Situation 1

Une variable aléatoire G prend ses valeurs dans l'ensemble {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. La probabilité de chaque issue est proportionnelle à la valeur de cette issue.

Déterminer la loi de probabilité de la variable aléatoire G. Déterminer « à la main » les paramètres de la variable aléatoire : espérance, variance et écart-type.

On propose ci-contre deux fonctions « esperance(x,p) » et « dispersion(x,p) » permettant de calculer les paramètres de la variable aléatoire G. Compléter ce script…

```
from math import *

def esperance(x,p):
  e=0
  for k in range(...):
    e=...
  return(e)

def dispersion(x,p):
  s=0
  e=esperance(x,p)
  for k in range(...):
    s=...
  return(s,sqrt(s))

x=[i for i in range(...)]
p=[i/55 for i in range(...)]

print(esperance(x,p))
print(dispersion(x,p))
```

Situation 2

Un robot se déplace sur une droite graduée de façon aléatoire et équiprobable de la manière suivante : à chaque seconde, soit il avance d'une graduation, soit il recule d'une graduation, soit il reste à sa place. Un trajet du robot est une succession de 5 déplacements, le robot étant initialement situé au point d'abscisse 10. On note X la variable aléatoire égale à l'abscisse du robot après un trajet.

1. Ecrire une fonction « deplacement(a) » qui simule un déplacement du robot à partir de l'abscisse a.

2. Ecrire une fonction « trajet(a) » qui simule un trajet du robot et qui renvoie la valeur de X.

3. Ecrire une fonction « echantillon(n) » qui renvoie un échantillon simulé de taille de n de la variable aléatoire X.

4. Simuler un échantillon de 10 trajets

5. Ajouter une fonction « etendue(n) » qui renvoie la valeur minimale et maximale dans un échantillon de taille n.

```
from random import *

def deplacement(a):
  h=randint(...)
  if h==...:
    a=...
  if h==...:
    a=...
  return(a)

def trajet(a):
  for i in range(...):
    a=...
  return(a)

def echantillon(n):
  l=[]
  for k in range(...):
    l.append(...)
  return(...)

print(echantillon(10))
```

6. Taper plusieurs fois « etendue(10) », puis plusieurs fois « etendue(1000) ». Commenter…

Situation 3

Le principe du jeu est le suivant : on mise 2 euros et on lance une pièce équilibrée au plus 4 fois. En cas d'apparition d'un « Pile », le gain est égal au rang de sortie de ce « Pile » (moins la mise initiale de 2 euros qui est automatiquement perdue). Si aucun « Pile » ne sort au cours des quatre lancers, on ne gagne rien et on perd la mise initiale de 2 euros.

On appelle G la variable aléatoire égale au gain algébrique du joueur, c'est-à-dire à la différence entre le gain obtenu à l'issue des lancers et la mise initiale de 2 euros automatiquement perdue.

1. Déterminer la loi de probabilité de la variable aléatoire G.

2. Déterminer « à la main » l'espérance de la variable aléatoire G.

3. Compléter le script de la fonction « jeu() » proposée ci-contre permettant de simuler une partie de ce jeu et de renvoyer le gain algébrique du joueur à l'issue de cette partie.

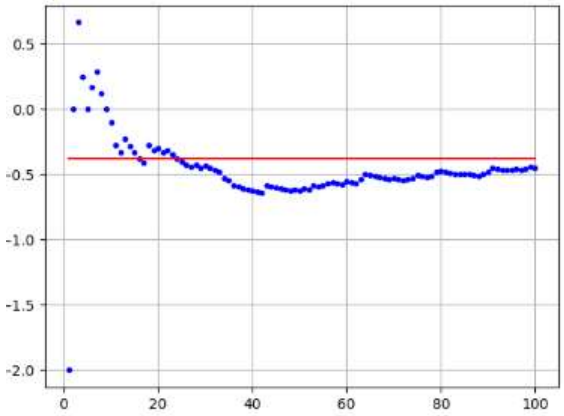
4. Compléter le script de la fonction « simulation(n) » proposée ci-contre permettant de simuler n parties de ce jeu et de renvoyer le gain algébrique moyen du joueur par partie à l'issue de ces n parties simulées.

```
1  from random import *
2
3  def jeu():
4    g=-2
5    for i in range(...):
6      if randint(0,1)==0 and g==-2:
7        g=...
8    return(g)
9
10 def simulation(n):
11   g=0
12   for k in range(...):
13     g=...
14   return(...)
15
16 print(simulation(1000000))
```

5. Utiliser la fonction « simulation(n) » pour n=1000000. Commenter le résultat obtenu…

6. On propose ci-dessous une fonction « evolmoy(nbexp) » ainsi qu'une capture d'écran de ce qu'on obtient pour nbexp=100. Programmer cette fonction puis observer et commenter les résultats obtenus afin d'être en mesure de décrire le rôle de cette fonction.

```
11 import matplotlib.pyplot as plt
12
13 def evolmoy(nbexp):
14   s=0
15   n=1
16   l=[]
17   while n<=nbexp:
18     s=s+jeu()
19     l.append(s/n)
20     n=n+1
21   plt.plot(list(range(1,nbexp+1)),l,'b.')
22   plt.plot([1,nbexp],[-3/8,-3/8],'r-')
23   plt.grid()
24   plt.show()
25
26 evolmoy(100)
```



Capture d'écran pour evolmoy(100)

Situation 4

Le jeu américain « Chuck a luck » consiste à parier sur un nombre de 1 à 6 puis à lancer trois dés équilibrés. Si le nombre sur lequel on a parié sort trois fois, on gagne 3 euros. S'il sort deux fois, on gagne 2 euros. S'il sort une fois, on gagne 1 euro. S'il ne sort pas on perd 1 euro. On choisit de parier sur le numéro « 3 ». On appelle G la variable aléatoire égale au gain algébrique.

1. Déterminer la loi de probabilité de la variable aléatoire G. Calculer son espérance.

2. Simuler à l'aide d'un tableur n parties de ce jeu et calculer le gain algébrique moyen par partie à l'issue de ces n parties simulées. Lancer la simulation pour diverses valeurs de n…

Situation 5

En France il naît environ 105 garçons pour 100 filles. On s'intéresse à des familles comportant deux enfants issus de grossesses non gémellaires. Pour une telle famille, on note F la variable aléatoire égale au nombre de filles nées dans cette famille.

1. Déterminer la loi de probabilité de la variable aléatoire F.
   Démontrer que son espérance $\mu = \frac{40}{41}$ et que son écart type $\sigma = \sqrt{\frac{840}{1681}}$.

On souhaite simuler « N » échantillons de « n » familles et étudier la proportion de cas où l'écart entre la moyenne de l'échantillon et la moyenne théorique μ est inférieure ou égale à $\frac{2\sigma}{\sqrt{n}}$.

2. Créer une fonction « famille() » qui simule la naissance de deux enfants issues de grossesses non gémellaires et renvoie le nombre de filles nées.

3. Compléter la fonction « simulation(n) » afin qu'elle renvoie le nombre moyen de filles sur n familles simulées.

4. Expliquer le rôle de la fonction « ecart(moyenne,n,mu,sigma) »

5. Compléter la fonction « proportion(n,N) » afin qu'elle renvoie pour N échantillons de n familles simulées, la proportion des cas où l'écart entre la moyenne de l'échantillon et la valeur théorique μ est inférieure ou égale à $2\sigma/\sqrt{n}$

6. Exécuter la fonction « proportion(n,N) » pour n=1000 et N=100. Interpréter et commenter…

```
1  from random import *
2  from math import *
3
4  def famille():
5    f=0
6    if ...:
7      f=...
8    if ...:
9      f=...
10   return(f)
11
12 def simulation(n):
13   filles=0
14   for k in range(n):
15     filles=...
16   moyenne=...
17   return(moyenne)
18
19 def ecart(moyenne,n,mu,sigma):
20   if abs(moyenne-mu)<=2*sigma/sqrt(n):
21     return(True)
22   else:
23     return(False)
24
25 def proportion(n,N):
26   s=0
27   for j in range(...):
28     s=s+ecart(simulation(n),n,...,...)
29   p=s/N
30   return(p)
```

Situation 6

Le bandit manchot est un jeu de machine à sous. Lorsqu'on tire le levier, trois rouleaux tournent et s'arrêtent sur un chiffre compris entre 1 et 9 pour le rouleau le plus à gauche, entre 0 et 9 pour les deux rouleaux suivants. Lorsque le joueur, qui a misé 1 euro pour jouer, obtient trois fois le même chiffre, il gagne 70 euros. On appelle G le gain algébrique du joueur.

1. Déterminer la loi de probabilité de G, son espérance μ et son écart type σ.

2. A l'aide d'un tableur, simuler 100 échantillons de 500 parties et étudier la proportion des cas où l'écart entre la moyenne d'un échantillon et μ est inférieure à $2\sigma/\sqrt{500}$.