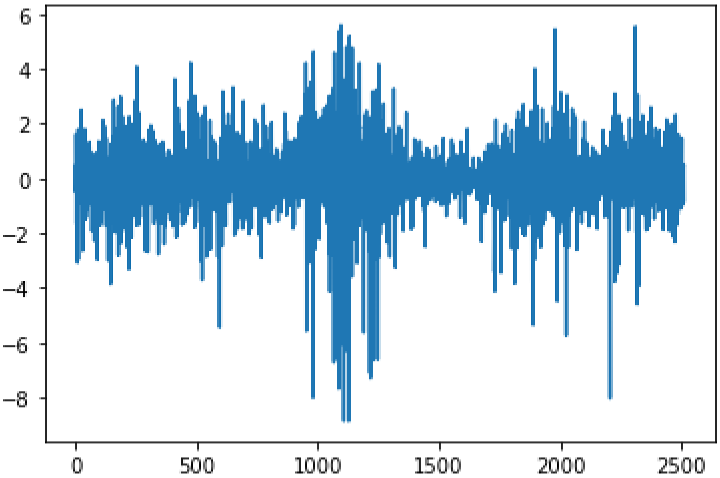
Python

In [3]:
```
import numpy as np
import pandas as pd
import requests
from scipy.stats import norm, chi2, genpareto
import matplotlib.pyplot as plt
from arch import arch_model
%matplotlib inline
```

In [2]:
```
res = requests.get('http://money.finance.sina.com.cn/quotes_service/api/
json_v2.php/CN_MarketData.getKLineData?symbol=sh000001&scale=240&ma=no&d
atalen=10000')
# scale
# ma
# datalen
data_json = res.json()
data = pd.DataFrame(data_json)
data.to_csv('data_ssec.csv')
print(data)
```

```
             day      open      high       low     close       volume
0     1990-12-19    96.050    99.980    95.790    99.980       126000
1     1990-12-20   104.300   104.390    99.980   104.390        19700
2     1990-12-21   109.070   109.130   103.730   109.130         2800
3     1990-12-24   113.570   114.550   109.130   114.550         3200
4     1990-12-25   120.090   120.250   114.550   120.250         1500
...          ...       ...       ...       ...       ...          ...
7417  2021-04-26  3484.107  3497.121  3438.572  3441.166  27697077100
7418  2021-04-27  3440.091  3443.854  3417.265  3442.611  25303239300
7419  2021-04-28  3432.161  3457.068  3423.325  3457.068  24738112000
7420  2021-04-29  3458.082  3478.228  3447.588  3474.901  27663138700
7421  2021-04-30  3468.302  3469.087  3426.902  3446.856  31266035400

[7422 rows x 6 columns]
```

In [4]:
```
data = pd.read_csv('data_ssec.csv')
data['return'] = np.log(data['close']) - np.log(data['close'].shift(peri
ods=1))
data['day'] = pd.to_datetime(data['day'], format='%Y-%m-%d')
ind = data['day'] >= pd.to_datetime('2011-01-01', format='%Y-%m-%d')
r = data[ind]['return'].values*100
plt.plot(r)
plt.show()
```



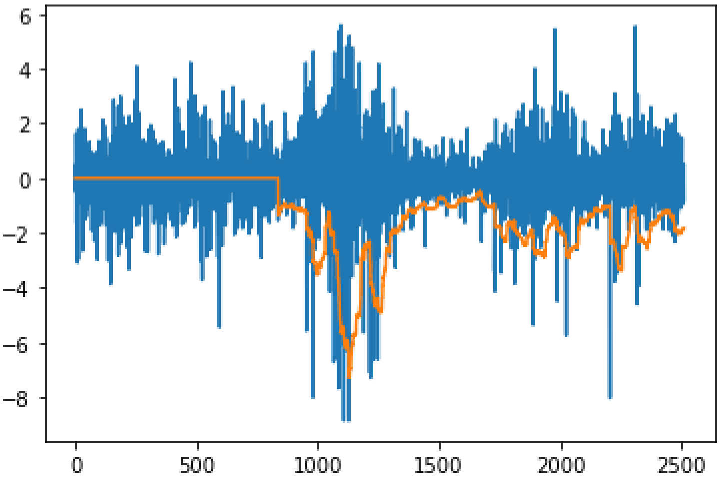
In [6]:

```
# RiskMetrics
l = np.fix(len(r)/3).astype(int)
VaR_RM = np.zeros(len(r))
qalpha = norm.ppf(0.05)
for i in range(l, len(r)):
    mhat, shat = norm.fit(r[i-50:i])
    VaR_RM[i] = -(mhat + qalpha*shat)
plt.plot(r)
plt.plot(VaR_RM*-1)
plt.show()
```

NP.FIX m m 0 -3.5 -3 4.5 4 L r

这里三分之一表示样本内外数据比是1/2（这代表样本内数据占1/3而不是1/2）

上



In [8]:

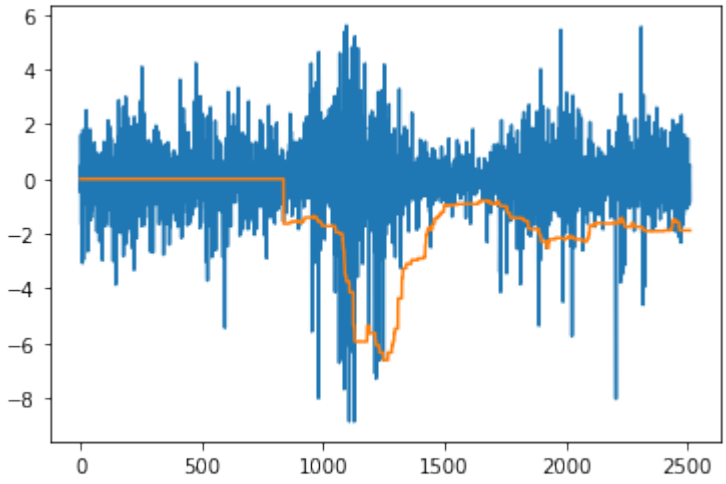
```
l = np.fix(len(r)/3).astype(int)
VaR_GN = np.zeros(len(r))
qalpha = norm.ppf(0.05)
for i in range(l, len(r)):
    am_ar_garch = arch_model(r[:i], mean='ar', lags=1, vol='garch', dist
='normal', p=2, q=2)
    res_ar_garch = am_ar_garch.fit()
    a = res_ar_garch.forecast(horizon=1, align='origin')
    mu = a.mean['h.1'].iloc[-1]
    sigma = a.variance['h.1'].iloc[-1]
    VaR_GN[i] = -(mu + qalpha * np.sqrt(sigma))

plt.plot(r)
plt.plot(VaR_GN*-1)
plt.show()
```

同上 同上 同上 1 horizon=1 上 同

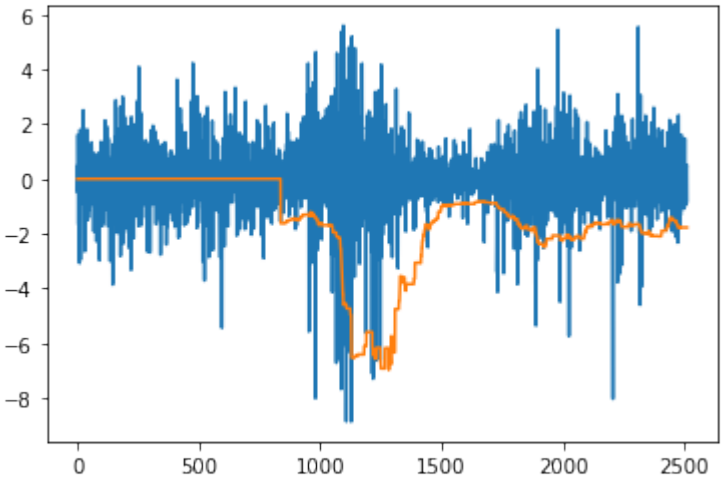
```
Iteration:      1,   Func. Count:      9,   Neg. LLF: 1775472581.7961023
Iteration:      2,   Func. Count:     20,   Neg. LLF: 40316.00309103527
```

KeyboardInterrupt:

In [9]:
```
#
l = np.fix(len(r)/3).astype(int) 同上，注意根据题设改变
VaR_HS = np.zeros(len(r))同上
qalpha = int(200*0.05)
for i in range(l, len(r)):
    his_sample = r[i-200:i] 200
    his_sample = np.sort(his_sample)
    VaR_HS[i] = -his_sample[qalpha-1]
plt.plot(r)
plt.plot(VaR_HS*-1) 同上
plt.show()
```



In [11]:
```
#POT
l = np.fix(len(r)/3).astype(int) 同上，注意根据题设改变
VaR_EVT = np.zeros(len(r))同上
alpha = 0.05
for i in range(l, len(r)):
    his_sample = r[i-200:i] 历史数据量设置为200，请根据题设改变，如果没独立要求应当是沿用历史模拟法的
    his_sample = np.sort(his_sample)默认升序排序
    ind = np.ceil(len(his_sample)*0.1).astype(int)这里np.ceil是向上取整（-1.1变-1）和np.fix不同。阈值设置u为样本内数据十分位对应的值0.1，请根据题设条件改变
    evt_sample = np.abs(his_sample[:ind])
    u = evt_sample[-1]因为evt_sample来自于his_sample，所以是升序排序的，最后一项就是其中最大项
    evt_sample = evt_sample - u  各个亏损超过u所对应的超额亏损
    evt_sample = np.delete(evt_sample, -1)把最大项u给删除
    n = len(his_sample)历史数据量个数
    Nu = len(evt_sample)超出阈值的样本个数，删除了一个u。如果存在和u等大的，也称作超出阈值的样本个数就行了

    parmhat = genpareto.fit(evt_sample, floc=0)用于拟合广义帕累托分布GPD尾部风险PPT第十五页
    kHat = parmhat[0]; # Tail index parameter 提取形状参数 尾部风险PPT第十五页
    sigmaHat = parmhat[2]; # Scale parameter 提取尺度参数  尾部风险PPT第十五页
尺度参数 σ 大于0形状参数 ε 大于0时, GPD具有厚尾的特点  尾部风险PPT第十五页 注意可能出题要求判断
    VaR_EVT[i] = u + sigmaHat / kHat * (((1-alpha) * n / Nu) ** -kHat - 1)

    公式来自于尾部风险PPT第16页 这里老师原来的代码是alpha，但是应该用置信水平而不是显著性水平我直接就改了
plt.plot(r)
plt.plot(VaR_EVT*-1)
plt.show()
```



In [ ]:
```
data = pd.DataFrame({'return': r, 'VaR_RM': VaR_RM, 'VaR_GN': VaR_GN, 'VaR_HS': VaR_HS, 'VaR_EVT': VaR_EVT})
data.to_csv('Data_VaR.csv')
```

In [12]:
```
def myfun_Kupiec(r, VaR, pstar):
    N = np.sum(r > VaR)
    T = len(r)
    LRuc = -2*((T-N)*np.log(1-pstar)+N*np.log(pstar)) + 2*((T-N)*np.log(1-N/T)+N*np.log(N/T))
    pvalue_LRuc = 1 - chi2.cdf(LRuc, 1)
    return LRuc, pvalue_LRuc

def myfun_Christoffersen(r, VaR):
    ind = r > VaR
    ind1 = ind[:-1]
    ind2 = ind[1:]
    n00 = np.sum((ind1==0) & (ind2==0))
    n01 = np.sum((ind1==0) & (ind2==1))
    n10 = np.sum((ind1==1) & (ind2==0))
    n11 = np.sum((ind1==1) & (ind2==1))

    Pi01 = n01/(n01+n00)
    Pi11 = n11/(n10+n11)
    Pi2 = (n01+n11)/(n00+n01+n10+n11)

    LRind = (n00+n10)*np.log(1-Pi2) + (n01+n11)*np.log(Pi2) - \
            n00*np.log(1-Pi01) - n01*np.log(Pi01) - n10*np.log(1-Pi11) - n11*np.log(Pi11)
    LRind = LRind*-2
    pvalue_LRind = 1 - chi2.cdf(LRind, 1)
    return LRind, pvalue_LRind

def myfun_Kupiec_Christoffersen(LRuc, LRind):
    LRcc = LRuc + LRind
    pvalue_LRcc = 1 - chi2.cdf(LRcc, 2)
    return LRcc, pvalue_LRcc

data = pd.read_csv('Data_VaR.csv')
ind = data['VaR_RM'] > 0
r = data.loc[ind, ['return']].values*-1
VaR_RM = data.loc[ind, ['VaR_RM']].values
VaR_GN = data.loc[ind, ['VaR_GN']].values
VaR_HS = data.loc[ind, ['VaR_HS']].values
VaR_EVT = data.loc[ind, ['VaR_EVT']].values
```

```
pstar = 0.05;
[LRuc_RM, pvalue_LRuc_RM] = myfun_Kupiec(r, VaR_RM, pstar)
[LRind_RM, pvalue_LRind_RM] = myfun_Christoffersen(r, VaR_RM)
[LRcc_RM, pvalue_LRcc_RM] = myfun_Kupiec_Christoffersen(LRuc_RM, LRind_R
M)

[LRuc_GN, pvalue_LRuc_GN] = myfun_Kupiec(r, VaR_GN, pstar)
[LRind_GN, pvalue_LRind_GN] = myfun_Christoffersen(r, VaR_GN)
[LRcc_GN, pvalue_LRcc_GN] = myfun_Kupiec_Christoffersen(LRuc_GN, LRind_G
N)

[LRuc_HS, pvalue_LRuc_HS] = myfun_Kupiec(r, VaR_HS, pstar)
[LRind_HS, pvalue_LRind_HS] = myfun_Christoffersen(r, VaR_HS)
[LRcc_HS, pvalue_LRcc_HS] = myfun_Kupiec_Christoffersen(LRuc_HS, LRind_H
S)

[LRuc_EVT, pvalue_LRuc_EVT] = myfun_Kupiec(r, VaR_EVT, pstar)
[LRind_EVT, pvalue_LRind_EVT] = myfun_Christoffersen(r, VaR_EVT)
[LRcc_EVT, pvalue_LRcc_EVT] = myfun_Kupiec_Christoffersen(LRuc_EVT, LRin
d_EVT)


print('{:12s}, {:>12s}, {:>12s}, {:>12s}, {:>12s}, {:>12s}, {:>12s}'.for
mat('', 'LRuc', 'pLRuc', 'LRind', 'pLRind', 'LRcc', 'pLRcc'))
print('{:12s}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f
}'.format('RiskMetrics', LRuc_RM, pvalue_LRuc_RM, LRind_RM, pvalue_LRind
_RM, LRcc_RM, pvalue_LRcc_RM))
print('{:12s}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f
}'.format('GarchNormal', LRuc_GN, pvalue_LRuc_GN, LRind_GN, pvalue_LRind
_GN, LRcc_GN, pvalue_LRcc_GN))
print('{:12s}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f
}'.format('HisSim', LRuc_HS, pvalue_LRuc_HS, LRind_HS, pvalue_LRind_HS,
LRcc_HS, pvalue_LRcc_HS))
print('{:12s}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f}, {:12.4f
}'.format('EVT GPD', LRuc_EVT, pvalue_LRuc_EVT, LRind_EVT, pvalue_LRind_
EVT,LRcc_EVT, pvalue_LRcc_EVT))
```

```
             ,         LRuc,        pLRuc,        LRind,        pLRind,
      LRcc,        pLRcc
RiskMetrics ,       1.2853,       0.2569,       3.9733,       0.0462,
    5.2587,       0.0721
GarchNormal ,       0.2829,       0.5948,       0.4341,       0.5100,
    0.7170,       0.6987
HisSim      ,       1.8203,       0.1773,      15.7443,       0.0001,
   17.5646,       0.0002
EVT GPD     ,       1.0515,       0.3052,      12.1244,       0.0005,
   13.1759,       0.0014
```