Basic Statistics with Python

Data Types

- There are two main types of data:
    - Qualitative (or 'categorical') and
    - quantitative (or 'numerical').

Qualitative Data

- Information about something that can be sorted into different categories, and that can't be described directly by numbers.
- For xxamples:
    - Nationality
    - Professions

Quantitative Data

- Information about something that is described by numbers.
- For examples:
    - Income
    - Height

Numerical Data

- Numerical data are numbers, and can be split into two numerical categories:
    - Discrete Data

      -Continuous Data

Discrete Data

- Numbers that are limited to integers. Example: The number of cars passing by.

Continuous Data

- Numbers that are of infinite value. Example: The price of an item, or the size of an item.

Categorical Data

- Categorical data are values that cannot be measured up against each other. Example: a color value, or any yes/no values.

Mean

- The mean value is the average value of a dataset.
- To calculate the mean, find the sum of all values, and divide the sum by the number of values.
- We can use `mean()` method from NumPy or Statistics module to find the average.

In [19]:
```
import numpy as np
import statistics

my_data = np.random.randint(10, size=7)

my_mean_np = np.mean(my_data).round(2)

print(f"Data: {my_data}")
print(f"Mean: {my_mean}")
```

```
Data: [8 5 5 7 6 3 5]
Mean: 4.71
```

Median

- The median is the middle value in a data set ordered from low to high.
- We can use the NumPy `median()` method to find the median.

In [21]:
```
my_data = np.random.randint(10, size=7)
my_data_sorted = np.sort(my_data)

my_median = np.median(my_data)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Median: {my_median}")
```

```
Data: [9 7 1 8 7 1 3]
Sorted Data: [1 1 3 7 7 8 9]
Median: 7.0
```

- If there are two numbers in the middle, then median will be the average of those two middle numbers.

In [20]:
```
my_data = np.random.randint(10, size=4)
my_data_sorted = np.sort(my_data)

my_median = np.median(my_data)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Median: {my_median}")
```

```
Data: [6 2 3 2]
Sorted Data: [2 2 3 6]
Median: 2.5
```

Mode

- The Mode value is the value that appears the most number of times in the dataset.
- We can use `mode()` method from Statistics module.

In [22]:
```
my_data = np.random.randint(10, size=7)
my_data_sorted = np.sort(my_data)

my_mode = statistics.mode(my_data)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Mode: {my_mode}")
```

```
Data: [7 2 1 4 6 4 9]
Sorted Data: [1 2 4 4 6 7 9]
Mode: 4
```

Standard Deviation

- Standard deviation is a number that describes how spread out the values are in the dataset from the mean value.
- A low standard deviation means that most of the numbers are close to the mean (average) value of the dataset.
- A high standard deviation means that the values are spread out over a wider range from the mean value.
- We can use the NumPy `std()` method to calculate standard deviation.

In [32]:
```
# Example of low standard deviation
my_data = np.random.randint(10, size=7)
my_data_sorted = np.sort(my_data)

my_mean = np.mean(my_data).round(2)
my_std = np.std(my_data).round(2)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Mean: {my_mean}")
print(f"Standard deviation: {my_std}")
```

```
Data: [3 6 5 4 7 6 5]
Sorted Data: [3 4 5 5 6 6 7]
Mean: 5.14
Standard deviation: 1.25
```

In [33]:
```
# Example of high standard deviation
my_data = np.random.randint(100, size=7)
my_data_sorted = np.sort(my_data)

my_mean = np.mean(my_data).round(2)
my_std = np.std(my_data).round(2)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Mean: {my_mean}")
print(f"Standard deviation: {my_std}")
```

```
Data: [31 62 38 72 20 34  6]
Sorted Data: [ 6 20 31 34 38 62 72]
Mean: 37.57
Standard deviation: 21.19
```

Variance

- Variance is another number that indicates how spread out the values are in the dataset.
- We can get variance by multiplying standard deviation by itself.
- Or, on the other hand, if you take the square root of the variance, you get the standard deviation!
- However, we can use the NumPy `var()` method to calculate variance.

In [51]:

```
# Example of high standard deviation
my_data = np.random.randint(10, size=7)
my_data_sorted = np.sort(my_data)

my_mean = np.mean(my_data).round(2)
my_std = np.std(my_data).round(2)
sq_std = pow(my_std, 2).round(2)
my_var = np.var(my_data).round(2)
sqrt_var = statistics.sqrt(my_var)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"Mean: {my_mean}")
print(f"Std: {my_std}")
print(f"Square of Std: {sq_std}")
print(f"Variance: {my_var}")
print(f"Sqrt of Variance: {sqrt_var:.2f}")
```

```
Data: [1 2 0 8 4 8 5]
Sorted Data: [0 1 2 4 5 8 8]
Mean: 4.0
Std: 2.98
Square of Std: 8.88
Variance: 8.86
Sqrt of Variance: 2.98
```

Percentiles

- Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than.
- That means, percentiles are values that separate the data into `100` equal parts.
- For example, The 95th percentile separates the lowest `95%` of the values from the top `5%`.
- The 25th percentile is the same as the first quartile (Q1).
- The 50th percentile is the same as the second quartile (Q2).
- THe 75th percentile is the same as the third quartile (Q3)
- We can use the NumPy `percentile()` method to find the percentiles.

In [53]:

```
# Example of percentiles
my_data = np.random.randint(100, size=100)
my_data_sorted = np.sort(my_data)

p_25 = np.percentile(my_data, 25)
p_50 = np.percentile(my_data, 50)
p_75 = np.percentile(my_data, 75)
p_90 = np.percentile(my_data, 90)

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print(f"25th percentile: {p_25:.2f}")
print(f"50th percentile: {p_50:.2f}")
print(f"75th percentile: {p_75:.2f}")
print(f"90th percentile: {p_90:.2f}")
```

```
Data: [15  1 89 14 42 12 53 72 12 37 59 90 40 11 66  6 50 14  4 27 82 29 30 35
 95 38 66  8 16 48 75 67 49 64 18 76 69 51  4 10 84 89 59 42 32 32  7 87
 89 95 66 49 91 54  2 95 19 82 92 82 46 99 82 78 66 16 54 56 87 31 83 88
 66 16 53 75 85 38 37 55 25 69 54 62 87 46 41 10  3 26 75 86 36 82 14 14
 64 99 55 38]
Sorted Data: [ 1  2  3  4  4  6  7  8 10 10 11 12 12 14 14 14 14 15 16 16 16 18 19 25
 26 27 29 30 31 32 32 35 36 37 37 38 38 38 40 41 42 42 46 46 48 49 49 50
 51 53 53 54 54 54 55 55 56 59 59 62 64 64 66 66 66 66 66 67 69 69 72 75
 75 75 76 78 82 82 82 82 82 83 84 85 86 87 87 87 88 89 89 89 90 91 92 95
 95 95 99 99]
25th percentile: 26.75
50th percentile: 53.00
75th percentile: 76.50
90th percentile: 89.00
```

Quartiles

- Quartiles are values that separate the data into four equal parts.
- 0 percentile = 0 quartile
- 25th percentile = 1st quartile
- 50th percentile = 2nd quartile
- 75th percentile = 3rd quartile
- 100th percentile = 4th quartile
- We can use the NumPy `percentile()` or `quantile()` method to find the quartiles.

In [83]:

```
# Example of quartiles
my_data = np.random.randint(100, size=100)
my_data_sorted = np.sort(my_data)

# using parcentile()
q0 = np.percentile(my_data, 0)
q1 = np.percentile(my_data, 25)
q2 = np.percentile(my_data, 50)
q3 = np.percentile(my_data, 75)
q4 = np.percentile(my_data, 100)

# using quantile()
q = np.quantile(my_data, [0, 0.25, 0.5, 0.75, 1])

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print()
print("Using percentile() method:")
print(f"0 Quartile: {q0:.2f}")
print(f"1st Quartile: {q1:.2f}")
print(f"2nd Quartile: {q2:.2f}")
print(f"3rd Quartile: {q3:.2f}")
print(f"4th Quartile: {q4:.2f}")
print()
print("Using quantile() method:")
print(f"0 Quartile: {q[0]:.2f}")
print(f"1st Quartile: {q[1]:.2f}")
print(f"2nd Quartile: {q[2]:.2f}")
print(f"3rd Quartile: {q[3]:.2f}")
print(f"4th Quartile: {q[4]:.2f}")
```

```
Data: [68 21 22 31 73 14 72 32 80 10 96 10 97  1 38 98 85 74 39 12  1 42 71 13
 84 36 14 53 85  5 87 10 39 87 69 10 29 34 33 33 82 50 88 48  2  3 98 62
  2 86 46 91  1 95 89 27 33 63 57 25 79 23 86 30 81 22 39 29 73 98 91 94
 18 81 12 91 26 21 15 20  6 74 45 66 97 53 49 14 50 29 29 79 78 48 13 47
 73 43 15 84]
Sorted Data: [ 1  1  1  2  2  3  5  6 10 10 10 10 12 12 13 13 14 14 14 15 15 18 20 21
 21 22 22 23 25 26 27 29 29 29 29 30 31 32 33 33 33 34 36 38 39 39 39 42
 43 45 46 47 48 48 49 50 50 53 53 57 62 63 66 68 69 71 72 73 73 73 74 74
 78 79 79 80 81 81 82 84 84 85 85 86 86 87 87 88 89 91 91 91 94 95 96 97
 97 98 98 98]

Using percentile() method:
0 Quartile: 1.00
1st Quartile: 21.75
2nd Quartile: 45.50
3rd Quartile: 79.25
4th Quartile: 98.00

Using quantile() method:
0 Quartile: 1.00
1st Quartile: 21.75
2nd Quartile: 45.50
3rd Quartile: 79.25
4th Quartile: 98.00
```

- 0 quartile is the smallest value in the data.
- 1st quartile is the value separating the first quarter from the second quarter of the data.
- 2nd quartile is the middle value (median), separating the bottom from the top half.
- 3rd quartile is the value separating the third quarter from the fourth quarter
- 4th quartile is the largest value in the data.

Interquartile Range

- Interquartile range is the difference between the first and third quartiles (Q1 and Q3).
- The 'middle half' of the data is between the first and third quartile.
- The first quartile is the value in the data that separates the bottom `25%` of values from the top `75%`.
- The third quartile is the value in the data that separates the bottom `75%` of the values from the top `25%`.
- We can use the the SciPy `iqr()` method to find the interquartile range.

In [86]:

```
# Example of inerquartile range

from scipy import stats

my_data = np.random.randint(100, size=100)
iqr = stats.iqr(my_data)

q = np.quantile(my_data, [0, 0.25, 0.5, 0.75, 1])

print(f"Data: {my_data}")
print(f"Sorted Data: {my_data_sorted}")
print()
print(f"Interquartile Range: {iqr:.2f}")
print()
print(f"1st Quartile: {q[1]:.2f}")
print(f"3rd Quartile: {q[3]:.2f}")
print(f"Difference between Q1 and Q3: {q[3]-q[1]:.2f}")
```

```
Data: [49 20 61 54 49 49 43 73  2 71 27 97 75 30 20 70 37  4 65 84 77 79  8 74
 13 83 63 17  5 63 69 82 41 65 47  0 65 29 83 77 82 33  0 85 57 75 14 82
  3 15 46 40 93 74 16 11 77 94 33 33 34 39 93 30 53 21 25 20  0 88 10 30
 56  9 88 77 36 89 84 40 10 36 32 97 41 24 72 31  7 75 78  4 77 42 38 34
 37 95 10  4]
Sorted Data: [ 1  1  1  2  2  3  5  6 10 10 10 10 12 12 13 13 14 14 14 15 15 18 20 21
 21 22 22 23 25 26 27 29 29 29 29 30 31 32 33 33 33 34 36 38 39 39 39 42
 43 45 46 47 48 48 49 50 50 53 53 57 62 63 66 68 69 71 72 73 73 73 74 74
 78 79 79 80 81 81 82 84 84 85 85 86 86 87 87 88 89 91 91 91 94 95 96 97
 97 98 98 98]

Interquartile Range: 51.75

1st Quartile: 23.25
3rd Quartile: 75.00
Difference between Q1 and Q3: 51.75
```

In [ ]: