```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  1 01:26:25 2021

@author: 11327
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy.matlib

# Exercise 2
# b
'''
coin_num = 1000
flip_times = 10

run = 100000
V1 = np.zeros(run)
Vrand = np.zeros(run)
Vmin = np.zeros(run)
for i in range(run):
    COIN = np.random.randint(0,2,(flip_times,coin_num))
    # count the number of heads of each coin
    head = np.zeros(coin_num)
    for kk in range(flip_times):
        for ii in range(coin_num):
            if COIN[kk,ii] == 1:
                head[ii] = head[ii] + 1
    # calculating V1
    V1[i] = head[0] / flip_times
    # calculating Vrand
    rand = np.random.randint(0,1000)
    Vrand[i] = head[rand] / flip_times
    # calculating Vmin
    Vmin[i] = min(head) / flip_times

# plot histogram
plt.figure()
plt.hist(V1,bins=11)

plt.figure()
plt.hist(Vrand,bins=11)

plt.figure()
plt.hist(Vmin,bins=11)
'''
'''
# c
N = 10
epsilon = np.linspace(0,0.5,11)
Hb = 2 * np.exp(-2 * epsilon**2 * N) # Hoeffding's bound
# plot Hoeffding's bound
plt.figure()
l1, = plt.plot(epsilon,Hb)
# plot P(|V1-miu1| > epsilon)
miu = 0.5
P1 = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
```

```
    for i in range(run):
        if np.abs(V1[i]-miu) > epsilon[j]:
            count = count + 1
    P1[j] = count / run
l2, = plt.plot(epsilon,P1)
# plot P(|Vrand-miu| > epsilon)
Prand = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
    for i in range(run):
        if np.abs(Vrand[i]-miu) > epsilon[j]:
            count = count + 1
    Prand[j] = count / run
l3, = plt.plot(epsilon,Prand)
# plot P(|Vmin-miu| > epsilon)
Pmin = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
    for i in range(run):
        if np.abs(Vmin[i]-miu) > epsilon[j]:
            count = count + 1
    Pmin[j] = count / run
l4, = plt.plot(epsilon,Pmin)
plt.legend(handles=[l1,l2,l3,l4],labels = ['Hoeffdings bound','V1','Vrand','Vmin'],loc='upper ri
plt.show()
'''
# mean1 = np.mean(V1)
# meanrand = np.mean(Vrand)
# meanmin = np.mean(Vmin)

# 3
# b
# Sum of Bernoulli = Binomial
p = 0.5
epsilon = 0.01
Nset = np.round(np.logspace(2,5,100)).astype(int)
x = np.zeros((10000,Nset.size))
prob_simulate  = np.zeros(100)
prob_chernoff = np.zeros(100)
prob_hoeffding = np.zeros(100)
beta = 1+(0.5+epsilon)*np.log2(0.5+epsilon)+(0.5-epsilon)*np.log2(0.5-epsilon)
for i in range(Nset.size):
  N = Nset[i]
  x[:,i] = stats.binom.rvs(N, p, size=10000)/N
  prob_simulate[i]  = np.mean((x[:,i]-p>=epsilon).astype(float))
  # prob_chebyshev[i] = p*(1-p)/(N* (epsilon**2) )
  prob_chernoff[i]  = 2**(-beta*N)
  prob_hoeffding[i] = np.exp(-2*N*epsilon**2)

plt.figure()
l1, = plt.loglog(Nset, prob_simulate,'x')
l2, = plt.loglog(Nset, prob_chernoff)
l3, = plt.loglog(Nset, prob_hoeffding)
plt.legend(handles=[l1,l2,l3],labels=['Simulation','chernoff','hoeffding'])
```