A picture containing logo

Description automatically generated

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

ANL252

Python for Data Analysis

Tutor-Marked Assignment

July 2021 Presentation

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Neo Hui Ying Joey** | **B1882288** |

**Submission Date: 15 Aug 2021, 2355hrs**

**Question 1(a)**

import math

Fx(X) = (1/math.sqrt(2\*math.pi\*variance))\*math.exp(-(x-mean)\*\*2/(2\*variance))

**Question 1(b)**

Code:

# Employ a program to ask user to enter the mean and variance of distribution

# Inform user on the values of mean and variance

print("The mean can be any value between minus infinity and plus infinity.")

print("The variance must be larger than 0.")

# Ask for user inputs

while True:

try:

mean = float(input("Please enter mean: ") or "0")

print(f"The mean is {mean}.")

break

except ValueError:

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

while True:

try:

variance = float(input("Please enter variance: ") or "1")

if variance > 0:

print(f"The variance is {variance}.")

break

else:

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

except ValueError:

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

Upon the initiation of the program, users will be provided with the information on the accepted values of the mean and variance. In this python program, the while loop is adopted. The purpose of while loop is to repeat instructions until a specific condition is met. In this case, the condition is true if the value entered by the user fulfils the conditions of a mean and variance. The program will then print the number inputted through formatted printing. The purpose of the “break” loop is to stop the repetition of instructions upon the generation of an accepted value. However, if the condition is false, meaning the user has inputted an invalid number such as letters, it will be considered as a “ValueError” and user will be requested to enter the number again. As shown in the codes above, there is an ‘or “0”’ for mean and ‘or “1”’ for variance in the same line for user input. This is to inform the program to generate “0” to mean and “1” to variance should the user not have any input. As for variance, the if-condition is applied as the variance must both be true and greater than 0. Therefore, if the user were to enter a valid number that is less than 0, the else-statement will be generated, and user will be requested to input another number until the conditions of a variance are met.

A picture containing text

Description automatically generated

*Figure 1*

In Figure 1, it shows that should the user press “Enter” and not input any number for both variables, the program will automatically assign the value “0” to mean and “1” to variance.

A picture containing text

Description automatically generated

*Figure 2*

In Figure 2, it shows that should the user enter values that satisfies the conditions of a mean and variance, the values generated would be the user inputs.

Text

Description automatically generated

*Figure 3*

In Figure 3, it shows that should the numbers inputted by the user are not valid, the program will inform the user the error present and request for a new user input. As for variance, the second round of user input is a value less than 0, which does not satisfy the conditions of a variance, hence user will be informed of the error and must enter another valid number. This loop will then stop upon the input of an accepted value.

**Question 1(c)**

Code

# Inform user on the value of X

print("X can be any value between minus infinity and plus infinity.")

# Ask for user input

while True:

try:

X = float(input("Please enter X: "))

print(f"X is {X}.")

break

except ValueError:

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

Upon the initiation of the program, users will be provided with the information on the accepted values of X. In this python program, the while loop is adopted. The purpose of while loop is to repeat instructions until a specific condition is met. In this case, the condition is true if the value entered by the user fulfils the conditions of X. The program will then print the number inputted through formatted printing. The purpose of the “break” loop is to stop the repetition of instructions upon the generation of an accepted value. However, if the condition is false, meaning the user has inputted an invalid number such as letters, it will be considered as a “ValueError” and user will be requested to enter the number again.



*Figure 4*

In Figure 4, it shows that should the user enter values that satisfies the conditions of X, the values generated would be the user input.

A picture containing chart

Description automatically generated

*Figure 5*

In Figure 5, it shows that should the number inputted by the user is not valid, the program will inform the user the error present and request for a new user input.

**Question 1(d)**

Code

# Inform users on the values of X, mean and variance

print("X can be any value between minus infinity and plus infinity.")

print("The mean can be any value between minus infinity and plus infinity.")

print("The variance must be larger than 0.")

import math

# Construct user-defined function

def fx(X,mean,variance):

fx = (1/math.sqrt(2\*math.pi\*variance))\*math.exp(-(X-mean)\*\*2/(2\*variance))

return fx

while True:

try:

X = float(input("Please enter X: "))

print(f"X is {X}.")

mean = float(input("Please enter mean: ") or "0")

print(f"The mean is {mean}.")

variance = float(input("Please enter variance: ") or "1")

if variance > 0:

print(f"The variance is {variance}.")

break

else:

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

except ValueError:

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

Before we construct a user-defined function, the math package must be imported to run the program since a math formula is applied. The def-syntax defines the math function, which in this case, is fx. Since we want to apply the user inputs of X, mean and variance into the function, we must also state these parameters. The program will pass these values to fx and return the value of fx. The following steps would be to combine the codes stated in Question 1(b) and 1(c). These values of X, mean and variance entered by the user will then be passed in as parameters before the program generates the value of fx.

**Question 1(e)**

Code

print("The probability is", fx(X,mean,variance))

By using formatted printing, the inputted values of X, mean and variance will be passed in as parameters and the program will return the value of fx, which is also the probability of the normal distribution when X is of the entered value.

Text

Description automatically generated

*Figure 6*

In Figure 6, upon assigning 0 to X and mean, and 1 to variance, the program returns the result of the function, which is a probability of 0.39894.

**Question 1(f)**

Code

# Inform users on the values of X, mean and variance

print("X can be any value between minus infinity and plus infinity.")

print("The mean can be any value between minus infinity and plus infinity.")

print("The variance must be larger than 0.")

import math

# Construct user-defined function

def fx(X,mean,variance):

fx = (1/math.sqrt(2\*math.pi\*variance))\*math.exp(-(X-mean)\*\*2/(2\*variance))

return fx

while True:

try:

X = float(input("Please enter X: "))

print(f"X is {X}.")

mean = float(input("Please enter mean: ") or "0")

print(f"The mean is {mean}.")

variance = float(input("Please enter variance: ") or "1")

if variance > 0:

print(f"The variance is {variance}.")

break

else:

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

except ValueError:

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

sum = 0

while X >= -100:

fx = (1/math.sqrt(2\*math.pi\*variance))\*math.exp(-(X-mean)\*\*2/(2\*variance))

sum += fx\*0.01

X -= 0.01

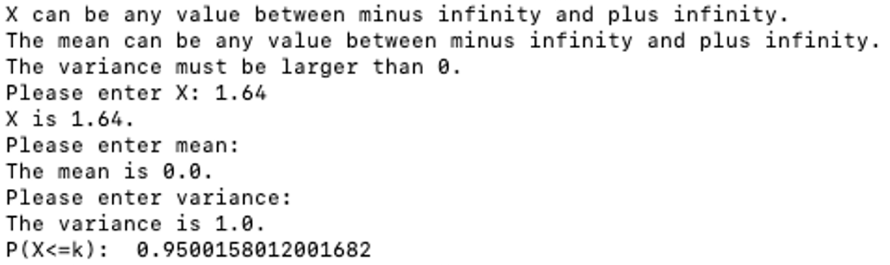
print("P(X<=k): ", sum)

Text

Description automatically generated

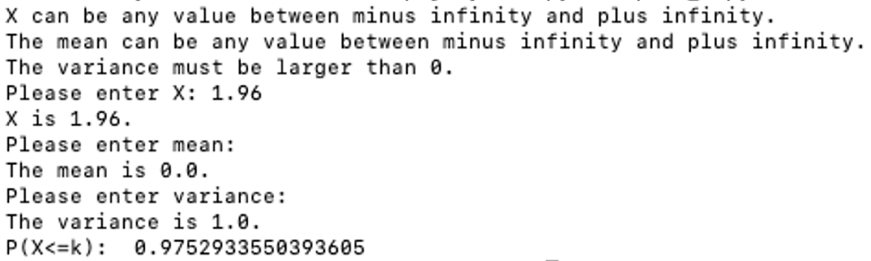
*Figure 7*

In Figure 7, it shows that with a mean of 0 and variance of 1, for k = 0, P(X<=k) = 0.50199.



*Figure 8*

In Figure 8, it shows that with a mean and variance of 1, for k = 1.64, P(X<=k) = 0.95001.

  
*Figure 9*

In Figure 9, it shows that with a mean and variance of 1, for k = 1.96, P(X<=k) = 0.97529.

**Question 1(g)**

P(X<=k) represents the total sum of f(x) multiplied by α as the value of X decreases from k to a. In this case, we assign -100 to a and 0.01 to α, and k is a value of X entered by the user. To find the summation of this series where X decreases from k to -100, we must first initialize the sum with 0. This will then be followed by the adoption of the while loop. In this instance, the condition would be X>= -100, since the value of X ranges from -100 to a user input. If the condition is true, then the sum will keep increasing by (fx x 0.01) as X decreases by 0.01 until X becomes -100. The sum reflected by the program would be the sum of fx between X = -100 and X = the value inputted by the user multiplied by 0.01. As shown in the above screenshot, when k = 0, 1.64 and 1.96 with mean of 0 and variance of 1, P(X<=k) would be 0.50199, 0.95001 and 0.97529 respectively.

**Question 1(h)**

# Create a dictionary to store the probabilities

dicts = {}

sum = 0

X = -5

while -5<=X<=5:

fx = (1/math.sqrt(2\*math.pi\*variance))\*math.exp(-(X-mean)\*\*2/(2\*variance))

sum += fx\*0.01

dicts[X] = sum

X = round(X+0.1,1)

# Print all probabilities of x’s between -2 and 2 with a step width of 0-.5

X = -2

while -2<=X<=2:

print(f"{X},{dicts[X]}")

X = round(X+0.5,1)

A dictionary is a collection of data values and holds key:value pair. In this case, the key is the X-values which range from -5.0 to 5.0 and the values are the corresponding probabilities. The curly brackets are used to indicate the construction of a new dictionary. We must also state the initial value of X to inform python to work on this stated value instead of the default value of 0. The while loop is adopted to generate the corresponding probabilities for every increment of X by 0.1 from -5.0 to 5.0. The round() function is to inform the program to round off the X-value to 1 decimal place. The new dictionary will have all the probabilities stored in it.

Upon creating the new dictionary, we will have to print all the probabilities of X-values ranging from -2.0 to 2.0. Similarly, we state the initial value of X, which in this case is -2. The while loop is adopted and due to the change of step width from 0.1 to 0.5, the program will be informed to generate the corresponding probabilities for every increment of X by 0.5 from -2.0 to 2.0. The formatted printing will enable the program to refer to the dictionary created to generate these probabilities. The results are shown in Figure 10 below.

Text

Description automatically generated

*Figure 10*

Appendix

Text

Description automatically generated

*Screenshot for Q1(b)*

Text

Description automatically generated

*Screenshot for Q1(c)*

Text

Description automatically generated

*Screenshot for Q1(d) and (e)*

Text

Description automatically generated

*Screenshot for Q1(f)*

Text

Description automatically generated

*Screenshot for Q1(h)*