

# Data Analytics Algorithms

Python is one of the most popular languages for Analytics in 3rd level and industry. One of the main reasons for this is the speed at which you can conduct Analysis such as getting the maximum and minimum values of a list.

We can also use the statistics library to find the mean, median and mode or we can code these without the use of a library.

# Max and Min Values

## Method 1: Using Inbuilt functions

```
myList = [1, 19, 27, 8, 5, 9]  
minValue = min(myList)  
maxValue = max(myList)  
print(minValue)  
print(maxValue)
```

The output is:

```
1  
27
```

# Max and Min Values

## Method 2: Using Sorting and extracting the First and Last Value

```
myList = [1, 19, 27, 8, 5, 9]
myList.sort()
minValue = myList[0]
maxValue = myList[-1]
print(minValue)
print(maxValue)
```

The output again is:

```
1
27
```

# Max and Min Values

## Method 3: Manually Coding

```
myList = [1, 19, 27, 8, 5, 9]  
minValue = myList[0]  
maxValue = myList[0]
```

```
for item in myList:  
    if item < minValue:  
        minValue = item  
    if item > maxValue:  
        maxValue = item
```

```
print(minValue)  
print(maxValue))
```

The output again:

```
1  
27
```

The algorithm sets the first value in the list to the maximum number. Then you iterate through the list, and if any number is bigger than the current maximum number you replace the maximum number with the new number.

# Mean

## Method 1: Using sum() and len()

```
myList = [1, 19, 27, 8, 5, 9]  
average = sum(myList) / len(myList)  
print(average)
```

The output is:

```
11.5
```

# Mean

## Method 2: Using statistics library

```
import statistics  
  
myList = [1, 19, 27, 8, 5, 9]  
average = statistics.mean(myList)  
print(average)
```

The output is:

```
11.5
```

# Mean

## Method 3: Manually Coding

```
myList = [1, 19, 27, 8, 5, 9]

sumValues = 0
for item in myList:
    sumValues += item
average = sumValues / len(myList)
print(average)
```

The output is:

```
11.5
```

# Median

## Method 1: Using the statistics library

```
myList = [1, 19, 27, 8, 5, 9]
myList.sort()
median = statistics.median(myList)
print(median)
```

The output is:

8.5



# Median

## Method 2: Manually Coding

```
myList = [1, 19, 27, 8, 5, 9]
myList.sort()
if len(myList) % 2 == 0:
    middlePlusOne = len(myList) // 2
    median = (myList[middlePlusOne - 1] +
              myList[middlePlusOne]) / 2
else:
    middle = len(myList) // 2
    median = myList[middle]
print(median)
```

The output is:

8.5

Note: The median is the middle number of a sorted list.

If there is an even number of elements in the list then the median is the middle two numbers added together and divided by 2.

# Frequency

Frequency is the number of times elements appear in a list, this can be for strings or numbers.

# Frequency

```
myList = ["red", "blue", "blue", "red",  
"green", "red", "red"]
```

```
colourNames = []  
colourCounts = []
```

```
for item in myList:  
    if item not in colourNames:  
        colourNames.append(item)
```

```
for colour in colourNames:  
    total = myList.count(colour)  
    colourCounts.append(total)
```

```
print(colourCounts)  
print(colourNames)
```

Script output:

```
[4, 2, 1]  
['red', 'blue', 'green']
```

Creates two empty lists

Loops through myList. If the item is not already in the colourNames list then add it to the list.

Loops through the colourNames list and counts the amount of times each colour is in myList

# Mode

Mode is the value that occurs most often in the dataset.

## Method 1: Using statistics library

```
import statistics

myList = ["red", "blue", "blue", "red",
"green", "red", "red"]

mode = statistics.mode(myList)
print(mode)
```

Script output:

```
red
```

# Mode

## Method 2: Manually Coding

```
colourCounts = [4, 2, 1]
colourNames  = ['red', 'blue', 'green']

maxFreq = max(colourCounts)
maxFreqLoc = colourCounts.index(maxFreq)

mode = colourNames[maxFreqLoc]

print(mode)
```

Script output:

```
red
```