

**ANL252**

**Python for Data Analytics**

**Tutor-Marked Assignment**

**July 2021 Presentation**

|  |  |
| --- | --- |
| Name: | Tan Yi Hao |
| Student Number: | H2070659 |
| Submissions Date: | 9th August 2021 |

**Question 1(a)**

Import.math

Math.pi

3.141592653589793

A picture containing text, orange

Description automatically generated

**Question 1(b) (Refer to Appendix 1 for screenshot of code)**

|  |
| --- |
| import math  # ensure variance is more than 0 and if user enter without value, mean = 0 and variance = 1  # no restrictions / control mechanism for mean  mean = float(input("Enter the mean of the distribution ([–∞, +∞]): "))  while True:  # User input  variance = (input("Enter the variance of the distribution (Value must be larger than 0): "))  if variance and float(variance) < 0:  print("Try again (Variance value must be larger than 0): ")  elif variance == "":  mean = 0  variance = 1  print("Mean is 0 & variance is 1.")  break  else:  # convert float from string input to numeric  variance = float(variance)  print(f"Mean is {mean} and variance is {variance} ")  break |
|  |

**Screenshot of Output**

**Text

Description automatically generated**

**Question 1(c) (Refer to Appendix 1 for screenshot of code)**

|  |
| --- |
| while True:  try:  x = float(input("Enter X ([–∞, +∞]): "))  print(f"X = {x}.")  break  except ValueError:  print("Invalid input, please enter a number ([–∞, +∞]) ") |
|  |

**Screenshot of output**

Text

Description automatically generated

**Question 1(d) (Refer to Appendix 1 for screenshot of code)**

|  |
| --- |
| import math  # Define function of (x)  def f(x, mean, variance):  #Calculates the probability density function of the normal distribution.  # x: The value of the variable  # mean: The mean of the distribution  # sd: The std deviation of the distribution  # return: The probability density function  return 1 / (variance \* math.sqrt(2 \* math.pi)) \* math.exp(-(x - mean) \*\* 2 / (2 \* variance \*\* 2))  while True:  try:  # User input  x = float(input("Enter X ([–∞, +∞]): "))  mean = float(input("Enter the mean of the distribution ([–∞, +∞]): "))  variance = input("Enter the variance of the distribution (Value must be larger than 0): ")  # Test for variance control mechanism  if variance and float(variance) < 0:  print("Try again (Value must be larger than 0): ")  elif variance == "":  # ensure variance is more than 0 and if user enter without value, mean = 0 and variance = 1  print("As you entered no value, the value of Mean = 0 and Variance = 1.")  mean = 0  variance = 1  #density\_function= (1 / (math.sqrt(2 \* math.pi \* (variance \*\* 2)))) \* math.exp(-((x - mean) \*\* 2) / (2 \*(variance \*\* 2)))  # 4 decimal points as pointed out by prof Alfred in the TMA Q&A  print(f"Therefore, the function of (x) is {f(x, mean, variance):.4f}")  break  else:  variance=float(variance)  # density\_function= (1 / (math.sqrt(2 \* math.pi \* (variance \*\* 2)))) \* math.exp(-((x - mean) \*\* 2) / (2 \* (variance \*\* 2)))  # 4 decimal points as pointed out by prof Alfred in the TMA Q&A  print(f"Therefore, the function of (x) is {f(x, mean, variance):.4f}")  break  except ValueError:  print("Pls try again([–∞, +∞]): ") |
|  |

**Question 1(e)**

Text

Description automatically generated

**Question 1(f) (Refer to Appendix 1 for screenshot of code)**

|  |
| --- |
| # PART (a)  import math  # PART (d)  def f(x, mean, sd):  #Calculates the probability density function of the normal distribution.  # x: The value of the variable  # mean: The mean of the distribution  # sd: The std deviation of the distribution  # return: The probability density function  return 1 / (sd \* math.sqrt(2 \* math.pi)) \* math.exp(-(x - mean) \*\* 2 / (2 \* sd \*\* 2))  # PART (b)  # gets the mean from the user as instructed, assuming mean is a number  mean = float(input("Enter the mean of the distribution ([–∞, +∞]): "))  # this while loop gets the variance from the user as instructed  while True:  variance\_value = input("Enter the variance of the distribution (Value must be larger than 0): ")  if variance\_value == "":  mean = 0  variance = 1  break  else:  try:  variance = float(variance\_value)  if variance < 0:  print("Try again (Variance must be larger than 0): ")  else:  break  except:  print("Try again (please enter a number): ")  # PART (c)  # this while loop gets the variable x from the user as instructed  while True:  try:  x = float(input("Enter the value of x (k) [–∞, +∞]: "))  break  except:  print("Invalid input, please enter a number ([–∞, +∞]): ")  # PART (e)  #find the probability desity function and print it  variance = math.sqrt(variance)  print(f"The value of probabilty density function at x = {x} is {f(x, mean, variance):.4f}")  # PART (f)  for k in [x, 0,1.64,1.96]:  a = -100  step = 0.001  sum = 0  while a <= k:  a += step  sum += f(a, mean, variance)  sum \*= step  print(f"The value of the cumulative distribution function at k = {k} is {sum:.4f}")  mean = 0  variance = 1  sd = math.sqrt(variance) # sd is the standard deviation |
|  |

**Question 1(g)**

Firstly, I have used the code in part c for the user to enter value k. The user can then use the mean and variance value in part b and to display the result.

During the first iteration of this loop, k = x, and this iteration completes the first half of part (f). After the first iteration, mean and variance change as instructed. The next 3 iterations go with mean = 0, variance = 1 and k is picked from the list iteratively. This is the second half of part (f). I have also attached the screenshot output as well.

**A picture containing text, plaque

Description automatically generated**

**Question 1(h) (Refer to Appendix 1 for screenshot of code)**

|  |
| --- |
| import math  # Mean & Variance given in the qns  mean = 0  sd = 1  # For generating x = {-5 to 5}, storing the range of -5.0 to 5.0  X1 = [0.1 \* x for x in range(-50,51)]  # creating the dictionary name store  normal\_distribution = {}  for x in X1:  # get the probability p(x) of the normal distribution  p = 1 / (sd \* math.sqrt(2 \* math.pi)) \* math.exp(-(x - mean) \*\* 2 / (2 \* sd \*\* 2))  # storing the resultant probability of x in the dictionary  normal\_distribution[x] = p  # displaying and generating x’s between -2 and 2 having step width of 0.5  X2 = [0.1 \* x for x in range(-20,21,5)]  # displaying the probabilities at those selected x's from -2.0 to 2.0 in steps of .5  # as advised by prof Alfred in TMA Q&A  for x in X2:  print(f"x = {x:.4f}, p(x) = {normal\_distribution[x]:.4f}") |
|  |

**Output of Code**

**Text

Description automatically generated**

**APPENDIX 1: SCREENSHOT OF CODE**

**QNS 1B:**

**Text

Description automatically generated**

**QNS 1C:**

**Text

Description automatically generated**

**QNS 1D:**

**Text

Description automatically generated**

**APPENDIX 1: SCREENSHOT OF CODE**

**QNS 1F:**

**Text

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Qns 1H:**

**Text

Description automatically generated**