****

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

**ANL252**

**Python for Data Analytics**

**Tutor-marked Assignment**

**July 2021 Presentation**

Name: Chiu Li Wen Renee

PI Number: K1981620

Tutorial Group: T09

***Declaration:***

“*I declare that this assignment is my own work, unless otherwise acknowledged or credited by appropriate referencing. I have read and abide by the SUSS Honour Code and I am aware of the penalties associated with plagiarism and collusion listed in the SUSS Student Handbook*.”

**Embed Full Code:**

****

**Question 1a**

Code:

#Qn Part A - Import math package in python

import math

**Question 1b**

Code:

#Qn Part B - Codes to gather mean input

#Ask user to input mean

mean = input("Input your mean (must be a numeric value): ")

#If user clicks enter without inputting a value

if mean == "":

print("You have not inputted a mean value.")

meanNumeric = float(0)

print("Mean is automatically set to:", meanNumeric,"\n")

else:

#If user has inputted a value

try:

#convert input value to float ensure value is numeric

meanNumeric = float(mean)

print("This is your inputted mean value:", meanNumeric,"\n")

except ValueError:

#If user keys in a value that is not numeric, which cant be converted to float

print("Your mean value is not numeric, try again!","\n")

#To loopback the code in case of a non-numeric input, to prevent the user to restart the code again

continue

#-------------------------------------------------------------------------------

#Qn Part B - Codes to gather variance input

#Ask user to input variance

variance = input("Input variance (numeric > 0): ")

#If user clicks enter without inputting a value

if variance == "":

print("You have not inputted a variance value.")

varianceNumeric = float(1)

print("Variance is automatically set to:", varianceNumeric, "\n")

else:

try:

#Set inputted variance value as float (can accept both decimals and integers)

varianceNumeric = float(variance)

#Variance control mechanism codes

if (varianceNumeric <= 0):

print("Variance must be larger than 0. Try again.", "\n")

else:

print("This is your variance:", varianceNumeric,"\n")

except ValueError:

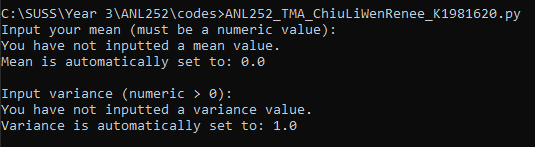
#If user keys in a value that is not numeric, which cant be converted to float

print("Your variance value is not numeric. Try again", "\n")

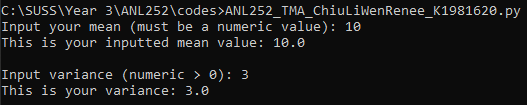
#To loopback the code in case of a non-numeric input, to prevent the user to restart the code again

continue

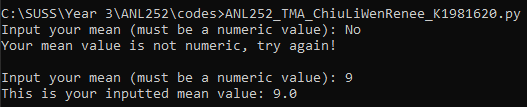
Q1b’s Output (when user presses ENTER without providing any values):

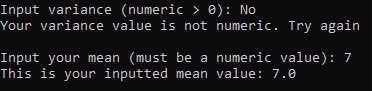


Q1b’s Output (when user enters their own values mean or variance):



Q1b’s Output (when user enters a non-numeric value for mean or variance):





**Question 1c**

Code:

#Qn Part C - Codes to gather X value input

value\_x = input("Input your value of x (can be any value between negative and positive infinity, and must be a numeric value): ")

#convert input value to float ensure value is numeric

try:

value\_x\_Numeric = float(value\_x)

print("This is your inputted x value:", value\_x\_Numeric,"\n")

except ValueError:

#If user keys in a value that is not numeric, which cant be converted to float

print("Your x value is not numeric, try again!","\n")

continue

Q1c’s Output (when user enters a numeric X value):



Q1c’s Output (when user enters a non-numeric X value):



**Question 1d**

Code:

#------------- User defined function --------------

#Qn Part D - Create a user defined function to calculate probabilty density f(x) based on user inputs

def formula\_prob\_dens(varianceNumeric,value\_x\_Numeric,meanNumeric):

#Make a variable for the formula of probility density

FX = (1/math.sqrt(2 \* math.pi \* varianceNumeric)) \* math.exp(-(math.pow((value\_x\_Numeric - meanNumeric),2)/2 \* varianceNumeric))

FX = round(FX,3)

return FX

**Question 1e**

Code:

#Part E - Formatted printing to display the result calculated in D

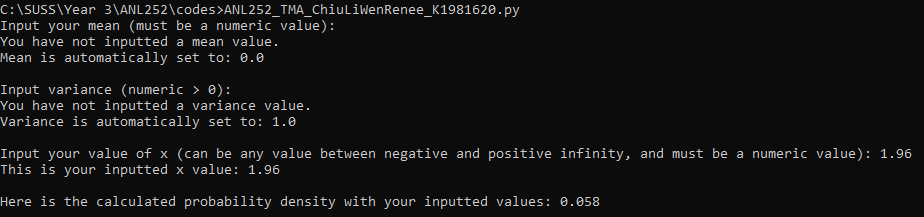
#Call the user defined function and input its value into a variable

probability\_density\_amount = formula\_prob\_dens(varianceNumeric,value\_x\_Numeric,meanNumeric)

#Print the user defined function's value with formatted printing

print(f"Here is the calculated probability density with your inputted values: {probability\_density\_amount}", "\n")

Q1d and e’s Output (if mean = 0, variance = 1, and x = 1.96):



**Question 1f**

Code:

#-------------------------------------------------------------------------------

#Part F - Creation of the codes to calculate CDF with user's inputs

#Get user input for new variable k, which is also the value of x

value\_k\_numeric = value\_x\_Numeric

print(f"The value of k will now be set to: ", value\_k\_numeric, "\n")

#Get user input for new variable alpha

value\_alpha = input("Enter the value of alpha (will be automatically set to 0.01 if you click enter with no input):")

if value\_alpha == "":

print("You have not inputted the value for alpha.")

value\_alpha = 0.01

alphaNumeric = float(value\_alpha)

print("The value of alpha is automatically set to:", alphaNumeric, "\n")

else:

try:

#convert input value to float ensure value is numeric

alphaNumeric = float(value\_alpha)

print("Your inputted alpha value is:", alphaNumeric, "\n")

except ValueError:

#If user keys in a value that is not numeric, which cant be converted to float

print("Your alpha value is not numeric, try again!","\n")

quit()

#Get user input for new variable a, which must always be less than or equals to variable k

value\_a = input("Enter the value of a (will be automatically set to 0.01 if you click enter with no input, and it must be lower or equals to k value):")

if value\_a == "":

print("You have not inputted the value for a.")

value\_a = -100

value\_a\_Numeric = float(value\_a)

print("The value of a is automatically set to:", value\_a\_Numeric, "\n")

else:

try:

#Set inputted value a as float (can accept both decimals and integers)

value\_a\_Numeric = float(value\_a)

except ValueError:

#If user keys in a value that is not numeric, which cant be converted to float

print("Your value of a is not numeric. Try again", "\n")

quit()

#To prevent any inputs of a that are larger than k

if value\_a\_Numeric > value\_k\_numeric:

print("Value of a cannot be more than k. Try again")

quit()

else:

print(f"Your inputted value of a is: {value\_a\_Numeric}","\n")

#Create a new variable to store new FX terms everytime the while loop passes

FXFormula = formula\_prob\_dens(varianceNumeric,value\_x\_Numeric,meanNumeric)

#Set the first term of the formula: fx(a)

#To sub the value of a into the user defined function of formula\_prob\_dens, let the numeric value of x = numeric value of a

value\_x\_Numeric = value\_a\_Numeric

FXFormula = formula\_prob\_dens(varianceNumeric,value\_x\_Numeric,meanNumeric)

#Increase the numeric value of a for the next loop

value\_a\_Numeric = value\_a\_Numeric + alphaNumeric

#If the numeric value of a is less than or equals to numeric value of k

while value\_a\_Numeric <= value\_k\_numeric:

#Set the numeric value x as the current numeric value of a, to sub into the user defined function

value\_x\_Numeric = value\_a\_Numeric

#Add a new term into the variable used to store the formula

FXFormula = (FXFormula + formula\_prob\_dens(varianceNumeric,value\_x\_Numeric,meanNumeric))

#Increase the numeric value of a for the next loop

value\_a\_Numeric = value\_a\_Numeric + alphaNumeric

#If the numeric value of a is more than its maximum value (numeric value of k)

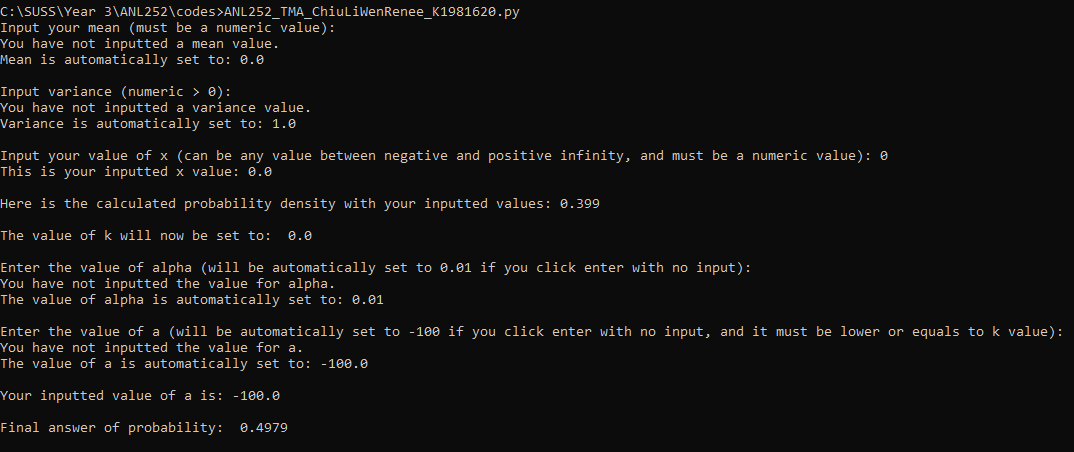
else:

#Assemble the terms stored in FXFormula variable and combine with the numeric value of alpha to finish the final formula

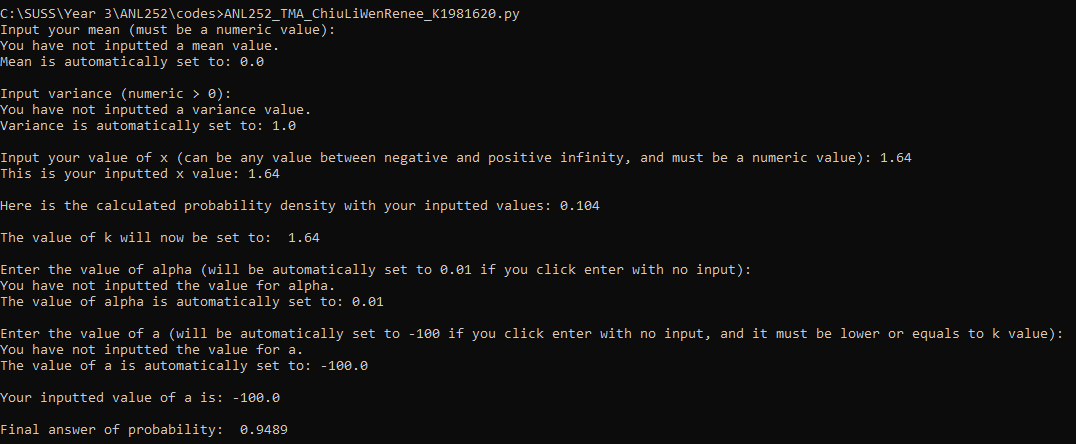
FinalForm = alphaNumeric \* FXFormula

print(f"Final answer of probability: ", round(FinalForm,4), "\n")

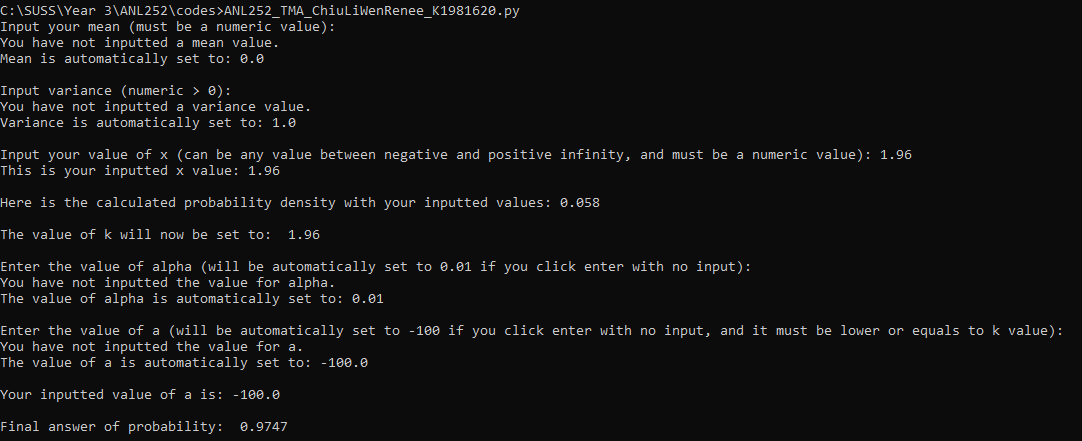
Q1f’s Output (if mean = 0, variance = 1, x/k = 0, a = -100, alpha ‘𝛼’ = 0.01):



Q1f’s Output (if mean = 0, variance = 1, x/k = 1.64, a = -100, alpha ‘𝛼’ = 0.01):



Q1f’s Output (if mean = 0, variance = 1, x/k = 1.96, a = -100, alpha ‘𝛼’ = 0.01):



**Question 1g (200 words)**

Before the calculation of P(X ≤ k), the program will set the value of k as the value of x inputted by the user previously. The program then prompts users to enter their “alpha” and “a” value, and also gives them the option to click ENTER and use default values set at alpha = 0.01 and a = -100. If users enter a non-numeric value for either value, the program will alert them and prompt them to try again.

With the necessary inputs, the program creates a new variable “FXFormula”, storing the user defined function (UDF) containing the FX formula from part 1b. The value of x will first be set to the current value of “a”, which will then be inputted into the UDF to create the first term of the P(X ≤ k) formula’s FX portion. The program then runs a while loop, reducing the value of a by alpha, substituting the new value of a into the UDF, until the value of a equals the value of k. Finally, the program creates a new variable “FinalForm”, which stores the multiplication of alpha with finalised “FXFormula” variable, format printed to show the result for P(X ≤ k).

**Question 1h**

Before the start of part 1b’s code, which includes a “while loop” that allows the user to re-run the whole program, I initialised the dictionary which will be storing the probabilities, in order to prevent the dictionary wipe from being looped.

Code (to initialise dictionary at the start):

#initialize a dictionary for part h outside of rerun loop, to allow for initialization of the dictionary at start, and prevention of clearing dictionary in while loop

prob\_dictionary = {}

Code (for the remaining part of the part 1h):

#Part H - Creation of a dictionary to store certain results from Part F, and print only the results from x values between -2 and 2, with alpha of 0.5

#Reset numeric x value to its original entered x value

value\_x\_Numeric = float(value\_x)

#If -5 <= Numeric value of x <= 5, store the values

if (value\_x\_Numeric >= -5 and value\_x\_Numeric <= 5):

prob\_dictionary[value\_x\_Numeric] = FinalForm # replace calculate probability function here

print("You have currently entered -","Key: ", value\_x\_Numeric, "- Value: ", prob\_dictionary[value\_x\_Numeric],"\n")

else:

#To allow the user to decide whether or not to re-run the entire code

print("Value of x entered is not between -5 and 5, therefore its value will not be entered into the dictionary.", "\n")

#-------------------------------------------------------------------------------

#To allow the user to decide whether or not to re-run the entire code

continueorend = str(input("Do you want to add new values into the dictionary, or quit the program? (Enter Y to continue, N to show results of the dictionary and end the program, and any other key to just end the whole program): "))

if continueorend.upper() == "Y":

print("\n")

continue

elif continueorend.upper() == "N":

#Print only values from -2 to 2:

print("Final Dictionary:","\n")

for k, v in prob\_dictionary.items():

if k == '-2':

print("Key: ", k, "- Value: ", v,"\n")

k += 0.5

elif k <= 2:

print("Key: ", k, "- Value: ", v,"\n")

k += 0.5

else:

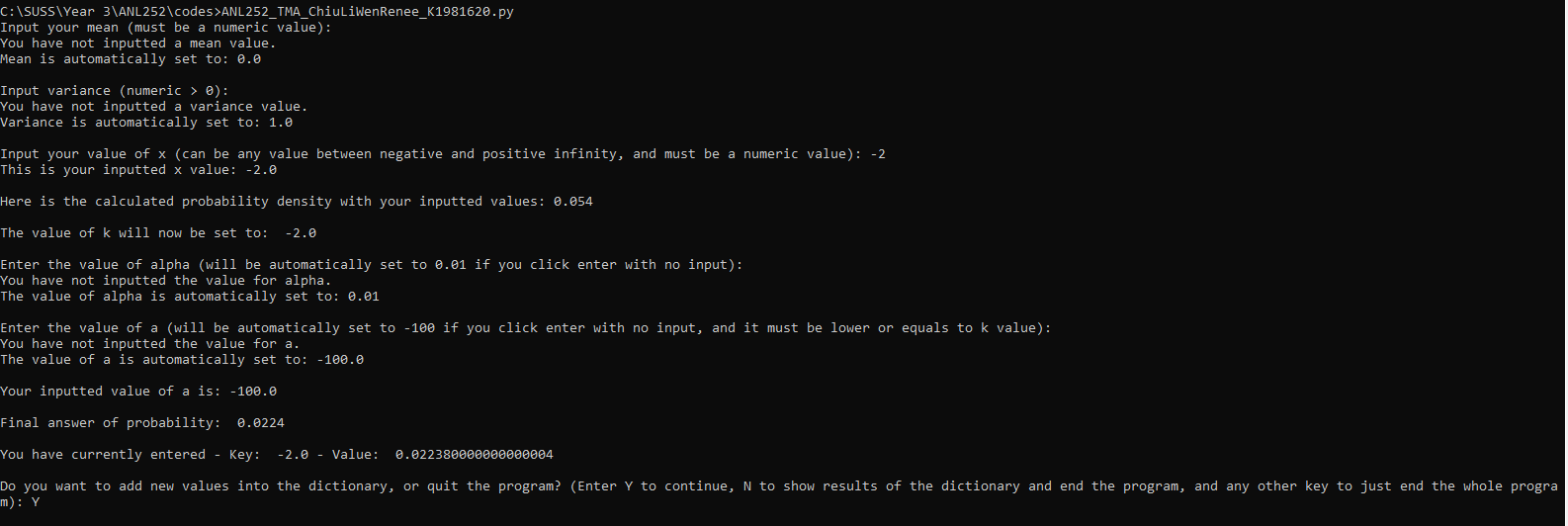
quit()

else:

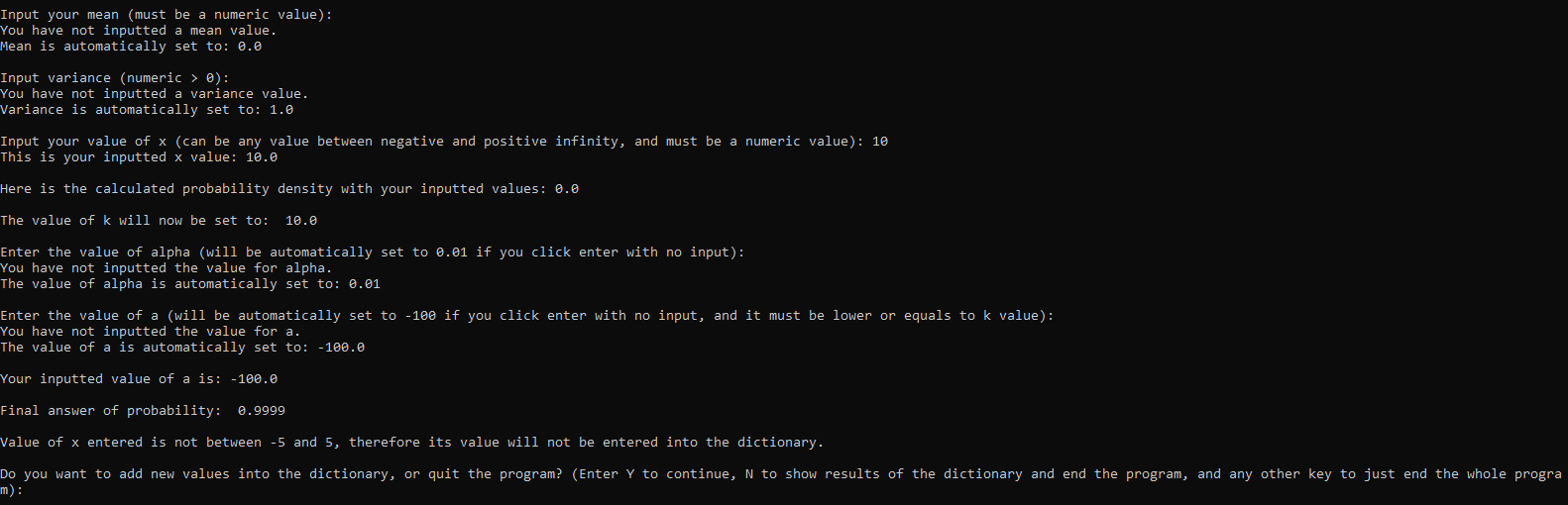
print("Bye bye.")

quit()

Output (If user enters in a X input that is between -5 and 5):

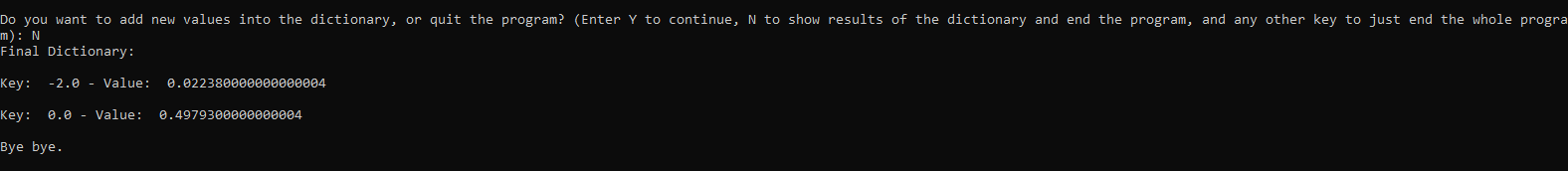


Output (If user enters in a X input that not is between -5 and 5):

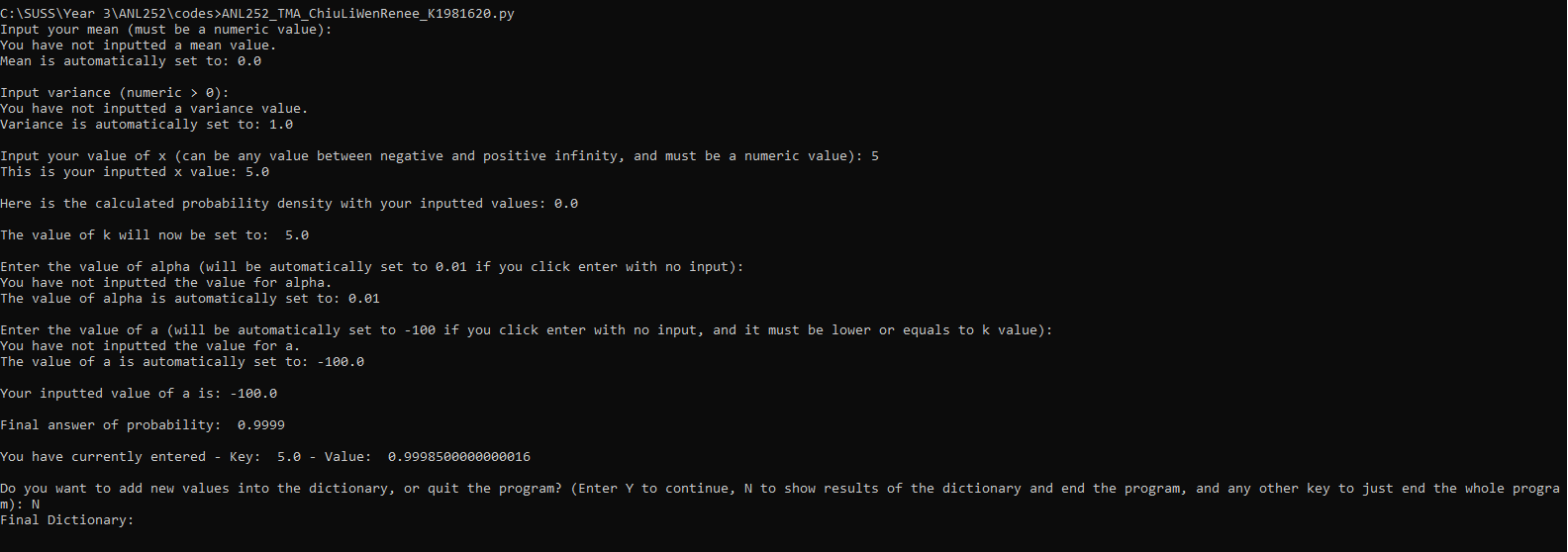


Output (print the dictionary probabilities for X between -2 and 2, and step width of 0.5):

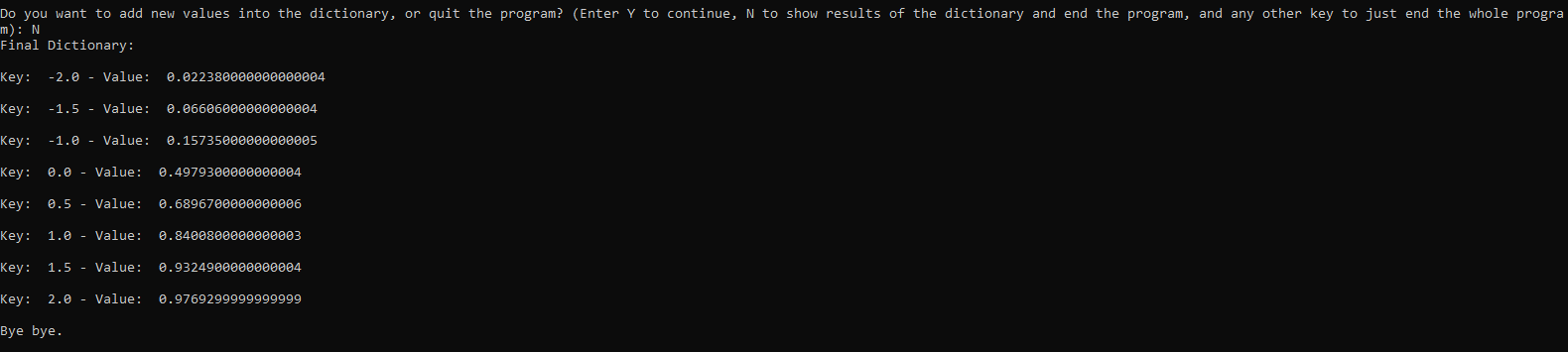
If only a few X inputs between -2 and 2 and a step width of 0.5 has been recorded:



If there are only probability values for X inputs outside of -2 and 2 and a step width of 0.5:



If there are probability values for all x inputs between -2 and 2 and a step width of 0.5:



**References**

Wu, K. Y. (2021). ANL252 Python for data analytics (study guide). Singapore University of

Social Sciences.