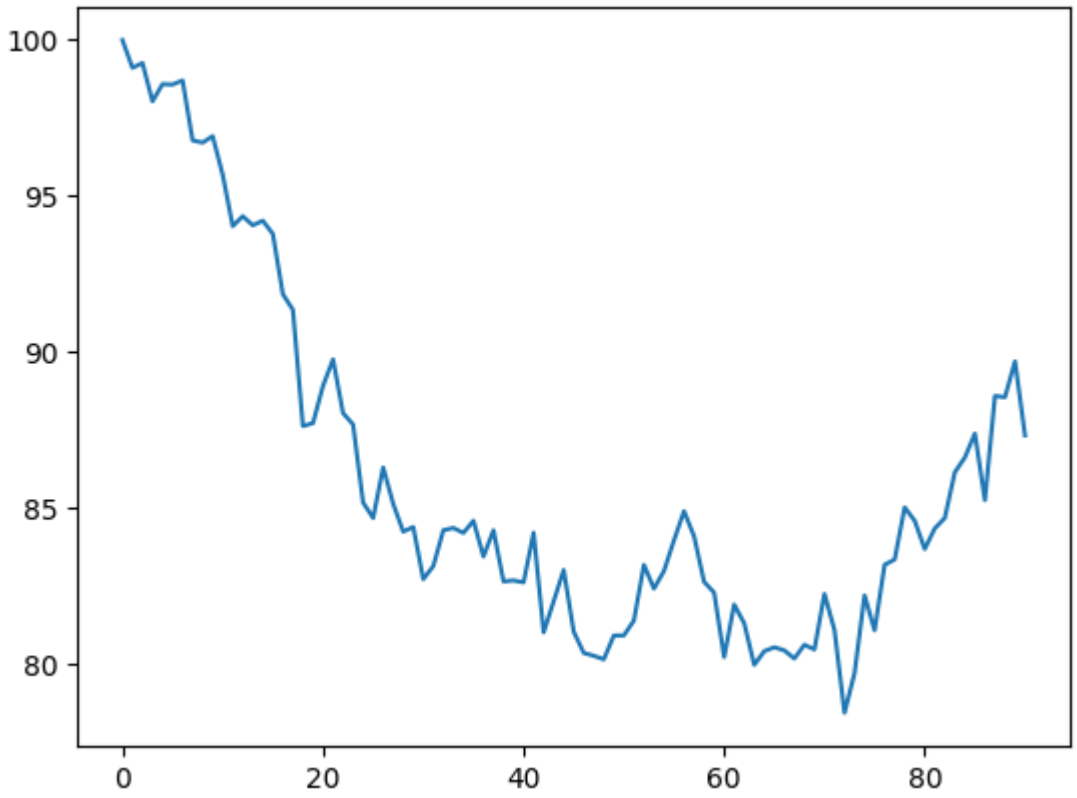
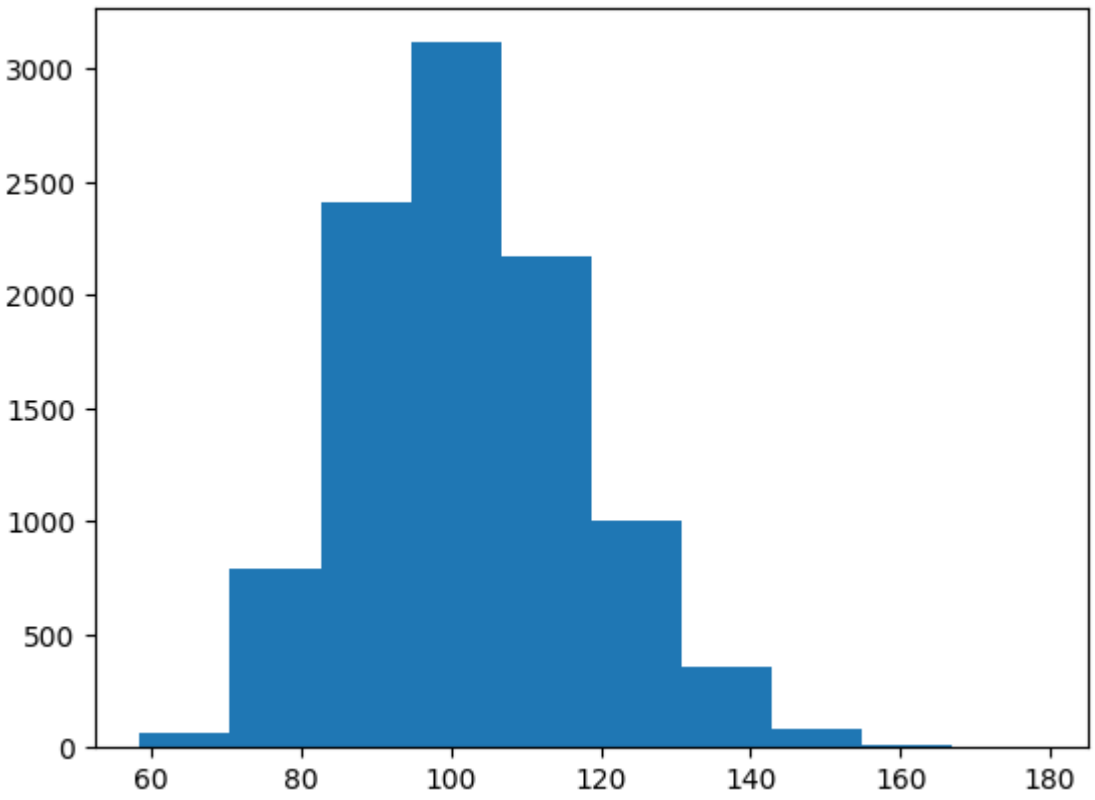
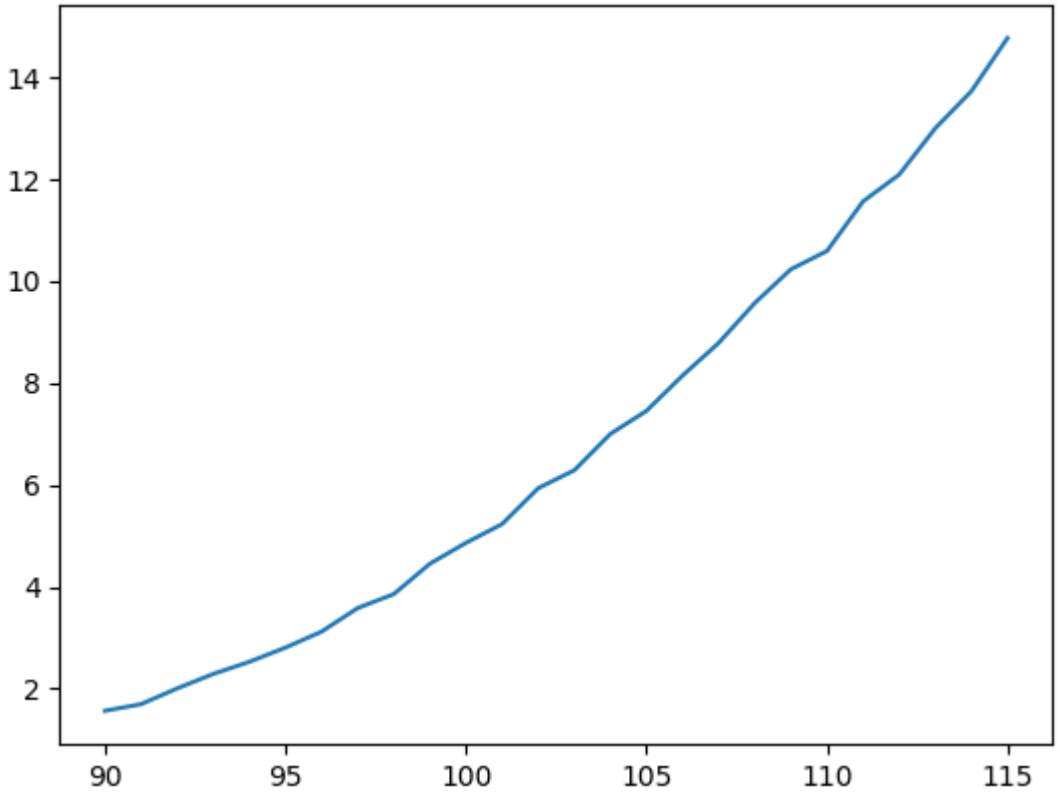
In [1]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
import math
```

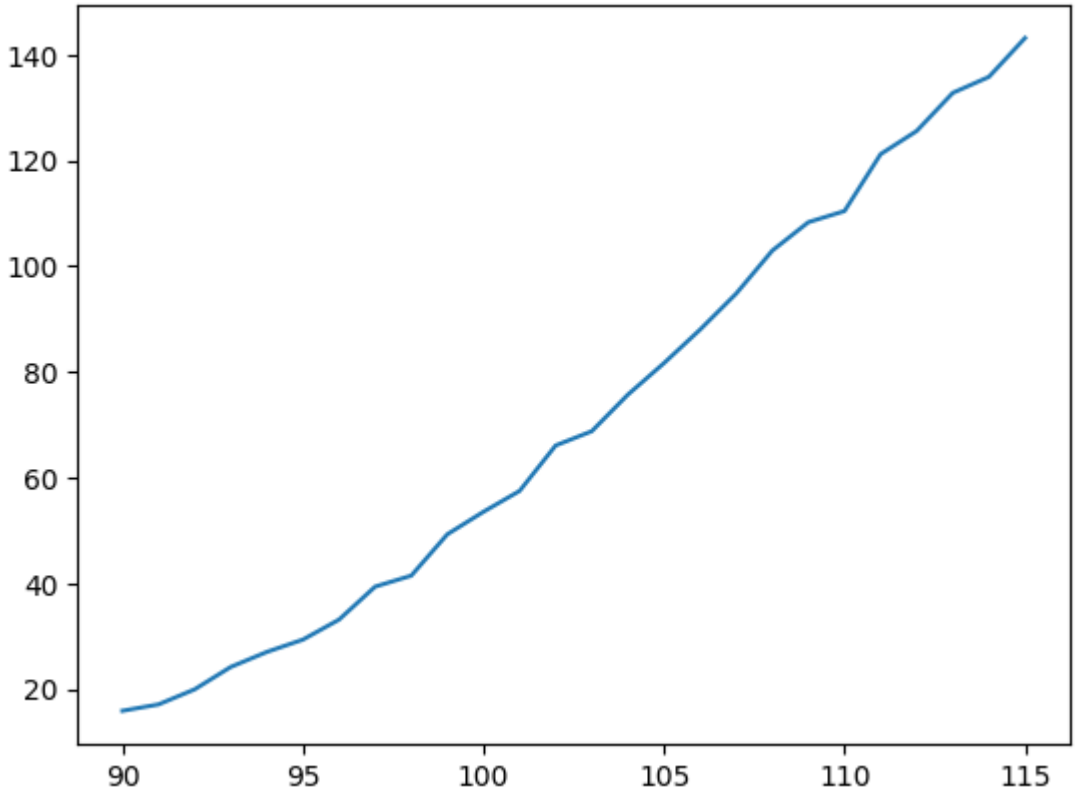
In [27]:
```
def Stock(datanum,Simulateday):
    S=datanum
    for t in range(Simulateday):
        delta_s=1/3650+0.3*math.sqrt(1/365)*random.normalvariate(0,1)
        S=delta_s*S+S
    return S
def StockShow(dataX,Simulateday):
    data=dataX
    for t in range(Simulateday):
        delta_s=1/3650+0.3*math.sqrt(1/365)*random.normalvariate(0,1)
        S=data[-1]
        data.append(delta_s*S+S)
    plt.plot(range(Simulateday+1),data)
    plt.show()
def Simulate(Simulatetimes, Simulateday,firstprice):
    data=[]
    for t in range(Simulatetimes):
        data.append(Stock(firstprice,Simulateday))
    plt.hist(data)
    plt.show()
def Price(datanum,Simulateday,Xprice):
    S=datanum
    for t in range(Simulateday):
        delta_s=1/3650+0.3*math.sqrt(1/365)*random.normalvariate(0,1)
        S=delta_s*S+S
    return max(Xprice-S,0)
def SimulatePrice(Simulatetimes, Simulateday,firstprice,priceset,Profitper):
    mean=[]
    Priceexp=[]
    var=[]
    for price in priceset:
        data=[]
        for t in range(Simulatetimes):
            data.append(Price(firstprice,Simulateday,price))
        mean.append(np.mean(data))
        Priceexp.append(np.mean(data)*math.exp(-Profitper*Simulateday/365))
        var.append(np.var(data))
    plt.plot(priceset,mean)
    plt.show()
    plt.plot(priceset,var)
    plt.show()
    plt.plot(priceset,Priceexp)
    plt.show()
```

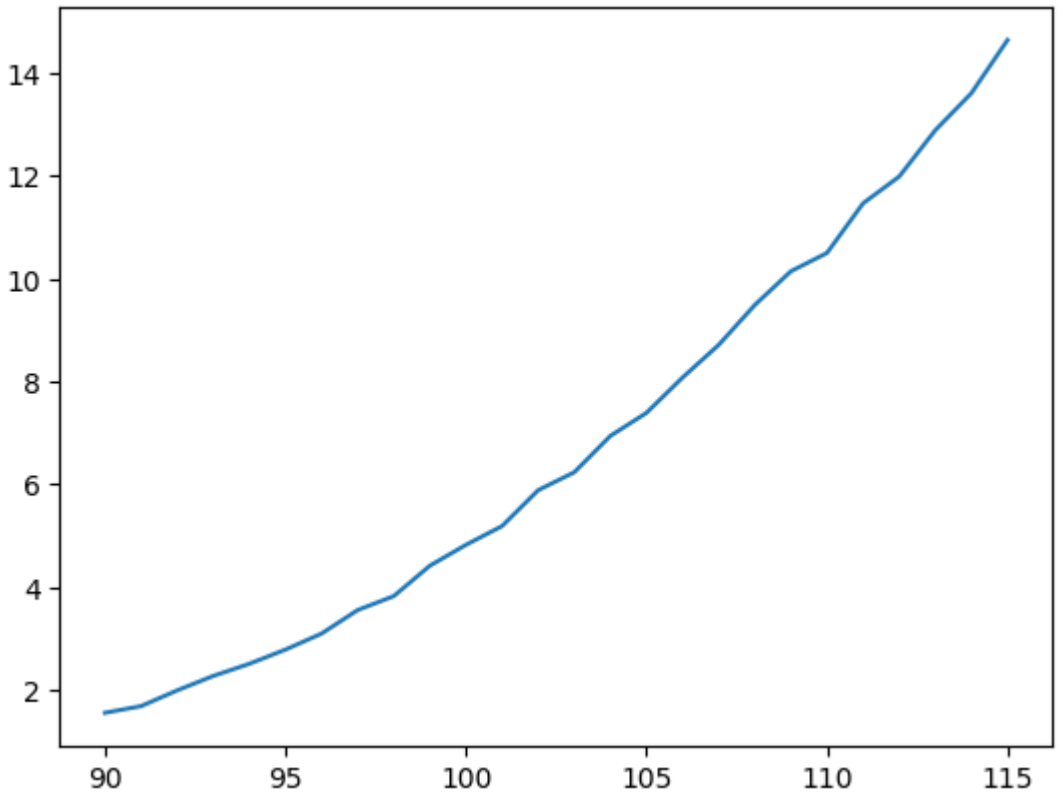
In [28]:
```
dataX=[100]
StockShow(dataX,90)
Simulate(10000,90,100)
set=np.arange(90,116,1).tolist()
SimulatePrice(10000,90,100,set,0.031)
```


100
95
90
85
80
0
20
40
60
80


3000
2500
2000
1500
1000
500
0
60
80
100
120
140
160
180


14
12
10
8
6
4
2
90
95
100
105
110
115


140
120
100
80
60
40
20
90
95
100
105
110
115



在这里模拟以90元定价为基础

In [47]:
```
def StockX(datanum,Simulateday,mu,sigma,Xprice):
    S=datanum
    for t in range(Simulateday):
        delta_s=1/365*mu+sigma*math.sqrt(1/365)*random.normalvariate(0,1)
        S=delta_s*S+S
    return max(Xprice-S,0)
def Simulatemu(times, start,stop):
    X=[]
    z=start
    sigma=0.3
    length=0.001
    while(z<stop ):
        data=[]
        for  t in range(times):
            data.append(math.exp(-0.031*90/365)*StockX(100,90,z,sigma,90))
        X.append(np.mean(data))
        z+=length
        #print(z)
    #print(X)
    #print(len(X))
    plt.plot(np.arange(start,stop,length),np.array(X))
    plt.show()
def Simulatesigma(times, start,stop):
    X=[]
    z=start
    mu=0.1
    length=0.001
    while(z<stop ):
        data=[]
        for  t in range(times):
            data.append(math.exp(-0.031*90/365)*StockX(100,90,mu,z,90))
        X.append(np.mean(data))
        z+=length
        #print(z)
    #print(X)
```

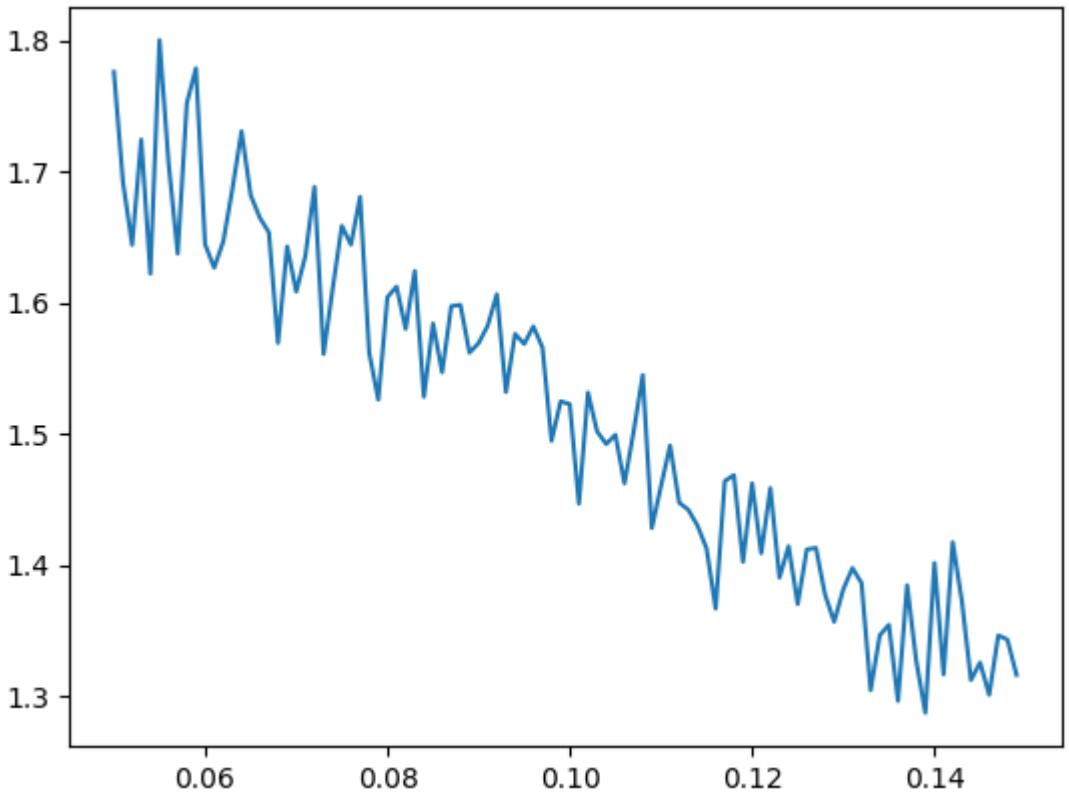
```
    #print(len(X))
    plt.plot(np.arange(start,stop,length),np.array(X))
    plt.show()
def Simulatetime(times, start,stop):
    X=[]
    z=start
    mu=0.1
    sigma=0.3
    length=1
    while(z<stop ):
        data=[]
        for  t in range(times):
            data.append(math.exp(-0.031*z/365)*StockX(100,z,mu,sigma,90))
        X.append(np.mean(data))
        z+=length
        #print(z)
    #print(X)
    #print(len(X))
    plt.plot(np.arange(start,stop,length),np.array(X))
    plt.show()
def SimulateRisk(times, start,stop):
    X=[]
    no=[]
    z=start
    mu=0.1
    sigma=0.3
    time=90
    length=0.0001
    while(z<stop ):
        data=[]
        no.append(z)
        for  t in range(times):
            data.append(math.exp(-z*time/365)*StockX(100,time,mu,sigma,90))
        X.append(np.mean(data))
        z+=length

        #print(z)
    #print(X)
    #print(len(X))
    plt.plot(no,np.array(X))
    plt.show()
```

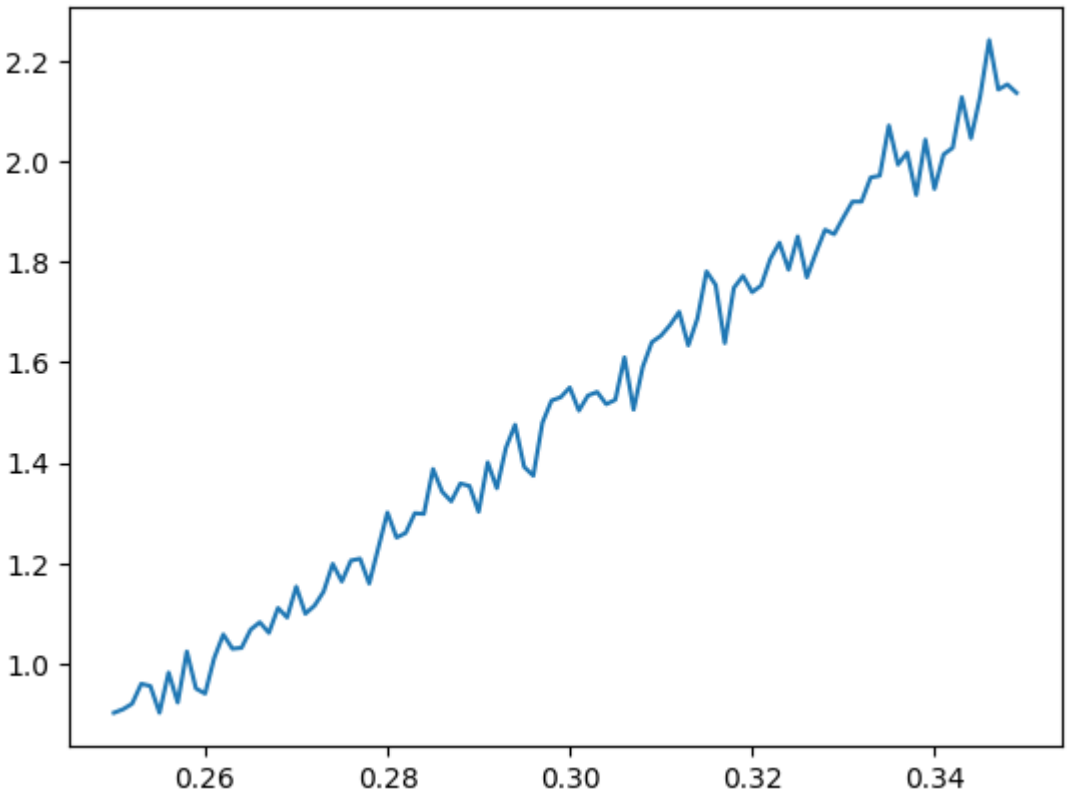
In [44]:
```
Simulatemu(10000,0.05,0.15)
```

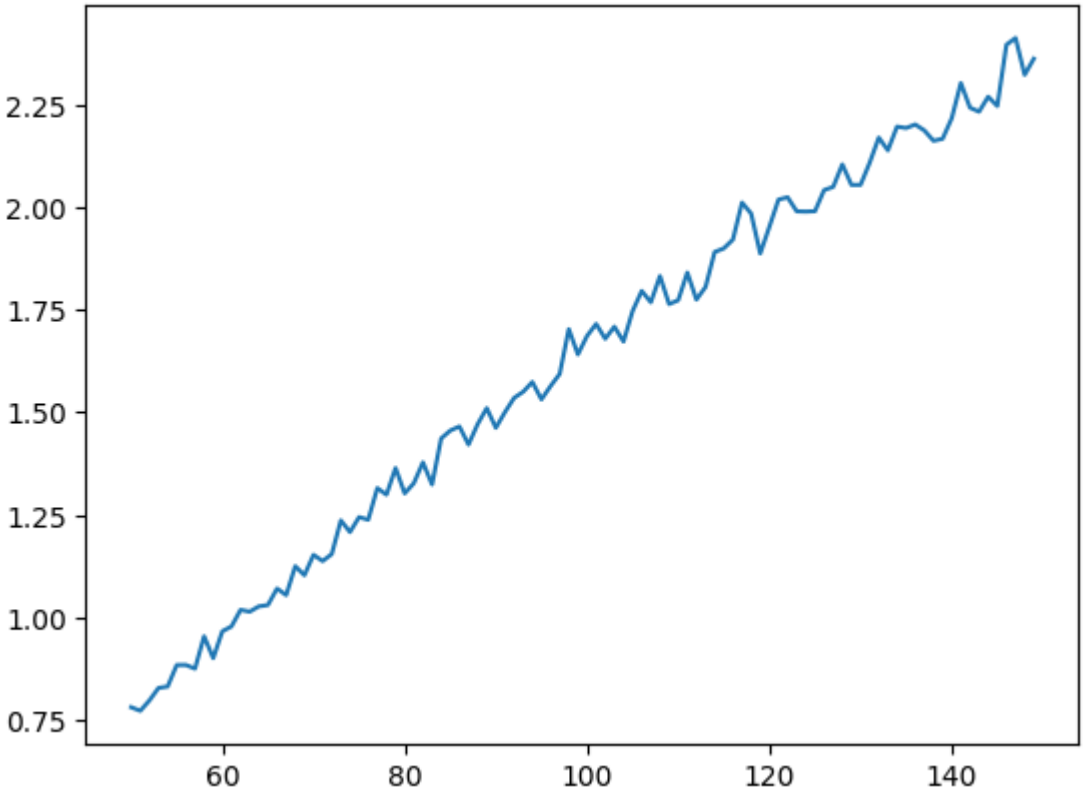
- 0.051000000000000004
- 0.052000000000000005
- 0.053000000000000005
- 0.054000000000000006
- 0.055000000000000001
- 0.056000000000000001
- 0.057000000000000001
- 0.058000000000000001
- 0.059000000000000001
- 0.060000000000000001
- 0.061000000000000001
- 0.062000000000000001
- 0.063000000000000001
- 0.064000000000000002
- 0.065000000000000002
- 0.066000000000000002
- 0.067000000000000002
- 0.068000000000000002
- 0.069000000000000002
- 0.070000000000000002
- 0.071000000000000002
- 0.072000000000000002
- 0.073000000000000002
- 0.074000000000000002
- 0.075000000000000002
- 0.076000000000000003
- 0.077000000000000003
- 0.078000000000000003
- 0.079000000000000003
- 0.080000000000000003
- 0.081000000000000003
- 0.082000000000000003
- 0.083000000000000003
- 0.084000000000000003
- 0.085000000000000003
- 0.086000000000000003
- 0.087000000000000004
- 0.088000000000000004
- 0.089000000000000004
- 0.090000000000000004
- 0.091000000000000004
- 0.092000000000000004
- 0.093000000000000004
- 0.094000000000000004
- 0.095000000000000004
- 0.096000000000000004
- 0.097000000000000004
- 0.098000000000000005
- 0.099000000000000005
- 0.10000000000000005
- 0.10100000000000005
- 0.10200000000000005
- 0.10300000000000005
- 0.10400000000000005
- 0.10500000000000005
- 0.10600000000000005
- 0.10700000000000005
- 0.10800000000000005
- 0.10900000000000006
- 0.11000000000000006
- 0.11100000000000006
- 0.11200000000000006
- 0.11300000000000006
- 0.11400000000000006

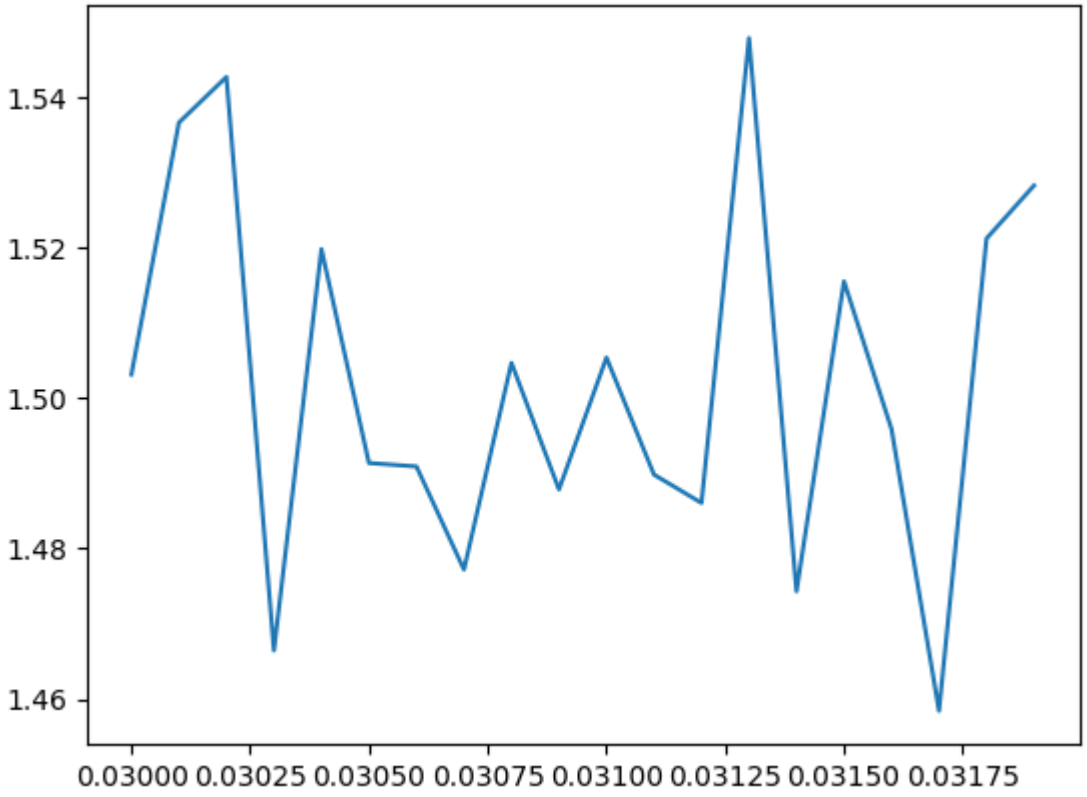
```
0 115000000000000006
0 116000000000000006
0 117000000000000006
0 118000000000000006
0 119000000000000006
0 120000000000000006
0 121000000000000007
0 122000000000000007
0 123000000000000007
0 124000000000000007
0 125000000000000006
0 126000000000000006
0 127000000000000006
0 128000000000000006
0 129000000000000006
0 130000000000000006
0 131000000000000006
0 132000000000000006
0 133000000000000006
0 134000000000000006
0 135000000000000006
0 136000000000000007
0 137000000000000007
0 138000000000000007
0 139000000000000007
0 140000000000000007
0 141000000000000007
0 142000000000000007
0 143000000000000007
0 144000000000000007
0 145000000000000007
0 146000000000000007
0 147000000000000008
0 148000000000000008
0 149000000000000008
0 150000000000000008
[1 7758940977425925 1 6916581070297505 1 6439094850940827 1 7243012092084894 1 0
219807057375324 1 8001739119473195 1 7061938403769854 1 6373098788827317 1 751919
01666646648 1 7786540468421903 1 6444846707807388 1 6265450838313893 1 6466364986
442756 1 6867037596179995 1 7306211034362662 1 6818882088117723 1 6647347647228126
1 6534410999379447 1 5692209798895551 1 6427805047820279 1 6082962748888386 1 6355
289653848157 1 6881654425520762 1 5607477239292227 1 6109172689632567 1 65845724
14291767 1 6440895415220002 1 6805846275762637 1 5616794335940011 1 525958100949957
97 1 6041126061403734 1 6121019065852 47 1 5798538619536102 1 6240394761023438
1 5281310617046 63 1 5839638079467795 1 5468001431446738 1 5972088481636415 1 598
261742139978 1 5618050584333727 1 5690135297632424 1 5823950074531 71 1 6063325855
8144701 1 5317585765391042 1 5760571795650187 1 5684612770921 3 1 5817538177877753
5 1 5658238815008362 1 4946015033034483 1 5246871315234376 1 5234449707145908
1 4466411671174735 1 5313211806039853 1 5016334497823716 1 492113533302 15 1 4991
253087652971 1 4620669230230 69 1 5015010825427826 1 5447092099317 88 1 4280113256
761942 1 4598900107727621 1 4909987955622 87 1 4474292686272 45 1 4419431686850 83
7 1 4300191271396718 1 4124998113066023 1 3666155579823263 1 4638442174499298
1 4683513666006873 1 4022898666079269 1 4621326316931877 1 4088331352569 93 1 458
4875849533052 1 3902271322293833 1 4141858124864 83 1 3700770018651 56 1 4117320 30
2597192 1 4128988877219113 1 3772404880538807 1 3565208331866816 1 3813321351584
59 1 3975184468726267 1 3860792681565 99 1 3043888075275 05 1 3462760507882154 1
3542527538892406 1 2962374968288186 1 3844793581140 77 1 3268247935545443 1 28713
81208816526 1 4013731445158 97 1 3164316899911 27 1 4172520975745566 1 372716956 62
4758 1 3121080032151961 1 3255325244080038 1 3008218104157727 1 3461824146707 71
5 1 342986036328164 1 3160721453643 42]
10 0
```



In [48]:
```
Simulatesigma(10000,0.25,0.35)
Simulatetime(10000,50,150)
SimulateRisk(10000,0.03,0.032)
```




2.25
2.00
1.75
1.50
1.25
1.00
0.75
60
80
100
120
140


1.54
1.52
1.50
1.48
1.46
0.03000 0.03025 0.03050 0.03075 0.03100 0.03125 0.03150 0.03175

In [ ]: