

**ANL252**

**Python for Data Analytics**

# **TMA**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Elijah Song Ming Fu** | **K2070438** |

Q1

|  |  |
| --- | --- |
| a) | #this code ‘Import math’ to be at the top of the python script to import math modules  Import math |
| b) | #this program prompts user to enter the value for the mean and variance of the distribution  def main():  #while loop to re-prompt users input value till it meets the conditions  while True:  #or float(0) set the default value of mean to 0 if no values are entered  mean = (input("Enter the value of Mean [Value between negative infinity to infinity]: ") or float(0) )  #try to convert mean to a float class if its a numeric digits, otherwise remind user to input only numeric values  try:  mean = float(mean)  break  except ValueError:  print("Only numeric values")    #while loop to re-prompt users input value till it meets the condition  while True:  #or float(1) set the default value of variance to 1 if no values are entered  variance = (input("Enter the value of Variance [Must be larger than 0]: ") or float(1))  try:  variance = float(variance)  #if variance is more than 0, the loops end  if variance > 0:  break  #else if variance less than or equals 0, display the message and re-prompt user to enter value > 0  elif variance <= 0:  print("Variance value must be larger than 0")  except ValueError:  print("Must only be numeric values") |
| c) | #this prompts user to enter the value for X which can be minus infinity and plus infinity  #while loop to prompt user to enter the value of X until it meets certain conditions  while True:  x = input("Enter the value of X [Value between negative infinity to infinity]: ")  #try to convert x to float class, if it is not a numeric value, it will remind user to key in a numeric value  try:  x = float(x)  break  except ValueError:  print("Must only be numeric value") |
| d) | #this user-defined function “probability” takes in 3 parameters, x, mean and variance to compute the probability density function  def probability(x, mean, variance):  probability = (1/(math.sqrt(2\*3.14159\*variance)))\*math.exp(-pow((x-mean),2)/(2\*variance))  return probability  #taking in the values of x, mean and variance entered by user above to compute the probability density fx(x)  result = probability(x, mean, variance) |
| e) | #this line of code is to display the result of (d) to the user using formatted printing  print(f"The probability density (Result) is {result}") |
| f) | # part (f)  while True:  range\_step = input("Enter the value of range of each step: ")  try:  range\_step = float(range\_step)  break  except ValueError:  print("Must only be numeric value")  while True:  a = input("Enter the value for a: ")  try:  a = float(a)  break  except ValueError:  print("Must only be numeric value")  k = x  cdf = 0  while a <= k:  pdf = probability(a, mean, variance)  cdf = cdf + pdf  cdf\_value = round(cdf \* range\_step, 4)  # line 76 to end the loop when a value is same as k value  a = a + range\_step  print(f"The cdf is {cdf\_value}") |
| g) | Firstly, we start it off with allowing user to input the range\_step value representing alpha symbol with control mechanism such as ‘try’ and ‘except’ method to only allow numeric values. Similarly, prompt user for the value for variable ‘a’.  Secondly, we take k = x as previously user has keyed in the value of x. After that we use a while loop while a is less than or equals to k, cdf will be a summation of the probability results. As every time the while loop is looping, a value will increase by adding the range\_step until it becomes equals to k and break the while loop. Followed by cdf multiplied with the range\_step to increase the accuracy of the approximation.  Lastly, print out the cdf result in a formatted string. |
| h) | # part h  # creating an empty dictionary for keys and values to be added in later  store\_result = {}  key\_x = -5  value\_x = 0  cdf1 = 0  # generating key values from -5 to 5 by 0.10 step  while key\_x <= 5:  pdf1 = probability(key\_x, mean, variance)  cdf1 = cdf1 + pdf1  value\_x = cdf1 \* range\_step  store\_result[round(key\_x, 2)] = value\_x  key\_x += 0.10    keys = list(filter(lambda x: x <= 2 and x >= -2 and x%0.5 == 0, store\_result.keys()))  for x in keys:  if x >= 0:  space = " "  else:  space = ''  print(space, x, round(store\_result[x], 5)) |

Appendix A:

Graphical user interface, text, application, email

Description automatically generated

Appendix B:

Text

Description automatically generated