TP 6 : Programmer l'aléatoire - Corrigé

2 BCPST 2 - TP Informatique

```
import random as rd
import matplotlib.pyplot as plt

#Q1
def bernoulli(p):
    if rd.random() < p :
        return 1
    return 0

#Q2
def binomiale(n,p): #Version r cursive
    if n == 0 :
        return 0
    return binomiale(n-1,p)+bernoulli(p)


#Q3
def geometrique(p): #Version r cursive
    if bernoulli(p) == 1 :
        return 1
    return 1+geometrique(p)

#Q4
def uniforme1(n):
    x = rd.random()
    for i in range(1,n+1):
        if x <= i/n:
            return i

#Version recursive
def uniforme1(n):
    if bernoulli(1/n) :
        return n
    return uniforme(n-1)

#Q5
def uniforme2(a,b):
    return uniforme1(b-a+1)+a-1
```

```
#Q6
import numpy as np

def poisson(lda):

    x = rd.random()

    p = np.exp(-lda);
    s = p; k = 0;

    while(x > s):
        k += 1
        p *= lda/k
        s += p

    return k

#Q7
N = 1000
E = 0;
for i in range(N):
    E += poisson(5)

E = E/N
print("esperance estim e :", E)

#Q8
for i in range(N):
    V += (poisson(5)-E)**2

V = V/N
print("variance estim e :", V)

#%%
#Q9
L = []
for i in range(N):
    L.append(poisson(5))

plt.hist(L,bins = 15, range=(0,14))
plt.show()
#%%
#Q10
def simulation1():
    L = []; compteur = 0

    while(len(L)<980):

        compteur += 1

        for _ in range(7):
            x = rd.randint(1,980)
            if x not in L :
                L.append(x)
    return compteur

print(simulation1())

#%%
```

```
#Q11

N = 100
E = 0;
for i in range(N):
    E += simulation1()

E = E/N
print("esperance estim e :", E)
#On trouve environs 1000 pochettes achet es soit 1200 euros environs.

#%%
#Q12
def simulation2():
    L1 = []; L2 = []; compteur = 0

    while(len(L1)<980 and len(L2)<980):

        compteur += 2

        for _ in range(7):
            x1 = rd.randint(1,980)
            if x1 not in L1 :
                L1.append(x1)
            elif x1 not in L2 :
                L2.append(x1)

            x2 = rd.randint(1,980)
            if x2 not in L2 :
                L2.append(x2)
            elif x1 not in L1 :
                L1.append(x2)


    return compteur

print(simulation2())

#Q11

N = 100
E = 0;
for i in range(N):
    E += simulation1()

E = E/N
print("esperance estim e :", E)
#On trouve l   encore une esp rance autour de 1000. Ce qui veut dire que
   les deux fans
#vont d penser environs deux fois moins d'argent.
```

```
#%%
#Exercice 2
# Q14

def f(x) :
    return x**2

def Monte_Carlo(f,n):

    compte = 0
    for i in range(n):
        x = rd.random()
        y = rd.random()

        if(f(x) >  y):
            compte += 1
    return compte/n

#-Q15
def Graphique_Monte_Carlo(f,n):

    compte = 0

    X1 = []; Y1 = []
    X2 = []; Y2 = []
    for i in range(n):
        x = rd.random()
        y = rd.random()

        if(f(x) >  y):
            compte += 1
            X1.append(x); Y1.append(y)
        else :
            X2.append(x); Y2.append(y)

    plt.plot(X1,Y1, "xr")
    plt.plot(X2,Y2, "xb")

    X = np.linspace(0,1,100)
    Y = f(X)

    plt.plot(X,Y, color = "orange")

    plt.plot([0,0],[0,1], "k", linewidth=0.5)
    plt.plot([0,1],[0,0], "k", linewidth=0.5)

    plt.axis("equal")

    plt.show()
    return compte/n


print(Monte_Carlo(f,100))
#%%
```

```
# Q16

def Monte_Carlo2(f,n,a,b):

    compte = 0
    for i in range(n):
        x = a+rd.random()*(b-a)
        y = rd.random()

        if(f(x) >  y):
            compte += 1

    return compte/n*(b-a)


def f(x) :
    return 1/x

print("valeur estim e :", Monte_Carlo2(f,1000,1,5))
print("valeur exacte :",np.log(5))


#%%
# Q17

def Monte_Carlo3(f,n,a,b,m,M):

    compte = 0
    for i in range(n):
        x = a+rd.random()*(b-a)
        y = m+rd.random()*(M-m)

        if(y > 0):
            if(f(x) >  y):
                compte += 1
        else :
            if(f(x) <  y):
                compte -= 1


    return compte/n*(b-a)*(M-m)


def f(x) :
    return x**2

n = 10000
print("int grale de x**2 entre 0 et 2:")
print("valeur estim e :", Monte_Carlo3(f,n,0,2,0,4))
print("valeur exacte :",8/3)


print("int grale de sin entre -pi et pi:")
print("valeur estim e :", Monte_Carlo3(np.sin,n,-np.pi,np.pi,-1,1))
print("valeur exacte :",0)
```

```
#%%
#Q18
def simul():
    D1  =rd.randint(1,6)
    D2 = rd.randint(1,6)
    return D1+D2, D1*D2

#Q19
n = 100; X,Y = [],[]

for _ in range(n):
    x,y = simul()
    X.append(x)
    Y.append(y)

plt.plot(X,Y,"xr")
plt.show()

#Q20-Q21
EX = 0; EY = 0
Cov = 0; VX = 0; VY = 0
N = 1000

for _ in range(n):
    x,y = simul()
    EX += x ; EY += y

EX /= N; EY /= N

for _ in range(n):
    x,y = simul()

    Cov += (x-EX)*(y-EY)
    VX += (x-EX)**2
    VY += (y-EY)**2

Cov /= N; VX /= N; VY /= N;

Corr = Cov/np.sqrt(VX*VY)

print("covariance estim e : ", Cov)
print("Corr lation estim e : ", Corr)
```