```python
"""
Registration : xxxx
Description  : Autocorrelation of a Random Time-series
Author       : AKB
"""

import numpy as np
#import random
import matplotlib.pyplot as plt

def autocorr1(x,lags):  #Using formula
    mean = np.mean(x)
    var  = np.var(x)
    xp   = x - mean
    corr = [1. if l==0 else np.sum(xp[l:]*xp[:-l])/len(x)/var for l in lags]
    return np.array(corr)

def autocorr2(x,lags):  #Using numpy.corrcoef
    corr = [1. if l==0 else np.corrcoef(x[l:],x[:-l])[0][1] for l in lags]
    return np.array(corr)

def autocorr3(x,lags):  #Using numpy.correlate
    mean = x.mean()
    var  = np.var(x)
    xp   = x-mean
    corr = np.correlate(xp,xp,'full')[len(x)-1:]/var/len(x)
    return corr[:len(lags)]

# Main
# Create a uniformly distributed time-series
#y = np.random.uniform(low=0, high=1, size=10000)
#y = np.array(y).astype('float')

# Create a normal distributed time-series (mean=0, var=0.1)
y = np.random.normal(loc=0.0, scale=0.1, size=100)
y = np.array(y).astype('float')
lags = np.arange(21);

# Calculate ACF using different methods
acf = autocorr3(y,lags)
print 'rk(=ck/c0) for k=2 is ', acf[2], ' and for k=10 is ', acf[10]

# Plot
plt.figure(1)
plt.plot(lags, acf, '-o', lw='2', color="teal", label=r'ACF = $\frac{1}{n\sigma}
\sum_{i=k+1}^{n}(x_i-\bar{x})(x_{i-k}-\bar{x})}$')
plt.legend(loc='best', prop={'size':20})
plt.xlabel('Lag', size=20); plt.xticks(size = 20);
plt.ylabel('Correlation Coefficient', size=20); plt.yticks(size = 20)
plt.title('ACF of a Randomly distributed Time Series', size=20)
plt.xlim([-1.0, max(lags)+1])
plt.ylim([min(acf)-0.1, 1.1])
plt.grid()
plt.show()
```