First watch - List Comprehension || Python Tutorial || Learn Python Program… Python zip function 🤐

1. Mean , median , mode

```
def calculate_mean(numbers):
    return sum(numbers) / len(numbers)

def calculate_median(numbers):
    sorted_numbers = sorted(numbers)
    n = len(sorted_numbers)

    if n % 2 == 0:
        mid1 = sorted_numbers[n // 2 - 1]
        mid2 = sorted_numbers[n // 2]
        median = (mid1 + mid2) / 2
    else:
        median = sorted_numbers[n // 2]

    return median

def calculate_mode(numbers):
    frequency = {}

    for num in numbers:
        frequency[num] = frequency.get(num, 0) + 1

    max_freq = max(frequency.values())
    mode = [key for key, value in frequency.items() if value == max_freq]

    return mode

# Input the data
data = [int(x) for x in input("Enter the space-separated numbers: ").split()]

# Calculate and print the mean, median, and mode
mean_value = calculate_mean(data)
median_value = calculate_median(data)
mode_value = calculate_mode(data)

print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
```

2. GM, HM

```
def calculate_geometric_mean(numbers):
    product = 1

    for num in numbers:
        product *= num

    geometric_mean = product**(1/len(numbers))
    return geometric_mean

def calculate_harmonic_mean(numbers):
    reciprocal_sum = sum(1/num for num in numbers)

    harmonic_mean = len(numbers) / reciprocal_sum
    return harmonic_mean

# Input the data
data = [float(x) for x in input("Enter the space-separated numbers: ").split()]

# Calculate and print the geometric and harmonic mean
geometric_mean_value = calculate_geometric_mean(data)
harmonic_mean_value = calculate_harmonic_mean(data)

print(f"Geometric Mean: {geometric_mean_value}")
print(f"Harmonic Mean: {harmonic_mean_value}")
```

Ranked corr.

```
def calculate_rank(data):
    ranked_data = [sorted(data).index(x) + 1 for x in data]
    return ranked_data

def calculate_spearman_rank_correlation(x, y):
    n = len(x)

    # Calculate ranks for both variables
    rank_x = calculate_rank(x)
    rank_y = calculate_rank(y)

    # Calculate squared differences of ranks
    d = [rx - ry for rx, ry in zip(rank_x, rank_y)]
    d_squared = [d_i**2 for d_i in d]

    # Calculate Spearman's rank correlation coefficient
    r_s = 1 - (6 * sum(d_squared)) / (n * (n**2 - 1))

    return r_s

# Input the data for two variables
x_data = [float(x) for x in input("Enter the space-separated values for variable X: ").split()]
y_data = [float(y) for y in input("Enter the space-separated values for variable Y: ").split()]

# Calculate Spearman's rank correlation
correlation_coefficient = calculate_spearman_rank_correlation(x_data, y_data)

# Display the ranks and result
print("\nRanks:")
print(f"Variable X: {calculate_rank(x_data)}")
print(f"Variable Y: {calculate_rank(y_data)}")

print(f"\nSpearman's Rank Correlation Coefficient: {correlation_coefficient:.4f}")
```

Skewness

```
def calculate_mean(numbers):
    return sum(numbers) / len(numbers)

def calculate_median(numbers):
    sorted_numbers = sorted(numbers)
    n = len(sorted_numbers)

    if n % 2 == 0:
        mid1 = sorted_numbers[n // 2 - 1]
        mid2 = sorted_numbers[n // 2]
        median = (mid1 + mid2) / 2
    else:
        median = sorted_numbers[n // 2]

    return median

def calculate_standard_deviation(numbers, mean_value):
    n = len(numbers)
    squared_diff = sum((x - mean_value)**2 for x in numbers)
    variance = squared_diff / n
    std_deviation = variance**0.5
    return std_deviation

def calculate_karl_pearson_skewness(numbers):
    mean_value = calculate_mean(numbers)
    median_value = calculate_median(numbers)
    std_deviation = calculate_standard_deviation(numbers, mean_value)

    skewness = 3 * (mean_value - median_value) / std_deviation
    return skewness
```

```
# Input the data
data = [float(x) for x in input("Enter the space-separated numbers: ").split()]

# Calculate and print the Karl Pearson skewness
skewness_value = calculate_karl_pearson_skewness(data)

print(f"Karl Pearson Skewness: {skewness_value:.4f}")
```

Chi square

```
def chi_square_test(observed, expected):
    chi_square = 0

    for o, e in zip(observed, expected):
        chi_square += ((o - e)**2) / e

    return chi_square

# Input the observed and expected frequencies
observed_values = [int(x) for x in input("Enter the observed frequencies (space-separated):
").split()]
expected_values = [float(x) for x in input("Enter the expected frequencies (space-separated):
").split()]

# Perform the chi-square test
chi_square_statistic = chi_square_test(observed_values, expected_values)

# Critical value (you can replace this with the provided critical value)
critical_value = float(input("Enter the critical value for your significance level: "))

# Compare the test statistic with the critical value
if chi_square_statistic > critical_value:
    print(f"Reject the null hypothesis. There is evidence of a significant relationship.")
else:
    print("Fail to reject the null hypothesis. There is no significant relationship.")
```

OR

```
data = []
for i in range(int(input("Enter the number of terms: "))):
    ob = int(input(f"Enter the observed frequency {i + 1}: "))
    ex = int(input(f"Enter the expected frequency {i + 1}: "))
    data.append((ob, ex))

print("Items:", data)

chi = 0
for o, e in data:
    chi += ((o - e) ** 2 / e)

print("Calculated Chi-square:", chi)

critical_value = float(input("Enter the critical value for chi-square test: "))

if chi > critical_value:
    print("Reject the null hypothesis")
else:
    print("Accept the null hypothesis")
```