****

**Qn1a)**

|  |  |
| --- | --- |
| **Code:** | # ANL252\_TMA01\_Y1981340\_DanialLokeZhiWeng.py  # Danial loke Zhi Weng, Y1981340  # TMA01, ANL252, Submission Deadline: 15 August 2021, 2355.  # (a).  # Import "math" package into program.  import math  # Start the program (pdf) by print to tell the user what the program is about.  print("This program is to compute the probability of a normally distributed random variable (X) given the distribution mean and variance which can be determined by the probability density function (pdf).")  print("Please enter the following values. (Note: if you do not provide any values for the mean and the variance, the values will be 0 and 1 respectively.)")  print("") |
| **Output:** |  |

**Qn1b)**

|  |  |
| --- | --- |
| **Code:** | # (b).  # This part is to create user inputs for the mean and variance of the distribution.  # ----------------------------------------------------------------------------------------  # User to enter the mean of the distribution.  # Use try, except and else for error handling in case user enter the wrong input types. This is to ensure that the user's input is numeric.  # Use while loop so that if user enters the wrong inputs, program does not crash and instead, allows user a chance to enter the right input.  # Use 'or' to ensure that if user press ENTER without providing any values, program will automatically set mean to 0.  # Print the value entered to display to the user.  while True:  try:  mean\_input = float(input("Enter the mean of the distribution (minus infinity (–∞) and plus infinity (+∞), default is mean = 0): ") or "0")  except ValueError:  print("")  print("The value of the mean must be a number.\nPlease try again.")  print("")  continue  else:  print(f"The mean entered is {mean\_input}")  break |
| **Output:** | The output above shows if user pressed ENTER without any input    The output above shows the error handling, Except. |

|  |  |
| --- | --- |
| **Code:** | # User to enter the variance of the distribution.  # Use try, except and else for error handling in case user enter the wrong input types. This is to ensure that the user's input is numeric.  # Use if function to ensure that the input value is larger than 0. If input value is less than or equal to 0, it will print a statement to tell the user to try again.  # Use while loop so that if user enters the wrong inputs, program does not crash and instead, allows user a chance to enter the right input.  # Use 'or' to ensure that if user press ENTER without providing any values, program will automatically set variance to 1.  # Print the value entered to display to the user.  while True:  try:  variance\_input = float(input("Enter the variance of the distribution (value must be larger than 0, default is variance = 1): ") or "1")  except ValueError:  print("")  print("The value of the variance must be a number.\nPlease try again.")  print("")  continue  if variance\_input <= 0:  print("")  print("The value of the variance must be larger than 0.\nPlease try again.")  print("")  continue  else:  print(f"The variance entered is {variance\_input}")  break |
| **Output:** | The output above shows if user pressed ENTER without any input    The output above shows the error handling, Except, and if function. |

**Qn1c)**

|  |  |
| --- | --- |
| **Code:** | # (c).  # This part is create user inputs for for the x value.  # ----------------------------------------------------------------------------------------  # User to enter the value of X.  # Use try, except and else for error handling in case user enter the wrong input types. This is to ensure that the user's input is numeric.  # Use while loop so that if user enters the wrong inputs, program does not crash and instead, allows user a chance to enter the right input.  # Print the value entered to display to the user.  while True:  try:  x\_input = float(input("Please enter the value of X (minus infinity (–∞) and plus infinity (+∞)): "))  except ValueError:  print("")  print("The value of X must be a number.\nPlease try again.")  print("")  continue  else:  print(f"The value of x entered is {x\_input}")  break  print("") |
| **Output:** | The output above shows if user entered 0 as input.    The output above shows the error handling, Except. |

**Qn1d)**

|  |  |
| --- | --- |
| **Code:** | # (d).  # This part is to contruct a user-defined function using the pdf formula.  # ----------------------------------------------------------------------------------------  # Creating a pdf calculator using def.  # return the answer of the formula (fx\_value) so that when we print this function, the output will be the outcome of the pdf.  def pdf\_function(mean, variance, X):  fx\_value = (1 / math.sqrt(2\*math.pi\*variance))\*math.exp(-(X-mean)\*\*2 / 2\*variance)  return fx\_value  # Using the earlier inputs for the pdf\_function.  pdf = pdf\_function(mean\_input, variance\_input, x\_input) |
| **Output:** | No output. |

**Qn1e)**

|  |  |
| --- | --- |
| **Code:** | #(e).  # Formatted printing to display the result of (d) to the user.  print("-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  print(f"The probability of a normally distributed random variable (X) = {x\_input} given the distribution mean = {mean\_input} and variance = {variance\_input} is:\nfx(x): {pdf}")  print("-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  print("") |
| **Output:** | The output above shows the formatted printing. |

**Qn1f)**

|  |  |
| --- | --- |
| **Code:** | #(f).  # This part is to use cumulative distribution function (cdf) to compute P(X ≤ k).  # ----------------------------------------------------------------------------------------  # Start the next part of program (cdf) by print to tell the user what the program is about.  print("The next program is to compute the probability of a normally distributed random variable X smaller equal to a value k P(X ≤ k) which can be determined by the cumulative distribution function (cdf).")  print("Please enter the following values. (Note: if you do not provide any values for the alpha and a, the values will be 0.01 and -100 respectively.)")  print("")  # User to enter the alpha value (step range).  # Use try, except and else for error handling in case user enter the wrong input types. This is to ensure that the user's input is numeric.  # Use while loop so that if user enters the wrong inputs, program does not crash and instead, allows user a chance to enter the right input.  # Use 'or' to ensure that if user press ENTER without providing any values, program will automatically set alpha to 0.01.  # Print the value entered to display to the user.  while True:  try:  alpha\_input = float(input("Enter Alpha (step range, default is alpha = 0.01): ") or "0.01")  except ValueError:  print("")  print("The value of Alpha must be a number.\nPlease try again.")  print("")  continue  else:  print(f"The Alpha entered is {alpha\_input}")  break |
| **Output:** | The output above shows the printing and the user input.    The output above shows if user press ENTER without input.    The output above shows the error handling, Except. |
| **Code:** | # User to enter the a value.  # Use try, except and else for error handling in case user enter the wrong input types. This is to ensure that the user's input is numeric.  # Use if function to ensure that the input value is negative.  # Use while loop so that if user enters the wrong inputs, program does not crash and instead, allows user a chance to enter the right input.  # Use 'or' to ensure that if user press ENTER without providing any values, program will automatically set the value of a to -100.  # Print the value entered to display to the user.  while True:  try:  a\_input = float(input("Enter value a (a number close to minus infinity (–∞), default is a = -100): ") or "-100")  except ValueError:  print("")  print("The value of a must be a number.\nPlease try again.")  print("")  continue  if a\_input >= 0:  print("")  print("The value of a must be negative.\nPlease try again.")  print("")  continue  else:  print(f"The value of a entered is {a\_input}")  break  print("") |
| **Output:** | The output above shows if user press ENTER without input.    The output above shows the error handling, Except, and the IF function. |
| **Code:** | # Creating a CDF calculator using def.  # CDF is the sum of cdf1, cdf2 and cd3 multiplied by alpha.  def cdf\_function(mean, variance, K, alpha, a):  cdf1 = pdf\_function(mean, variance, a)  # alpha is the step width.  # terms\_cdf2 is the number of terms we can put between a and k.  # To find the number of terms, we take the difference of a and k and divide it by the step width (alpha).  terms\_cdf2 = ((K - a) / alpha)  cdf2 = 0  while terms\_cdf2 > 0:  variable\_cdf2 = K - terms\_cdf2\*alpha  terms\_cdf2 = terms\_cdf2 - 1  cdf2 = cdf2 + pdf\_function(mean, variance, variable\_cdf2)  cdf3 = pdf\_function(mean, variance, K)  fx\_value2 = alpha\*(cdf1 + cdf2 + cdf3)  return fx\_value2  # Using the earlier inputs for the cdf\_function.  CDF = cdf\_function(mean\_input, variance\_input, x\_input, alpha\_input, a\_input)  # Formatted printing to show the result of the cdf.  print("-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  print(f"The probability of a normally distributed random variable X of smaller equal to a value k (P(X ≤ k)) when k = {x\_input}, with mean = {mean\_input}, variance = {variance\_input}, alpha = {alpha\_input} and a = {a\_input} is:\nP(X ≤ {x\_input}): {CDF}")  print("-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  print("")  # Checking the results for k from the the given k values with a mean and variance of 0 and 1.  # Used a list for the k values.  # Use the for loop to print the results of the different k values  # Mean and variance is 0 and 1 respectively, thus hard code 0 and 1 into cdf\_function  k\_variables = [0, 1.64, 1.96]  print(f"Furthermore, the probability results (P(X ≤ k)) when k = {k\_variables}, with mean = 0, variance = 1, alpha = {alpha\_input} and a = {a\_input} is: ")  for e in k\_variables:  CDF\_check = cdf\_function(0, 1, e, alpha\_input, a\_input)  print(f"P(X ≤ {e}): {CDF\_check}")  print("") |
| **Output:** | The output above shows the formatted printing. |

**Qn1g)**

I created a user-defined function for the cdf which requires the arguments: mean, variance, K, alpha and a.

I broke down the cdf into 3 parts: (1) cdf1: ‘𝑓𝑋(𝑎)’ (2) cdf2: ‘fX(K – terms(alpha)’ (3) cdf3: ‘𝑓𝑋(𝑘)’.

The fx in the formula refers to the pdf, which I defined earlier in the program (pdf\_function(mean, variance, X)). Hence, I used it to calculate cdf1, cdf2 and cdf3.

Since the objects for the mean and variance arguments are the user inputs in the earlier part of the program, the object for argument K will differ among the 3 parts. For cdf1, the object is the user input for a. For cdf3, the object is the user input for X.

For cdf2, the object is (K – terms(alpha)). I multiplied the terms and the alpha because the number of functions is same as the number of terms there are between a and K. Hence, I calculated the terms (terms\_cdf2) by taking the difference of a and k, then dividing it by the step range (alpha). After which, I used a while loop to sum the functions together.

Lastly, I sum the 3 parts together and multiplied it by alpha to return the cdf value.

**Qn1h)**

|  |  |
| --- | --- |
| **Code:** | # (h).  # Create a dictionary to store the probabilities of the normal distribution for the given x values  # ----------------------------------------------------------------------------------------  # x values are the keys, and the corresponding probabilities are the values of the dictionary.  # Mean and variance is 0 and 1 respectively, thus hard code 0 and 1 into cdf\_function  # Store x values from -5 to 5 with a step of 0.1 into the dictionary.  d\_P = {}  x\_keys = -5.0  while round(x\_keys, 1) <= 5.0:  CDF2\_values = cdf\_function(0, 1, round(x\_keys,1), alpha\_input, a\_input)  d\_P[round(x\_keys, 1)] = CDF2\_values  x\_keys = x\_keys + 0.1  # print(d\_P). This is to check if the dictionary is working.  # Print all the probabilities (with the corresponding x) of those x’s between -2 and 2 with a step width of 0.5 from the dictionary onto the screen.  print(f"The probabilities (with the corresponding x) of those x’s between -2 and 2, with mean = 0, variance = 1, alpha = {alpha\_input} and a = {a\_input} is: ")  x\_keys = -2.0  while round(x\_keys, 1) <= 2.0:  print(f"P(X ≤ {round(x\_keys, 1)}): {d\_P[round(x\_keys, 1)]}")  x\_keys = x\_keys + 0.5  print("-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  print("")  print("Thank you for using this program.") |
| **Output:** | The output above shows the formatted printing of the dictionary keys and values. |