```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

#%% simulate Brownian Motion
T = 4
N = 101

t = np.linspace(0,T,N)
k = t[1]

num_trajectories = 1000

dWt = np.random.randn(num_trajectories,N)*np.sqrt(k)
dWt[:,0] = 0

Wt = np.cumsum(dWt,axis=1)

plt.figure()
plt.plot(t,Wt.T,color='k',alpha=0.01)
plt.show()

#%% check statistics
mean = np.mean(Wt,axis=0)
var = np.var(Wt,axis=0)

plt.figure()
plt.plot(t,mean)
plt.show()

plt.figure()
plt.plot(t,var)
plt.show()

plt.figure()
plt.hist(Wt[:,-1])
plt.show()

#%% euler-maruyama method
def euler_maruyama(mu,sigma,x0,t,Wt):
    N = len(t)
    x = np.zeros(N)
    x[0] = x0
    for i in range(N-1):
        dt = t[i+1] - t[i]
```

```
        dWt = Wt[i+1] - Wt[i]
        x[i+1] = x[i] + mu(t,x[i])*dt + sigma(t,x[i])*dWt
    return x

#%% Geometric Brownian Motion
u = 1
s = 1

def mu(t,x):
    return u*x

def sigma(t,x):
    return s*x

def x_true_(x0,t,Wt):
    return x0*np.exp((u-s**2/2)*t+s*Wt)

#%% Solve over generated BM trajectories
x0 = 1
x_gbm = euler_maruyama(mu,sigma,x0,t,Wt[0])
x_true = x_true_(x0,t,Wt[0])

plt.figure()
plt.plot(t,x_gbm)
plt.plot(t,x_true)

#%% OU Process
theta = 10
u = 1
s = 0.5

def mu(t,x):
    return theta*(u-x)

def sigma(t,x):
    return s

#%% Solve over generated BM trajectories
x0 = 0
plt.figure()
for wt in Wt:
    x_ou = euler_maruyama(mu,sigma,x0,t,wt)
    plt.plot(t,x_ou,color='k',alpha=.01)
plt.show()
```