

CSC-632_Team 3_Project 2

April 26, 2021

1 MS Excel link: <https://drive.google.com/file/d/1oTw5DbHKMq1bhsYdIN>

```
[ ]: import math
class NormalDist:
    def __init__(self, mu=0.0, sigma=1.0):
        "NormalDist where mu is the mean and sigma is the standard deviation."
        if sigma < 0.0:
            print('sigma must be non-negative')
        self._mu = float(mu)
        self._sigma = float(sigma)

    def inv_cdf(self, p):
        if p <= 0.0 or p >= 1.0:
            print('p must be in the range 0.0 < p < 1.0')
        if self._sigma <= 0.0:
            print('cdf() not defined when sigma at or below zero')
        return _normal_dist_inv_cdf(p, self._mu, self._sigma)

def _normal_dist_inv_cdf(p, mu, sigma):
    q = p - 0.5
    if math.fabs(q) <= 0.425:
        r = 0.180625 - q * q
        num = ((((((2.50908_09287_30122_6727e+3 * r + 3.
↪34305_75583_58812_8105e+4) * r + 6.72657_70927_00870_0853e+4) * r + 4.
↪59219_53931_54987_1457e+4) * r + 1.37316_93765_50946_1125e+4) * r + 1.
↪97159_09503_06551_4427e+3) * r + 1.33141_66789_17843_7745e+2) * r + 3.
↪38713_28727_96366_6080e+0) * q
        den = ((((((5.22649_52788_52854_5610e+3 * r + 2.
↪87290_85735_72194_2674e+4) * r + 3.93078_95800_09271_0610e+4) * r + 2.
↪12137_94301_58659_5867e+4) * r + 5.39419_60214_24751_1077e+3) * r + 6.
↪87187_00749_20579_0830e+2) * r + 4.23133_30701_60091_1252e+1) * r + 1.0)
        x = num / den
        return mu + (x * sigma)
    r = p if q <= 0.0 else 1.0 - p
    r = math.sqrt(-math.log(r))
    if r <= 5.0:
```

```

    r = r - 1.6
    num = ((((((7.74545_01427_83414_07640e-4 * r + 2.
↪27238_44989_26918_45833e-2) * r + 2.41780_72517_74506_11770e-1) * r + 1.
↪27045_82524_52368_38258e+0) * r + 3.64784_83247_63204_60504e+0) * r + 5.
↪76949_72214_60691_40550e+0) * r + 4.63033_78461_56545_29590e+0) * r + 1.
↪42343_71107_49683_57734e+0)
    den = ((((((1.05075_00716_44416_84324e-9 * r + 5.
↪47593_80849_95344_94600e-4) * r + 1.51986_66563_61645_71966e-2) * r + 1.
↪48103_97642_74800_74590e-1) * r + 6.89767_33498_51000_04550e-1) * r + 1.
↪67638_48301_83803_84940e+0) * r + 2.05319_16266_37758_82187e+0) * r + 1.0)
    else:
        r = r - 5.0
        num = ((((((2.01033_43992_92288_13265e-7 * r + 2.
↪71155_55687_43487_57815e-5) * r + 1.24266_09473_88078_43860e-3) * r + 2.
↪65321_89526_57612_30930e-2) * r + 2.96560_57182_85048_91230e-1) * r + 1.
↪78482_65399_17291_33580e+0) * r + 5.46378_49111_64114_36990e+0) * r + 6.
↪65790_46435_01103_77720e+0)
        den = ((((((2.04426_31033_89939_78564e-15 * r + 1.
↪42151_17583_16445_88870e-7) * r + 1.84631_83175_10054_68180e-5) * r + 7.
↪86869_13114_56132_59100e-4) * r + 1.48753_61290_85061_48525e-2) * r + 1.
↪36929_88092_27358_05310e-1) * r + 5.99832_20655_58879_37690e-1) * r + 1.0)
    x = num / den
    if q < 0.0:
        x = -x
    return mu + (x * sigma)

```

2 Q1

```

[ ]: import statistics
import numpy as np
from scipy import stats

def interval(p, N, S, mu):
    ta = -NormalDist(mu=0, sigma=1).inv_cdf((1-p)/2)
    print((mu - ta * S/(N ** .5)), 'to', (mu + ta * S/(N ** .5)))

def main(NumCarDemandF):

    # B
    best = 0
    numb = 0

    profitBest = []

    # find the optimal solution
    for NumOrderdCar in range(20, 40+1):

```

```

profit = 0
profitS = []
for loop in range(10000):
    NumCarDemand = NumCarDemandF()
    carsSoldBestPrice = np.min([NumOrderdCar, NumCarDemand])
    carsSoldWostprice = NumCarDemand - carsSoldBestPrice
    carsSoldLossPrice = np.max([0, NumOrderdCar - NumCarDemand])

    Lprofit = (
        (carsSoldBestPrice * 5) +
        (carsSoldWostprice * 3) -
        (carsSoldLossPrice * 1)
    )

    profitS.append(Lprofit)

    profit += Lprofit

profit = profit/10000

if(profit > best):
    best = profit
    numb = NumOrderdCar
    profitBest = profitS

print(NumOrderdCar, profit * 1000, numb)

# c
profitBest = profitBest[0:1000]
print(statistics.pstdev(profitBest), statistics.mean(profitBest))

interval(.95, 1000, statistics.pstdev(profitBest), statistics.
↪mean(profitBest))

# D
profitBest = profitBest[0:30]
oneSampleTtest = stats.ttest_1samp(profitBest, 30)
print(oneSampleTtest)
oneSidedPvalue = (oneSampleTtest.pvalue/2)
print("The p-value is", oneSidedPvalue)

# 97% confidence interval
alpha = 0.03

# state the hypothesis test
if oneSidedPvalue <= alpha:

```

```

    print("With the small p-value, We have 97% confident to reject the null_
↪hypothesis")
else:
    print("With the large p-value, We have 97% confident fail to reject the_
↪null hypothesis")

```

```

[ ]: # B
main(lambda: np.random.choice(
    [20, 25, 30, 35, 40], p=[0.3, 0.15, 0.15, 0.2, 0.2]))

```

```

20 127343.5 20
21 128958.9 21
22 129667.1 22
23 131371.0 23
24 132117.6 24
25 133343.0 25
26 134069.5 26
27 134837.1 27
28 134897.9 28
29 136003.3 29
30 136131.0 30
31 137051.3 31
32 136566.4 31
33 137372.4 33
34 137604.19999999998 34
35 137455.0 34
36 136470.0 34
37 137133.19999999998 34
38 135489.4 34
39 136143.6 34
40 135536.0 34
40.27406113120453 137
134.50383369801492 to 139.49616630198508
Ttest_1sampResult(statistic=14.089221249955889, pvalue=1.6709097688014774e-14)
The p-value is 8.354548844007387e-15
With the small p-value, We have 97% confident to reject the null hypothesis

```

```

[ ]: # D
main(lambda: int(np.random.normal(30, 7)))

```

```

20 127462.59999999999 20
21 129327.59999999999 21
22 130749.2 22
23 132465.4 23
24 133873.5 24
25 135102.8 25
26 136021.3 26

```

```

27 137020.800000000002 27
28 137937.800000000002 28
29 139112.8 29
30 138866.4 29
31 139673.300000000002 31
32 140159.8 32
33 139785.0 32
34 139264.7 32
35 138746.800000000002 32
36 139494.300000000002 32
37 138829.4 32
38 137280.4 32
39 137544.3 32
40 136112.599999999998 32
34.88552708502482 139
136.83781236133754 to 141.16218763866246
Ttest_1sampResult(statistic=14.128396184315568, pvalue=1.556995000986134e-14)
The p-value is 7.78497500493067e-15
With the small p-value, We have 97% confident to reject the null hypothesis

```

#Problem 2

Six months before its annual convention, The Computer Science Conference (CSC) must determine how many rooms to reserve. CSC believes the number of people attending the convention will be normally distributed with a mean of 5000 and a standard deviation of 1000.(Clearly, this demand must be rounded to an integer since only whole people attend the conference.) At this time, CSC can reserve rooms at a cost of \$50 per room. CSC can reserve up to 8000 rooms, but it must reserve rooms in blocks of 100. CSC must pay the \$50 room cost even if the room is not occupied. If the number of people attending the convention exceeds the number of rooms reserved, extra rooms must be reserved at a cost of \$80 per room. (Do the following steps by hand or using MS Excel, and run the simulation using Python and include the .py file in the submission folder.)

3 a) Develop a simulation model for this problem. You can use MS Excel to depict the model.

4 Excel Link:

<https://docs.google.com/spreadsheets/d/1Ixzctwm52LUKEqy2GMHFIvFayYj4lwwON9eWs3r5q4E/edit?usp=sha>

- 5 b) Using the trial-and-error method, determine the number of rooms that should be reserved to minimize the expected cost.

6 Initialization

```
[ ]: import math
import random
import statistics
import numpy as np
import seaborn as sns
from scipy import stats
```

7 Parameters

```
[ ]: NumberMaxGuests = 8001
DesiredMarginError20 = 20#####We assume the desire margin
↳error is 20.
DesiredMarginError25 = 25#####We assume the desire margin
↳error is 25.
DesiredMarginError30 = 30#####We assume the desire margin
↳error is 30.
DesiredMarginError40 = 40#####We assume the desire margin
↳error is 40.
NumberRoomMeanValue = 5000
NumberRoomStandardDeviation = 1000
ConfidenceIntervalQd = 0.93
LowRoomCost = 50
HighRoomCost = 80
ReserveRoomsBlock = 100

MiniEntireNumberRoom = 0

LowNumberRoom = 0
LowRoomAdvance = 0#

NormalNumberRoom = 0
NormalRoomAdvance = 0#
NotNormalNumberRoom = 0
NotNormalRoomAdvance = 0#

AllNumberRoom = 0#b)I
ExceedingNumberRoom = 0#b)I

NormalExceedingNumberRoom = 0
NormalExceedingRoomAdvance = 0#
NotNormalExceedingNumberRoom = 0
```

```
NotNormalExceedingRoomAdvance = 0#
```

#b) (ALL) This part could show all the results of trial-and-error method when we assume that .95 confidence and 20 as desire margin error.

```
[ ]: # We use 95% as the confidence interval from c) and we assume that the desired
    ↪margin error would be 20, 30, and 40.
def TrialAndErrorNumberRoom():
    # .95 confidence -> Z(/2) equal to 1.96
    MiniEntireNumberRoom = math.ceil((1.96 * 1.96) * (NumberRoomStandardDeviation
    ↪* NumberRoomStandardDeviation) / (DesiredMarginError20 *
    ↪DesiredMarginError20))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i%ReserveRoomsBlock
        b = MiniEntireNumberRoom - i
        c = i//ReserveRoomsBlock
        if (i <= NumberRoomMeanValue):
            if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
            ↪so I seperate it
                LowNumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error
            ↪is 20." %(MiniEntireNumberRoom))
                else:
                    print("The appropriate sample size is Not less than the room blocks")
                    pass
                else:
                    if (a == 0):#To identify whether the num of people over the minimum
            ↪block
                        NormalNumberRoom = c * ReserveRoomsBlock
                        if (b == 0):
                            print("The appropriate sample size is:%.1f when desired margin error
            ↪is 20." %(MiniEntireNumberRoom))
                        else:
                            print("The appropriate sample size is between the room blocks and
            ↪the mean number of guests, but it is an integer multiple of room blocks")
                            pass
                        else:
                            NotNormalNumberRoom = (c + 1) * ReserveRoomsBlock #Room number should
            ↪+100 if >=1 person come to the conference
                            if (b == 0):
                                print("The appropriate sample size is:%.1f when desired margin error
            ↪is 20." %(MiniEntireNumberRoom))
                            else:
```

```

        print("The appropriate sample size is between the room blocks and
↳the mean number of guests and is Not an integer multiple of room blocks")
        pass
    else:
        if (a == 0):
            NormalExceedingNumberRoom = c * ReserveRoomsBlock
            #print(ExceedingNumberRoom)
            if (b == 0):
                print("The appropriate sample size is: %.1f when desired margin error
↳is 20." %(MiniEntireNumberRoom))
            else:
                print("The appropriate sample size is over the mean number of guests
↳and is an integer multiple of room blocks")
                pass
            else:
                NotNormalExceedingNumberRoom = (c + 1) * ReserveRoomsBlock
                if (b == 0):
                    print("The appropriate sample size is: %.1f when desired margin error
↳is 20." %(MiniEntireNumberRoom))
                else:
                    print("The appropriate sample size is over the mean number of guests
↳and is not an integer multiple of room blocks")
                    pass
        TrialAndErrorNumberRoom()

```

Streaming output truncated to the last 5000 lines.

The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks
 The appropriate sample size is between the room blocks and the mean number of
 guests and is Not an integer multiple of room blocks

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

[illegible]

[illegible]

The appropriate sample size is over the mean number of guests and is not an integer multiple of room blocks

[illegible]

[illegible]

8 b) I

The desired margin error would be 20

```
[ ]: # We use 95% as the confidence interval from c) and we assume that the desired
    ↪ margin error would be 20, 30, and 40.
def TrialAndErrorNumberRoom():
    # .95 confidence -> Z(/2) equal to 1.96
    MiniEntireNumberRoom = math.ceil((1.96 * 1.96) * (NumberRoomStandardDeviation
    ↪ * NumberRoomStandardDeviation) / (DesiredMarginError20 *
    ↪ DesiredMarginError20))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i%ReserveRoomsBlock
        b = MiniEntireNumberRoom - i
        c = i//ReserveRoomsBlock
        if (i <= NumberRoomMeanValue):
            if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
    ↪ so I separete it
                AllNumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error
    ↪ is 20." %(MiniEntireNumberRoom))
                else:
                    if (a == 0):#To identify whether the num of people over the minimum
    ↪ block
                        AllNumberRoom = c * ReserveRoomsBlock
                        if (b == 0):
                            print("The appropriate sample size is:%.1f when desired margin error
    ↪ is 20." %(MiniEntireNumberRoom))
                        else:
                            AllNumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100
    ↪ if >=1 person come to the conference
                            if (b == 0):
                                print("The appropriate sample size is:%.1f when desired margin error
    ↪ is 20." %(MiniEntireNumberRoom))
                            else:
                                if (a == 0):
                                    ExceedingNumberRoom = c * ReserveRoomsBlock
                                    #print(ExceedingNumberRoom)
                                    if (b == 0):
                                        print("The appropriate sample size is:%.1f when desired margin error
    ↪ is 20." %(MiniEntireNumberRoom))
                                    else:
                                        ExceedingNumberRoom = (c + 1) * ReserveRoomsBlock
```

```

        if (b == 0):
            print("The appropriate sample size is:%.1f when desired margin_
↳error is 20." %(MiniEntireNumberRoom))

TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 9604.

Warning: The appropriate room size is over acceptable capability.

9 b) II

Using the trial-and-error method, determine the number of rooms that should be reserved to minimize the expected cost.

The desired margin error would be 25

```

[ ]: # We use 95% as the confidence interval from c) and we assume that the desired_
↳margin error would be 20, 30, and 40.
def TrialAndErrorNumberRoom():
    # .95 confidence -> Z(/2) equal to 1.96
    MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation *
↳NumberRoomStandardDeviation) / (DesiredMarginError25 * DesiredMarginError25))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i % ReserveRoomsBlock
        b = MiniEntireNumberRoom - i
        c = i // ReserveRoomsBlock
        if (i <= NumberRoomMeanValue):
            if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
↳so I sepearte it
                AllNumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error_
↳is 25." %(MiniEntireNumberRoom))
                else:
                    pass
            else:
                if (a == 0):#To identify whether the num of people over the minimum_
↳block
                    AllNumberRoom = c * ReserveRoomsBlock
                    if (b == 0):
                        print("The appropriate sample size is:%.1f when desired margin error_
↳is 25." %(MiniEntireNumberRoom))
                        #print(AllNumberRoom)
                    if (b == 0):

```

```

        print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
    else:
        AllNumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100
↳if >=1 person come to the conference
        if (b == 0):
            print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
        else:
            if (a == 0):
                ExceedingNumberRoom = c * ReserveRoomsBlock
                #print(ExceedingNumberRoom)
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
                else:
                    ExceedingNumberRoom = (c + 1) * ReserveRoomsBlock
                    if (b == 0):
                        print("The appropriate sample size is:%.1f when desired margin
↳error is 25." %(MiniEntireNumberRoom))

TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 6147.

The appropriate sample size is:6147.0 when desired margin error is 25.

10 b) III

Using the trial-and-error method, determine the number of rooms that should be reserved to minimize the expected cost.

The desired margin error would be 30

```

[ ]: # We use 95% as the confidence interval from c) and we assume that the desired
↳margin error would be 20, 30, and 40.
def TrialAndErrorNumberRoom():
    # .95 confidence -> Z(/2) equal to 1.96
    MiniEntireNumberRoom = math.ceil((1.96 * 1.96) * (NumberRoomStandardDeviation
↳* NumberRoomStandardDeviation) / (DesiredMarginError30 *
↳DesiredMarginError30))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i % ReserveRoomsBlock
        b = MiniEntireNumberRoom - i
        c = i // ReserveRoomsBlock

```

```

    if (i <= NumberRoomMeanValue):
        if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
→so I seperate it
            AllNumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
            if (b == 0):
                print("The appropriate sample size is:%.1f when desired margin error,
→is 30." %(MiniEntireNumberRoom))
            else:
                pass
        else:
            if (a == 0):#To identify whether the num of people over the minimum
→block
                AllNumberRoom = c * ReserveRoomsBlock
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error,
→is 30." %(MiniEntireNumberRoom))
                    #print(AllNumberRoom)
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error,
→is 30." %(MiniEntireNumberRoom))
                else:
                    AllNumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100
→if >=1 person come to the conference
                    if (b == 0):
                        print("The appropriate sample size is:%.1f when desired margin error,
→is 30." %(MiniEntireNumberRoom))
                    else:
                        if (a == 0):
                            ExceedingNumberRoom = c * ReserveRoomsBlock
                            #print(ExceedingNumberRoom)
                        if (b == 0):
                            print("The appropriate sample size is:%.1f when desired margin error,
→is 30." %(MiniEntireNumberRoom))
                        else:
                            ExceedingNumberRoom = (c + 1) * ReserveRoomsBlock
                            if (b == 0):
                                print("The appropriate sample size is:%.1f when desired margin,
→error is 30." %(MiniEntireNumberRoom))

TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 4269.

The appropriate sample size is:4269.0 when desired margin error is 30.

11 b) IV

Using the trial-and-error method, determine the number of rooms that should be reserved to minimize the expected cost.

The desired margin error would be 40

```
[ ]: # We use 95% as the confidence interval from c) and we assume that the desired
    ↪margin error would be 20, 30, and 40.
def TrialAndErrorNumberRoom():
    # .95 confidence -> Z(/2) equal to 1.96
    MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation *
    ↪NumberRoomStandardDeviation) / (DesiredMarginError40 * DesiredMarginError40))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i%ReserveRoomsBlock
        b = MiniEntireNumberRoom - i
        c = i // ReserveRoomsBlock
        if (i <= NumberRoomMeanValue):
            if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
            ↪so I seperate it
                AllNumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
                if (b == 0):
                    print("The appropriate sample size is:%.1f when desired margin error
            ↪is 40." %(MiniEntireNumberRoom))
                else:
                    pass
            else:
                if (a == 0):#To identify whether the num of people over the minimum
            ↪block
                    AllNumberRoom = c * ReserveRoomsBlock
                    if (b == 0):
                        print("The appropriate sample size is:%.1f when desired margin error
            ↪is 40." %(MiniEntireNumberRoom))
                        #print(AllNumberRoom)
                    if (b == 0):
                        print("The appropriate sample size is:%.1f when desired margin error
            ↪is 40." %(MiniEntireNumberRoom))
                    else:
                        AllNumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100
            ↪if >=1 person come to the conference
                        if (b == 0):
                            print("The appropriate sample size is:%.1f when desired margin error
            ↪is 40." %(MiniEntireNumberRoom))
                        else:
```

```

if (a == 0):
    ExceedingNumberRoom = c * ReserveRoomsBlock
    #print(ExceedingNumberRoom)
    if (b == 0):
        print("The appropriate sample size is:%.1f when desired margin error is 40." %(MiniEntireNumberRoom))
    else:
        ExceedingNumberRoom = (c + 1) * ReserveRoomsBlock
        if (b == 0):
            print("The appropriate sample size is:%.1f when desired margin error is 40." %(MiniEntireNumberRoom))

TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 2401.

The appropriate sample size is:2401.0 when desired margin error is 40.

12 c) For your optimal order quantity in part b, provide a 95% confidence interval for the expected cost.

We choose 25 as the desire margin error.

```

[ ]: import math
NumberMaxGuests = 8001
DesiredMarginError20 = 20#####We assume the desire margin error is 20.
DesiredMarginError25 = 25#####We assume the desire margin error is 25.
DesiredMarginError30 = 30#####We assume the desire margin error is 30.
DesiredMarginError40 = 40#####We assume the desire margin error is 40.
NumberRoomMeanValue = 5000
NumberRoomStandardDeviation = 1000

LowRoomCost = 50
HighRoomCost = 80
ReserveRoomsBlock = 100

MiniEntireNumberRoom = 0
NumberRoom = 0
RoomAdvance = 0

def TrialAndErrorNumberRoom():
    # 95% confidence -> Z(/2) equal to 1.96

```

```

#MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation_
↳ * NumberRoomStandardDeviation) / (DesiredMarginError20 * _
↳ DesiredMarginError20))
MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation * _
↳ NumberRoomStandardDeviation) / (DesiredMarginError25 * DesiredMarginError25))
#MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation_
↳ * NumberRoomStandardDeviation) / (DesiredMarginError30 * _
↳ DesiredMarginError30))
#MiniEntireNumberRoom = math.ceil((1.96 * 1.96)*(NumberRoomStandardDeviation_
↳ * NumberRoomStandardDeviation) / (DesiredMarginError40 * _
↳ DesiredMarginError40))
print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
if (MiniEntireNumberRoom > NumberMaxGuests):
    print("Warning: The appropriate room size is over acceptable capability.")
for i in range(NumberMaxGuests):
    a = i % ReserveRoomsBlock #To
    b = MiniEntireNumberRoom - i # To identify the minimum number of room equal_
↳ to the number in the whole list when b == 0.
    c = i // ReserveRoomsBlock
    if (i <= NumberRoomMeanValue):
        if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,_
↳ so I seperate it
            NumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
            if (b == 0):
                RoomAdvance = ReserveRoomsBlock * LowRoomCost
                print("The appropriate sample size is:%.1f when desired margin error_
↳ is 25." %(MiniEntireNumberRoom))
                print(".95 confidence interval for the expected cost is: %d." _
↳ %(LowRoomAdvance))
            else:
                pass
        else:
            if (a == 0):#To identify whether the num of room need to add 100 or not.
                NumberRoom = c * ReserveRoomsBlock
                if (b == 0):
                    RoomAdvance = NumberRoom * LowRoomCost
                    print("The appropriate sample size is:%.1f when desired margin error_
↳ is 25." %(MiniEntireNumberRoom))
                    print(".95 confidence interval for the expected cost is: %d." _
↳ %(NormalRoomAdvance))
                else:
                    pass
            else:
                NumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100 if_
↳ >=1 person come to the conference
                if (b == 0):

```

```

        RoomAdvance = NumberRoom * LowRoomCost
        print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
        print(".95 confidence interval for the expected cost is: %d."
↳%(NotNormalRoomAdvance))
    else:
        pass
else:
    if (a == 0):
        NumberRoom = c * ReserveRoomsBlock
        if (b == 0):
            RoomAdvance = LowRoomCost*NumberRoomMeanValue + (NumberRoom -
↳NumberRoomMeanValue) * HighRoomCost
            print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
            print(".95 confidence interval for the expected cost is: %d."
↳%(RoomAdvance))
        else:
            pass
    else:
        NumberRoom = (c + 1) * ReserveRoomsBlock
        if (b == 0):
            RoomAdvance = LowRoomCost * NumberRoomMeanValue + (NumberRoom -
↳NumberRoomMeanValue) * HighRoomCost
            print("The appropriate sample size is:%.1f when desired margin error
↳is 25" %(MiniEntireNumberRoom))
            print(".95 confidence interval for the expected cost is: %.1f."
↳%(RoomAdvance))
        else:
            pass
TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 6147.

The appropriate sample size is:6147.0 when desired margin error is 25

.95 confidence interval for the expected cost is: 346000.0.

- d) CSC would like to know if the expected cost exceeds \$300,000. Conduct a statistical analysis to answer the question with 93% confidence coefficient. Clearly determine the null and alternative hypothesis.

```

[ ]: import math
NumberMaxGuests = 8001
DesiredMarginError20 = 20#####We assume the desire margin
↳error is 20.
DesiredMarginError25 = 25#####We assume the desire margin
↳error is 25.

```



```

DesiredMarginError30 = 30#####We assume the desire margin
↳error is 30.
DesiredMarginError40 = 40#####We assume the desire margin
↳error is 40.
NumberRoomMeanValue = 5000
NumberRoomStandardDeviation = 1000

LowRoomCost = 50
HighRoomCost = 80
ReserveRoomsBlock = 100

MiniEntireNumberRoom = 0
NumberRoom = 0
RoomAdvance = 0

def TrialAndErrorNumberRoom():
    # 93% confidence -> Z(/2) equal to 2.17
    MiniEntireNumberRoom = math.ceil((2.17 * 2.17) * (NumberRoomStandardDeviation
↳* NumberRoomStandardDeviation) / (DesiredMarginError25 *
↳DesiredMarginError25))
    print("Appropriate number of rooms is: %d." %(MiniEntireNumberRoom))
    if (MiniEntireNumberRoom > NumberMaxGuests):
        print("Warning: The appropriate room size is over acceptable capability.")
    for i in range(NumberMaxGuests):
        a = i % ReserveRoomsBlock #To
        b = MiniEntireNumberRoom - i # To identify the minimum number of room equal
↳to the number in the whole list when b == 0.
        c = i // ReserveRoomsBlock
        if (i <= NumberRoomMeanValue):
            if (i <= ReserveRoomsBlock):#The num of people <=100 is a special part,
↳so I seperate it
                NumberRoom = ReserveRoomsBlock#100 rooms is the minimum block
            if (b == 0):
                RoomAdvance = ReserveRoomsBlock * LowRoomCost
                print("The appropriate sample size is:%.1f when desired margin error
↳is 25." %(MiniEntireNumberRoom))
                print(".95 confidence interval for the expected cost is: %d."
↳%(RoomAdvance))
            if (RoomAdvance>3000000):
                print("The expected cost exceeds $3000000.")
            else:
                pass
        else:
            pass
    else:
        if (a == 0):#To identify whether the num of room need to add 100 or not.

```

```

        NumberRoom = c * ReserveRoomsBlock
        if (b == 0):
            RoomAdvance = NormalNumberRoom * LowRoomCost
            print("The appropriate sample size is:%.1f when desired margin error_
→is 25." %(MiniEntireNumberRoom))
            print(".95 confidence interval for the expected cost is: %d."_
→%(RoomAdvance))
            if (RoomAdvance>300000):
                print("The expected cost exceeds $3000000.")
            else:
                pass
        else:
            pass
    else:
        NumberRoom = (c + 1) * ReserveRoomsBlock #Room number should +100 if_
→>=1 person come to the conference
        if (b == 0):
            RoomAdvance = NumberRoom * LowRoomCost
            print("The appropriate sample size is:%.1f when desired margin error_
→is 25." %(MiniEntireNumberRoom))
            print(".95 confidence interval for the expected cost is: %d."_
→%(RoomAdvance))
            if (RoomAdvance>300000):
                print("The expected cost exceeds $3000000.")
            else:
                pass
        else:
            pass
    else:
        if (a == 0):
            NumberRoom = c * ReserveRoomsBlock
            if (b == 0):
                RoomAdvance = LowRoomCost * NumberRoomMeanValue + (NumberRoom -_
→NumberRoomMeanValue) * HighRoomCost
                print("The appropriate sample size is:%.1f when desired margin error_
→is 25." %(MiniEntireNumberRoom))
                print(".95 confidence interval for the expected cost is: %d."_
→%(RoomAdvance))
                if (RoomAdvance>300000):
                    print("The expected cost exceeds $3000000.")
                else:
                    pass
            else:
                pass
        else:
            NumberRoom = (c + 1) * ReserveRoomsBlock

```

```

    if (b == 0):
        RoomAdvance = LowRoomCost * NumberRoomMeanValue + (NumberRoom -
        ↪NumberRoomMeanValue) * HighRoomCost
        print("The appropriate sample size is:%.1f when desired margin error
        ↪is 25 and .93 confidence interval for the expected cost is: %.1f."
        ↪%(MiniEntireNumberRoom, RoomAdvance))
        if (RoomAdvance>3000000):
            print("The expected cost exceeds $3000000.")
        else:
            pass
    else:
        pass
TrialAndErrorNumberRoom()

```

Appropriate number of rooms is: 7535.
 The appropriate sample size is:7535.0 when desired margin error is 25 and .93
 confidence interval for the expected cost is: 458000.0.
 The expected cost exceeds \$3000000.

13 The null and alternative hypothesis.

$$\{H_0 : > 300,000\}$$

$$\{H_1 : \leq 300,000\}$$

According to 93% confidence coefficient. The level of significance = .07

$$P - \text{value } 0.0556173352$$

With the large p-value, We have 93% confident fail to reject the null hypothesis

14 Problem 3

14.1 Author: Yunting Chiu

14.2 email: yc6705a@american.edu

A ticket from Indianapolis to Orlando on Deleat Airlines sells for \$150. The plane can hold 100 people. It costs \$8000 to fly an empty plane. The airline incurs variable costs of \$30 (food and fuel) for each person on the plane. If the flight is overbooked, anyone who cannot get a seat receives \$300 in compensation. Not everyone who has a reservation shows up for the flight. The number of people with reservations who actually show up for the flight is binomial with $p = 0.95$. (Do the following steps by hand or using MS Excel, and run the simulation using Python and include the .py file in the submission folder).

```

[ ]: # install the required libraries
import math
from scipy import stats

```

```
import statistics
```

15 3a

- a) To maximize expected profit, how many reservations for the flight should be taken by Deleast?
- The maximum profit can be obtained by selling 105 flight tickets.

```
[ ]: ticketPrice = 150
planeCapacity = 100
fixedFlightCost = 8000
variableFlightCostPerPerson = 30
overBookCostPerPerson = 300
chanceOfShowingUp = 0.95
# revenueEach = []
profitEach = []

# Let us simulate the number of tickets sold from 100 to 129. That is we
↳simulate 30 samples only
NumberOfTicketsSoldEnd = 129
NumberOfTicketsSold = 100
dict = {}
while NumberOfTicketsSold <= NumberOfTicketsSoldEnd:
    passengersArrived = math.ceil(NumberOfTicketsSold * chanceOfShowingUp) #
↳professor used `roundup` in MS excel
    passengersBoardingThePlane = min(passengersArrived, planeCapacity)
    if passengersArrived > 100:
        numberOfRebooking = passengersArrived - passengersBoardingThePlane
    else:
        numberOfRebooking = 0
    costToDeleast = fixedFlightCost + variableFlightCostPerPerson *
↳passengersBoardingThePlane + numberOfRebooking * overBookCostPerPerson
    revenue = NumberOfTicketsSold * ticketPrice
    profit = revenue - costToDeleast
    #revenueEach.append(revenue)
    profitEach.append(profit)
    dict[NumberOfTicketsSold] = profit # save the NumberOfTicketsSold and profit
↳as keys and values
    NumberOfTicketsSold += 1
maxProfit = max(profitEach)

# find out the nest number of tickets sold
for i, j in dict.items():
    if j == maxProfit:
        print("The maximum profit can be obtained by selling {} flight tickets.".
↳format(i))
print("We have {} sample sizes for the simulation".format(len(dict)))
```

The maximum profit can be obtained by selling 105 flight tickets.
We have 30 sample sizes for the simulation

16 3b

- b) What is the associated expected profit for this number of reservations?
- \$4750

```
[ ]: print("The associated expected profit for this number of reservations is {}_\n\n↪dollars".format(maxProfit))
```

The associated expected profit for this number of reservations is 4750 dollars

17 3C

- c) Provide a Hypothesis test with $\alpha=0.05$ for the associated expected profit (from part a).

17.1 Hypothesis Test (t test: with 30 samples)

- H_0 : profit = 4750
- H_a : profit \neq 4750

17.2 Interpretation

With the small p-value 1.3161944583304568e-08, we have evidence to reject the null hypothesis, meaning that the the expected profit is not equal to \$4750 with 95 % confidence interval.

```
[ ]: oneSampleTtest = stats.ttest_1samp(profitEach, 4750)
print(oneSampleTtest)
```

Ttest_1sampResult(statistic=-7.804797247267218, pvalue=1.3161944583304568e-08)

```
[ ]: # for small samples (<50) we use t-statistics
oneSampleTtest = stats.ttest_1samp(profitEach, 4750)
print(oneSampleTtest) # [0] is a T-stat, [1] is a p-value

# 95% confidence interval
alpha = 0.05

# state the hypothesis test
if oneSampleTtest[1] <= alpha:
    print("With the small p-value, We have 95 % confident to reject the null_\n\n↪hypothesis")
else:
```

```
print("With the large p-value, We have 95 % confident fail to reject the null_
↪hypothesis")
```

```
Ttest_1sampResult(statistic=-7.804797247267218, pvalue=1.3161944583304568e-08)
```

With the small p-value, We have 95 % confident to reject the null hypothesis

18 3E

- e) Deleat Airlines would like to know if the expected profit is bellow \$5000. Conduct a statistical analysis to answer the question with 97% confidence coefficient. Clearly determine the null and alternative hypothesis.

18.1 Hypothesis Test

- H_0 : profit < 5000
- H_a : profit \geq 5000

18.2 Interpretation

With the large p-value 0.999999997570608, we fail to reject the null hypothesis, meaning that the the expected profit is not bellow \$5000 with 97 % confidence interval.

19 Reference:

<https://towardsdatascience.com/one-tailed-or-two-tailed-test-that-is-the-question-1283387f631c>

```
[ ]: # for small samples (<50) we use t-statistics

# print(profitEach)
oneSampleTtest = stats.ttest_1samp(profitEach, 5000)
oneSidedPvalue = 1 - (oneSampleTtest.pvalue/2) # alternative = "greater"
print("The p-value is", oneSidedPvalue)

# 97% confidence interval
alpha = 0.03

# state the hypothesis test
if oneSidedPvalue <= alpha:
    print("With the small p-value, We have 97 % confident to reject the null_
↪hypothesis")
else:
    print("With the large p-value, We have 97 % confident fail to reject the null_
↪hypothesis")
# print(profitEach)
```

The p-value is 0.999999997570608

With the large p-value, We have 97 % confident fail to reject the null hypothesis