```python
##############################################################################
###Code for figure 4.2
###Generating m paths of the Ornstein-Uhlenbeck process (OU)
###on the time intervall [0,T]
###SDE_SamplingOU.py
###Python 2.7
##############################################################################

import numpy as np
import numpy.matlib
import matplotlib.pyplot as plt
from NumericalSDE import *

##################################################
### Ornstein-Uhlenbeck process (OU)
### dXt = a(Xt)dt + b(Xt)dWt
### X0 = x0
### a(x) = -beta*x, b(x) = sigma
### beta, sigma positive constants
### True solution:
###
##################################################
#Parameter
sigma = 1.5
beta = 1.0
#starting value x0
x0 = 1
#Parameters for the discretization
n =2**8
t = timegrid(n)
#m discretized Wiener processes
m = 5
w = np.zeros((n+1,m))
for k in range(0,m):
    w[:,k] = wiener(n)
#m sample paths
Xt = np.zeros((n+1,m))
stochIntApprox = np.zeros((n+1,m))
temp = np.zeros(n+1)
for s in range(0,m):
    Xt[0,s] = x0
    for k in range(0,n):
        for j in range(0,k):
            temp[j+1] =(w[j+1,s]-w[j,s])*np.exp((-beta)*((t[k+1])-t[j]))
        stochIntApprox[k+1,s] = np.sum(temp)
        temp = np.zeros(n)
        Xt[k+1,s] = Xt[0,s]*np.exp((-beta)*(t[k+1])) + sigma*stochIntApprox[k+1,s]


#The code below is needed to test if the approximated stoch. integral is
#a good approximation or not (can be removed)
##def a(x):
##    return -beta*x
##def b(x):
##    return sigma
##Yt  = sde_euler(x0, a, b, w[:,1])
##plt.scatter(t, Yt, 1, c='b')

#Plot
for sample_path in Xt.T:
    plt.plot(t, sample_path,'r',linewidth=0.5)
plt.xlabel('t', fontsize=16)
plt.ylabel('x', fontsize=16)
plt.show()
```