ANL252

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

TMA01

Z1970078

Muhammad Syairazi Bin Mahdhar

14 Aug 2021

**Question 1(a)**

***Code Start***

# Importing the Math package.

import math

***Code End***

**Question 1(b)**

***Code Start***

# The User will be asked to input the Mean.

# Inputs accepted: any number that is between –∞ and +∞.

# If the input is non-numeric, the user will be prompted to re-enter.

# If ENTER is pressed without any input, the Mean value will be set to 0.

# If the input is accepted, the program prints the value to inform the User of their Mean.

valid\_input = False

while valid\_input == False:

try:

mean = float(input("Please enter the Mean (any number between –∞ and +∞): ") or "0")

except ValueError:

print("Your input is not numeric. Please try again.")

else:

valid\_input = True

print(f'Thank you. Your Mean is {mean}.')

# The User will be asked to input the Variance.

# Inputs accepted: any number that is larger than 0.

# If the input is non-numeric, a negative number or 0 number, the user will be prompted to re-enter.

# If ENTER is pressed without any input, the Variance value will be set to 1.

# If the input is accepted, the program prints the value to inform the User of their Variance.

valid\_input = False

while valid\_input == False:

try:

variance = float((input("Please enter the Variance (any number larger than 0): ") or "1"))

except ValueError:

print("Your input is not numeric. Please try again.")

else:

if variance <= 0.0:

print("Your input must be larger than 0. Please try again.")

else:

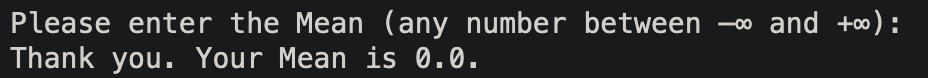
valid\_input = True

print(f'Thank you. Your Variance is {variance}.')

***Code End***

**Outputs**:

**For the Mean, if ENTER is pressed without any input:**

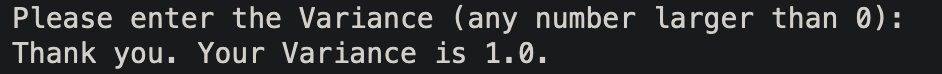


**For the Mean, for all the other values:**

Graphical user interface, text

Description automatically generated

**For the Variance, if ENTER is pressed without any input:**



**For the Variance, for all the other values:**

Text

Description automatically generated

**Question 1(c)**

***Code Start***

# The User will be asked to input x.

# If the input is non-numeric, the user will be prompted to re-enter.

# If the input is accepted, the program prints the value to inform the User of their X value.

valid\_input = False

while valid\_input == False:

try:

x = float((input("Please enter the value of X (any number between –∞ and +∞): ")))

except ValueError:

print("Your input is not numeric. Please try again.")

else:

valid\_input = True

print(f'Thank you. Your X is {x}.')

***Code End***

**Output:**

Text

Description automatically generated

**Question 1(d)**

***Code Start***

# Constructing a user-defined function, based on the probability density function, to compute the probability density.

# Let x: the Parameter for x.

# Let m: the Parameter for Mean.

# Let v: the Parameter for Variance.

# Let fx: the evaluated Probability Density, with the user's previous inputs for Mean, Variance and x as Arguments.

def prob\_density(x, m, v):

return (1/math.sqrt(2\*math.pi\*v))\*math.exp(-(x-m)\*\*2/(2\*v))

fx = prob\_density(x, mean, variance)

***Code End***

**Question 1(e)**

***Code Start***

# The User will see the display for the value of the Probability Density (fx) that was computed using their input values for Mean, Variance and x.

# The fx value will be rounded off to 4 decimal places.

print(f'Your Probability Density value is {fx:.4f} (for X = {x}, Mean = {mean}, Variance = {variance})')

***Code End***

**Output:**

**The output is based on the User Input values of Mean = 5 and Variance = 4**



**Question 1(f)**

***Code Start***

# Constructing a program to compute the Cumulative Distribution Function, P(X <= k).

# Let k be the Parameter for x, m be the Parameter for Mean, and v be the Parameter for Variance.

# Let alpha = 0.001, hard-coded as the step-size.

# Let a = -1000, hard-coded as the lower boundary, a value close to minus infinity.

# Let n be the approximate number of steps between a and x, to the nearest integer. The value of n will change depending on alpha.

# Let fx be the Probability Density Function from (d). If k = x, the first function defined is fx(k).

# Let r be the domain of the CDF in the first iteration. Hence the domain changes as follows: r = k, r = k - alpha, r = k - 2\*alpha, ... , r = a.

# With each For-Loop iteration, fx will accumulate eventually to: f(k) + f(k - alpha) + f(k - 2\*alpha) + ... + f(a).

def cdf(k, m, v):

n = int((k + 1000)/0.001)

fx = prob\_density(k, m, v)

for i in range(1,n+1):

r = k - i\*0.001

fx += prob\_density(r, m, v)

return 0.001\*fx

# Let cdf\_1 be the CDF function when k = x, Mean = User-input from (b), Variance = User-input from (b).

cdf\_1 = cdf(x, mean, variance)

print(f'The Cumulative Distribution Function is {cdf\_1:.4f} (for Step-Size = 0.001, k = {x}, Mean = {mean}, Variance = {variance})')

# Let cdf\_2 be the CDF function when k = 0, Mean = 0, Variance = 1.

# Let cdf\_3 be the CDF function when k = 1.64, Mean = 0, Variance = 1.

# Let cdf\_4 be the CDF function when k = 1.96, Mean = 0, Variance = 1.

cdf\_2 = cdf(0, 0, 1)

cdf\_3 = cdf(1.64, 0, 1)

cdf\_4 = cdf(1.96, 0, 1)

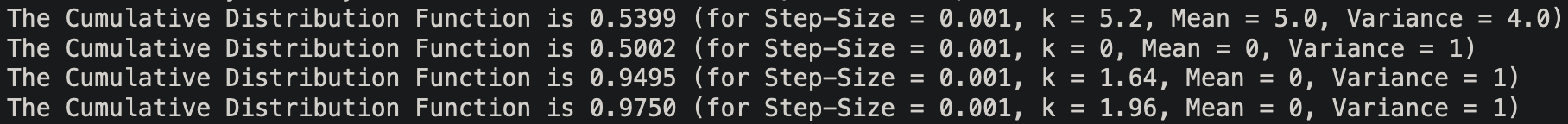
print(f'The Cumulative Distribution Function is {cdf\_2:.4f} (for Step-Size = 0.001, k = 0, Mean = 0, Variance = 1)')

print(f'The Cumulative Distribution Function is {cdf\_3:.4f} (for Step-Size = 0.001, k = 1.64, Mean = 0, Variance = 1)')

print(f'The Cumulative Distribution Function is {cdf\_4:.4f} (for Step-Size = 0.001, k = 1.96, Mean = 0, Variance = 1)')

***Code End***

**Output:**



**Question 1(g)**

Variables assigned: *k* = *x*, *alpha* = 0.001, *a* = -1000, and *n* is the number of steps to the nearest integer.

A function cdf(k, m, v) is defined. *n* is calculated to the nearest integer. The first *fx* is defined, representing the pdf *fx*(*k*), with *k* = *x*. A For-Loop is deployed, with a range from 1 to *n* + 1, to include *i* = *n* in the final iteration.

**First iteration:** The dummy variable *r* is evaluated to *r* = *k* – 1\*0.001 = *k* – 0.001. *r* is each pdf’s domain, and changes in each iteration, following the cdf formulation. Then *fx* is evaluated with the updated *r*, added to the previous fx, to accumulate to *fx*(*k*) + *fx*(*k* – 0.01).

**Next iteration:** *r* is re-evaluated as *r* = *k* – 2\*0.001 = *k* – 0.002. *fx* accumulates with the updated *r* to *fx*(*k*) + *fx*(*k* – 0.01) + *fx*(*k* – 0.02).

Iterations continue until *i* = *n*, for the cumulative sum: *fx* = *fx*(*k*) + *fx*(*k* – 0.01) + *fx*(*k* – 0.02) + … + *fx*(*a*).

Finally, the cumulative *fx* multiplies 0.001 for the cdf value.

[181 words]

**Question 1(h)**

***Code Start***

# Let x\_values be an empty List

# Let x\_prob be an empty Dictionary

x\_values = []

x\_prob = {}

# Populate the x\_values list with a lower boundary of -2 and upper boundary of 2, and a step-width of 0.5

t = -2

while t <= 2:

x\_values.append(t)

t += 0.5

# Populate the x\_prob dictionary with Keys from the x\_values list, and Values from the probabilities calculated by its cdf function.

# The probability will be rounded to 4 decimal places.

for i in x\_values:

x\_prob[i] = round(cdf(i, 0, 1), 4)

# Print the x\_prob dictionary. First, to present the dictionary in full. Secondly, to format it for the user for each x-value.

print(f'The dictionary is {x\_prob}')

x\_prob\_show = list(x\_prob.items())

for key, val in x\_prob\_show:

print(f'When x is {key}, the probability is {val}.')

***Code End***

**Output:**

****

**Text

Description automatically generated**

**Python File**

[[As per the updated announcement, embedding the Python script file is now optional only, due to Mac users reporting errors during the embedding process.]]