Objective: This worksheet explains important functions that are available with the module ***statistics*** to calculate the central tendency and spread. The module DOES NOT use any third part library like NumPy, SciPy etc. The sheet does not explain few new functions which are made available in Python-3.8.

Reference: https://docs.python.org/3/library/statistics.html

- **Import the statistics module**: It is a common practice in Python programming to import a module using an alias. The import statement is done for this purpose that uses the following syntax. It helps to keep a short name for the module as per your convenience.

```
import [module] as [alias]
```

- The following example shows that ***statistics*** module is imported as ***stat*** alias.

Statistics

```
In [1]: import statistics as stat
```

- Now different functions available in the ***statistics*** module will be reviewed. They will be accessed using the alias ***stat***.

1. ***mean (data):*** calculates the mean (average) through a list of data or iterable.

```
stat.mean([30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110])
```

```
58
```

```
stat.mean(range(1, 10, 3))
```

```
4
```

2. ***harmonic_mean (data):*** calculates the harmonic_mean through a list of data or iterable. Harmonic mean for n real positive numbers (x_i, x_2,….x_n) is defined as:

$$Harmonic\ Mean = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}$$

```
stat.harmonic_mean([40.0, 60.2, 25.2, 98.6])
```

```
43.746530192519934
```

Exercise:

Harmonic Mean is defined only for positive numbers. Observe what happens when you provide a negative number. Handle the error (if there is any) using ***Try/Except*** block of code.

3. ***median (data):*** It returns the middle value of the numeric data points when organized in the increasing order. When the number of data points is even, the median is calculated by taking the average of the two middle values, otherwise the middle value is returned.

```
data1 = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110]
stat.median(data1)
```

```
54.0
```

```
data2 = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70]
stat.median(data2)
```

```
52
```

Exercise:

i. Verify if the ***median ()*** function works properly when the input data points are not arranged in the increasing order?
ii. Why in the first example above the result is reported as a float value and as an integer in the second example?

4. ***median_low (data):*** When the number of data points is odd, the middle value is returned. When it is even, the smaller of the two middle values is returned.

```
data1 = [30, 36, 47, 50, 52, 52, 56, 110, 63, 70, 70, 60]
stat.median_low(data1)
```

```
52
```

```
data2 = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70]
stat.median_low(data2)
```

```
52
```

Exercise:

There is a corresponding function ***median_high ().*** Experiment and find out what it does.

5. ***median_grouped (data, interval):*** Let us say there is a grouped data shown below for which approximate median is to be calculated. From the methods explained in the session slides it would be = 13.33.

Class	5-10	10-15	15-20
Frequency	5	6	7

Using Python ***statistics*** module and ***median_grouped ()*** function, it can be calculated in the following way:

```
data = [7.5, 7.5, 7.5, 7.5, 7.5, 12.5,12.5,12.5,12.5,12.5,12.5,17.5, 17.5, 17.5, 17.5, 17.5, 17.5, 17.5]
stat.median_grouped(data, interval=5)
```

```
13.333333333333334
```

Note that, 7.5, 12.5 and 17.5 are the mid-points for the given classes and they are occurring in the data frequency many times (7.5 is present 5 times, 12.5 is present 6 times and 17.5 is present 7 times) with interval as the class difference of 5. The default value of optional parameter ***interval*** is 1, which it also takes when it is not supplied.

6. ***mode (data):*** it returns the most frequently occurred value from the discrete or nominal data. This function is not capable to handle multimodal data.

```
data = [1.14, 2.14, 2.14, 2.14, 3.14, 4.14]
stat.mode(data)
```

```
2.14
```

```
data = ['Mars', 'Venus', 'Venus', 'Jupiter']
stat.mode(data)
```

```
'Venus'
```

7. ***pvariance (data, mu):*** calculates the population variance. Variance is also called second moment about the mean. The second argument is optional. If it is provided and if it is not the population mean, the function calculates the second moment about that number.
8. ***pstddev (data, mu):*** calculates the population standard deviation.

```
data = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110]
print(stat.pvariance(data))
print(stat.pstdev(data))
```

```
379.1666666666667
19.47220240924654
```

9. ***variance (data, xbar):*** calculates the sample variance. The value of xbar can be missing or it can be set to None. Any other value may lead to an invalid result.
10. ***stddev (data, xbar):*** calculates the sample standard deviation.

```
data = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110]
print(stat.variance(data))
print(stat.stdev(data))
```

```
413.6363636363636
20.338052110179177
```