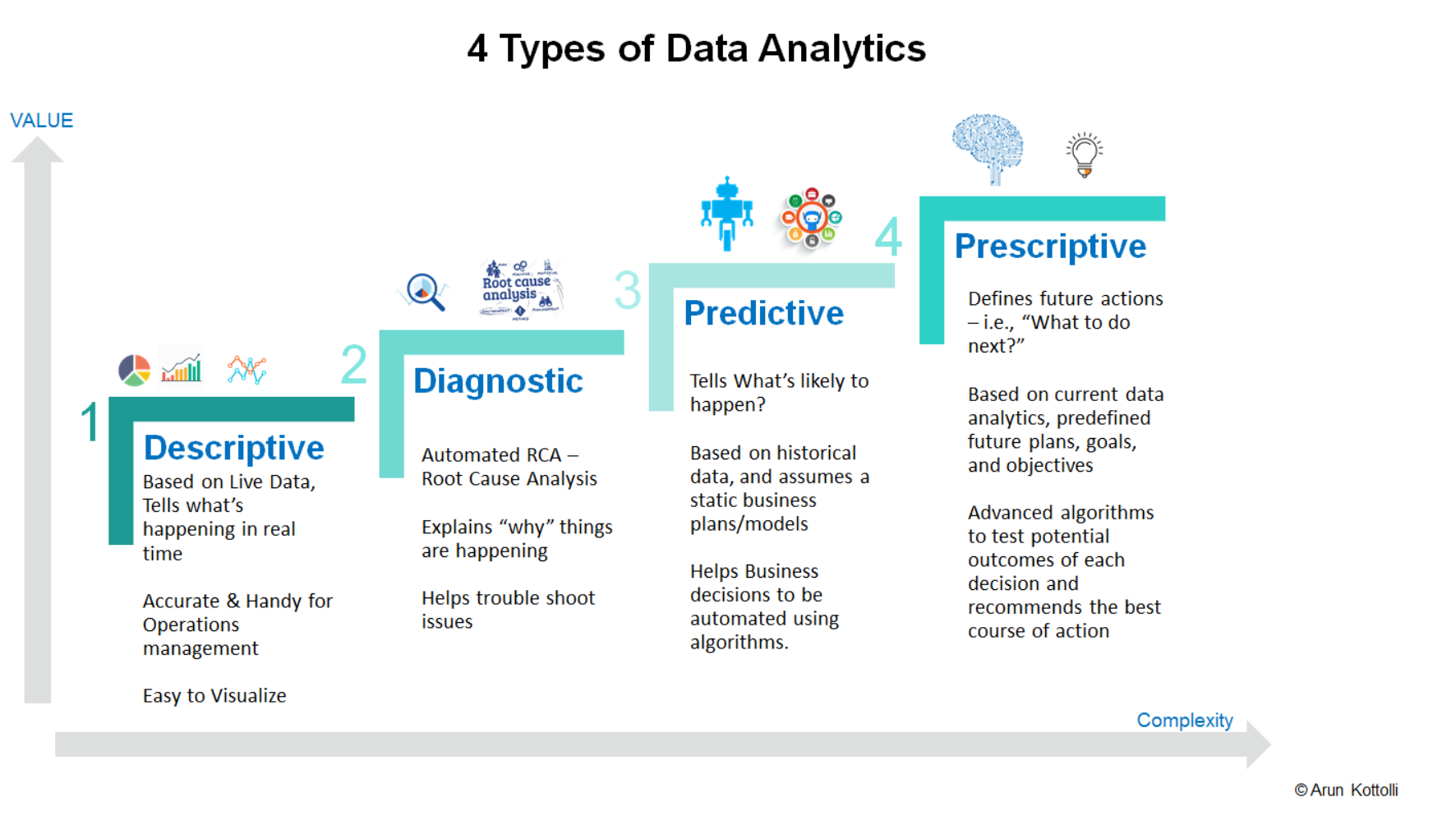
Today's Agenda

- Types of Data Analytics
- Python Libraries for DA
- Codes and Examples of various statistical measurements
- Data Visualization
- Outlier Detection

In [ ]:



Credits - Image from Internet

In [ ]:

Python tools for Data Analysis

Credits - Image from Internet

Other libraries

- OpenCV
- Pandas Profiling
- GeoPandas
- Plotly
- Statsmodel
- Statistics
- Tensorflow
- Pytorch
- -
- Dash
- Flask

Basic Statistics

- Mean → Average of all the data values
  - Sum of all data values divied by total number of data values
- Median → The value separating the higher half from the lower half of the data
- Mode → The value that appears most frequently in the data set
- Standard deviation → Used to measure of the amount of variation or dispersion of a set of values
  - Low standard deviation → All values very close to mean
  - High standard deviation → All values are far from the mean

`import` the necessary packages

In [1]:
```
import pandas as pd
import numpy as np
from collections import Counter
```

Mean

In [2]:
```
def calculate_mean(data_values):
    return sum(data_values) / len(data_values)
```

In [3]:
```
# show example

x_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
m_x = calculate_mean(data_values=x_list)
print(m_x)
```

```
5.0
```

Median

In [4]:
```
int(5 / 2)
```

Out[4]: 2

In [5]:
```
5 // 2
```

Out[5]: 2

In [6]:
```
def calculate_median(data_values):
    mid_index = len(data_values) // 2
    sorted_values = sorted(data_values)

    if len(data_values) % 2 != 0:
        median = sorted_values[mid_index]
    else:
        mid_index_l = mid_index - 1
        mini_data = [sorted_values[mid_index_l], sorted_values[mid_index]]
        median = calculate_mean(mini_data)

    return median
```

In [7]:
```
# show example

x_list = [3, 2, 6 , 38, 45, 1, 67]
med_x = calculate_median(data_values=x_list)
print(med_x)
```

```
6
```

Mode

In [8]:
```
# show dictionary example
d = [1, 1, 2, 3, 4, 4, 5, 6, 7]
```

In [9]:
```
def calculate_mode(data_values):
    data_counter = Counter(data_values)
    max_freq = max(list(data_counter.values()))

    if max_freq == 1:
        return "Mode doesn't exist"

    mode = [i for i, j in data_counter.items() if j == max_freq]
    return min(mode)
```

In [10]:
```
# show example
x_list = [1, 2, 3]
mo_x = calculate_mode(data_values=x_list)
print(mo_x)
```

```
Mode doesn't exist
```

Standard deviation

Formula

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}} \rightarrow i = 1, 2, 3, \dots n$$

where

- σ = Standard deviation
- x_i = each data value
- μ = Mean
- N = Total size of the data

In [11]:
```
def calculate_stddev(data_values):
    return np.std(a=data_values)
```

In [12]:
```
# show example

x_list = [1, 1, 2, 3, 4, 4, 5, 6, 7]
std_v = calculate_stddev(data_values=x_list)
print(std_v)
```

```
2.0
```

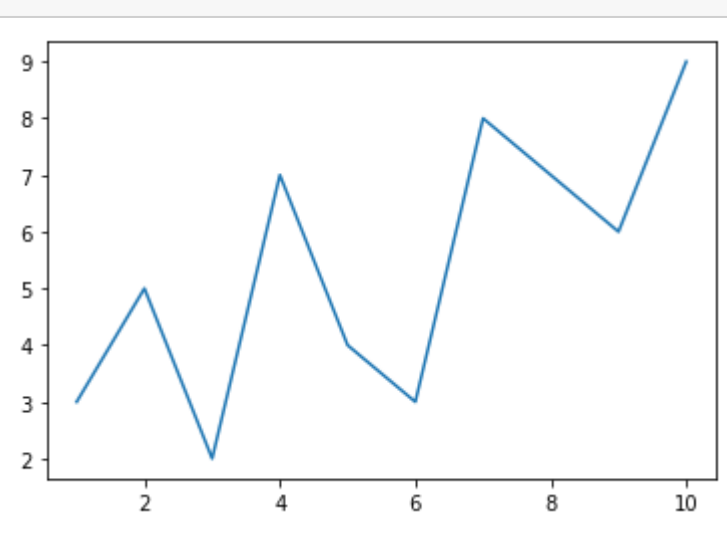
In [ ]:

Data visualization

In [13]:
```
from matplotlib import pyplot as plt
```

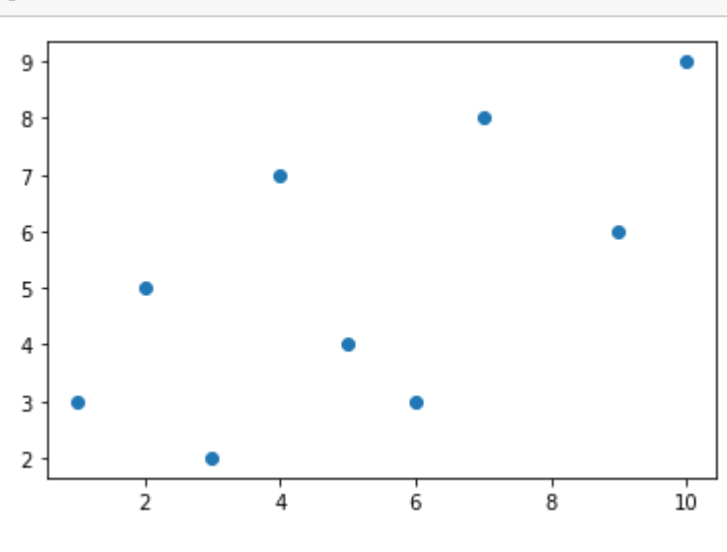
Line Plot

In [14]:
```
x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

# show example
plt.plot(x, y)
plt.show()
```



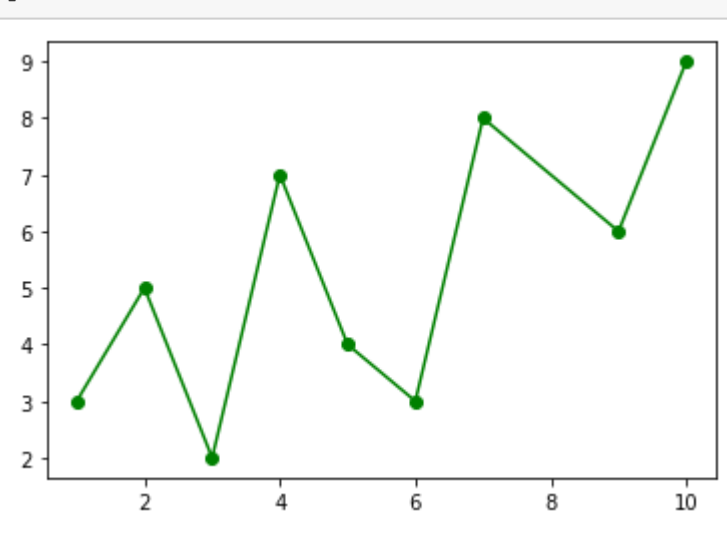
Scatter Plot

In [15]:
```
x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

# show example
plt.scatter(x, y)
plt.show()
```



Line and Scatter together

In [16]:
```
x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

# show example
plt.plot(x, y, 'o-g')
plt.show()
```



Read data

In [17]:
```
df = pd.read_csv('students_hw.csv')
df.head()
```

Out[17]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

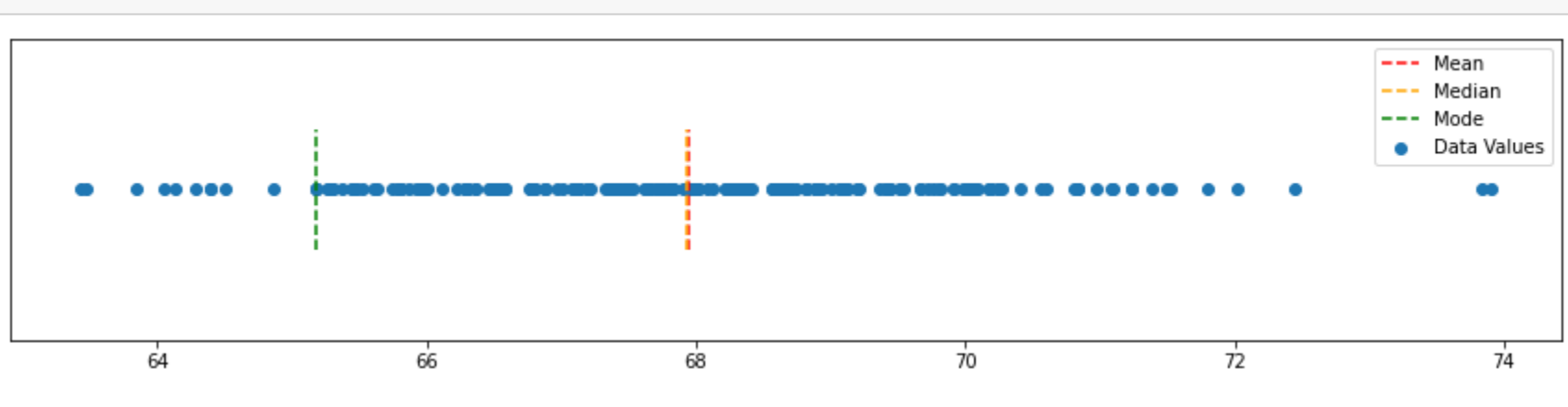
Heights → Mean, Median, Mode

In [18]:
```
x_list = df['Height(Inches)'].to_list()
y_list = [0 for i in range(len(x_list))]
```

In [19]:
```
mean_val = calculate_mean(data_values=x_list)
median_val = calculate_median(data_values=x_list)
mode_val = calculate_mode(data_values=x_list)
print(mean_val)
print(median_val)
print(mode_val)
```

```
67.94979999999998
67.935
65.18
```

In [20]:
```
plt.figure(figsize=(15, 3))

plt.yticks([])

plt.scatter(x_list, y_list, label='Data Values')

plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='red', label='Mean')
plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='orange', label='Median')
plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='green', label='Mode')

plt.legend()

plt.show()
```



In [ ]:

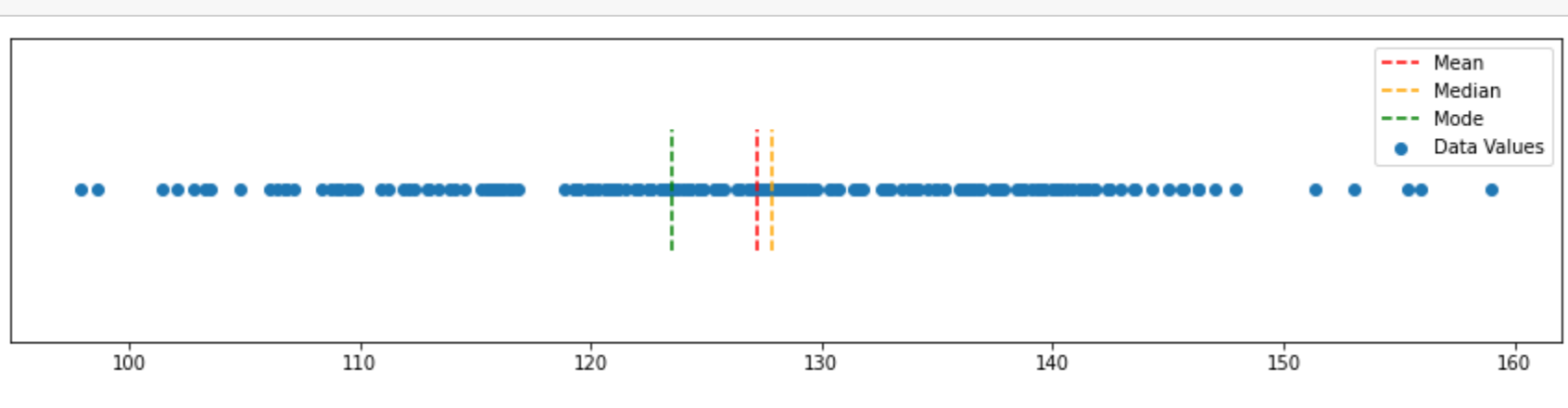
Weights → Mean, Median, Mode

In [21]:
```
x_list = df['Weight(Pounds)'].to_list()
y_list = [0 for i in range(len(x_list))]
```

In [22]:
```
mean_val = calculate_mean(data_values=x_list)
median_val = calculate_median(data_values=x_list)
mode_val = calculate_mode(data_values=x_list)
print(mean_val)
print(median_val)
print(mode_val)
```

```
127.2219500000001
127.875
123.49
```

In [23]:
```
plt.figure(figsize=(15, 3))

plt.yticks([])

plt.scatter(x_list, y_list, label='Data Values')

plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='red', label='Mean')
plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='orange', label='Median')
plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='green', label='Mode')

plt.legend()

plt.show()
```



In [ ]:

Outliers

- An outlier is a data point that differs significantly from other data values

In [24]:
```
x = [1, 2, 3, 4, 5, 6, 7, 9, 10, 50, 5, 6, 9, 4, 7]
y = [3, 5, 2, 7, 4, 3, 8, 6, 9, 65, 6, 8, 3, 6, 9]
```

How can we detect outliers?

1. By graphing
   - scatter plot
   - box plot
2. By calculating z_score values
   - if z_score value is > 3 → reject
   - if z_score value is < -3 → reject

Formula

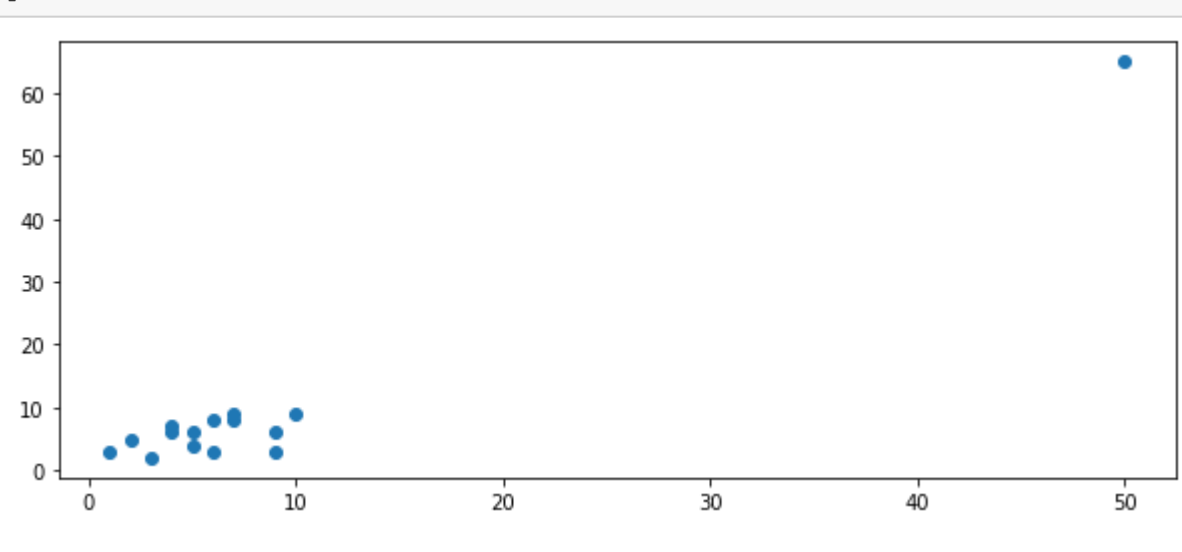
$$z = \frac{(x_i - \mu)}{\sigma} \rightarrow i = 1, 2, 3 \dots n$$

where

- μ = Mean
- σ = Standard deviation
- x_i = each data value

1 - a) scatter plot

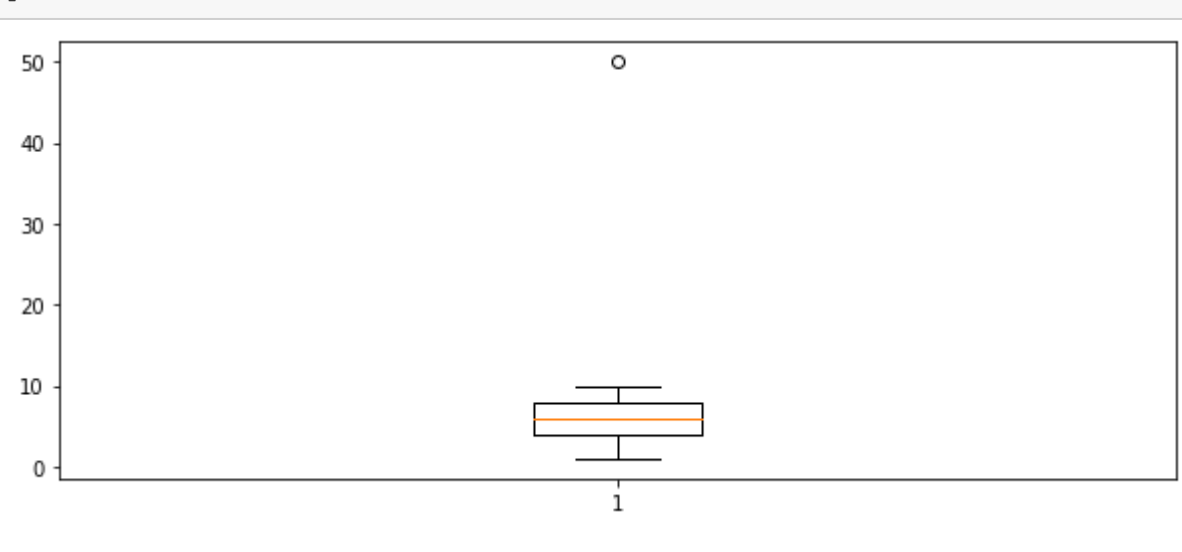
In [25]:
```
plt.figure(figsize=(10, 4))
plt.scatter(x, y)
plt.show()
```



1 - b) box plot

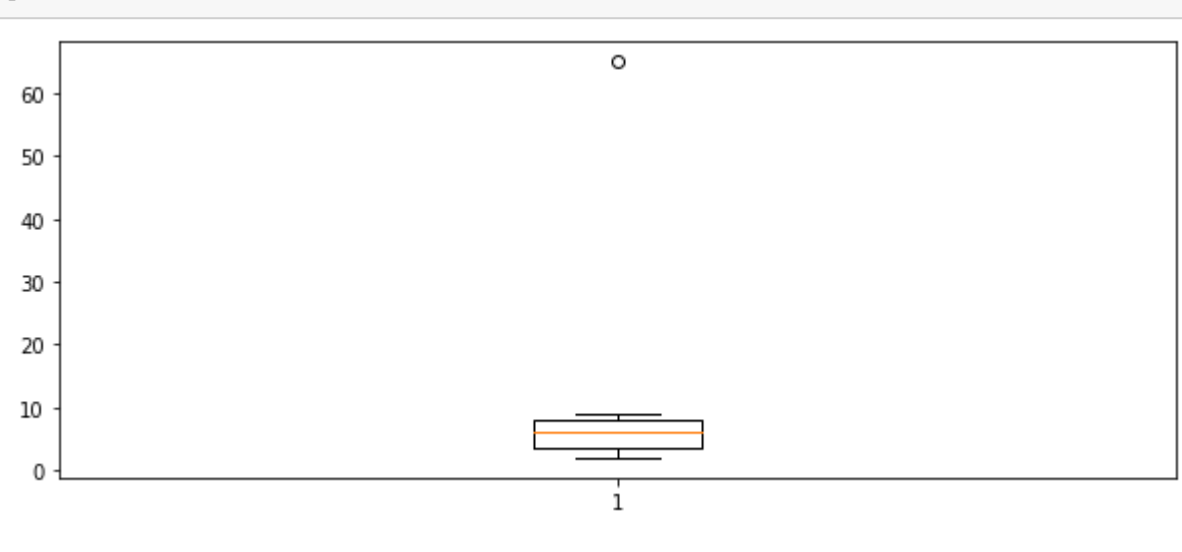
Heights

In [26]:
```
plt.figure(figsize=(10, 4))
plt.boxplot(x)
plt.show()
```



Weights

In [27]:
```
plt.figure(figsize=(10, 4))
plt.boxplot(y)
plt.show()
```



2 - zscore method

In [28]:
```
def calculate_zscore(data_values):
    # mean
    mean_vals = calculate_mean(data_values)
    # standard deviation
    std_dev = calculate_stddev(data_values=data_values)
    # applying the formula for all the values
    zscore = [(i - mean_vals)/std_dev for i in data_values]
    return zscore
```

In [36]:
```
z_x = calculate_zscore(data_values=x)
print(z_x)
```

```
[-0.6631473670024701, -0.5751189554534697, -0.4870905439044692, -0.3990621323554687, -0.3110337208064
683, -0.2230053092574678, -0.13497689770846735, 0.041079925389533554, 0.12910833693853402, 3.65024479
88985525, -0.3110337208064683, -0.2230053092574678, 0.041079925389533554, -0.3990621323554687, -0.134
97689770846735]
```

In [30]:
```
def get_outlier_indices(data_values):
    # z_score values of all the data values
    z_score = calculate_zscore(data_values=data_values)
    # get the index of the outlier
    # whose value is > 3
    # whose value is < -3
    outlier_indices = [i for i, j in enumerate(z_score) if j > 3 or j < -3]
    return outlier_indices
```

In [31]:
```
x_index = get_outlier_indices(data_values=x)
print(x_index)
```

```
[9]
```

In [32]:
```
y_index = get_outlier_indices(data_values=y)
print(y_index)
```

```
[9]
```

In [33]:
```
heights = df['Height(Inches)'].to_list()
weights = df['Weight(Pounds)'].to_list()
```

In [34]:
```
out_heights = get_outlier_indices(data_values=heights)
print(out_heights)
```

```
[138, 174]
```

In [35]:
```
out_weights = get_outlier_indices(data_values=weights)
print(out_weights)
```

```
[]
```

In [ ]:

What did we learn?

- Statistical Measurements
- Introduction to DA
- Visualization
- Visualization of the dataset
- Outlier detection
- Live streaming visualization