****

**ANL252**

**Python For Data Analytics**

**Tutor-Marked Assignment (TMA)**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **LOO CHENKAI** | **Y2071132** |

**Submission Date:** 09 August 2021

**Extended Deadline:** 15 August 2021

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

# TMA

# Q1 A)--------------------

# Preparing the program by importing the "Math" Package

import math

print("\nMath package has been imported! ")

# Printing statement to let user know the math package is imported

# Q1 B)--------------------

# Ask user to enter Mean & Variance of distribution

# For Mean:

# Ask user to enter Mean of the distribution

# Ensure that Mean can be any value between minus infinity (–∞) and plus infinity (+∞).

# Ensure that input is numeric.

# Ensure that if user press ENTER without providing any values, the program will automatically set Mean to 0 and Variance to 1.

# Using while-loop, with try and except, to ensure user input is numeric

x1 = True

while x1 == True:

mean\_qns=input("\nPlease enter the Mean of the distribution \n(Press ENTER to set default value): ")

if mean\_qns == "":

mean\_qns = 0

print("The mean is set to default 0! ")

# Ensure user can press ENTER with the default 0

# Printing statement to let user know if the default value is set to 0

try:

mean\_qns=float(mean\_qns)

# Ensure that the program will capture numeric decimal place

break

except ValueError:

print("\nError: Please input numeric number! \n\*Note that Mean of the distribution can be any value between minus infinity (–∞) and plus infinity (+∞) ")

# For Variance:

# Ask user to enter variance of the distribution

# Ensure that Variance must be a value larger than 0.

# Ensure that input is numeric.

# Ensure that if user press ENTER without providing any values, the program will automatically set Mean to 0 and Variance to 1.

x2 = True

while x2 == True:

var\_qns=input("\nPlease enter the Variance of the distribution \n(Press ENTER to set default value): ")

if var\_qns == "":

var\_qns = 1

print("The Variance is set to default 1! ")

# Similarly, user can press ENTER with default value 1

break

try:

var\_qns=float(var\_qns)

if var\_qns <= 0:

print("\nNote that Variance of the distribution must be larger than 0! ")

else:

break

except ValueError:

print("\nPlease input numeric number! \n\*Note that Variance of the distribution must be larger than 0. ")

# Ensure user input value larger than 0 with if, else statement

# Ensure user input numeric number and not letters or symbols

# Ensure mechanism works for both mean and variance to be fulfilled

# Q1 C) --------------------

# Ask user to input value of X

# Ensure the value of X can be any value between minus infinity (–∞) and plus infinity (+∞).

# Ensure that input is numeric.

x3 = True

while x3 == True:

try:

x\_qns=float(input("\nPlease enter the value of X: "))

except ValueError:

print("\nPlease input numeric number! \n\*Note that the value of X can be any value between minus infinity (–∞) and plus infinity (+∞) ")

else:

break

# Similarly with part b, just need to ensure input for the value of X is numeric

# Q1 D) --------------------

# Using formula to compute probability density function (pdf)

# To set math calculation for easier reference

# User-defined function to calculate probability density function (pdf)

# Using formula based on pdf formula

pi = math.pi

sqrt = math.sqrt

exp = math.exp

# Trying to make it neater with indicated math calculation to a variable

def pdf(mean\_qns,var\_qns,x\_qns):

output\_pdf\_answer = (1/sqrt(2\*pi\*var\_qns)\*exp(-1\*((x\_qns-mean\_qns)\*\*2)/(2\*var\_qns)))

return output\_pdf\_answer

# using user-defined function, return with answer

# Q1 E) --------------------

# Showing output result

# Printing probability density function result

# Printing using f-string formatting

output\_pdf\_answer = pdf(mean\_qns,var\_qns,x\_qns)

print(f"\nThe probability density function of normal distribution is {output\_pdf\_answer} \n")

# Q1 F) --------------------

# Design program to compute P(X ≤ k) where k is the value that the user entered in part (c)

# Formula given for cumulative distribution function (cdf) is given:

# α is the range of each step, close to –∞.

# k is the value from part (c), value of X.

alpha=0.01 #using this as step range

end\_final\_value=100 #using this as fixed value

range\_count=int((end\_final\_value+x\_qns)/alpha)-1

part\_function=0

full\_function=0

each\_count=0

for number in range(0,range\_count):

each\_count = each\_count + 1

# Creating exit condition with while loop

while True:

part\_function=(x\_qns-(each\_count\*alpha))

break

full\_function+=pdf(mean\_qns,var\_qns,part\_function)

full\_function = pdf(mean\_qns,var\_qns,-(end\_final\_value))+full\_function+pdf(mean\_qns,var\_qns,x\_qns)

# Using the given cdf formula as given

output\_cdf\_answer=full\_function\*alpha

# The answer mutiply the step range

print(f"The results for P(X≤{x\_qns}) is {output\_cdf\_answer}. \n")

# display result of with user input

# Checking of results:

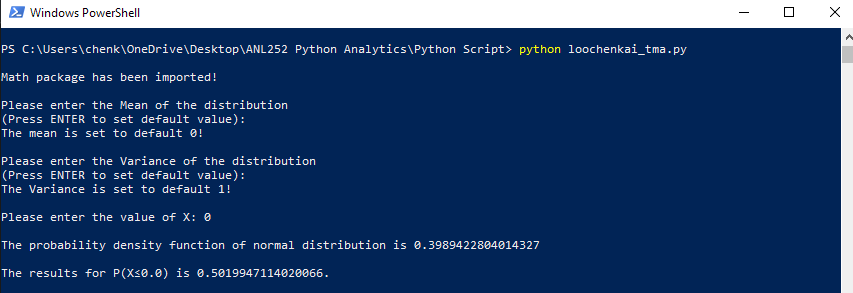
# 1) Mean = 0

# Variance = 1

# k = 0

# The probability density function of normal distribution is 0.3989422804014327.

# The results for P(X≤0.0) is 0.5019947114020066.



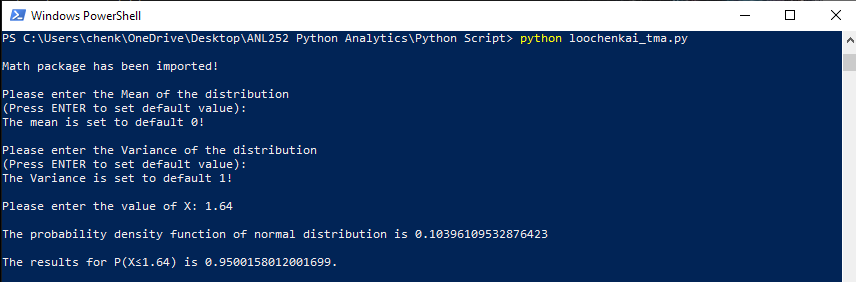
# 2) Mean = 0

# Variance = 1

# k = 1.64

# The probability density function of normal distribution is 0.10396109532876423

# The results for P(X≤1.64) is 0.9500158012001699.



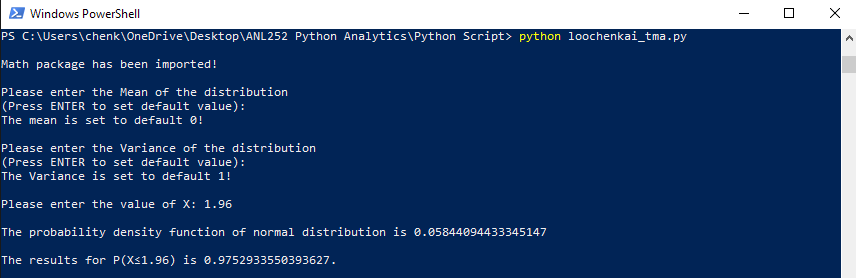
# 3) Mean = 0

# Variance = 1

# k = 1.96

# The probability density function of normal distribution is 0.05844094433345147

# The results for P(X≤1.96) is 0.9752933550393627.



# Q1 G) --------------------

# Calculation of P(X ≤ k) explanation as per part

I input the alpha as 0.01 as α act as a range of each step, where it runs continuously throughout the program as we will be calculating based on the given formula as given:



…depending on the user input for the value of k (part c) because a is a number close to –∞ (0, -0.01, -0.02… all the way until negative infinity). Hence, I used the end final value as 100 as a guide. I input a while loop range from 0 + each count based on the given formula because the accuracy of the approximation increases with the decrease of step range α, the more accurate our results is, and input a break function to compute formula because I did not want to keep looping nonstop. With the given value of k, we input them into the formula, to compute P(X ≤ k), with the given mean and variance by the user (as per part b). After computing the cumulative distribution function (cdf), the answer will multiply the step range, and printing the results to user.

# Q1 H) --------------------

# Create dictionary to store probabilities

# Creating list to store probabilities for x

# Print all probabilities onto screen for user

x\_dict = {}

x\_list = []

x\_dist\_feedback = []

x\_assign = []

count\_row = -5.1

count\_start = -2.5

count\_step = 0.5

for number in range(101):

count\_row = count\_row + 0.1

x\_list.append(round(count\_row,2))

# Using range as we are are storing the probabilities by creating a list

# Using append() to add in single item to existing list

# Using round() to returns floating numeric to a specific number

x\_dist\_feedback.append(pdf(mean\_qns,var\_qns,count\_row))

x\_dict = dict(zip(x\_list,x\_dist\_feedback))

for number in range(9):

count\_start += count\_step

index\_1 = x\_list.index(-2.5)

x\_assign.append(round(count\_start,2))

a\_dict = {key:x\_dict[key] for key in x\_assign}

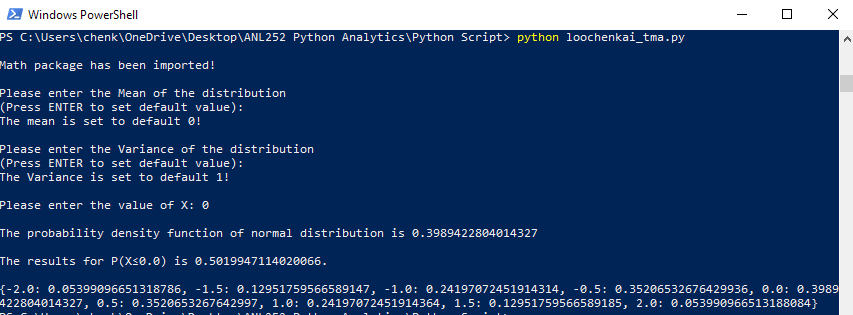
print(a\_dict)

# Printing all probabilities between -2 and 2, with step width of 0.5 onto screen

# Mean = 0

# Variance = 1

# X = 0 (screenshot as example)



Embed python script:



Screenshot of Python Script (Atom):

