```
# Andrew Morris

from random import gauss
from random import gammavariate
from Andrew_functions import *

def getRandomReturnRate():
    x = gauss(11.5,20)
    y = 2.0*gammavariate(1,2.0)
    ans = x/(y/2)**(1/2)
    while ans > 50 or ans < -50:
        x = gauss(11.5,20)
        y = 2.0*gammavariate(1,2.0)
        ans = x/(y/2)**(1/2)
    return round(x,2)

def nestEggFixed(salary, save, growthRate, years):
    F = []
    F.append(salary*save*0.01)
    for i in range(years-1):
        F.append(F[-1]*(1+ 0.01*growthRate) + salary * save * 0.01)
    return F

def testNestEggFixed():
    print("------------------------------------------------------------------------")
    print("Testing nestEggFixed...\n")
    success = True
    salary     = 10000
    save       = 10
    growthRate = 15
    years      = 5
    savingsRecord = [1000.0, 2150.0, 3472.5, 4993.375, 6742.3812499999995]
    if savingsRecord == nestEggFixed(salary, save, growthRate, years):
        print("Nest Egg Fixed Test 1: PASS!")
    else:
        print("Nest Egg Fixed Test 1: FAIL!")
        print("Output should have values close to: \n[1000.0, 2150.0, 3472.5, 4993.375, 6742.3812499999
995]")
        print('Your output was:\n',nestEggFixed(salary, save, growthRate, years),sep='')
        success = False

    salary     = 50000
    save       = 10
    growthRate = 21
    years      = 14
    savingsTest = []
    savingsRecord = [5000.0, 11050.0, 18370.5, 27228.305, 37946.24905,
    50914.96135, 66607.10323, 85594.59491, 108569.45985, 136369.04641,
    170006.54616, 210707.92085, 259956.58423, 319547.46692]
    for num in nestEggFixed(salary, save, growthRate, years):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Fixed Test 2: PASS!")
    else:
        print("Nest Egg Fixed Test 2: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False

    salary     = 50000
    save       = 10
    growthRate = 3
    years      = 20
    savingsTest = []
```

```
    savingsRecord = [5000. , 10150. , 15454.5 , 20918.135 , 26545.67905
    , 32342.04942 , 38312.3109 , 44461.68023 , 50795.53064 , 57319.39656
    , 64038.97845 , 70960.14781 , 78088.95224 , 85431.62081 ,
    92994.56943 , 100784.40652 , 108807.93871 , 117072.17687 ,
    125584.34218 , 134351.87244]
    for num in nestEggFixed(salary, save, growthRate, years):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Fixed Test 3: PASS!")
    else:
        print("Nest Egg Fixed Test 3: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False
    salary     = 15000
    save       = 20
    growthRate = 7
    years      = 20
    savingsTest = []
    savingsRecord = [3000. , 6210. , 9644.7 , 13319.829 , 17252.21703 ,
    21459.87222 , 25962.06328 , 30779.40771 , 35933.96625 , 41449.34388
    , 47350.79796 , 53665.35381 , 60421.92858 , 67651.46358 ,
    75387.06603 , 83664.16065 , 92520.6519 , 101997.09753 , 112136.89436
    , 122986.47696]
    for num in nestEggFixed(salary, save, growthRate, years):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Fixed Test 4: PASS!")
    else:
        print("Nest Egg Fixed Test 4: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False
    if success == True:
        print("\nSUCCESS: testNestEggFixed()",sep='')
    else:
        print("\nFAILURE: testNestEggFixed()",sep='')

def nestEggVariable(salary, save, growthRates):
    F = []
    F.append(salary*save*0.01)
    i = 0
    for gr in growthRates[1:]:
        F.append(F[-1]*(1+ 0.01*gr) + salary * save * 0.01)
        i = i+1
    return F

def testNestEggVariable():
    print("------------------------------------------------------------------------")
    print("Testing nestEggVariable...\n")
    success = True
    salary      = 10000
    save        = 10
    growthRates = [3, 4, 5, 0, 3]
    savingsRecord = nestEggVariable(10000, 10, [3, 4, 5, 0, 3])
    #print(savingsRecord)
    #print("Output should have values close to: \n[1000.0, 2040.0, 3142.0, 4142.0, 5266.2600000000002]"
)

    if savingsRecord == nestEggVariable(salary, save, growthRates):
        print("Nest Egg Variable Test 1: PASS!")
    else:
        print("Nest Egg Variable Test 1: FAIL!")
        print("Output should have values close to: \n[1000.0, 2040.0, 3142.0, 4142.0, 5266.260000000000
```

```
2]")
        print('Your output was:\n',nestEggVariable(salary, save, growthRates),sep='')

    salary     = 50000
    save       = 10
    years      = 20
    savingsRecord = [5000., 10700., 21050., 30681., 36908.24, 45599.064,
    54702.97976, 73925.7545, 79665.01204, 110954.46602, 127049.91262,
    158730.39427, 181190.73764, 204309.8114, 225654.59631, 305120.6131,
    313171.81923, 336962.12838, 432941.90304, 524530.28365,
    655417.55173, 948801.27449, 1342809.79703, 1737224.63817,
    1759596.88455, 2169304.168, 2369541.54312, 2872145.26718,
    3710067.39466, 4902288.96095, 7260387.66221, 9370900.08425,
    12093461.10868]
    growthRates = [8 , 14 , 50 , 22 , 4 , 10 , 9 , 26 , 1 , 33 , 10 , 21
    , 11 , 10 , 8 , 33 , 1 , 6 , 27 , 20 , 24 , 44 , 41 , 29 , 1 , 23 ,
    9 , 21 , 29 , 32 , 48 , 29 , 29]
    savingsTest = []
    for num in nestEggVariable(salary, save, growthRates):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Variable Test 2: PASS!")
    else:
        print("Nest Egg Variable Test 2: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False

    salary     = 60000
    save       = 10
    years      = 20
    savingsRecord = [6000. , 12840. , 14859.6 , 20116.62 , 19478.1354 ,
    19245.13207 , 23897.97283 , 27269.19582 , 44176.87414 , 54152.79282
    , 66109.60002 , 56904.39202 , 54937.77714 , 44456.444 , 59347.73279
    , 60599.91417 , 44783.94507 , 47201.22946 , 65001.53683 ,
    87251.92104]
    growthRates = [-29 ,14 ,-31 ,-5 ,-33 ,-32 ,-7 ,-11 ,40 ,9 ,11 ,-23
    ,-14 ,-30 ,20 ,-8 ,-36 ,-8 ,25 ,25]
    savingsTest = []
    for num in nestEggVariable(salary, save, growthRates):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Variable Test 3: PASS!")
    else:
        print("Nest Egg Variable Test 3: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False
    salary     = 60000
    save       = 10
    years      = 25
    savingsRecord = [6000. , 12840. , 23847.6 , 21977.892 , 30395.46012
    , 39738.96073 , 48918.07759 , 44645.2813 , 64485.3185 , 78868.4099 ,
    66728.67563 , 48039.06564 , 71333.12928 , 70199.81635 , 83921.79615
    , 84886.48838 , 58629.62279 , 71078.8813 , 105510.43382 ,
    89353.24272 , 99820.90486 , 137763.59441 , 182337.40085 ,
    184690.65283 , 211006.62464]
    growthRates = [34 ,14 ,39 ,-33 ,11 ,11 ,8 ,-21 ,31 ,13 ,-23 ,-37 ,36
    ,-10 ,11 ,-6 ,-38 ,11 ,40 ,-21 ,5 ,32 ,28 ,-2 ,11]
    savingsTest = []
    for num in nestEggVariable(salary, save, growthRates):
        savingsTest.append(round(num,5))
    if savingsRecord == savingsTest:
        print("Nest Egg Variable Test 4: PASS!")
```

```
    else:
        print("Nest Egg Variable Test 4: FAIL!")
        print("Output should have values close to: \n",savingsRecord,sep='')
        print('Your output was:\n',savingsTest,sep='')
        success = False
    if success == True:
        print("\nSUCCESS: testNestEggVariable()",sep='')
    else:
        print("\nFAILURE: testNestEggVariable()",sep='')

def nestEggRandom(salary, save, years):
    F = []
    F.append(salary*save*0.01)
    years-1
    dictlist = []
    for i in range(years):
        F.append(F[-1]*(1+ 0.01*getRandomReturnRate()) + salary * save * 0.01)
    return F

def monteCarlo(num,salary,save,years,goal=0):
    runlist = []
    Results = {}
    for i in range(num):
        runlist.append(nestEggRandom(salary, save, years)[-1])
    sr = 0
    for end in runlist:
        if end > goal:
            sr = sr+1
    successRate = sr/len(runlist)*100
    Results['min'] = get_low(runlist)
    Results['q1'] = get_percentile(runlist, 25)
    Results['med'] = get_percentile(runlist, 50)
    Results['q3'] = get_percentile(runlist,75)
    Results['max'] = get_high(runlist)
    Results['successRate'] = successRate
    return Results

def nestEggRandom2(salary, save, years, growth):
    F = {}
    F[0] = salary*save*0.01
    years-1
    dictlist = []
    for i in range(years):
        salary = salary*(1+.01*growth)
        F[i+1] = F[i]*(1+ 0.01*getRandomReturnRate()) + salary * save * 0.01
    for key, value in F.items():
        dictlist.append(value)
    return dictlist

def monteCarlo2(num,salary,save,years, growth, goal=0):
    runlist = []
    Results = {}
    for i in range(num):
        runlist.append(nestEggRandom2(salary, save, years, growth)[-1])
    sr = 0
    for end in runlist:
        if end > goal:
            sr = sr+1
    successRate = sr/len(runlist)*100
    #print('high:',get_high(runlist))
    Results["min"] = get_low(runlist)
    Results["q1"] = get_percentile(runlist, 25)
    Results['med'] = get_percentile(runlist, 50)
    Results['q3'] = get_percentile(runlist,75)
```

```
    Results['max'] = get_high(runlist)
    Results['successRate'] = successRate
    return Results


def main():
    testNestEggFixed()
    testNestEggVariable()
    print("\n-----------------------------------------------------------------------")
    print('Problem 1 $1,000,000  7% annual growth:')
    print("-----------------------------------------------------------------------")
    for i in range(100):
        if nestEggFixed(50000, i, 7, 20)[19] > 1000000:
            print("Smallest savings percentage:",i)
            break
        else:
            pass
    print("\n-----------------------------------------------------------------------")
    print('Problem 2 $1,000,000 growth given by list:')
    print("-----------------------------------------------------------------------")
    for i in range(100):
        if nestEggVariable(50000, i, [5, 7, -3, 2, 5, -2, 9, 11, -7, 3, 5, -2, 4, 8, 12, -3, -5, 9, 2,
7])[19] > 1000000:
            print("Smallest savings percentage:",i)
            break
        else:
            pass
    print("\n-----------------------------------------------------------------------")
    print('Problem 4 $1,000,000 success rate:')
    ten30 = monteCarlo(1000,50000,10,30,1000000)['successRate']
    fteen20 = monteCarlo(1000,50000,15,20,1000000)['successRate']
    print("----------------------------------------------------------------------- \nSaving 10% per year
for 30 years:",ten30)
    print('Saving 15% per year for 20 years:',fteen20)
    if fteen20 < ten30:
        print("\n  Therefore you have a higher chance of accumulating over $1000000\nif you save 10% pe
r year for 30 years than if you save 15% per\nyear for 20 years.")
    else:
        print("\n  Therefore you have a higher chance of accumulating over $1000000\nif you save 15% pe
r year for 20 years than if you save 10% per\nyear for 30 years.")
    print("\n-----------------------------------------------------------------------")
    print('Problem 5 $1,000,000 success rate salary increase:')
    print("----------------------------------------------------------------------- ")
    ten25 = monteCarlo(1000,50000,10,25,1000000)['successRate']
    ten252 = monteCarlo2(1000,50000,10,25,2,1000000)['successRate']
    print('Saving 10% per year for 25 years:',ten25)
    print('Saving 10% per year for 25 years salary increase 2%:',ten252)
    print("\n  Therefore you have a ",round((ten252/ten25-1)*100,2),"% better chance of accumulating ov
er\n$1000000 if you have a 2% salary increase.",sep='')
    print("\n-----------------------------------------------------------------------")
    print('Additional Questions:')
    print("-----------------------------------------------------------------------")
    print('1. Saving 5% per year for 30 years median:',monteCarlo(1000,50000,5,30)['med'])
    print('1. Saving 10% per year for 30 years median:',monteCarlo(1000,50000,10,30)['med'])
    print('\n  If you double your savings rate your median final balance about\ndouble the balance from
saving 5%.')
    print('\n2. Saving 5% per year for 15 years median:',monteCarlo(1000,50000,5,15)['med'])
    print('2. Saving 5% per year for 30 years median:',monteCarlo(1000,50000,5,30)['med'])
    print("\n  If you are earning $50,000 per year and save 5% per year for\njust 15 years your median
final balance will be less than\nhalf the balance from saving 5% for 30 years")
    print('\n3a. Saving 5% per year for 30 years median:',monteCarlo(1000,50000,5,30)['med'])
    print('3b. Saving 15% per year for 10 years median:',monteCarlo(1000,50000,15,10)['med'])
    print("\n  You end up with about 3 times as much money when saving 5% per\nyear for 30 years than w
hen saving 15% per year for 10 years.")
```

```
    print("\n4. Salary increase 3%:")
    print('  1. Saving 5% per year for 30 years median:',monteCarlo2(1000,50000,5,30,3)['med'])
    print('  1. Saving 10% per year for 30 years median:',monteCarlo2(1000,50000,10,30,3)['med'])
    print('  2. Saving 5% per year for 15 years median:',monteCarlo2(1000,50000,5,15,3)['med'])
    print('  2. Saving 5% per year for 30 years median:',monteCarlo2(1000,50000,5,30,3)['med'])
    print('  3a. Saving 5% per year for 30 years median:',monteCarlo2(1000,50000,5,30,3)['med'])
    print('  3b. Saving 15% per year for 10 years median:',monteCarlo2(1000,50000,15,10,3)['med'])
    print('''\n  I think that it would be able to catch up but it would be
very hard. The reason is that when saving Saving 5%
per year for 30 years with a 3% salary increase which
is that same as adding more money to catch up. The
result is close to if you saved 10% from the begining
and didn't have an increase of salary.''')
    print("-----------------------------------------------------------------------\n")

main()
'''
    >>> ================================ RESTART ================================
    >>>
    ------------------------------------------------------------------------
    Testing nestEggFixed...

    Nest Egg Fixed Test 1: PASS!
    Nest Egg Fixed Test 2: PASS!
    Nest Egg Fixed Test 3: PASS!
    Nest Egg Fixed Test 4: PASS!

    SUCCESS: testNestEggFixed()
    ------------------------------------------------------------------------
    Testing nestEggVariable...

    Nest Egg Variable Test 1: PASS!
    Nest Egg Variable Test 2: PASS!
    Nest Egg Variable Test 3: PASS!
    Nest Egg Variable Test 4: PASS!

    SUCCESS: testNestEggVariable()

    ------------------------------------------------------------------------
    Problem 1 $1,000,000  7% annual growth:
    ------------------------------------------------------------------------
    Smallest savings percentage: 49

    ------------------------------------------------------------------------
    Problem 2 $1,000,000 growth given by list:
    ------------------------------------------------------------------------
    Smallest savings percentage: 73

    ------------------------------------------------------------------------
    Problem 4 $1,000,000 success rate:
    ------------------------------------------------------------------------
    Saving 10% per year for 30 years: 30.4
    Saving 15% per year for 20 years: 6.3

      Therefore you have a higher chance of accumulating over $1000000
    if you save 10% per year for 30 years than if you save 15% per
    year for 20 years.

    ------------------------------------------------------------------------
    Problem 5 $1,000,000 success rate salary increase:
    ------------------------------------------------------------------------
    Saving 10% per year for 25 years: 8.7
    Saving 10% per year for 25 years salary increase 2%: 12.7
```

```
  Therefore you have a 45.98% better chance of accumulating over
$1000000 if you have a 2% salary increase.

------------------------------------------------------------------------
Additional Questions:
------------------------------------------------------------------------
1. Saving 5% per year for 30 years median: 343504.7592573985
1. Saving 10% per year for 30 years median: 697506.0166511632

  If you double your savings rate your median final balance about
double the balance from saving 5%.

2. Saving 5% per year for 15 years median: 79555.16698353193
2. Saving 5% per year for 30 years median: 323187.1664230658

  If you are earning $50,000 per year and save 5% per year for
just 15 years your median final balance will be less than
half the balance from saving 5% for 30 years

3a. Saving 5% per year for 30 years median: 335976.6714800863
3b. Saving 15% per year for 10 years median: 131075.8536223494

  You end up with about 3 times as much money when saving 5% per
year for 30 years than when saving 15% per year for 10 years.

4. Salary increase 3%:
  1. Saving 5% per year for 30 years median: 458851.7471079695
  1. Saving 10% per year for 30 years median: 935424.4940142343
  2. Saving 5% per year for 15 years median: 95593.04243176272
  2. Saving 5% per year for 30 years median: 465342.4736029893
  3a. Saving 5% per year for 30 years median: 443919.66960981605
  3b. Saving 15% per year for 10 years median: 148592.5573873332

  I think that it would be able to catch up but it would be
very hard. The reason is that when saving Saving 5%
per year for 30 years with a 3% salary increase which
is that same as adding more money to catch up. The
result is close to if you saved 10% from the begining
and didn't have an increase of salary.
------------------------------------------------------------------------
'''
```

```
import turtle

def main():
    randnum = readlist('randomnumbers copy.txt')
    #print(sorted(randnum))
    print('median:',get_median(randnum))
    print('mean:',get_mean(randnum))
    print('high:',get_high(randnum))
    print('low:',get_low(randnum))
    print('mode:', get_mode(randnum))
    print('percentile:', get_percentile(randnum,25))
    print('standard deviation:', standard_d(randnum))
    print('outliers:', outliers(randnum))
    graph(randnum,outliers(randnum),non_outliers(randnum,outliers(randnum)))

def write(graph, text, movement,pd=False):
                graph.up()
                graph.forward(movement)
                graph.write(text/4+50,False, align="center")
                graph.forward(-movement)
                if pd == True:
                        graph.down()

def graph(randnum,outliers,non_outliers):
        wn = turtle.Screen()
        graph = turtle.Turtle()
        graph.ht()
        graph.speed(0)
        graph.left(90)
        Q3 = 4*(get_percentile(randnum,75)-50)
        Q1 = 4*(get_percentile(randnum,25)-50)
        median= 4*(get_median(randnum)-50)
        low = get_low(non_outliers)
        high = get_high(non_outliers)
        height = 17
        graph.up()
        for i in outliers:
                graph.goto(4*(i-50),0)
                graph.dot(3, "black")
                graph.goto(4*(i-50),height+5)
                graph.write(i,False, align="left")
        graph.fillcolor('red')
        graph.goto(low,-height)
        graph.down()
        graph.goto(low,height)
        write(graph, low, 5)
        graph.goto(low,0)
        graph.down()
        graph.goto(Q1,0)
        graph.up()
        graph.goto(median,-height)
        graph.begin_fill()
        graph.down()
        graph.goto(Q1,-height)
        graph.goto(Q1,height)
        write(graph, Q1, 5,True)
        graph.goto(median,height)
        graph.end_fill()
        write(graph, median, 5, True)
        graph.fillcolor('green')
        graph.goto(median,height)
        graph.begin_fill()
        graph.goto(Q3,height)
        write(graph, Q3, 5,True)
```

```
        graph.goto(Q3,-height)
        graph.goto(median,-height)
        graph.goto(median,height)
        graph.end_fill()
        graph.up()
        graph.goto(Q3,0)
        graph.down()
        graph.goto(high,0)
        graph.goto(high,-height)
        graph.goto(high,height)
        write(graph, high, 5)
        for i in range(0,416,16):
                graph.goto(-200+i,-45)
                graph.down()
                graph.goto(-200+i,-35)
                graph.up()
                graph.goto(-200+i,-60)
                graph.write(int(i/4), False, align="center")
        graph.goto(200,-45)
        graph.down()
        graph.goto(-200,-45)


def outliers(randnum):
        outliers = []
        Q3 = get_percentile(randnum,75)
        Q1 = get_percentile(randnum,25)
        IQR = Q3 - Q1
        for num in randnum:
                if num > (1.5*IQR+Q3):
                        outliers.append(num)
                        print(num)
                elif num < (Q1 - 1.5 *IQR):
                        outliers.append(num)
        return outliers

def non_outliers(randnum,outliers):
        non_outliers = []
        for num in randnum:
                if num in outliers:
                        pass
                else:
                        non_outliers.append(4*(num-50))
        return non_outliers

def readlist(file):
    read = open(file,'r')
    numbers = []
    for line in read:
        numbers.append(float(line.strip()))
    return numbers

def get_mode(randnum):
    numbers = {}
    for i in randnum:
        try:
            numbers[i] += 1
        except(KeyError):
            numbers[i] = 1
    nums = numbers.keys()
    max = 0
    for num in nums:
        if numbers[num] > max:
            max = numbers[num]
```

```
    for num in nums:
        if numbers[num] == max:
            return num

def get_high(randnum):
    ahigh = randnum[0]
    for num in randnum:
        if num > ahigh:
            ahigh = num
    return ahigh

def get_low(randnum):
    alow = randnum[0]
    for num in randnum:
        if num < alow:
            alow = num
    return alow

def get_percentile(rand,percent):
    randnum = rand[:]
    for a in range(int(len(randnum)/100.*(100-percent)-1)):
        randnum.remove(get_high(randnum))
    avg = 0
    if len(rand)%2 == False:
        avg += randnum.pop(randnum.index(get_high(randnum)))
        avg += randnum.pop(randnum.index(get_high(randnum)))
        avg = avg/2.
        return avg
    return get_high(randnum)

def get_mean(randnum):
    mean = 0
    for num in randnum:
        mean += num
    return mean/float(len(randnum))

def get_median(rand):
    return get_percentile(rand,50)
    avg = 0
    randnum = rand[:]
    for a in range(int(len(randnum)/2-1)):
        randnum.remove(get_high(randnum))
        randnum.remove(get_low(randnum))
    if len(randnum)%2 == True:
        randnum.remove(get_high(randnum))
        randnum.remove(get_low(randnum))
        for i in randnum:
            avg += i
    else:
        for i in randnum:
            avg += i
        avg = avg/2.
    return avg

def standard_d(numlist):
    mean = get_mean(numlist)
    sigma2 = 0
    for x in numlist:
        sigma2 += (x-mean)**2.
    sigma2 = sigma2/len(numlist)
    sigma = sigma2**(1./2.)
    return sigma
```

```
if __name__ == "__main__":
    main()
```