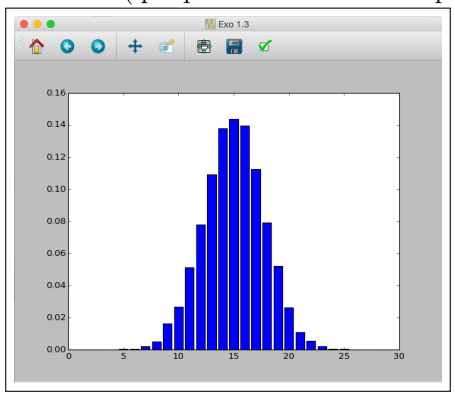
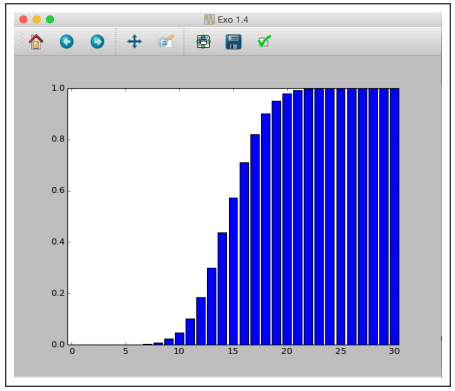
```
# Importation des modules nécessaire
import numpy as np
import matplotlib.pyplot as plt
from random import random


# On utilisera la fonction simulBernoulli du TD 2 :
def simulBernoulli(p):
    rnd = random()
    if rnd <= p:
        return 1
    else:
        return 0



# Exercice 1
# (1.1)
def simulSommeBernoulli(p,n):
    y = 0
    for k in range(n):
        y += simulBernoulli(p)
    return y


# (1.2)
def freqSommeBernoulli(p,n,N):
    R = [0] * (n+1)
    for i in range(N):
        y = simulSommeBernoulli(p,n)
        R[y]+= 1
    return [r/N for r in R]

# (1.3)
N = 10000
n = 30
p = 0.5
Y = freqSommeBernoulli(p,n,N)
X = [x - 0.4 for x in range(n+1)]
plt.figure("Exo 1.3")
plt.clf()
plt.bar(X,Y)
```

On obtient le graphique suivant (qui peut varier d'une implémentation à l'autre) :




```
# (1.4)
Z = [0] * (n+1)
Z[0] = Y[0]
for k in range(1,n+1):
    Z[k] = Z[k-1] + Y[k]
plt.figure("Exo 1.4")
plt.clf()
plt.axis([-0.4,30.4,0,1])
plt.bar(X,Z)
```

Ce qui produit :



```
# EXERCICE 2
from random import shuffle
from random import randint

# Parametres
N = 100
n = 10
p = 0.45
Np = int(N*p)
Nq = N - Np

L = [1] * Np + [0] * Nq
shuffle(L)

def tir():
    T = []
    for i in range(n):
        k = randint(0,N-1)
        T.append(L[k])
    return T

def frequence(m = 100000):
    Issues = [0] * (n+1)
    for i in range(m):
        T = tir()
        k = sum(T)
        Issues[k] += 1
    Freq = [ k/m for k in Issues ]
    return Freq

Y = frequence()
plt.figure(2)
plt.clf()
X = [ x-0.4 for x in range(n+1) ]
plt.bar(X,Y)
plt.show()
```

On obtient :