (float(S*n*n*n))/(10**6)
```

Une marche aléatoire : *le paradoxe des grenouilles*

Une grenouille veut rejoindre une mare située à un mètre devant elle. Elle effectue pour l'atteindre des sauts supposés en ligne droite de longueur aléatoire entre 0 et 1 mètre, suivant ainsi une répartition uniforme sur [0 ; 1]. Elle atteint la mare lorsque la distance totale parcourue devient strictement supérieure à 1m.

- Quelle est la distance moyenne parcourue en deux sauts ?
- Quel est le nombre moyen de sauts nécessaires pour atteindre la mare ?

Autant la première question trouve une réponse immédiate pour un élève de terminale, autant la seconde lève un "pseudo" paradoxe lié à des difficultés de conception. Dès lors, on peut orienter les élèves vers une approche de modélisation et leur faire conjecturer le résultat via une méthode algorithmique.

Une marche aléatoire : *le paradoxe des grenouilles*

Exemple :

1. Quelle est la probabilité que la grenouille atteigne la mare **au premier saut ?**
2. On cherche à déterminer la probabilité que la grenouille atteigne la mare **au deuxième saut.**
   a) À l'aide d'un programme en Python 3, simuler un grand nombre de trajectoires de grenouille et déterminer à quelle fréquence la grenouille atteint la mare en deux sauts.
   b) Conjecturer la probabilité cherchée.
   c) Déterminer cette probabilité.
   Indication: On pourra considérer la surface du plan délimitée par $0 \leq X \leq 1$, $0 \leq Y \leq 1$ et par le demi-plan $X+Y \leq 1$

Une marche aléatoire : *le paradoxe des grenouilles*

```
from random import *
n=int(input("Nombre de trajectoires :"))
s=0

for i in range(n) :
        d=random()+random()
        if d>1 :
                s=s+1

print(s/n)
```

Une marche aléatoire : *le paradoxe des grenouilles*

3. On cherche à déterminer la probabilité que la grenouille atteigne la mare **au troisième saut**.
   a) Conjecturer cette probabilité en effectuant plusieurs simulations.
   b) Déterminer cette probabilité.
   Indication: On pourra considérer la volume de l'espace délimité par $0 \leq X \leq 1$, $0 \leq Y \leq 1$, $0 \leq Z \leq 1$ et par les demi-espaces $X+Y \leq 1$ et $X+Y+Z < 1$
4. On cherche à déterminer **le nombre moyen de saut** nécessaires à la grenouille pour atteindre la mare.
   a) Concevoir un programme en Python 3 pour simuler n trajectoires jusqu'à la mare et estimer ce nombre moyen de sauts.
   b) Quelle conjecture peut-on émettre sur la valeur de ce nombre ?

Une marche aléatoire : *le paradoxe des grenouilles*

```
import random, numpy
def sauts():
    d, s = 0, 0
    while d < 1 :
        d+=random.random()
        s=S+1
    return s

n = int(input("Nombre de trajectoires :")
l = [ ]
for k in range(n):
    l.append(sauts())
print(numpy.mean(l))
```

Une marche aléatoire : *le paradoxe des grenouilles*

Éléments théoriques :

Pour tout $n \in \mathbb{N}^*$, désignons par X_n la v.a. qui donne la longueur du n^{eme} saut en mètre.

Ainsi $(X_n)_{n\in\mathbb{N}^*}$ une suite de v.a. indépendantes suivant $\mathscr{U}([0;1])$.

Pour tout $n \in \mathbb{N}^*$, désignons par S_n la v.a. qui donne longueur de la trajectoire en mètre au bout de n sauts. Ainsi $S_n = \sum_{i=1}^{n} X_i$.

Montrons par récurrence que : $\forall n \in \mathbb{N}^*, \forall x \in [0;1],\ \mathbb{P}(S_n \leq x) = \dfrac{x^n}{n!}$.

Remarque : S_n suit une loi de Irwin–Hall de paramètre n. L'expression cherchée correspond à la restriction sur $[0;1]$ de la fonction de répartition de S_n, soit $F_{S_n}|_{[0;1]}$.

Une marche aléatoire : *le paradoxe des grenouilles*

Soit $x \in [0;1]$.

Initialisation :

$$\mathbb{P}(S_1 \leq x) = \mathbb{P}(X_1 \leq x) = x = \frac{x^1}{1!}$$

Hérédité :

Soit $k \in \mathbb{N}^*$. Supposons que $\mathbb{P}(S_k \leq x) = \frac{x^k}{k!}$.

En remarquant que $S_{k+1} = S_k + X_{k+1}$ et que S_k et X_{k+1} sont indépendantes, on en déduit que la densité de S_{k+1} est donnée par le produit de convolution des densités de S_k et X_{k+1} autrement dit que :

$$f_{S_{k+1}}(x) = \int_{-\infty}^{+\infty} f_{S_k}(t) f_{X_{k+1}}(x-t)dt$$

Une marche aléatoire : *le paradoxe des grenouilles*

Sur $\mathbb{R}$, la fonction densité de X_{k+1} est :

$$f_{X_{k+1}}(t) = \mathbb{1}_{[0;1]}(t)$$

Sur $\mathbb{R}^-$, $\mathbb{P}(S_k \leq t) = 0$, donc la fonction densité de S_k s'écrit :

$$f_{S_k}(t) = 0$$

Sur $[0;1]$, d'après l'hypothèse de récurrence, $\mathbb{P}(S_k \leq t) = \dfrac{t^k}{k!}$ donc la fonction de répartition et la fonction densité de S_k s'écrivent :

$$F_{S_k}(t) = \frac{t^k}{k!} \text{ et } f_{S_k}(t) = \frac{t^{k-1}}{(k-1)!}$$

Une marche aléatoire : *le paradoxe des grenouilles*

Ainsi $f_{S_{k+1}}(x) = \int_{-\infty}^{+\infty} f_{S_k}(t) \cdot \mathbb{1}_{[0;1]}(x-t)dt$.

$\forall t \in \mathbb{R},\ 0 \leq x-t \leq 1 \iff x-1 \leq t \leq x$

et $0 \leq x \leq 1 \iff x-1 \leq 0 \leq x$.

Il vient :

$$\begin{aligned} f_{S_{k+1}}(x) &= \int_{x-1}^{x} f_{S_k}(t)dt \\ &= \int_{x-1}^{0} f_{S_k}(t)dt + \int_{0}^{x} f_{S_k}(t)dt \\ &= \int_{x-1}^{0} 0dt + \int_{0}^{x} \frac{t^{k-1}}{(k-1)!}dt \\ &= [\frac{t^k}{k!}]_0^x \\ &= \frac{x^k}{k!} \end{aligned}$$

Une marche aléatoire : *le paradoxe des grenouilles*

Puis que :

$$\begin{aligned} F_{S_{k+1}}(x) &= \int_{-\infty}^{x} f_{S_{k+1}}(t)dt \\ &= \int_{-\infty}^{0} f_{S_{k+1}}(t)dt + \int_{0}^{x} f_{S_{k+1}}(t)dt \\ &= \int_{0}^{x} \frac{t^k}{k!}dt \\ &= \frac{t^{k+1}}{(k+1)!} \end{aligned}$$

D'où $\mathbb{P}(S_{k+1} \leq x) = \dfrac{x^{k+1}}{(k+1)!}$

Une marche aléatoire : *le paradoxe des grenouilles*

Conclusion :

Par l'axiome de récurrence, pour tout $n \in \mathbb{N}^*, \mathbb{P}(S_n \leq x) = \dfrac{x^n}{n!}$

En particulier pour $x = 1$, on a : $\mathbb{P}(S_n \leq 1) = \dfrac{1}{n!}$

Une marche aléatoire : *le paradoxe des grenouilles*

Maintenant, désignons par N la v.a. qui donne le nombre de saut nécessaires à la grenouille pour dépasser 1 mètre. $N \in [\![2; +\infty[\![$.
Soit $k \in [\![2; +\infty[\![$.

L'événement $(N = k)$ est réalisé lorsque la somme des k premiers sauts est strictement supérieure à 1 sans que celle des $(k-1)$ premiers sauts le soit.
Ainsi on a : $(N = k) = (S_{k-1} \leq 1) \cap (S_k > 1)$.

Or $(S_k \leq 1) \subset (S_{k-1} \leq 1)$, donc

$$\begin{aligned} \mathbb{P}((N = k)) &= \mathbb{P}((S_{k-1} \leq 1) \cap (S_k > 1)) \\ &= \mathbb{P}(S_{k-1} \leq 1) - \mathbb{P}(S_k \leq 1). \end{aligned}$$

Une marche aléatoire : *le paradoxe des grenouilles*

Ainsi, d'après ce qu'il précède :

$$\mathbb{P}((N=k)) = \frac{1}{(k-1)!} - \frac{1}{k!} = \frac{k-1}{k!}$$

Par suite, le calcul de l'espérance de N donne :

$$\begin{aligned}
\mathbb{E}(N) &= \sum_{k=2}^{+\infty} k \cdot \frac{k-1}{k!} \\
&= \sum_{k=2}^{+\infty} \frac{1}{(k-2)!} \\
&= \sum_{k=0}^{+\infty} \frac{1}{k!} \\
&= e
\end{aligned}$$

En vrac

Programme mystère : que calcule ce script Python ?

```
from math import *
u=0
for k in range(100):
                u=cos(u)
print(u)
```

En vrac

On dispose d'une urne contenant 1000 boules numérotées de 1 à 1000.
On tire simultanément deux boules de l'urne.

Déterminer la probabilité que les numéros des boules tirées soient deux entiers premiers entre eux.

Le nombre possible de couples d'entiers formés par le tirage des deux boules, nous est donné par : $\binom{1000}{2} = 499500$
Le nombre de couples formés par des entiers premiers entre eux nous est donné par l'algorithme :

```
S ← 0
pour k de 1 à 1000 faire
    pour j de k+1 à 1000 faire
        si pgcd(k,j) = 1 alors S ← S+1;
    fin
fin
écrire S
```

La probabilité cherchée est alors égale à la valeur de S retournée par l'algorithme divisée par 499500.

En vrac

a et b désignent deux entiers naturels avec $a > b$.
Dans la division euclidienne de a par b :
on note q le quotient et r le reste r.
On sait que $a + b = 86$ et que $r = 9$.

Déterminer les couples $(a; b)$ possibles.

```
pour b de 1 à 86 faire
|   pour a de b+1 à 86 faire
|   |   si a+b=86 et a%b=9 alors afficher a et b;
|   fin
fin
```

```
pour b de 1 à 42 faire
|   si (86−b)%b=9 alors afficher a et b;
fin
```