TD n°4 de probabilité : Algorithmes et probabilités

1. Le jeu du chapeau

Les 40 élèves d'une fameuse classe de seconde ont décidé de jouer au jeu du chapeau : on fabrique 39 étiquettes « perdu » et une étiquette « gagné », les étiquettes étant indiscernables, et on les met dans un chapeau. Chacun met 1 € sur la table pour constituer une cagnotte et ensuite, à tour de rôle, selon un ordre fixé d'avance et en payant 2 € pour jouer, chaque élève retire une des étiquettes du chapeau (et l'y remet après avoir lu), jusqu'à ce que l'un d'eux tire l'étiquette « gagné ». Il y a donc 40 € sur la table avant que le 1er joue et 42 € après que celui-ci ait perdu (s'il perd). On suppose qu'aucun élève ne passe son tour.

a) Quelle est la probabilité P(E=i) que l'élève qui passe en $i^{ème}$, retire l'étiquette « gagné » et gagne ainsi la somme posée sur la table ? On donnera P(E=1), P(E=2), P(E=3), et la formule générale P(E=i).
Quelle somme gagne t-on en moyenne à la place i (somme gagnée=cagnotte–mise) ?
À quelle place a t-on le plus de chances de gagner à ce jeu (s'il y en a une) ?
Quelle est la probabilité qu'aucun élève de la classe ne gagne la cagnotte ?

Cette règle ressemble à la roulette Russe, ce jeu macabre sinon stupide, dans son scénario où on ferait tourner le barillet d'un revolver à 40 coups contenant 1 balle à chaque fois que quelqu'un n'a pas trouvé le papier « gagné » (l'équivalent ludique de la balle). Comme on l'a vu, dans ce scénario n°2, les probabilités de gagner (1 balle dans la tête ou une somme d'argent pour cette version *soft*) diminuent lentement, le 1er étant celui qui a le plus de chances de gagner :
$P(E=1)=\frac{1}{40}\approx 0{,}025$, $P(E=2)=\frac{39}{40}\times\frac{1}{40}=\frac{39}{40^2}=0{,}024375$, $P(E=3)=\left(\frac{39}{40}\right)^2\times\frac{1}{40}=\frac{39^2}{40^3}\approx 0{,}023766$, etc.
La valeur générale demandée est $P(E=i)=\left(\frac{39}{40}\right)^{i-1}\times\frac{1}{40}=\frac{39^{i-1}}{40^i}$ pour les différentes valeurs de i rencontrées jusqu'à ce que l'un des élèves gagne (pour i allant de 1 à 40 pour cette question).

La probabilité qu'aucun élève de la classe ne gagne à la fin du tour est $\left(\frac{39}{40}\right)^{40}\approx 0{,}363232$, donc il y a un peu moins de deux chances sur cinq (2/5=40%) qu'un $2^{ème}$ tour soit nécessaire pour trouver un gagnant.

Le premier, s'il gagne, gagne 39 € (42 dans le chapeau dont 3 qui viennent de sa poche), le $2^{ème}$ gagne 41 €, etc. Le $i^{ème}$, s'il gagne, gagne 37+2i. Dressons un tableau avec les probabilités et les gains à chaque place.
En multipliant la probabilité par le gain, on trouve ce que la personne de rang i peut espérer gagner en moyenne en jouant de multiples fois à ce jeu (si elle gagne et si elle est toujours à la même place i) : ce nombre croit avec i. Cela n'aurait pas été le cas si, pour jouer, on devait avancer 1 € au lieu de 2. Cette propriété permet donc d'accroître l'intérêt du jeu.
Le premier, s'il gagne, gagne 39 € avec la probabilité $\frac{1}{40}$, donc il gagne en moyenne $\frac{39}{40}=0{,}975$ € ; le $2^{ème}$ gagne 41 € avec la probabilité $\frac{39}{40^2}$, donc il gagne en moyenne $\frac{41\times 39}{40^2}\approx 1$ € ; etc.

place au tour 1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
probabilité de gagner	0,0250	0,0244	0,0238	0,0232	0,0226	0,0220	0,0215	0,0209	0,0204	0,0199	0,0194	0,0189	0,0184	0,0180	0,0175	0,0171	0,0167	0,0163	0,0158	0,0155
somme gagnée	39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	73	75	77
probabilité de gain +	0,98	1	1,02	1,04	1,06	1,08	1,1	1,11	1,12	1,13	1,15	1,15	1,16	1,17	1,18	1,18	1,18	1,187	1,189	1,1899
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
	0,0151	0,0147	0,0143	0,0140	0,0136	0,0133	0,0129	0,0126	0,0123	0,0120	0,0117	0,0114	0,0111	0,0108	0,0106	0,0103	0,0100	0,0098	0,0096	0,0093
	79	81	83	85	87	89	91	93	95	97	99	101	103	105	107	109	111	113	115	117
	1,1903	1,1899	1,189	1,187	1,18	1,18	1,18	1,17	1,17	1,16	1,16	1,15	1,15	1,14	1,13	1,12	1,12	1,11	1,1	1,09

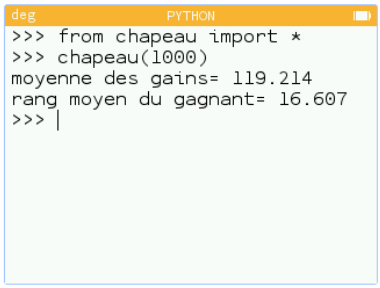
En calculant ces valeurs pour les différentes valeurs de i, on trouve (voir le tableau plus haut) que la $21^{ème}$ personne est celle qui a l'espoir maximum de gain (1,1903 €).

b) Un des élèves propose alors que si personne n'a gagné la cagnotte à la fin du 1er tour, on continue un $2^{ème}$ tour et ainsi de suite, jusqu'à ce que quelqu'un gagne. Écrire un algorithme qui simule cette version du jeu et calcule la somme moyenne gagnée à ce jeu (pour cela on simulera au moins 1 000 fois ce jeu pour estimer la valeur théorique par une valeur expérimentale assez précise).
Programmer cet algorithme sur votre calculatrice et donner le gain moyen pour 1 000 jeux.
Si on continue avec cette règle, sans rien changer : le 1er rejoue après le $40^{ème}$ si personne n'a gagné au 1er tour, et ainsi de suite. La question portant uniquement sur la somme moyenne gagnée, on n'a pas besoin du rang J du joueur dans l'algorithme. On le renseigne tout de même ici pour en déterminer la moyenne : on augmente J à chaque tirage et on le remet à 0 à la fin d'un tour. Le rang moyen du gagnant est trouvé en enregistrant la valeur de J du gagnant dans une variable notée G (rang du Gagnant). J'ai aussi déterminé le montant M des sommes engagées par le gagnant pour calculer son gain algébrique (la différence entre ce qu'il gagne et ce qu'il a engagé dans le jeu). Dans une première approche (celle qu'on a faite en cours), on peut se passer de

cette subtilité, la somme engagée étant négligeable comparée à la somme gagnée.
Réalisons donc l'algorithme suivant dans lequel on a directement assigné 10 000 au Nombre *N* de jeux. Pour tester votre programme, il vaut mieux mettre 100 à la place de 10 000. On peut également mettre le nombre *E* d'Élèves (ici *E*=40) dans une mémoire, pour le changer plus facilement si nécessaire.

```
N=10000          (N est le nombre de jeux joués, c'est la taille de notre échantillon)
T=0              (T indique la somme totale gagnée pour les N jeux)
G=0              (G va cumuler le rang des gagnants pour en déterminer la moyenne)
E=40             (E est le nombre d'élèves pour toute la partie)
Pour I allant de 1 à N
    R=0          (la valeur de R est fixée arbitrairement ici pour entrer dans la boucle)
    J=0          (J est le rang de la personne qui joue dans un tour)
    M=3          (M est le montant des sommes Misées par le joueur qui gagne)
    S=E          (S est la somme contenue dans la cagnotte au moment présent)
    tant que R≠1 (quand R=1 cela simule le fait que le joueur gagne)
        S=S+2
        R=nombre entier aléatoire entre 1 et E
        J=J+1
        si J>E alors
            J=1      (on arrive à la fin d'un tour, donc au début d'un nouveau tour)
            M=M+2    (on remet 2 € dans la cagnotte à chaque tour)
    fin du tant que
    T=T+S−M      (le gagnant a un gain algébrique égal à S-M)
    G=G+J        (le gagnant a un rang égal à J)
fin du pour
affichage de T/N
affichage de G/N
```

Programmons cet algorithme sur la calculatrice en langage Python (nous utilisons la Workshop de Numworks pour pouvoir copier l'écran) : Cela ne pose pas de difficultés particulières (sauf peut-être si on entre N=10000). On trouve un gain moyen espéré à ce jeu d'environ 119 €!



chapeau.py

créé par ph-moutou

créé le 7 mai 2018

273 octets

Envoyer sur ma calculatrice

Modifier Supprimer

Jeu du chapeau

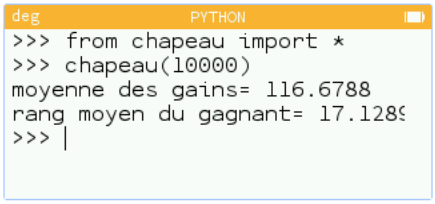
```
from random import randint
def chapeau(N):
  E,T,G=40,0,0
  for I in range(N):
    R,J,S=E,0,E
    while R>1:
      R=randint(1,E)
      S=S+2
      J+=1
    T+=S
    G+=((J-1)%E+1)
  print("moyenne des gains=",T/N)
  print("rang moyen du gagnant=",G/N)
```

Vous remarquerez que je n'ai pas implémenté la solution « si J>E alors J=1 » donnée dans l'algorithme ni la solution proposée en cours (où après la fin du « tant que », on trouvait le rang du joueur en faisant Rang=J%E et si Rang=0 alors Rang=40). J'ai choisi une solution à la fois plus économique et plus élégante : après la fin du « tant que », je trouve le rang du joueur en faisant Rang=(J-1)%E+1. Le problème de trouver 0 quand c'est le 40ème joueur (si J=40 alors J%40=0) est ainsi supprimé : si J=40 alors J-1=39 et (J-1)%40+1=39+1=40.

On peut aussi retrouver le numéro du tour en utilisant le quotient entier de J par E, plus précisément en faisant ent((J-1)/E)+1 (par exemple si J=41 alors ent((J-1)/40)+1=2 qui montre bien qu'on est dans le 2ème tour et, vous vérifierez également que si J=40 alors ent((J-1)/40)+1=1 qui montre bien qu'on est dans le 1er tour).

Vous trouvez que c'est bien compliqué d'enlever 1 pour ajouter 1 ensuite ? Vous commencez à comprendre pourquoi on note les rangs des objets à partir de 0... Dans une 2ème version du programme, j'ai donc calculé le gain algébrique du gagnant en enlevant, après la fin du « tant que », autant de fois 2 euros qu'il y a de tours dans J.

```
def chapeau(N):#tient compte des sommes M engagees
  E,T,G=40,0,0
  for I in range(N):
    R,J,S,M=E,0,E,1
    while R>1:
      R=randint(1,E)
      S=S+2
      J+=1
    T+=S-2*((J-1)//E+1)
    G+=((J-1)%E+1)
  print("moyenne des gains=",T/N)
  print("rang moyen du gagnant=",G/N)
```



Ceux qui sont devenu expert dans l'art d'utiliser les listes (ou tableaux) en programmation peuvent également obtenir des statistiques expérimentales sur la place du gagnant : il suffit d'ajouter une liste L dont les différentes valeurs L[J] comptabilisent les fois où c'est au rang J que l'on a gagné à ce jeu. Rappelez vous que le rang d'un élément dans une liste commence à 0... il faut donc utiliser L[J-1] pour enregistrer le nombre de fois où le gagnant avait le rang J. Ce n'est pas une grand difficulté d'ajouter cette liste. Pour déterminer le rang de la valeur qui a le plus grand effectif, on peut utiliser la fonction index qui donne le rang d'une valeur et max() qui identifie la valeur maximum, en écrivant L.index(max(L))+1 (l'ajout de 1 est pour restituer le véritable rang qui commence à 1 (et non à 0)). Si vous voulez en apprendre davantage sur les listes en Python Numworks, lisez la fiche sur ce sujet qui est dans le chapitre 0 (algorithmique) du cours de seconde : elle contient 10 exercices (les corrigés sont également sur cette page) qui vous permettront de maîtriser un peu mieux le sujet.

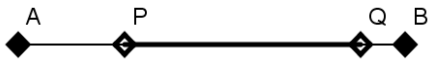
```
def chapeau(N):#determine le rang le plus frequent du gagnant
  E,T,G=40,0,0
  L=E*[0] #liste contenant l'effectif des gagnants au rang J
  for I in range(N):
    R,J,S,M=E,0,E,1
    while R>1:
      R=randint(1,E)
      S=S+2
      J+=1
    T+=S-2*((J-1)//E+1)#gain algebrique
    G+=((J-1)%E+1)
    L[(J-1)%E]+=1 #incremente l'effectif des gagnants du rang J
  print("moyenne des gains=",T/N)
  print("rang moyen du gagnant=",G/N)
  print("rang le plus frequent du gagnant=",L.index(max(L))+1)
  print(L)
```

```
deg                    PYTHON
>>> from chapeau import *
>>> chapeau(10000)
moyenne des gains= 116.6788
rang moyen du gagnant= 17.128
rang le plus frequent du gagnant= 1
[404, 387, 369, 305, 373, 357,
, 225, 242, 230, 194, 212, 218
5]
>>>
```

Conclusion : dans ce jeu, on gagne le plus souvent en étant classé 1er, mais la moyenne du rang des gagnant n'est pas 1, évidemment, puisque souvent on gagnera à des rangs plus avancés. La moyenne se déplace forcément vers les rangs supérieurs et atteint environ 17 (il y a une certaine asymétrie dans la distribution, l'étalement sur la droite est net).

2. Segment au hasard

a) On tire au hasard deux nombres x et y de l'intervalle [0 ; 1[. On considère qu'il s'agit des abscisses de deux points $P(x)$ et $Q(y)$ sur un segment $[AB]$ de milieu I avec $AB=1$.



On se demande quelle est la longueur moyenne du segment $[PQ]$. Pour répondre à cette question, réaliser un algorithme qui simule les tirages de x et y un nombre n de fois.

Ici, on va tirer deux nombres p et q de l'intervalle [0;1[et calculer la valeur absolue de leur différence (c'est le sens de « longueur du segment »).

L'algorithme est très simple ici :

Saisir N
Total=0 *(Total indique la somme totale gagnée pour les N jeux)*
Pour I allant de 1 à N
    P=nombre aléatoire de [0;1[
    Q=nombre aléatoire de [0;1[
    Total=Total+abs(P−Q)
fin du pour
affichage de Total/N

```
from random import random
N=int ( input ( "Combien d'essais? ") )
Total=0
for I in range ( N) :
   P=random ()
   Q=random ()
   Total+=abs ( P-Q)
print ( "Moyenne des longueurs de segment PQ= {}  ".format ( Total/N) )
```

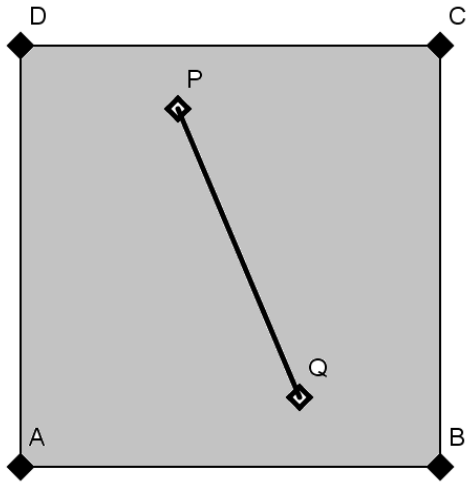
```
Combien d'essais? 1000000
Moyenne des longueurs de segment PQ= 0.3335313801553211

Combien d'essais? 10
Moyenne des longueurs de segment PQ= 0.3620555011118465
```

On obtient les résultats expérimentaux suivants (chacun trouvera des valeurs légèrement différentes, mais plus N augmente, plus les valeurs seront proches)

n	10	100	1000	10000	100000	1000000
Longueur moyenne	0,36205	0,37838	0,32884	0,33276	0,33304	0,33353

Voilà, ce n'était pas bien difficile. On trouve donc que la moyenne des longueurs de segment est environ égale à 1/3 (on trouve 0,3335 pour N=1 000 000). Peut-être en aviez-vous eu l'intuition, cela semble assez raisonnable en effet, comme valeur. Si vous avez ce genre d'intuition (le 6ème sens des probabilités...), vous trouverez sans doute la valeur suivante de façon intuitive ?

b) On tire au hasard deux fois deux nombres de [0 ; 1[et on considère que ce sont les coordonnées de deux points P et Q du carré $ABCD$ avec $A(0 ; 0)$, $B(1 ; 0)$, $C(1 ; 1)$ et $D(0 ; 1)$.
Quelle est la longueur moyenne du segment $[PQ]$?
Répondre à cette question en programmant un algorithme qui simule les tirages de x et y un nombre n de fois.



L'algorithme n'est pas très différent ici. Pour déterminer la longueur du segment, il faut utiliser la relation de Pythagore $PQ^2=(x_Q-x_P)^2+(y_Q-y_P)^2$. Afin de ne pas faire quatre tirages, mais seulement deux, on a opté pour une 2ème boucle. J'espère que cela ne vous perturbera pas trop (on aurait pu tirer quatre nombres aléatoire), la 1ère fois P et Q sont les abscisses de P et Q, la 2de fois ce sont les ordonnées.

Saisir N
Total=0 *(Total indique la somme totale gagnée pour les N jeux)*
Pour I allant de 1 à N
    Long=0
    Pour J allant de 1 à 2
        P=nombre aléatoire de [0;1[
        Q=nombre aléatoire de [0;1[
        Long=Long+(P-Q)²
        fin du pour
    Total=Total+racine(Long)
    fin du pour
affichage de Total/10000

```
from random import random
from math import sqrt
N=int ( input ( "Combien d'essais? ") )
Total=0
for I in range ( N) :
   Long=0
   for J in range ( 2) :
      P=random ()
      Q=random ()
      Long=Long+ ( P-Q)  **2
   Total=Total+sqrt ( Long)
print ( "Moyenne des longueurs de segment PQ=  {}  ".format ( Total/N) )
```

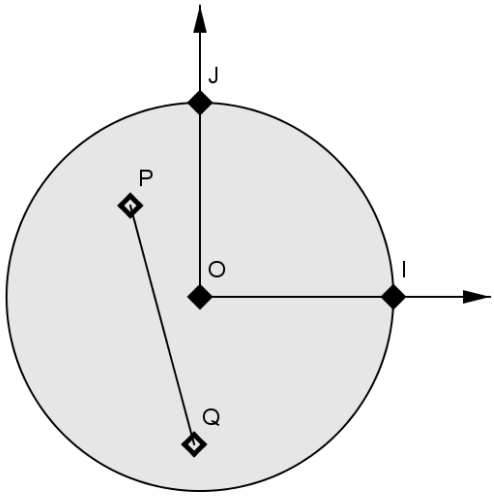
```
Combien d'essais? 1000000
Moyenne des longueurs de segment PQ= 0.521134157510314
```

Passons à la programmation et aux résultats.

n	10	100	1000	10000	100000	1000000
Longueur moyenne	0,49835	0,52499	0,51266	0,51991	0,52019	0,52113

Alors ? Ce résultat est-il conforme à votre intuition première ? Ce n'est pas 1/2=0,5 que l'on trouve : j'ai refait l'exécution 4 fois avec 1 000 000 essais à chaque fois, on trouve 0,521...
Étonnant, n'est-ce pas ? Cela correspond à quel nombre selon vous ?
Il s'agit du nombre transcendant, pas intuitif du tout, égal à $\frac{2+\sqrt{2}}{15}+\frac{\ln(1+\sqrt{2})}{3}\approx 0{,}5214054332$.

c) On tire au hasard les coordonnées de deux points P et Q du disque de rayon 1 centré sur l'origine.
Pour réaliser ces tirages, on tire deux nombres, x et y, dans $[-1 ; 1[$ jusqu'à ce que la distance entre le point de coordonnées $(x ; y)$ et l'origine soit inférieure ou égale à 1. Cette distance est égale, on le rappelle à $\sqrt{x^2+y^2}$.
Déterminer la moyenne pour n tirages aléatoires.
On se demande quelle longueur moyenne aurait un segment dont les extrémités sont choisies à l'intérieur d'un disque (et non d'un carré). Pour simuler le tirage d'un point P dans un tel disque de rayon 1, il suffit d'ajouter une petite boucle « Tant que » qui s'assure que les deux points tirés ne s'écartent pas du centre de plus d'une unité (le rayon) :



XP=1
YP=1
Tant que $XP^2+YP^2>1$
    XP=nombre aléatoire de [0;1[×2−1
    YP=nombre aléatoire de [0;1[×2−1

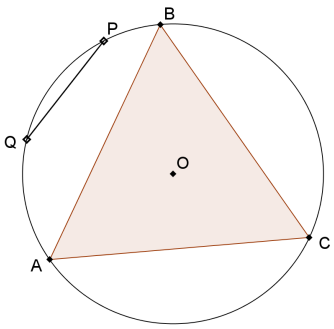
Inutile d'utiliser la racine carrée car si la somme XP^2+YP^2 est inférieure à 1, sa racine carrée aussi. Nous avons évité la boucle « pour J allant de 1 à 2 » ici afin de clarifier ce qu'on calcule : nous avons appelé xP et yP les coordonnées de P, de même pour Q, cela est donc plus facile à comprendre. Il y a cependant deux fois deux boucles « While » pour s'assurer du tirage des points à l'intérieur du disque.

```
from random import random
from math import sqrt
N=int ( input ( "Combien d'essais? ") )
Total=0
for I in range ( N) :
    Long=0
    xP,yP=1,1
    while xP**2+yP**2>1:
        xP=random () *2-1
        yP=random () *2-1
    xQ,yQ=1,1
    while xQ**2+yQ**2>1:
        xQ=random () *2-1
        yQ=random () *2-1
    Long= ( yP-yQ) **2+ ( xP-xQ) **2
    Total=Total+sqrt ( Long)
print ( "Moyenne des longueurs de segment PQ= {} ".format ( Total/N) )
```

```
Combien d'essais? 1000000
Moyenne des longueurs de segment PQ= 0.9055086735822079
```

n	10	100	1000	10000	100000	1000000
Longueur moyenne	0,86028	0,85551	0,89807	0,90319	0,90571	0,90551

Une question d'intuition : va t-on trouver plus ou moins de 0,5214... Vous pensiez plus et vous aviez bien raison car le disque est bien plus grand que le carré. Le carré a pour aire 1 alors que le disque a pour aire π. Le segment sera forcément plus long, du moins en moyenne. Mais la vraie valeur est difficile à deviner. D'une façon algorithmique, je trouve 0,90551... (moyenne obtenue pour 1 000 000 de segments). Un essai pour N=10 000 000 donne 0,905479... On peut donc raisonnablement penser qu'il s'agit d'environ 90547.
La valeur exacte est plus simple mais toute aussi transcendante.
Il s'agit de $\frac{128}{45\pi}\approx 0{,}9054147874$.



Pour prolonger cette passionnante activité, on peut s'intéresser au paradoxe de Bertrand. Le sujet est développé dans le cours (pp.10-11) : « *un triangle équilatéral ABC inscrit dans un cercle. On choisit P et Q, deux points au hasard sur le cercle et on s'interroge sur la probabilité que la corde [PQ] soit plus courte que le côté [AB] du triangle* ». On peut, au passage déterminer la longueur moyenne de la corde [*PQ*] du cercle trigonométrique, lorsqu'on choisit au hasard les points *P* et *Q* sur le cercle.
Selon la méthode employée pour tirer « au hasard » la corde, on trouve des probabilités différentes. Qu'en est-il d'une méthode algorithmique ?
Première remarque : l'emplacement du triangle *ABC* n'a pas d'importance, seul le côté importe. Comme *AO*=1 est égal au 2/3 de la médiane, celle-ci mesure 1,5. Or la médiane d'un triangle équilatéral est en même temps sa hauteur *h* qui vaut, on l'a montré souvent $h=\frac{\sqrt{3}c}{2}$. Le côté *c* du triangle vaut donc $c=\frac{2h}{\sqrt{3}}=\frac{2\times 1{,}5}{\sqrt{3}}=\sqrt{3}$.
Deuxième remarque : les points $M(x\,;y)$ d'un cercle de centre $O(0\,;0)$ et de rayon 1, vérifient l'égalité $x^2+y^2=1$ qui vient du théorème de Pythagore. Aussi, si on a déterminé *x* au hasard, on a $y^2=1-x^2$ et donc, une fois sur deux, de façon aléatoire, on aura $y=\sqrt{1-x^2}$ et une fois sur deux $y=-\sqrt{1-x^2}$.
Il ne reste plus qu'à écrire le programme. Celui nous donne une valeur d'environ 0,37 pour la probabilité...

```
from math import sqrt
from random import *
def bertrand(N):
  Total,Proba,Cote=0,0,sqrt(3)
  for i in range(N):
    Long,coord=0,[[0,0],[0,0]]
    for j in range(2):
      coord[j][0]=2*random()-1#abscisse dans [-1,1[
      coord[j][1]=(-1)**randint(0,1)*sqrt(1-coord[j][0]**2)#ordonnees
    Long=sqrt((coord[0][0]-coord[1][0])**2+(coord[0][1]-coord[1][1])**2)
    if Long>Cote:
      Proba+=1
    Total+=Long
  print("longueur moyenne=",Total/N)
  print("proba(long>cote)=",Proba/N)
  print("cote du triangle=",Cote)
```

```
deg                PYTHON
>>> from bertrand import *
>>> bertrand(1000)
longueur moyenne= 1.272200558(
proba(long>cote)= 0.386
cote du triangle= 1.732050807!
>>> bertrand(10000)
longueur moyenne= 1.261127716!
proba(long>cote)= 0.3685
cote du triangle= 1.732050807!
>>>
```

Le paradoxe est que la probabilité dépend de la façon dont on réalise le tirage des points qui sont les extrémités de la corde. Dans le cours, on a rappelé les trois valeurs données par Bertrand (0,25, 0,33 et 0,5). Ici on trouve donc 0,37 environ qui ne correspond à aucune de ces valeurs. Essayons une 5ème méthode : les coordonnées du point $M(x\,;y)$ sont déterminées en tirant au hasard, la valeur de l'angle $\alpha=\widehat{IOM}$ et on a alors $M(\cos\alpha\,;\sin\alpha)$.
Le résultat est différent de la méthode précédente : on trouve ici un nombre très proche de un tiers (0,333) alors qu'avec la méthode précédente on trouve 0,362. J'ai effectué pour ces deux fréquences expérimentales, le tirage de 100 000 cordes. D'après ce qu'on a vu au chapitre échantillonnage, le probabilité est comprise dans les intervalles de confiance $[\,0{,}362-\frac{1}{\sqrt{100000}}\,;\,0{,}362+\frac{1}{\sqrt{100000}}\,]$, soit [0,3588;0,3652] dans 95% des cas, selon la méthode 1 et $[\,0{,}333-\frac{1}{\sqrt{100000}}\,;\,0{,}333+\frac{1}{\sqrt{100000}}\,]$, soit [0,3298;0,3362] dans 95% des cas, selon la méthode 2. Les intervalles de confiance étant disjoints, les probabilités trouvées sont incompatibles.

```
from math import *
from random import *
def bertrand1(N):
  Total,Proba,Cote=0,0,sqrt(3)
  for i in range(N):
    Long,coord=0,[[0,0],[0,0]]
    for j in range(2):
      coord[j][0]=2*random()-1#abscisse dans [-1,1[
      coord[j][1]=(-1)**randint(0,1)*sqrt(1-coord[j][0]**2)#ordonnees
    Long=sqrt((coord[0][0]-coord[1][0])**2+(coord[0][1]-coord[1][1])**2)
    if Long>Cote:
      Proba+=1
    Total+=Long
  print("longueur moyenne=",Total/N)
  print("proba(long>cote)=",Proba/N)
  print("cote du triangle=",Cote)

def bertrand2(N):
  Total,Proba,Cote=0,0,sqrt(3)
  for i in range(N):
    Long,coord=0,[[0,0],[0,0]]
    for j in range(2):
      alpha=2*pi*random()
      coord[j][0]=cos(alpha)#abscisse
      coord[j][1]=sin(alpha)#ordonnees
    Long=sqrt((coord[0][0]-coord[1][0])**2+(coord[0][1]-coord[1][1])**2)
    if Long>Cote:
      Proba+=1
    Total+=Long
  print("longueur moyenne=",Total/N)
  print("proba(long>cote)=",Proba/N)
  print("cote du triangle=",Cote)
```

```
deg                    PYTHON
>>> from bertrand import *
>>> bertrand2(100000)
longueur moyenne= 1.273242933
proba(long>cote)= 0.33338
cote du triangle= 1.732050807
>>> bertrand1(100000)
longueur moyenne= 1.252878529
proba(long>cote)= 0.36162
cote du triangle= 1.732050807
>>>
```