**PRACTICAL NO- 1**

**AIM-** Compute measures of contract tendency to calculate Mean, Median and Mode (using numpy and scipy)

**BRIEFING OF THE AIM**

Measures of central tendency are summary statistics that represent the center point or typical value of a dataset. They provide a single value that summarizes the distribution of the data. The three main measures are:

**1. Mean:** The arithmetic average of all data points. It is calculated by summing all the values in the dataset and dividing by the number of values. It is sensitive to outliers.

**2. Median:** The middle value in a dataset that has been sorted in ascending order. If the dataset has an even number of values, the median is the average of the two middle values. It is less affected by outliers than the mean.

**3. Mode:** The value that appears most frequently in a dataset. A dataset can have one mode, more than one mode, or no mode at all.

**Libraries Used:**

**\* NumPy:** A fundamental package for numerical computation in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. We will use numpy.mean() and numpy.median().

**\* SciPy:** A library used for scientific and technical computing. It builds on NumPy and provides a large number of user-friendly and efficient numerical routines, such as routines for statistical analysis. We will use scipy.stats.mode().

INDIVIDUAL SERIES

CODE:

**import** numpy **as** np

**from** scipy **import** stats

data = [1, 2, 2, 3, 4, 5, 5, 5, 6, 7]

mean\_value = np.mean(data)

print(f"Individual Series Data: {data}")

print(f"Mean: {mean\_value}")

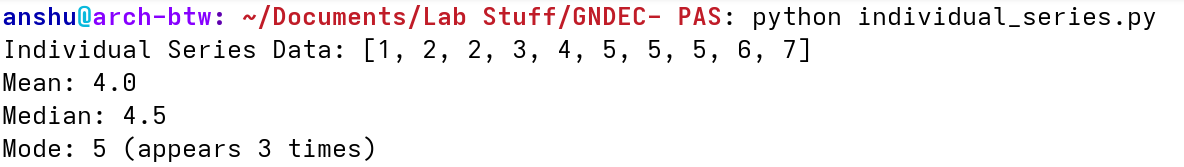
median\_value = np.median(data)

print(f"Median: {median\_value}")

mode\_result = stats.mode(data)

print(f"Mode: {mode\_result.mode} (appears {mode\_result.count} times)")

OUTPUT:



DISCRETE SERIES

CODE:

**import** numpy **as** np

**from** scipy **import** stats

x = np.array([10, 20, 30, 40, 50])

f = np.array([5, 12, 15, 8, 3])

data = np.repeat(x, f)

print(f"Discrete Series Data (x): {x}")

print(f"Frequencies (f): {f}")

print("-------------------------------------")

mean\_value = np.average(x, weights=f)

print(f"Mean: {mean\_value}")

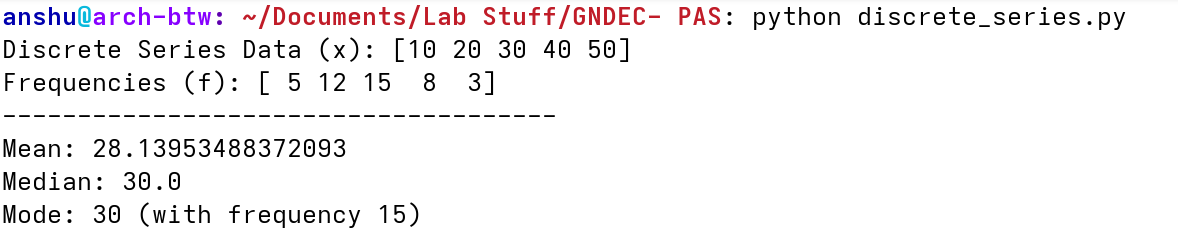
median\_value = np.median(data)

print(f"Median: {median\_value}")

mode\_value = x[np.argmax(f)]

print(f"Mode: {mode\_value} (with frequency {np.max(f)})")

OUTPUT:



CONTINUOUS SERIES

CODE:

**import** numpy **as** np

intervals = [(0, 10), (10, 20), (20, 30), (30, 40), (40, 50)]

frequency = np.array([5, 15, 25, 8, 7])

midpoints = np.array([(low + high) / 2 **for** low, high **in** intervals])

print(f"Intervals: {intervals}")

print(f"Frequencies: {frequency}")

print(f"Midpoints: {midpoints}")

print("-------------------------------------")

mean\_value = np.average(midpoints, weights=frequency)

print(f"Mean: {mean\_value:.2f}")

N = np.sum(frequency)

cumulative\_frequency = np.cumsum(frequency)

median\_class\_index = np.where(cumulative\_frequency >= N/2)[0][0]

L = intervals[median\_class\_index][0]

cf = cumulative\_frequency[median\_class\_index - 1] **if** median\_class\_index > 0 **else** 0

f = frequency[median\_class\_index]

h = intervals[median\_class\_index][1] - intervals[median\_class\_index][0]

median\_value = L + ((N/2 - cf) / f) \* h

print(f"Median: {median\_value:.2f}")

modal\_class\_index = np.argmax(frequency)

L\_mode = intervals[modal\_class\_index][0]

f1 = frequency[modal\_class\_index]

f0 = frequency[modal\_class\_index - 1] **if** modal\_class\_index > 0 **else** 0

f2 = frequency[modal\_class\_index + 1] **if** modal\_class\_index < len(frequency) - 1 **else** 0

h\_mode = intervals[modal\_class\_index][1] - intervals[modal\_class\_index][0]

mode\_value = L\_mode + ((f1 - f0) / (2\*f1 - f0 - f2)) \* h\_mode

print(f"Mode: {mode\_value:.2f}")

OUTPUT:

