2de Simulations et autres situations algorithmiques

1) QCM

Un QCM comporte cinq questions. Pour chacune d'entre elles, quatre réponses sont proposées mais une seule est exacte. Une réponse juste vaut 2 points, une réponse fausse enlève 1 point et lorsque le résultat est négatif on met 0. On veut simuler cette situation pour des candidats qui donnent une réponse au hasard pour chacune des cinq questions. L'algorithme suivant simule cette situation et calcule la note moyenne obtenue par N candidats sans tenir compte de la contrainte soulignée plus haut.

- Entrer N (le nombre de candidats) ; Affecter 0 à T.
- Pour I allant de 1 à N : Affecter 0 à S ;
  Pour J allant de 1 à 5 : Affecter *random*() à R
  Si R≤0,25 Alors Affecter S+2 à S Sinon Affecter S-1 à S
  Affecter T+S à T
- Afficher T÷N

a) Que doit-on corriger dans l'algorithme pour tenir compte de la contrainte soulignée ?

Pour que « lorsque le résultat est négatif on met 0 », il faut ajouter un test au bon endroit qui remette à 0 une note négative. Le bon endroit est évidemment après le calcul de la note au QCM pour un candidat, donc juste avant l'instruction « Affecter T+S à T » qui utilise la note S. Si on met ce test après cela ne sert à rien et on comptabilise les notes négatives dans le total T.

- Entrer N (le nombre de candidats) ; Affecter 0 à T.
- Pour I allant de 1 à N : Affecter 0 à S
  Pour J allant de 1 à 5 : Affecter *random*() à R
  Si R≤0,25 Alors Affecter S+2 à S Sinon Affecter S-1 à S
  Si S<0 alors Affecter 0 à S
  Affecter T+S à T
- Afficher T÷N

b) Expliquer l'instruction conditionnelle « Si R≤0,25 Alors Affecter S+2 à S Sinon Affecter S-1 à S ».
(Que réalise t-elle ? À quoi sert le test R≤0,25 ?)

Cette instruction conditionnelle examine le résultat R d'un tirage aléatoire entre 0 (0 compris) et 1 (1 non compris). Lorsque R≤0,25, c'est-à-dire dans 1 cas sur 4, on va augmenter la note de 2 points (le candidat répondant au hasard est tombé sur la bonne réponse) et sinon on va la diminuer de 1 point (il a donné une mauvaise réponse).

c) Programmer l'algorithme corrigé pour tenir compte de la contrainte soulignée, puis lancer trois fois le programme pour N=5, pour N=50 et pour N=500. Noter à chaque fois la note moyenne obtenue.

Programmons cet algorithme.

Cela ne pose aucun problème majeur si on connait la syntaxe de sa calculatrice... Je l'ai programmé sur Algobox qui est tout de même plus souple. Cela conduit aux résultats suivants (nous avons mis deux résultats pour montrer les fluctuations qui se réduisent lorsque N augmente) :

N	5	50	500	5000	50000
1er tirage	0,4	0,62	0,722	0,7162	0,73132
2d titage	0,2	0,64	0,624	0,7084	0,71856

Conclusion : alors que le candidat répond au hasard, on peut s'attendre à avoir des notes entre -5 et 10 (sans la correction). Comme le candidat moyen se trompe trois fois sur quatre, il marque en moyenne -1 dans 75% des cas et +2 dans 25% des cas, il obtient donc en moyenne $-0,25$ à chaque question $-1\times0,75+2\times0,25=-0,25$. Comme il y a 5 questions, on peut s'attendre à une moyenne de $5\times(-0,25)=-1,25$. La correction que nous avons apporté élimine les possibilités d'avoir une note négative, mais celles-ci arrivent souvent (en répondant au hasard). La moyenne des notes obtenues s'établit donc finalement aux alentours de 0,71 ou 0,72. Les premiers résultats obtenus (pour N=5) ne montrent pas bien cela car 5 est trop petit. Pour N=500, on s'approche un peu mieux. Nous avons continué jusqu'à N=50000 avec Algobox mais cette valeur est sans doute longue à obtenir avec une calculatrice...

d) Ajouter un compteur pour déterminer le pourcentage de candidats ayant 0 à ce QCM. Donner ce pourcentage pour N=5, pour N=50 et pour N=500.

Le compteur Z (comme Zéro) s'ajoute très aisément. Voici l'algorithme amélioré :

- Entrer N (le nombre de candidats) Affecter 0 à T ; Affecter 0 à Z.
- Pour I allant de 1 à N : Affecter 0 à S
  Pour J allant de 1 à 5 : Affecter *random*() à R
  Si R≤0,25 Alors Affecter S+2 à S Sinon Affecter S-1 à S
  Si S<0 alors Affecter 0 à S

Affecter Z+1 à Z
Affecter T+S à T
Afficher T÷N ; Afficher Z÷N
Ainsi modifié, on s'aperçoit que plus de la moitié des candidats répondant au hasard va avoir 0. Le tableau ci-dessous donne deux résultats pour différentes valeurs de N.

N	5	50	500	5000	50000
tirage	1,2	0,76	0,730	0,7028	0,73058
Fréquence de 0	0,4	0,6	0,642	0,6368	0,63164

On voit que 63% environ des candidats va avoir 0 à ce QCM en répondant au hasard. C'est donc un très bon QCM qui évite d'avoir des réussites dues au hasard. Celles-ci ne sont pas évitées car il se peut qu'un candidat obtienne 10/10 en répondant au hasard. Quelle est la fréquence d'une telle réussite injustifiée ? Pour répondre à cela, le plus simple est d'améliorer (encore) notre algorithme...
Nous avons fait cela et trouvons qu'il apparaît un telle réussite après 1887 candidats, 522 candidats, 16 candidats, 420, 314, 1343, 676, 84, 1114, 140, 1044, 300, 1944, 1077, 80, etc. soit une moyenne de 730 candidats entre deux candidats chanceux.

2) Inéquations

a) Déterminer à partir de quel entier n on a $1{,}01^n>100$ (justifier en donnant $1{,}01^{n-1}$ et $1{,}01^n$ arrondis au dixième).
On utilise ici un algorithme qui a été vu en cours et programmé sur la calculatrice.
Cet algorithme donne, pour $1{,}01^n>100$ (on entre A=1, B=1,01 et C=100) : n=463.
Vérifions : $1{,}01^{462}=99{,}19...$ alors que $1{,}01^{463}=100{,}18...$ On a donc bien, à partir de n=463, $1{,}01^n>100$.
Déterminer, de même, à partir de quel entier n on a $(\frac{6}{7})^n < 10^{-9}$ (justifier selon le même principe).
Ce même algorithme donne, pour $(\frac{6}{7})^n<10^{-9}$ (on entre A=1, B= $6\div7$ et C= 1×10^{-9}) : n=135.
Vérifions : $(\frac{6}{7})^{134}\approx1{,}069\times10^{-9}>10^{-9}$ alors que $(\frac{6}{7})^{135}\approx9{,}166\times10^{-10}<10^{-9}$.
On a donc bien, à partir de n=135, $(\frac{6}{7})^n<10^{-9}$.
b) Déterminer à partir de quel entier n on a :

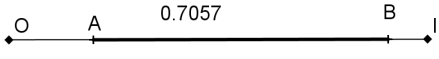
$$1{,}25+1{,}25^2+1{,}25^3+ \ ... \ +1{,}25^n>10^5$$

Pour répondre à cette question, il faut adapter un peu l'algorithme précédent qui donnerait sinon la réponse à la question $1{,}25^n>100\,000$ (on trouve alors n=52, cela nous indique seulement que la valeur de n cherchée doit être inférieure à cette valeur).
En effectuant la modification requise, on trouve n=45.
C'est-à-dire que $1{,}25+1{,}25^2+1{,}25^3+ \ ... \ +1{,}25^{45}>10^5$ alors que $1{,}25+1{,}25^2+1{,}25^3+ \ ... \ +1{,}25^{44}<10^5$.
On ne demandait pas de donner le résultat de S_{44} et S_{45} mais ce n'est pas bien difficile à obtenir (il suffit d'afficher S à la fin) : $S_{45}\approx 114\,789{,}37$ (nous avons enlevé le 1er terme 1 qui n'est pas compris dans S). Ensuite nous enlevons $1{,}25^{45}\approx22958{,}87$ à S_{45} pour trouver $S_{44}\approx 91\,830{,}50$.
c) Quel algorithme, écrit en pseudo-langage (pas de syntaxe pour calculatrice), permet de répondre à la question b ?
On fait la somme S des termes successifs de la suite géométrique $1{,}25^n$ (le 1er terme est A=1 mais il n'a pas été comptabilisé dans la somme S, et la raison de la suite est B= 1,25) jusqu'à dépasser un nombre C (ici C=100 000).
Cette méthode se programme à partir de l'algorithme suivant qui résume ce qu'on vient de dire :

1. [A, B, C, N sont des nombres] (déclarations inutiles pour les calculatrices)
2. Entrer A, B (A est le 1er terme et B la raison de la suite) et C le nombre à dépasser.
3. Affecter A à S. Affecter 0 à N.
4. Tant que S<C {Affecter A×B à A. Affecter N+1 à N. Affecter S+A à S.}
5. Afficher N et S (c'est N 1ère valeur pour laquelle $S=S_N$ est supérieur à C)

Bien sûr, il faudra adapter encore cela pour d'autres situations : cet algorithme ne permet pas de répondre à toutes les questions. Par exemple, si on entre un nombre B compris entre 0 et 1, les termes $A\times B^N$ vont diminuer jusqu'à des valeurs proches de 0, et la somme S ne va pas dépasser n'importe quel nombre C. Dans ce cas le programme va « boucler », il ne se terminerait jamais. Heureusement, la calculatrice le fermera au bout d'un moment avec un message du genre « break ».

3) Longueur au hasard



On choisit au hasard deux points A et B sur le segment [OI] de longueur 1.
On veut déterminer la probabilité p d'obtenir une longueur AB supérieure ou égale à 0,5. Faute de savoir déterminer directement p, nous allons simuler cette expérience un grand nombre de fois et comptabiliser les fois où cet événement « $E_{0,5}$: obtenir une longueur AB supérieure ou égale à 0,5 » se réalise.
On va ainsi estimer $p=p(E_{0,5})$ à partir de la fréquence expérimentale f_e obtenue sur un échantillon de n tirages.

a) En notant a et b les abscisses de A et B, la distance AB se calcule par la formule $AB=\sqrt{(a-b)^2}$
>Programmer l'algorithme suivant sur votre calculatrice :

Saisir n (le nombre de tirages)
Affecter 0 à s
Pour i allant de 1 à n : Affecter un nombre aléatoire de l'intervalle [0;1[à a
Affecter un nombre aléatoire de l'intervalle [0;1[à b ;
Affecter $\sqrt{(a-b)^2}$ à d *NB : on peut remplacer cette valeur par* $|a-b|$
Si $d\geq 0{,}5$ alors affecter $s+1$ à s
Affecter $s\div n$ à s
afficher s

Noter que s comptabilise les fois où l'événement $E_{0,5}$ se réalise.
On affiche finalement $s\div n$ (affecté à s pour réduire le nombre de variables) qui est notre fréquence expérimentale f_e.
Voici l'algorithme que chacun traduira dans le langage de sa calculatrice :

Saisir N M=0 Pour I allant de 1 à N A prend une valeur aléatoire de [0;1[B prend une valeur aléatoire de [0;1[Si la valeur absolue de A-B≥0,5	Alors M=M+1 Fin du Si Fin du Pour Afficher M M=M/N×100 Afficher M

Par exemple pour Casio :

```
?→N
0→M
For 1→I To N
Ran#→A
Ran#→B
If Abs(A-B)≥0,5
Then M+1→M
If-End
Next
M
M/N×100→M
M
```

pour Python :

```
from random import random
from math import sqrt
n=10000  #nombre de tirages
d=0.5  #nombre de tirages
m=0
for i in range(n) :
  a=random()
  b=random()
  e=abs(a-b)
  if e>=d :
    m=m+1
print("Fréquence des longueurs dépassant {} = {}%".format(d,m/n*100))
```

Fréquence des longueurs dépassant 0.5 = 24.79%

b) Lancer votre programme en prenant n=100, 1000, 10000 deux fois pour chaque valeur de n.
>Noter à chaque fois la valeur de f_e obtenue, puis calculer l'intervalle de confiance au seuil de 95%.

n	100	1000	10000
f_e pour le 1er tirage	0,24	0,244	0,2403
Intervalle de confiance 1	[0,14 ; 0,34]	[0,2124 ; 0,2756]	[0,2303 ; 0,2503]
f_e pour le 2ème tirage	0,19	0,247	0,2422
Intervalle de confiance 2	[0,09 ; 0,29]	[0,2154 ; 0,2786]	[0,2322 ; 0,2522]

L'intervalle de confiance donne les limites dans lesquelles varie la probabilité p cherchée au seuil de 5%. C'est-à-dire que $p\in[\ f_{expér.}-\frac{1}{\sqrt{n}}\ ;\ f_{expér.}+\frac{1}{\sqrt{n}}\ ]$, où $f_{expér.}$ est la fréquence expérimentale, trouvée par notre algorithme avec une taille de l'échantillon égale à n.

Quelle est alors la meilleure estimation de p compatible avec vos données au seuil de 95% ?
On peut conclure cette analyse statistique en disant que, au seuil de 95%, le segment [AB] mesure plus de 0,5 dans environ 24% ou 25% des cas. Les intervalles de confiance obtenus pour n=10000 ne permettent pas une meilleure précision.

Déterminer la taille N que doit avoir l'échantillon pour que l'intervalle de confiance ait une amplitude de 2×10^{-2} puis lancer votre programme avec cette valeur de N. En déduire la probabilité p au centième près.
Pour que cet intervalle ait une amplitude égale à 2×10^{-2}, on doit avoir :
$f_{expér.}+\frac{1}{\sqrt{n}}-(f_{expér.}-\frac{1}{\sqrt{n}})=2\times10^{-2}$, soit $\frac{2}{\sqrt{n}}=2\times10^{-2}$ ou encore $\sqrt{n}=10^2$, finalement on doit avoir $n=10^4$. Examinons la situation en lançant l'algorithme qui calcule cette fréquence pour différentes valeurs de n. Bien sûr, en recommençant les tirages aléatoires, on obtiendrait d'autres valeurs, c'est pour illustrer cela que nous avons donné 3 valeurs de f pour chaque valeur de n.

Seule, la ligne colorée en bleu donne des valeurs qui conviennent. Avec la 1ère valeur obtenue (24,95%), on déduit que pour n=10 000, le nombre M moyen est 2495 et la fréquence moyenne $f_{expér.} \approx 0{,}2495$. L'intervalle de confiance IC contenant p est donc

[0,2395 ; 0,2595]

La fréquence p peut-être 0,25 exactement (c'est compatible avec le résultat obtenu) ou bien une valeur assez proche contenue dans IC, par exemple 0,24.

n	Essai 1	Essai 2	Essai 3	moyenne
10	10	40	30	27
100	27	29	21	25,7
1 000	24,8	26,1	25,7	25,53
10 000	24,95	25,23	24,88	25,020
100 000	25,020	24,978	24,920	24,9727

Le résultat peut être amélioré sans peine si on utilise un ordinateur qui ne prend qu'un instant pour n=100 000. Pour avoir un intervalle de confiance ayant une amplitude égale à 2×10^{-3}, donc une estimation de p au millième, il faut augmenter la valeur de n jusqu'à 1 000 000. Avec mon programme en Python, sur l'ordinateur, on trouve le résultat en quelques secondes : 25.0058% ; une autre valeur obtenue est 24.9276%. Sur une calculatrice, il semble que cela soit inaccessible ; l'obtention d'un résultat pour n=10000 étant déjà relativement long.

Nous voudrions utiliser ce programme pour explorer davantage cette situation.

d) Quelles sont les probabilités des événements $E_{0,4}$ « obtenir une longueur AB supérieure ou égale à 0,4 », $E_{0,3}$ « obtenir une longueur AB supérieure ou égale à 0,3 », $E_{0,2}$, etc. ?

$p(E_{0,4})$=35,713 (puis 35,984) ; $(E_{0,3})$=48,852 (puis 48,959) ; $p(E_{0,2})$= 64,123 (puis 64,034)

L'idée est, ici, de réunir des points pour tracer le graphique donnant l'évolution de $p(E_x)$ en fonction de $x \in [0\ ;\ 1]$. Notons $E_{0,4}$ l'événement « obtenir une longueur AB supérieure ou égale à 0,4 ».

En changeant la distance d par 0,4 au lieu de 0,5, j'obtiens $p(E_{0,4})$=35,713%.

En relançant le programme une 2ème fois j'obtiens 35,984%.

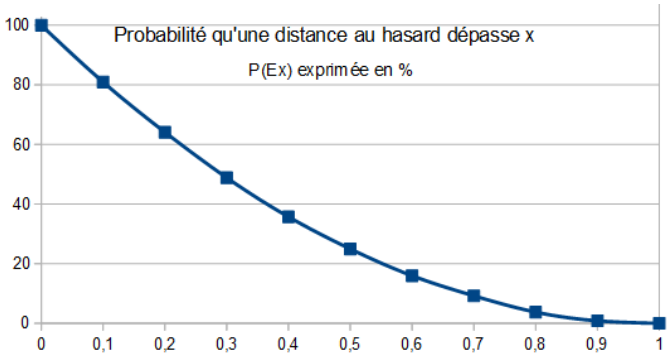
De même, je détermine $p(E_{0,3})$=48,852%, puis une 2ème fois j'obtiens 48,959% ;

De même, $p(E_{0,2})$= 64,123% (puis 64,034%)

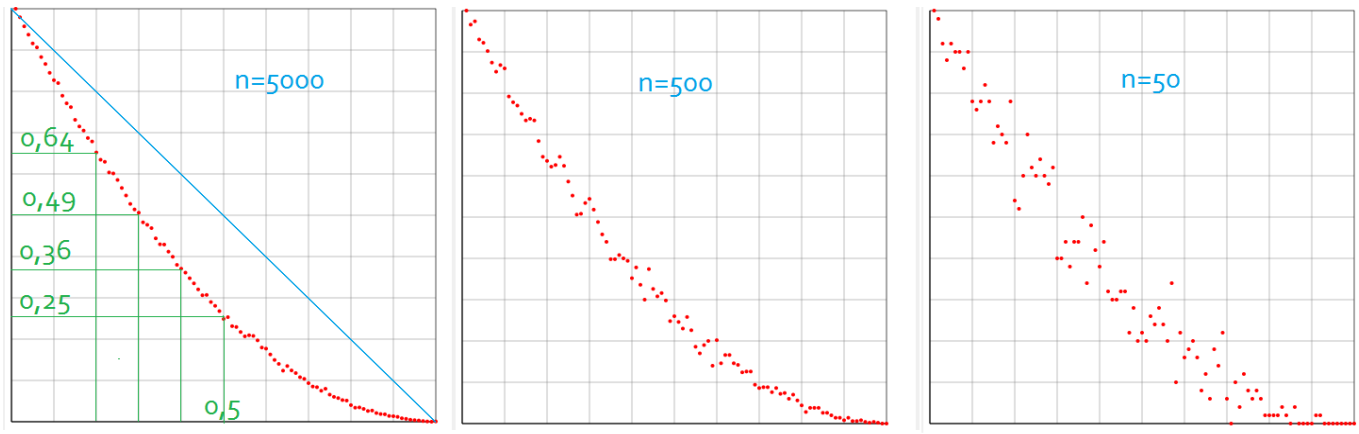
La question demande une valeur pour $p(E_x)$ lorsque $x \in \{0\ ;\ 0{,}1\ ;\ 0{,}2\ ;\ 0{,}3\ ;\ ...\ ;\ 0{,}9\ ;\ 1\}$.

x	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$p(E_x)$ - 1er essai	100%	80,94%	64,12%	48,85%	35,71%	24,95%	15,96%	9,22%	3,75%	0,84%	0,00%
$p(E_x)$ - 2ème essai	100%	81,25%	64,03%	48,96%	35,98%	25,23%	16,02%	8,72%	4,04%	0,88%	0,00%

Il faut maintenant tracer un graphique représentant cette situation. Voici un tel graphique, réalisé avec l'assistant graphique du tableur.



Les dix points du graphique suffisent pour se faire une bonne idée de la relation entre la distance d (notée x ici), mais on peut obtenir une autre courbe en faisant varier x de 0,01 en 0,01 et en exécutant le programme avec cette valeur. On va tracer le nuage de points que donne cet algorithme, pour différentes valeurs de n afin de montrer la dispersion des valeurs due à la nature aléatoire du phénomène. On a pris 50, 500 et 5000 pour valeur de n (le nombre de valeurs dans chaque échantillon).



En vert, sur le graphique de gauche, on retrouve les valeurs déterminées de $p(E_{0,5})$, $p(E_{0,4})$, $p(E_{0,3})$ et $p(E_{0,2})$.

e) Améliorer le programme afin de déterminer une estimation de la moyenne des longueurs AB.

Ce n'est pas difficile de calculer la moyenne des longueurs de segments : elle est de 0,333 soit environ un tiers.

La simulation permet d'obtenir cette valeur, à la place du raisonnement probabiliste qui devrait conduire à justifier cette valeur.
Voici l'algorithme que chacun traduira dans le langage de sa calculatrice :

```
N=10000
L=0      (on y calcule la somme des longueurs)
X=0      (on y calcule la somme des carrés des longueurs)
Pour I allant de 1 à N
  A prend une valeur aléatoire de [0;1[
  B prend une valeur aléatoire de [0;1[
  D= valeur absolue de A-B
  L=L+D
  X=X+D×D
Fin du Pour
L=L/N
Afficher L    (la moyenne des longueurs)
X=X/N         (la moyenne des carrés des longueurs)
X=X-L×L       (la variance des longueurs)
X=racine carrée de X (l'écart-type des longueurs)
Afficher X
```

Par exemple pour Casio :

```
10000→N
0→L
0→X
For 1→I To N
Ran#→A
Ran#→B
Abs(A-B)→D
L+D→L
L+D→L
Next
L/N→L
L
X/N→X
X-L×L→X
Sqrt(X)→X
X
```

pour Python :

```
from random import random
from math import sqrt
n=10000 #nombre de tirages
l,x=0,0
for I in range(n) :
    a=random()
    b=random()
    d=abs(a-b)
    l=l+d
    x=x+d*d
l=l=l/n
print("Moyenne des longueurs={}".format(l))
x=x=sqrt(x/n-l*l)
print("Ecart-type des longueurs={}".format(x))
```

```
Moyenne des longueurs=0.3317661445692329
Ecart-type des longueurs=0.2366908196478012
```

Voici, à droite, le programme Python qui permet la réalisation de cet objectif. Le résultat de l'exécution de ce programme est donnée en dessous (en bleu) : la moyenne des segments mesure 0,331766.. et l'écart-type vaut 0,23669.. mais, bien sûr, ces valeurs sont des estimations dont la précision est limitée par le nombre *n* d'essais dans l'échantillon.
Essayons avec une valeur cent fois supérieure : *n*=1 000 000. On pouvait aussi choisir *n*=100, ce qui était demandé était d'avoir un rapport de 100 entre les deux valeurs de *n* choisies. Les résultats sont dans le tableau.

n	Moyenne des longueurs	Écart-type
10 000	0,3318	0,2367
1 000 000	0,3333	0,2357

On s'approche de 1/3 mais l'écart-type reste le même (une légère tendance à diminuer tout de même). En prenant *n*=100 000 000, je trouve *m*=0,333352 et σ=0,23571 pour l'écart-type, comme quoi cela ne diminue pas avec la taille de l'échantillon. Pour *n*=100, je trouve *m*=0,3150 et σ=0,2518 environ, des valeurs déjà très proche de leur limite.

4) Tirages au sort

On veut simuler avec un algorithme, N tirages d'une pièce de monnaie équilibrée, en notant « 1 » l'apparition d'une « face » et « 0 » l'apparition d'un « pile ».
a) Écrivez un algorithme en pseudo-langage qui réalise ces tirages et détermine la valeur de la fréquence F d'apparition d'un « pile » pour N tirages.

1. I, N et R sont des entiers.
2. X=0,
   Lire N (nombre de tirages à effectuer),
3. Pour I allant de 1 à N {
   Tirer un nombre entier R au hasard entre 0 et 1.
   Si (R==0) {X=X+1}}
4. Afficher la fréquence finale de l'échantillon : 100X/N.

b) Pour N=10, est-il étonnant d'obtenir pour cette fréquence la valeur F=0,3 ?

L'intervalle de confiance au seuil de 95% est l'intervalle $[0{,}3-\frac{1}{\sqrt{10}};0{,}3+\frac{1}{\sqrt{10}}]$ où 0,3 est la fréquence expérimentale F d'apparition de « pile » pour n=10 tirages. Cet intervalle est approximativement égal à $[-0{,}016;0{,}616]$, ce qui signifie que l'on peut s'attendre à avoir une fréquence réelle comprise entre 0 (la valeur négative est impossible à atteindre) et 0,616 (61,6%). Ce résultat est donc compatible avec la réalité qui situe la fréquence réelle à 50%. On peut aussi partir de cette fréquence réelle de 50% et calculer l'intervalle de fluctuation où se situent 95% des fréquences expérimentales :
$[0{,}5-\frac{1}{\sqrt{10}};0{,}5+\frac{1}{\sqrt{10}}] = [0{,}184;0{,}816]$.
Une fréquence de 0,3 est donc tout-à-fait normale, puisqu'elle est contenue dans cet intervalle.

Même question si on procède à 100 tirages. On répondra à ces questions après avoir déterminé l'intervalle de confiance au seuil de 95% correspondant à l'expérience considérée.

Pour 100 tirages, l'intervalle de confiance au seuil de 95% est l'intervalle $[0{,}3-\frac{1}{\sqrt{100}};0{,}3+\frac{1}{\sqrt{100}}]$ où 0,3 est la fréquence expérimentale F d'apparition de « pile » pour n=100 tirages. Cet intervalle est approximativement égal à $[0{,}2;0{,}4]$, ce qui signifie que l'on peut s'attendre à avoir une fréquence réelle comprise entre 0 et 0,4 (40%). Ce résultat est incompatible avec la réalité d'une fréquence réelle égale à 50%. Ce résultat serait donc anormal.
On peut aussi partir de la fréquence réelle de 50% et calculer l'intervalle de fluctuation où se situent 95% des fréquences expérimentales : $[0{,}5-\frac{1}{\sqrt{100}};0{,}5+\frac{1}{\sqrt{100}}] = [0{,}4;0{,}6]$. Une fréquence de 0,3 est donc tout-à-fait anormale, puisqu'elle n'est pas contenue dans cet intervalle.

5) Décimales de pi

Si l'on ne dispose que des décimales du nombre π pour générer une suite de tirages aléatoires de notre pièce, on peut décider de noter 0 (pile) lorsque le chiffre est pair et 1 (face) lorsqu'il est impair. Voici les 200 premières décimales de π :

3.14159265358 97932384626 43383279502 88419716939 93751058209
74944592307 81640628620 89986280348 25342117067 98214808651
32823066470 93844609550 58223172535 94081284811 17450284102
70193852110 55596446229 48954930381 96...

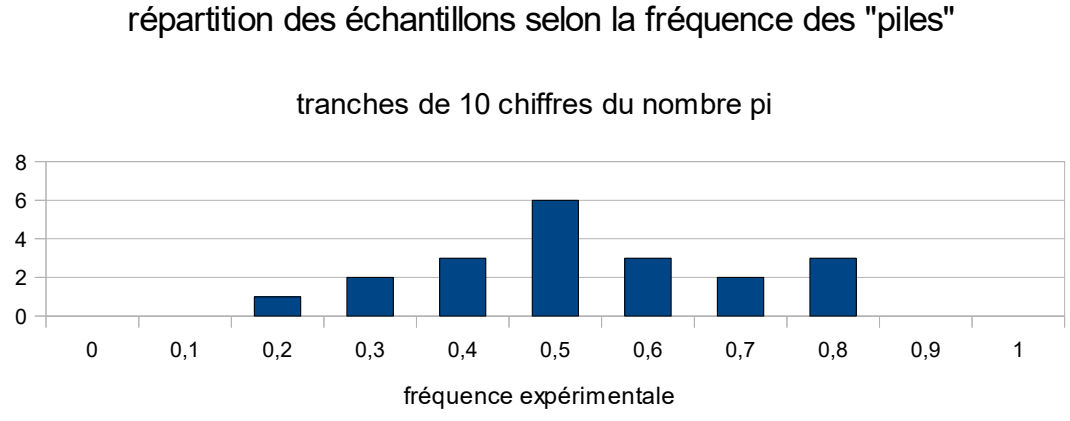
Les 10 premières décimales de la partie décimale (1415926535) contiennent ainsi 3 « piles » (les chiffres 4, 2 et 6). Complétez le tableau suivant qui donne la fréquence des piles dans les 20 premiers échantillons déterminés par cette série pseudo-aléatoire.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
N.Pile	3	4	5	5	2	6	6	8	8	4	7	7	6	5	5	5	5	3	8	4
Fréq.	0,3	0,4	0,5	0,5	0,2	0,6	0,6	0,8	0,8	0,4	0,7	0,7	0,6	0,5	0,5	0,5	0,5	0,3	0,8	0,4

Calculer la fréquence moyenne de ces 20 échantillons.
Compléter le tableau qui présente la répartition de ces 20 fréquences et commenter cette répartition.

	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Effectif	0	0	1	2	3	6	3	2	3	0	0
Fréq.	0	0	0,05	0,1	0,15	0,3	0,15	0,1	0,15	0	0

Ce qui paraît anormal dans cette répartition est la fréquence importante des échantillons (tranches de 10 chiffres du nombre pi) où il y a 8 « piles » dans l'échantillon. Ceci est une aberration qui doit sans doute disparaître lorsqu'on augmente le nombre de tranches examinées.



6) Approcher π par la méthode de Monte-Carlo

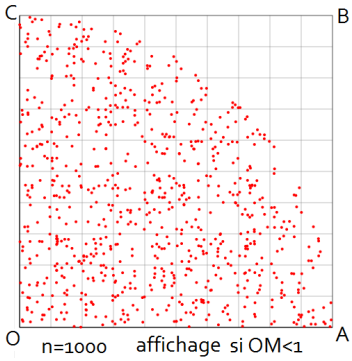
On place un grand nombre de points aléatoires dans un carré $OABC$ de côté 1. On note $(x;y)$ les coordonnées du point aléatoire M.

a) Montrer qu'en réalisant n tirages aléatoires et en déterminant la fréquence des points dans le quart de disque OAC, on peut obtenir une approximation du nombre π (montrer pour cela que la fréquence des points M tels que $OM<1$ est une estimation du nombre $\frac{\pi}{4}$; en déduire l'approximation cherchée).

b) Écrire un algorithme permettant de simuler ces n tirages et d'en déduire l'approximation de π.

c) Programmer cet algorithme et lancer ce programme pour n=100, 1 000 et 10 000. On donnera l'intervalle de confiance pour chacun de ces échantillons.

d) Déterminer le nombre de tirages minimum qu'il faut réaliser pour obtenir une estimation de π à 10^{-4} près au seuil de 5%. Simuler cet échantillon à l'aide de votre programme et conclure.

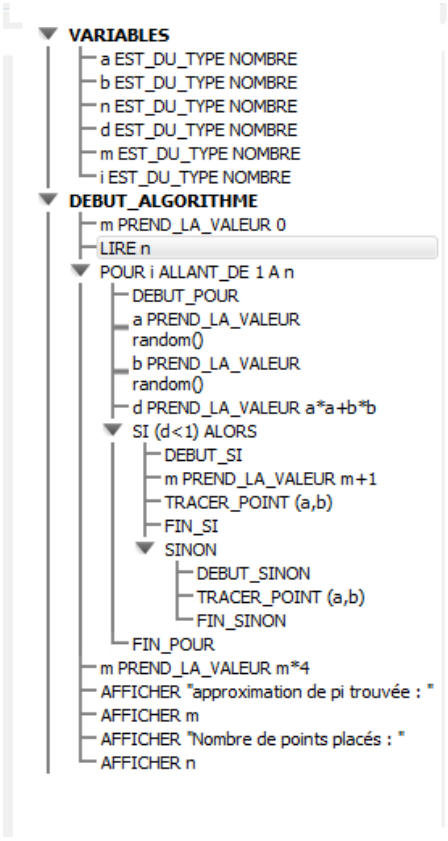


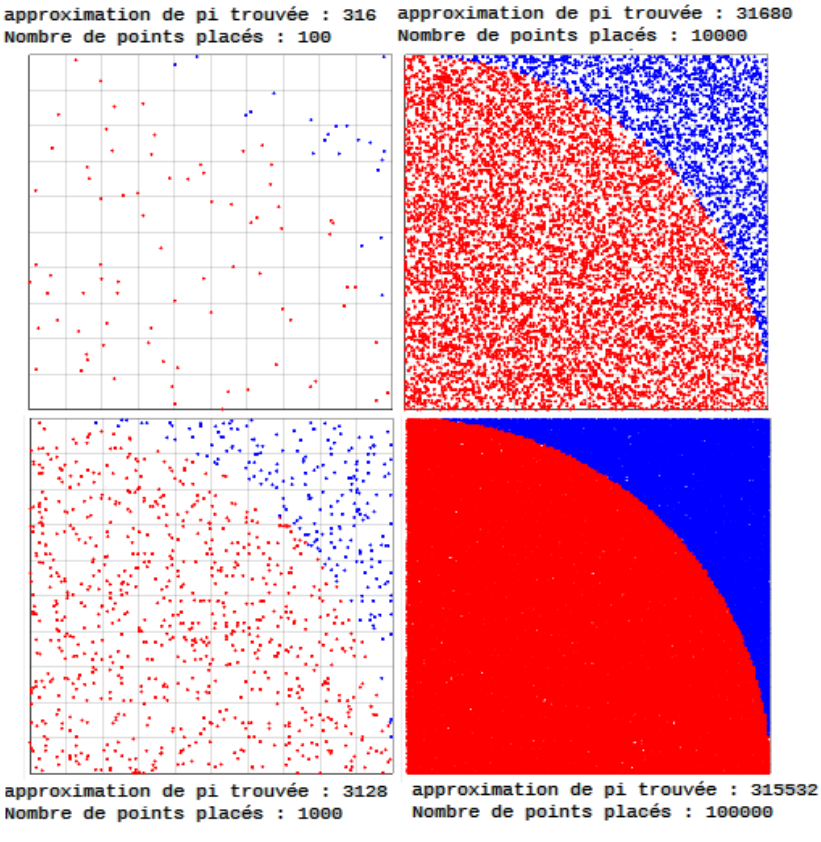
a) Dans cette expérience aléatoire, un point M de coordonnées $(x;y)$ tombe dans le quart de disque de centre O et de rayon 1 lorsque la distance $OM<1$. Or l'aire de ce quart de disque est $\frac{\pi\times1^2}{4}=\frac{\pi}{4}$. Les points tombent dans le carré de côté 1 et sont supposés se repartir de façon homogène dans ce carré. Or l'aire de ce carré est égale à $1^2=1$. Le rapport du nombre de points tombés dans le quart de disque relativement au nombre total de points tombés dans le carré est une estimation, de plus en plus précise au fur et à mesure que le nombre de points augmente, du rapport des aires de ces surface, donc de la fréquence des points M tels que $OM<1$. C'est donc une estimation du nombre $\frac{\frac{\pi}{4}}{1}=\frac{\pi}{4}\approx0{,}7854$.

b) On calcule OM à l'aide de la formule dérivée du théorème de Pythagore, $OM=\sqrt{x^2+y^2}$. Voici l'algorithme que nous allons programmer :

Lire N (le nombre de points que l'on va tirer), M=0.
Pour I allant de 1 à N : Tirer un nombre A entre 0 et 1 (1 non compris).
Tirer un nombre B entre 0 et 1 (1 non compris).
D=Racine carrée de A^2+B^2
Si (D<1) alors : M=M+1
M=4M/N
Afficher M

c) Voici la traduction sur Algobox de l'algorithme et quelques exécutions pour N=100, 1 000, 10 000 et 100 000 (sur cette version, la valeur de l'estimation donnée pour π n'a pas été divisée par N).





Nous avons effectué un tirage pour chaque valeur de n demandée. Comme on doit avoir $\frac{\pi}{4}\in[f-\frac{1}{\sqrt{n}};f+\frac{1}{\sqrt{n}}]$, on obtient en multipliant par 4 : $\pi\in[4f-\frac{4}{\sqrt{n}};4f+\frac{4}{\sqrt{n}}]$. Dans le tableau ci-dessous, les bornes de l'intervalle de confiance sont calculées avec cette méthode.

	Tirage	Borne 1	Borne 2
n=100	3,160000	2,760000	3,560000
n=1 000	3,128000	3,001509	3,254491
n=10 000	3,168000	3,128000	3,208000

Le nombre π figure bien dans ces trois intervalles de confiance.

d) On doit avoir $\frac{\pi}{4}\in[f-\frac{1}{\sqrt{n}};f+\frac{1}{\sqrt{n}}]$, soit $\pi\in[4f-\frac{4}{\sqrt{n}};4f+\frac{4}{\sqrt{n}}]$.
L'intervalle de confiance de $\frac{\pi}{4}$ a une amplitude égale à $f+\frac{1}{\sqrt{n}}-(f-\frac{1}{\sqrt{n}})=\frac{2}{\sqrt{n}}$, mais l'intervalle de confiance de π qui s'en déduit a une amplitude égale à $4f+\frac{4}{\sqrt{n}}-(4f-\frac{4}{\sqrt{n}})=\frac{8}{\sqrt{n}}$. Si on veut avoir une estimation de π à 10^{-4} près au seuil de 95%, il faut que l'on ait $\frac{4}{\sqrt{n}}\leq 10^{-4}$ ce qui revient à $\sqrt{n}\geq 4\times 10^{+4}$, soit $n\geq(4\times 10^{+4})^2$ ou encore $n\geq 1{,}6\times 10^{9}$ (près de 2 milliards!).

On va troquer la calculatrice contre l'ordinateur pour obtenir une estimation avec une telle valeur de n. Du même coup, j'améliore un peu le programme pour qu'il détermine automatiquement les bornes de l'intervalle de confiance de π qui s'en déduit (il suffit d'ajouter le calcul final des bornes 1 et 2).
Le petit programme ci-contre me donne le résultat suivant après un temps assez long qui en décourageraient plus d'un : un million de boucles toutes les secondes, cela fait tout de même 1600 secondes, soit près d'une demi-heure. J'attends patiemment que la machine achève ce travail titanesque, tout cela pour obtenir les 4 premières décimales de pi ! Cet algorithme décidément n'est pas très efficace. Voici ce que j'obtiens :

```
#estimation de pi par la méthode de Monte Carlo
from random import *
from math import *
n=1600000000
m=0
for I in range(n) :
  a= random()
  b= random()
  d=sqrt(a**2+b**2)
  if (d<1) : m=m+1
print( 4*m/n,4*m/n-4/sqrt(n),4*m/n+4/sqrt(n))
```

3.1415478075 3.1414478074999996 3.141647807

Même si l'approximation 3,1415478075 n'est pas fabuleuse (seuls 4 chiffres après la virgule sont corrects), il faut tout de même lui reconnaître le mérite d'être correcte, après ces 3,2 milliards de nombres aléatoires générés. L'intervalle de confiance obtenu, environ [3,14145 ; 3,14165], contient bien le nombre π et son amplitude est bien de 0,0001 comme on le souhaitait.

Ci-contre, en 4 instructions seulement, voici le même programme. Peut-on faire plus court ?

```
n,m=1600000000,0
for i in range(n) :
  if (sqrt(random()**2+random()**2)<1) : m=m+1
print( 4*m/n,4*m/n-4/sqrt(n),4*m/n+4/sqrt(n))
```

<u>Pour info</u> (extrait de Wikipédia) : Le terme méthode de Monte-Carlo, ou méthode Monte-Carlo, désigne une famille de méthodes algorithmiques visant à calculer une valeur numérique approchée en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de ces méthodes, qui fait allusion aux jeux de hasard pratiqués à Monte-Carlo, a été inventé en 1947 par Nicolas Metropolis, et publié pour la première fois en 1949 dans un article coécrit avec Stanislas Ulam.
Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier, pour calculer des surfaces et des volumes). Elles sont également couramment utilisées en physique des particules, où des simulations probabilistes permettent d'estimer la forme d'un signal ou la sensibilité d'un détecteur. La comparaison des données mesurées à ces simulations peut permettre de mettre en évidence des caractéristiques inattendues, par exemple de nouvelles particules.
La méthode de simulation de Monte-Carlo permet aussi d'introduire une approche statistique du risque dans une décision financière. Elle consiste à isoler un certain nombre de variables-clés du projet, tels que le chiffre d'affaires ou la marge, et à leur affecter une distribution de probabilité. Pour chacun de ces facteurs, un grand nombre de tirages aléatoires est effectué dans les distributions de probabilité déterminées précédemment, afin de trouver la probabilité d'occurrence de chacun des résultats.
Le véritable développement des méthodes de Monte-Carlo s'est effectué sous l'impulsion de John Von Neumann et Stanislas Ulam notamment, lors de la Seconde Guerre mondiale et des recherches sur la

fabrication de la bombe atomique. Notamment, ils ont utilisé ces méthodes probabilistes pour résoudre des équations aux dérivées partielles dans le cadre de la Monte-Carlo N-Particle transport (MCNP).

7) Tirages au sort

On veut simuler avec un algorithme, N tirages d'une pièce de monnaie équilibrée, en notant « 1 » l'apparition d'une « face » et « 0 » l'apparition d'un « pile ».

a) Écrivez un algorithme en pseudo-langage qui réalise ces tirages et détermine la valeur de la fréquence F d'apparition d'un « pile » pour N tirages.

1. I, N et R sont des entiers.
2. X=0,
   Lire N (nombre de tirages à effectuer),
3. Pour I allant de 1 à N {
   Tirer un nombre entier R au hasard entre 0 et 1.
   Si (R==0) {X=X+1}}
4. Afficher la fréquence finale de l'échantillon : 100X/N.

b) Pour N=10, est-il étonnant d'obtenir pour cette fréquence la valeur F=0,3 ?

L'intervalle de confiance au seuil de 95% est l'intervalle $[0{,}3-\frac{1}{\sqrt{10}};0{,}3+\frac{1}{\sqrt{10}}]$ où 0,3 est la fréquence expérimentale F d'apparition de « pile » pour n=10 tirages. Cet intervalle est approximativement égal à $[-0{,}016;0{,}616]$, ce qui signifie que l'on peut s'attendre à avoir une fréquence réelle comprise entre 0 (la valeur négative est impossible à atteindre) et 0,616 (61,6%). Ce résultat est donc compatible avec la réalité qui situe la fréquence réelle à 50%. On peut aussi partir de cette fréquence réelle de 50% et calculer l'intervalle de fluctuation où se situent 95% des fréquences expérimentales : $[0{,}5-\frac{1}{\sqrt{10}};0{,}5+\frac{1}{\sqrt{10}}] = [0{,}184;0{,}816]$. Une fréquence de 0,3 est donc tout-à-fait normale, puisqu'elle est contenue dans cet intervalle.

Même question si on procède à 100 tirages. On répondra à ces questions après avoir déterminé l'intervalle de confiance au seuil de 95% correspondant à l'expérience considérée.

Pour 100 tirages, l'intervalle de confiance au seuil de 95% est l'intervalle $[0{,}3-\frac{1}{\sqrt{100}};0{,}3+\frac{1}{\sqrt{100}}]$ où 0,3 est la fréquence expérimentale F d'apparition de « pile » pour n=100 tirages. Cet intervalle est approximativement égal à $[0{,}2;0{,}4]$, ce qui signifie que l'on peut s'attendre à avoir une fréquence réelle comprise entre 0 et 0,4 (40%). Ce résultat est incompatible avec la réalité d'une fréquence réelle égale à 50%. Ce résultat serait donc anormal.

On peut aussi partir de la fréquence réelle de 50% et calculer l'intervalle de fluctuation où se situent 95% des fréquences expérimentales : $[0{,}5-\frac{1}{\sqrt{100}};0{,}5+\frac{1}{\sqrt{100}}] = [0{,}4;0{,}6]$. Une fréquence de 0,3 est donc tout-à-fait anormale, puisqu'elle n'est pas contenue dans cet intervalle.