**E**stimation **A**nd **C**onfidence **I**ntervals

**Background**

In quality control processes, especially when dealing with high-value items, destructive sampling is a necessary but costly method to ensure product quality. The test to determine whether an item meets the quality standards destroys the item, leading to the requirement of small sample sizes due to cost constraints.

**Scenario**

A manufacturer of print-heads for personal computers is interested in estimating the mean durability of their print-heads in terms of the number of characters printed before failure. To assess this, the manufacturer conducts a study on a small sample of print-heads due to the destructive nature of the testing process.

**Data**

A total of 15 print-heads were randomly selected and tested until failure. The durability of each print-head (in millions of characters) was recorded as follows:

1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29

**Assignment Tasks**

**a. Build 99% Confidence Interval Using Sample Standard Deviation**

Assuming the sample is representative of the population, construct a 99% confidence interval for the mean number of characters printed before the print-head fails using the sample standard deviation. Explain the steps you take and the rationale behind using the t-distribution for this task.

**PROGRAM:**

**import numpy as np**

**from scipy.stats import t**

**# Data: Durability of print-heads in millions of characters**

**data = [1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29]**

**# Sample size**

**n = len(data)**

**# Sample mean and standard deviation**

**mean = np.mean(data)**

**std\_dev = np.std(data, ddof=1) # ddof=1 for sample standard deviation**

**# Degrees of freedom**

**df = n - 1**

**# Critical t-value for 99% confidence interval**

**confidence\_level = 0.99**

**t\_critical = t.ppf((1 + confidence\_level) / 2, df)**

**# Margin of error**

**moe = t\_critical \* (std\_dev / np.sqrt(n))**

**# Confidence interval**

**confidence\_interval = (mean - moe, mean + moe)**

**# Print results**

**print(f"Sample Mean: {mean}")**

**print(f"Sample Standard Deviation: {std\_dev}")**

**print(f"Critical t-value: {t\_critical}")**

**print(f"Margin of Error: {moe}")**

**print(f"99% Confidence Interval: ({confidence\_interval[0]}, {confidence\_interval[1]})")**

**OUTPUT:**

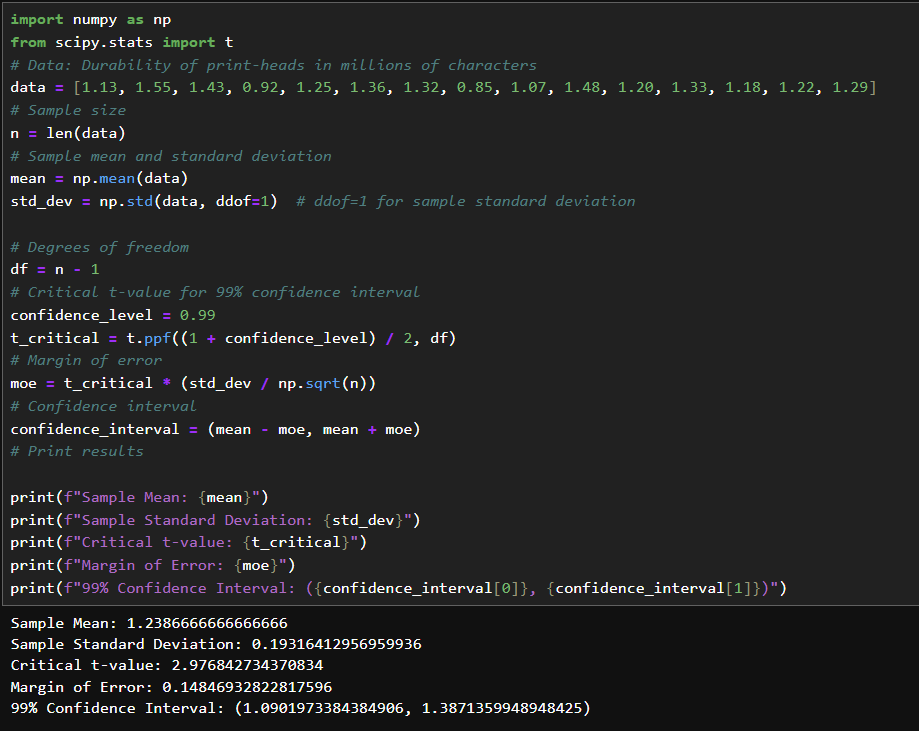
**Sample Mean: 1.2386666666666666**

**Sample Standard Deviation: 0.19316412956959936**

**Critical t-value: 2.976842734370834**

**Margin of Error: 0.14846932822817596**

**99% Confidence Interval: (1.0901973384384906, 1.3871359948948425)**



**STEPS:**

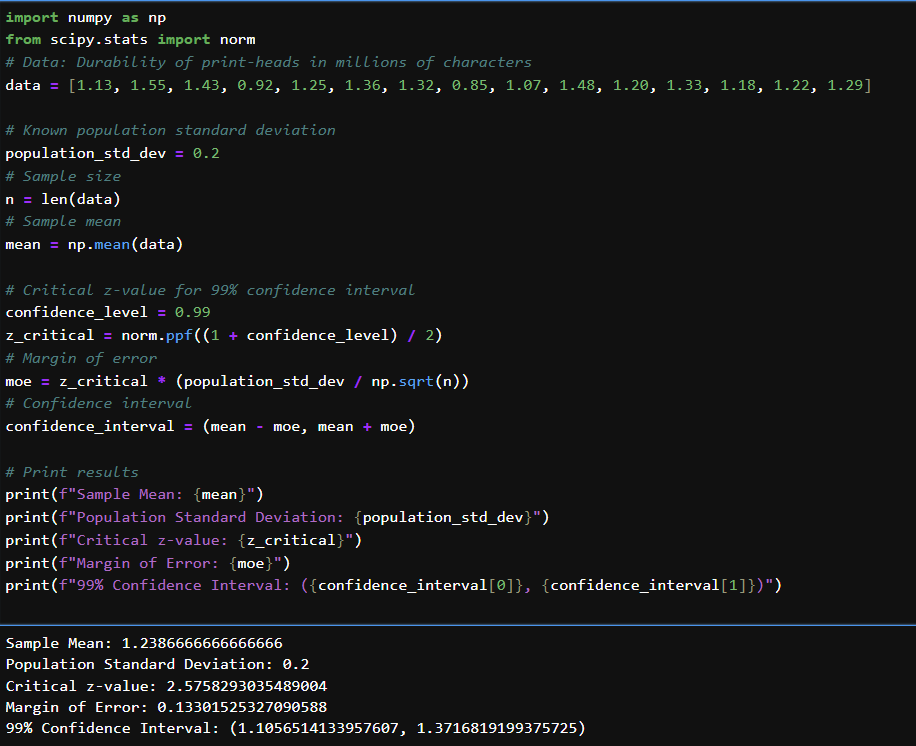
1. Understanding the Data and Context
2. Calculating Descriptive Statistics
3. Identifying the Appropriate Distribution
4. Finding the Critical t-Value
5. Calculating the Margin of Error (MOE)

**Rationale for Using the t-Distribution:**

The t-distribution is used in this task because it provides a more accurate representation of the uncertainty in the estimate of the population mean when the sample size is small and the population standard deviation is unknown. This ensures that the 99% confidence interval is reliable and accounts for the additional variability introduced by using the sample standard deviation

**b. Build 99% Confidence Interval Using Known Population Standard Deviation**

If it were known that the population standard deviation is 0.2 million characters, construct a 99% confidence interval for the mean number of characters printed before failure.



**PROGRAM:**

**import numpy as np**

**from scipy.stats import norm**

**# Data: Durability of print-heads in millions of characters**

**data = [1.13, 1.55, 1.43, 0.92, 1.25, 1.36, 1.32, 0.85, 1.07, 1.48, 1.20, 1.33, 1.18, 1.22, 1.29]**

**# Known population standard deviation**

**population\_std\_dev = 0.2**

**# Sample size**

**n = len(data)**

**# Sample mean**

**mean = np.mean(data)**

**# Critical z-value for 99% confidence interval**

**confidence\_level = 0.99**

**z\_critical = norm.ppf((1 + confidence\_level) / 2)**

**# Margin of error**

**moe = z\_critical \* (population\_std\_dev / np.sqrt(n))**

**# Confidence interval**

**confidence\_interval = (mean - moe, mean + moe)**

**# Print results**

**print(f"Sample Mean: {mean}")**

**print(f"Population Standard Deviation: {population\_std\_dev}")**

**print(f"Critical z-value: {z\_critical}")**

**print(f"Margin of Error: {moe}")**

**print(f"99% Confidence Interval: ({confidence\_interval[0]}, {confidence\_interval[1]})")**

**OUTPUT:**

**Sample Mean: 1.2386666666666666**

**Population Standard Deviation: 0.2**

**Critical z-value: 2.5758293035489004**

**Margin of Error: 0.13301525327090588**

**99% Confidence Interval: (1.1056514133957607, 1.3716819199375725)**