

**ANL252**

**Python for Data Analytics**

# **Tutor-Marked Assignment**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Lee Min Kang Bryan** | **E1771573** |

**Tutorial Group: ­­­­­­­­­­­­ T09**

**Instructor’s Name: Mr. Munish Kumar**

**Submission Date: 11/08/2021**

**a)** To import the math package into my program, I used the import math code.



**b)** Firstly, I set the variables default\_mean and default\_variance to 0 and 1 respectively and will call on them when the user did not enter a value for the mean and variance. Next, I used a while loop to check if the mean that the user has input is a numeric input, if it is not, I will print an error message saying, “Please enter a numeric input.”. I did not force the user input to be a float because if I do that, when the user does not enter any value, there will automatically be a ValueError and I would not be able to set the default mean value as 0. Therefore, I only converted the user input to a float after checking for a blank input.

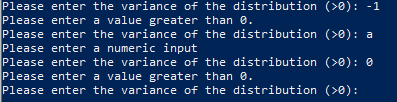
**



***Screenshot of mean error message***

Similar to the code for the mean, I also used another while loop to check if the variance the user input has any error. The first check I did is to check if the input is blank, if input is blank, the variable “variance” will be set to the default variance value of 1 as defined previously and the loop will end. If the user input is not blank, the next if statement will check if the variance entered is > 0. If yes, the user’s input will be stored in the variable “variance” and the loop will end. If not, it means the user has entered a value < 0 which will print the message “Please enter a value greater than 0.” and the loop will repeat. If throughout the code, there is a ValueError, which means that the value the user input is not numeric, I would also print the message “Please enter a numeric input.” And repeat the loop again. Similarly, I also converted the user input to a float after checking for a blank input.





***Screenshot of variance error messages***

**c)** The code that I wrote for the X value input is like the one I wrote for the input of the mean. However, for the variable X, I included a float at the input box to force the user to enter a numeric value. If a non-numeric value is entered or if there is no value entered, there will be a ValueError which will print “Please enter a numeric input.” and the code will repeat the loop again. If the input is numeric, the value the user entered will be stored under variable X.

**



***Screenshot of error message for X-value***

**d)** I used the 3 user inputs as above and substituted them into the probability distribution function (“PDF”) to calculate the probability density based on the user inputs. The formula is as below with the variable “mean” denoting the user defined mean, variable “variance” as the user defined variance and variable “X” as the random variable X. The function “math.sqrt” is to calculate the square root and the function “math.pi” is the value of the constant pi. I also formatted the output to 4 decimal places.

**

**e)** I used formatted printing to display the output to the user. Code as per below.

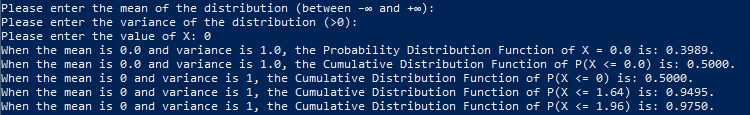




***Screenshot of formatted printing to display the PDF***

**f)**





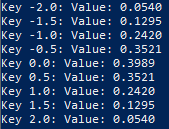
***Screenshot of user defined CDF output and the test results for k=0, 1.64 and 1.96***

g) As the CDF is a summation of all the individual PDF, I used the for loop to calculate the individual PDF from the “a” value to the user-defined X with a step width of 0.0001. According to the Empirical Rule, 99.7% of data observed following a normal distribution lies within 3 standard deviations of the mean. Therefore, I chose an “a” value of (mean – 4SD) to ensure that my formula can accurately calculate any value that the user input as opposed to if I hardcode the value of a. Instead of using a range for the for loop, I used the arange function from the NumPy package because the range function only allows integer as steps while arange allows for floats and decimals. Each time the for loop runs, it will calculate the individual PDF and add it into the variable CDF. When the for loop reaches the user-defined X value, the loop will stop automatically. The code will then divide the CDF by 10000 and store the value up to 4 decimal places. I then used formatted printing to show the user defined values of the mean, variance, and X value as well as the corresponding CDF value.

(199 words)

h) I used a for loop with range from -5 to 5 with a step width of 0.1 to create my dictionary and store the corresponding PDF values in it.





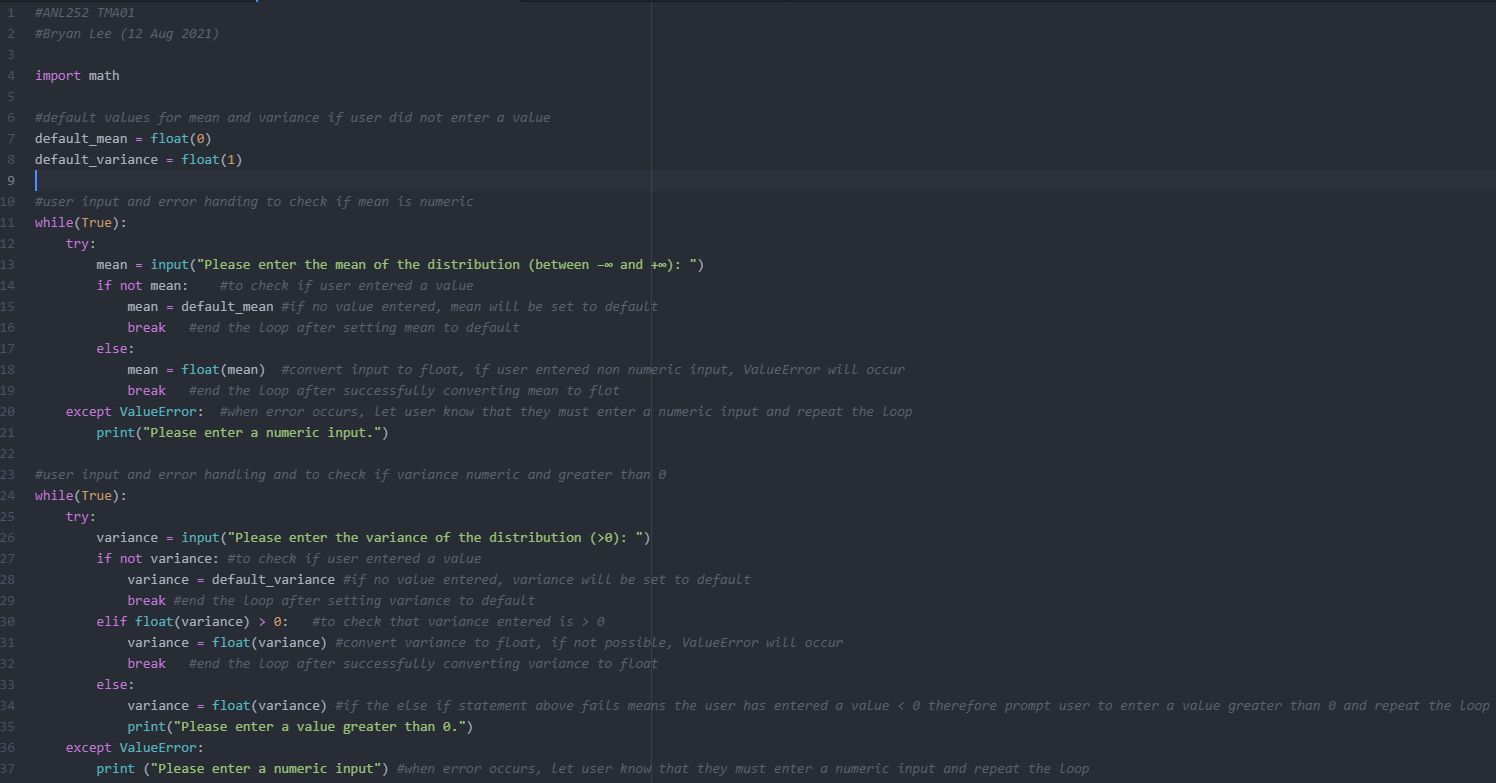
***Screenshot of dictionary output***

**Appendix**

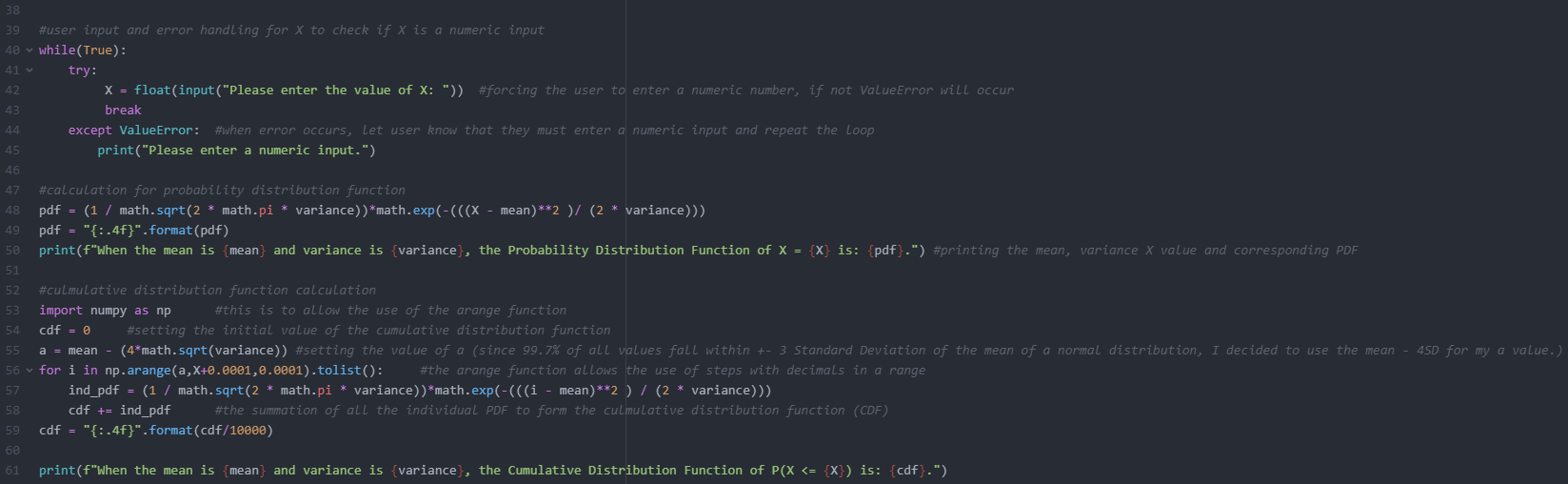
**Embedded code**

****

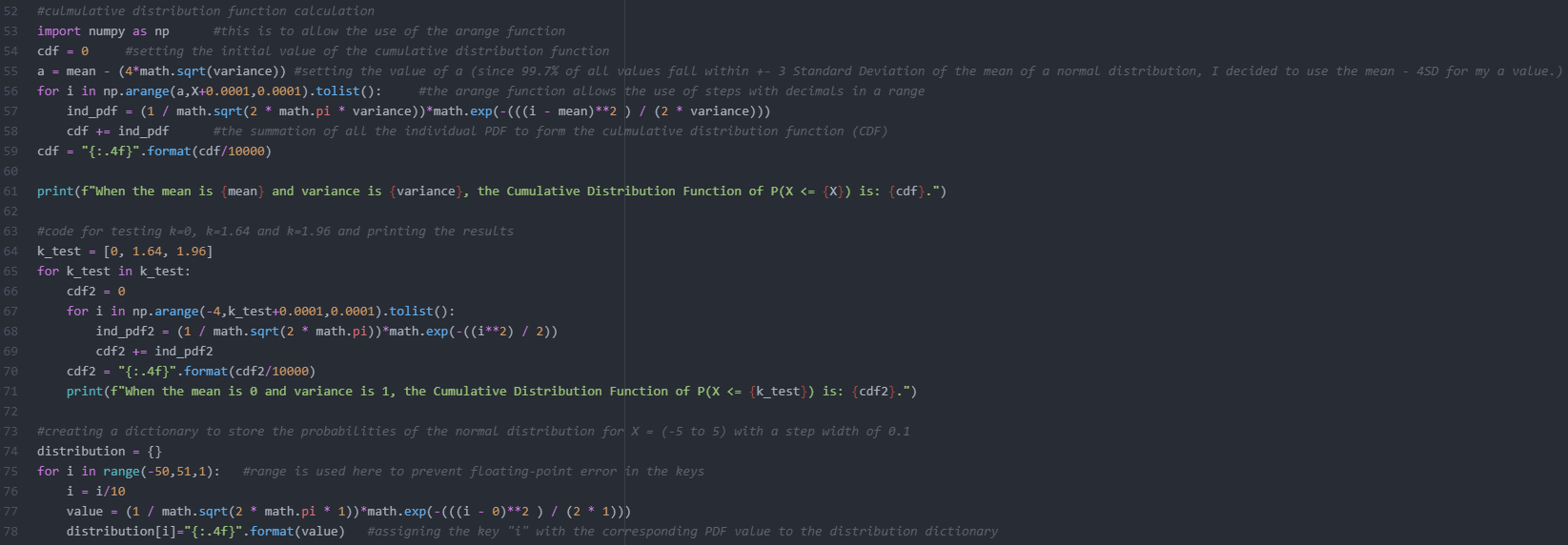
**Screenshot of code for question a and b**



**Screenshot of code for question c, d, and e**



**Screenshot of code for question f**



**Screenshot of code for question h**

