+

ANL252

Python for Data Analytics

Tutor Marked Assignment – July 2021 Semester

1. (a) Importing the Math Package

|  |
| --- |
| import math |

1. (b) Program to Enter Mean and Variance

|  |
| --- |
| sets the mean value and variance value as 0, to be able to take in a numeric value.  mean\_value = 0  variance\_value = 0  Error Handling for Mean Value using While Loop  Using while loop, we put in place error handling (try, except and else) to highlight issues if user enters an invalid figure.  Using Error Handling for Input  Under the try portion, we are allowing user to input the mean value between between minus and plus infinity. If user has indicated a valid mean figure, it will pop up the response "You have entered {mean\_qn} as the mean." Thereafter, it will break out of the loop.  However, if user does not enter anything since length of the input is "blank", the program will automatically set the value as float and as the value as "0".  If user does not enter a valid mean figure (i.e. invalid character symbol or characters), it will pop up a response saying that entering {mean\_qn} is not a number and will get user to try again until a valid response is correctly entered.  Error Handling for Variance Value using While Loop  Similar to Mean Value, we use we put in place error handling (try, except and else).  Using Error Handling for Input  Under the try portion, we are allowing user to input the variance value for a value larger than 0. If user has indicated a valid mean figure, it will pop up the response "You have entered {variance\_qn} as the variance." Thereafter, it will break out of the loop. However, if user indicates a negative number, the program will request user to enter a positive number instead.  However, if user does not enter anything since length of the input is "blank", the program will automatically set the value as float and as the value as "1".  If user does not enter a valid mean figure (i.e. invalid character symbol or characters), it will pop up a response saying that entering {variance\_qn} is not a number and will get user to try again until a valid response is correctly entered.  while True:  try:  mean\_qn = input(f"Please Enter the Mean (between minus infinity (–∞) and plus infinity (+∞)):\n")  mean\_value = float(mean\_qn)  print(f"You have entered {mean\_qn} as the mean.")  break  except ValueError:  if len(mean\_qn) == 0:  mean\_value = float(0)  print(f"You have set μ as 0.")  break  else:  print(f"You entered {mean\_qn}, which is not a number. Try again.")    print("\n")  while True:  try:  variance\_qn = input(f"Please Enter the Variance (larger than 0):\n")  if float(variance\_qn) > 0:  variance\_value = float(variance\_qn)  print(f"You have entered {variance\_qn} as the variance.")  break  else:  print(f"You entered {variance\_qn}, which is a negative number. Try again.")  except ValueError:  if len(variance\_qn) == 0:  variance\_value = float(1)  print(f"You have set σ^2 as 1.")  break  else:  print(f"You entered {variance\_qn}, which is not a number. Try again.") |

1. (c) Entering the value of X

|  |
| --- |
| Sets the x\_value as 0, to be able to take in a numeric value.  x\_value = 0  Error Handling for X-Value using While Loop  Using while loop, we put in place error handling (try, except and else) to highlight issues if user enters an invalid figure.  Using Error Handling for Input  Under the try portion, we are allowing user to input the x-value between minus and plus infinity. If user has indicated a valid x-value, it will pop up the response "You have entered {x\_value\_qn} as the mean." Thereafter, it will break out of the loop.  However, if user does not enter anything since length of the input is "blank", the program will ask for an input for you.  If user does not enter a valid mean figure (i.e. invalid character symbol or characters), it will pop up a response saying that entering {x\_value\_qn} is not a number and will get user to try again until a valid response is correctly entered.  while True:  try:  x\_value\_qn = input(f"Please Enter the X Value (between minus infinity (–∞) and plus infinity (+∞)):\n")  x\_value = float(x\_value\_qn)  print(f"You have entered {x\_value\_qn} as the X Value.")  break  except ValueError:  if len(x\_value\_qn) == 0:  print(f"You are required to input a figure. Try again.")  else:  print(f"You entered {x\_value\_qn}, which is not a number. Try again.") |

1. (d) User Defined Function to Compute Probability Distribution Function (pdf) Formula

|  |
| --- |
| For the user defined function (i.e probability distribution function formula), it requires 3 values to help compute (i.e. mean\_value, variance\_value, x\_value). Formula are broken down into 3 parts to ensure that the simplified internally for programmers to view. Returns the formula response via "output\_pdf\_answer". Print is hidden for user to confirm answer response.  def formula(mean\_value,variance\_value,x\_value):    pt\_a = 1/math.sqrt(2\*math.pi\*variance\_value)  pt\_b = (-1) \* ((x\_value-mean\_value)\*\*2)/(2\*variance\_value)  pt\_c = math.exp(pt\_b)  formula = pt\_a \* pt\_c  return(formula) |
| output\_pdf\_answer = formula(mean\_value,variance\_value,x\_value)  # print(output\_pdf\_answer) |

1. (e) Formatted Printing in Question (d) to Display Results to User

|  |
| --- |
| print(f"The results for probability density function (pdf) is {output\_pdf\_answer}.") |

1. (f) Computation of Cumulative Distribution Function (cdf)

|  |
| --- |
| print("---------------------------------------------------------------------------------------------------")  print("The fomula cumulative distribution function (cdf) is 𝑃(𝑋≤𝑘)≈𝛼[𝑓𝑋(𝑎)+⋯𝑓𝑋(𝑘−2𝛼)+𝑓𝑋(𝑘−𝛼)+𝑓𝑋(𝑘)]")  print("Where α(denoted alpha), is the range of each step")  print("Where a, is a number close to –∞")  print("Where k, the value derived from Question (c)")  print("---------------------------------------------------------------------------------------------------")  alpha = 0.01 #treated as fixed values, this represents the step range  final\_value = -100 #treated as fixed values  while True:  try:  alpha\_qn = input(f"Please Enter the α(denoted alpha) value: (default is 0.01)\n")  if float(alpha\_qn) > 0:  alpha = float(alpha\_qn)  print(f"You have entered {alpha\_qn} as the α(denoted alpha).")  break  else:  print(f"You entered {alpha\_qn}, which is a negative number. Try again.")  except ValueError:  if len(alpha\_qn) == 0:  alpha = float(0.01)  print(f"You have set α(denoted alpha) as 0.01.")  break  else:  print(f"You entered {alpha\_qn}, which is not a number. Try again.")  print("\n")  while True:  try:  final\_qn = input(f"Please Enter the a-value: (default is -100)\n")  final\_value = float(final\_qn)  print(f"You have entered {final\_qn} as a.")  break  except ValueError:  if len(final\_qn) == 0:  final\_value = float(-100)  print(f"You have set a as -100.")  break  else:  print(f"You entered {final\_qn}, which is not a number. Try again.")    sr\_count = int((-final\_value + x\_value) / alpha) - 1  inner\_function\_part = 0  inner\_function\_full = 0  count\_a = 0  for number in range(0,sr\_count):  count\_a += 1  # print(count\_a)  while True:  inner\_function\_part = (x\_value - (count\_a\*alpha))  break  # print(inner\_function\_part)  # print(formula(mean\_value,variance\_value,inner\_function\_part))  inner\_function\_full += formula(mean\_value,variance\_value,inner\_function\_part)  # print(inner\_function\_full)  full\_function = formula(mean\_value,variance\_value,-(final\_value)) + inner\_function\_full + formula(mean\_value,variance\_value,x\_value)  output\_cdf\_answer = full\_function \* alpha  print("\n")  print(f"The results for P(X≤{x\_value}) is {output\_cdf\_answer}.") |

1. (g) Explanation to Question (f)

|  |
| --- |
| User can then indicate the α-alpha and a-value by entering the values. It is stated that in order to increase the accuracy of the approximation, users can decrease the value of step range α-alpha. In the example, instead of indicating the value as 0.01, user can indicate as 0.001 to increase the accuracy.  Additionally, there has to be a sufficient amount before a-value on formula eventually returns 0. User has to ensure that there is sufficient buffer, taking into account values greater than 0 before it reaches 0.  To derive the n-th count for the range, we take the addition of a-value and k-value divided by α-alpha. Thereafter, it will run continuously which will get you the full count of n-th minus 2 because the 2 are the starting and ending values which are not in the loop. The while loop will compute values and put them all in the cdf formula function. Afterwards, it will add up the summation of values.  Finally, the function will churn out the formula with full\_function adding the first and final values of the amount in the formula. Thereafter, it will be multiplied by the α-alpha value.  *Total Word Count: 193* |

1. (h) Dictionary to store Probabilities of Normal Distribution & Display Subset of Dictionary

|  |
| --- |
| We create the an empty dictionary and list for the program to fill up.  normal\_dist\_dict = {}  normal\_dist\_list = []  normal\_dist\_formula\_response = []  normal\_dist\_assign = []  Here, we take a simplified approach to generate the counts to store the dictionary.  count\_line = -5.1 #The count\_line will proceed to add 0.10, with a range loop of 101 times to get from -5.0 to +5.0.  count\_start = -2.5 #The count start will allow us to show the dictionary between -2.0 to 2.0, with a range loop of 9 times.  count\_step = 0.5 # The step is in the intervals of 0.5, hence allowing us to generate it in 0.5 steps from keys in dictionary.  For loop is used to put the probabilities of the normal distribution in keys for x values where;  x = {-5, -4.9, -4.8, …, 0, 0.1, 0.2, …, 4.8, 4.9, 5} are the keys and,  From Question (f), the results are the values of the dictionary.  for number in range(101):  count\_line += 0.1  normal\_dist\_list.append(round(count\_line,2))  normal\_dist\_formula\_response.append(formula(mean\_value,variance\_value,count\_line))  normal\_dist\_dict = dict(zip(normal\_dist\_list, normal\_dist\_formula\_response))    For loop is used to extract the probabilities of normal distribution with step width of 0.5 from the dictionary. This is in tandem by using count\_step previously outlined in our code.  We would like to churn out a subset of our dictionary from -2.0 to +2.0. Hence the keys that we want are put under normal\_dist\_assign. Hence, only want keys by looking at normal\_dist\_assign. Program will refer to normal\_dist\_assign list to produce the subset in your keys and values in your dictionary.  for number in range(9):  count\_start += count\_step  index\_a = normal\_dist\_list.index(-2.5)  normal\_dist\_assign.append(round(count\_start,2))  # print(normal\_dist\_assign)  a\_subset = {key: normal\_dist\_dict[key] for key in normal\_dist\_assign}  print(f" The probabilities of X's between -2 and 2 with a step width of 0.5 are {a\_subset}")  For this case, we assume that user would like to see values of -2.0 to +2.0 every time with the step width of 0.5  Hence, we do not ask user for a question or to key in default responses to produce the outcome faster for user to view.  However, do note that the Python programming specialist can cater to the user's demands. |

**Workings:**

|  |
| --- |
| **Question (b) - Program to Enter Mean and Variance**  Default when pressing “Enter”  Mean    Variance |
| Allowing numbers only  Mean – Negative and Positive Numbers      Variance – Positive Numbers only |
| **Question (c) - Entering the value of X** |
| **Question (e) - Formatted Printing in Question (d) to Display Results to User** |

|  |
| --- |
| **Question (f) - Computation of Cumulative Distribution Function (cdf)**  For results (μ = 0, σ2 = 1, 𝑘 = 0)    For results (μ = 0, σ2 = 1, 𝑘 = 1.64)    For results (μ = 0, σ2 = 1, 𝑘 = 1.96) |
| **Question (h) - Dictionary to store Probabilities of Normal Distribution & Display Subset of Dictionary** |

**Jupyter Notebook Attachment:**

****