**Logo, company name

Description automatically generated**

**ANL252**

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

**Python for Data Analytics**

**Tutor-Marked Assignments**

**15th August 2021**

|  |  |
| --- | --- |
| **Names** | **PI Numbers** |
| Yeo Ker Woon, Jan | E1881560 |

**To start off the program**

print("Welcome to the program to calculate the probability density function and cumulative distribution function. This program is created by Jan.")

**#a)** To import math package

import math

**#b)** The mean is set to either a float or 0. Hence when spacing is inputted, mean is preset to 0. Additionally when a string is inputted or an error occur, the except function will prompt to re-enter.The variance is set to either a float or 1. Hence when spacing is inputted, mean is preset to 1.

Mean = True

while Mean:

try:

dis\_mean = float(input("Please enter the mean (Between minus infinity and plus infinity): ") or 0)

except ValueError:

print("Please try again and enter a number: ")

else:

Mean = False

Variance = True

while Variance:

try:

dis\_variance = float(input("Please enter the variance (Must be a value larger than 0):" ) or 1)

except ValueError:

print("Please try again and enter a number larger than 0: ")

else:

if dis\_variance >0:

Variance = False

**#c)** The X value is set to float for the user to enter a number. If a string is entered, the system will prompt the user to enter a number. Refer to appendix B picture 1.1 for the screen input.

X = True

while X:

try:

dis\_X = float(input("Please enter the value of X (Between minus infinity and plus infinity): "))

except ValueError:

print("Please try again and enter a number: ")

else:

X = False

**#d)** The user defined function to calculate the probability density function by imputing the function. The function can be places anywhere and input in as probability\_density\_function when needed.

def pdf(dis\_mean, dis\_variance, dis\_X):

probability\_density\_function = 1/math.sqrt(2\*math.pi\*dis\_variance)\*math.exp(-(dis\_X-dis\_mean)\*\*2/2\*dis\_variance)

return probability\_density\_function

probability\_density\_function = pdf(dis\_mean, dis\_variance, dis\_X)

**#e)** Formatted printing for the probability density function calculated. Refer to appendix B picture 1.1 for the results of formatted printing.

print(f"The probability density function is {probability\_density\_function}.")

**#f)** Explanation is in part g. Refer to appendix B for the results for k = 0,1.64 and 1.96.

dis\_k = dis\_X

dis\_alpha = 0.01

dis\_number\_a = -100

cdf = 0

if dis\_k > 0:

recursions = int(abs((dis\_number\_a/dis\_alpha)-(dis\_k/dis\_alpha)-1))

else:

recursions = int(abs((dis\_number\_a/dis\_alpha)-1))

print(f"The number of recursion is {recursions}.")

for i in range(0,recursions):

partcdf = pdf(dis\_mean, dis\_variance, dis\_number\_a)

cdf = cdf + partcdf

dis\_number\_a = dis\_number\_a + dis\_alpha

final\_cdf =round(cdf\*dis\_alpha,4)

print(f"The cumulative distribution function is {final\_cdf} with value k being {dis\_X}.")

**#g)**Based on the number inputted by the user in part C, we input that dis\_X is equal to dis\_k. The input for alpha and number a is set to 0.01 and -100 respectively. Next, the number of recursion is calculated by taking (dis\_number\_a/dis\_alpha)-(dis\_X/dis\_alpha)-1). The else function is used to input another formular for when X=K=0. Next, we format the calculation in accordance by using range to set the number of recursions in order to calculate the cumulative distribution function. The number of recursion is set to an absolute value by using abs(). We pull in the self-defined function from above in part d to calculate. For cdf + partcdf is for the calculation of the first to the last pdf to keep adding on until it reaches zero recussion. This is similar for dis\_number\_a + dis\_alpha for it to keep adding until the last number a. The final\_cdf is used to input the full equation by multiplying it against alpha and rounding it off to four decimal places.

**#h)** Use the for loop to set the criteria for the dictionary and print it out.

def the\_list(beg, end, int):

list =[]

iteration = int(((-beg) + end) / int) +1

for i in range(interation):

list.append(beg)

start = round((float(start)+int),1)

return list

def all\_cdf(list, dis\_mean,dis\_variance, dic):

for dis\_k in list:

final\_cdf = cdf(dis\_k, dis\_mean, dis\_variance)

dic[dis\_k]=final\_cdf

return dictionary

dic\_x = {}

Beg1 = -5

end1 = 5

interval1 =0.1

Beg2=-2

end2=2

interval2=0.5

list1 = the\_list(Beg1, end1, interval1)

list2 = the\_list(Beg2, end2, interval2)

dic\_x = final\_cdf(list, dis\_mean,dis\_variance, dic)

for key in list2:

keyvalue = dic\_x[key]

print(f"The cumulative function is x = {key} is {keyvalue}.")



**Please refer to appendix A for the screenshots of the program to check for the indentations.**

**Appendix A (Indention)**

Text

Description automatically generated

Text

Description automatically generated

**Text

Description automatically generated**

**Appendix B (Screenshots)**

Picture 1.1

Table

Description automatically generated with low confidence

Picture 1.2

A picture containing background pattern

Description automatically generated

Picture 1.3

A picture containing background pattern

Description automatically generated