1. function: arithmetic_mean
    a. input: list of numbers
    b. output: arithmetic mean of the given list
2. The function first checks to see if the input list any elements.
    a. if the list is empty, the function returns None.
3. After ensuring that the list is not empty, the function computes the sum of the elements of the list.
4. The len function counts the number of elements in the input list.
5. In both cases, the float function ensures that the number is not rounded or truncated.
6. The sum is divided by the length to give the arithmetic mean.

1. function: variance
    a. input: list of numbers
    b. output: variance of the given list
2. Outside the while loop:
    a. The empty list meanlist is provided for later appendation.
    b. The function makes use of a counter, and it is set at 0 to begin.
3. The while loop operates as long as the counter is less than the length of the list. Every time the while loop runs, the counter increases by one.
4. Inside the while loop:
    a. The empty list is appended with the mean of the input list. Basically what this accomplishes is that it creates a list of the same size as the input list, but all of the values are equal to the mean. This sets up the list to be able to be used with the zip function.
5. After the while loop completes, the input list and meanlist are zipped together. For loop specifies for each tuple in the zipped list, the difference between the two elements of the tuple is squared.
    a. The resulting value is appended to a new empty list.
6. The arithmetic mean of the new list is calculated using a called function. This is the variance.

1. function: lower_median
    a. input: list of numbers
    b. output: lower median of the given list
2. The function first checks to see if the input list any elements.
    a. if the list is empty, the function returns None.
3. After ensuring that the list is not empty, the input list is copied and the copied list is sorted in ascending order.
4. Modulo is used to determine whether or not the list has an even or odd number of elements. Based on this, the index of where the lower median value is located will be calculated.
5. Finally, the value at that index will be returned.

1. function: random_sample
    a. input: list of data and size of sample
    b. output: new list using elements of original
2. random module imported
3. Outside the while loop:
    a. To avoid mutating the input list, it is first copied into a new list.
    b. At the same time, an empty list is created for future appendation.
4. The while loop operates as long as the counter is less than the sample size. Every time the while loop runs, the counter increases by one.
5. Inside the while loop:
    a. A variable is assigned to randomly choose a value from the copied input list. This number is appended to the empty list.
    b. The chosen value is removed from the copied input list using the pop function so it cannot be chosen again when the while loop is repeated.
    c. The value of the counter increases by 1.
    d. This occurs until the value of the counter is no longer less than the sample size. At this time, the new list will have been appended the number of times needed for it to be the sample size specified by the input to the random_sample function.
6. At this time, newlist can be returned.

1. function: hist_data
    a. input: list of scores
    b. output: list specifying how many of the scores fall in each of 5 ranges
2. Outside the for loop:
    a. Five counters are created, to count for 5 if statements in the for loop.
    b. The reason each statement uses if instead of using elifs is because this is a “hard” if. The function checks every if statement regardless of if it finds one that the score fits the criteria of. This is necessary because each if statement has a secondary condition nested underneath. This could also be done by using and statements and elifs instead of nested ifs and ifs.
3. Each of the five if statements in the for loop works the same way, so I will only explain one of them.
    a. The first if statement sets the low value for the range. If the score passes this, the second if statement inside the first one sets the high value of the range.
    b. If the score fits both criteria, the value of the counter increases by 1.
4. The values of all 5 counters are returned in a list.

1. function: sma
    a. input: list of numbers and window size
    b. output: list of simple moving averages
2. Any empty list is created for future appendation.
3. A counter is used so that the while loop can be run a specific number of times.
4. An if statement specifies that if the window size is greater than the length of the list, the window size is changed to equal the length of the list. This is because sma requires the list to have at least as many components as the window size.
5. Inside the while loop:
    a. A variable is set to represent the arithmetic mean of a section of the list of numbers.
        i. This section is specified by using the index function. The index starts at the value of the counter and ends at the value of the counter plus the window size.
        ii. This number is appended to the empty list created. Then the value of the counter increases by one.
        iii. The while loop runs as long as the sum of the counter and the window size is less than 1 greater than the length of the list of numbers.
            1. This (the 1 greater) is because the index function runs until the specified end, but does not include the specified end.
6. At this point, newlist can be returned.

1. function: one_sample_t_test
    a. input: list of population and sample size
    b. output: t value and randomized sample list
2. math module imported
3. The sample list is defined using the previous random_sample function on the input population list.
4. Both the population list, sample list, and sample size are used to calculate the t value, using the components and equations below.
5. The components of the t-test equation are defined.
    a. x = arithmetic mean of sample
    b. o = standard deviation of population
    c. n = sample size
    d. s = standard deviation of sample
6. The equations to get t are defined.
    a. z = x / (o / sqrt(n))
    b. t = z / s
7. At this point, the value of t and the sample list can be returned.