Oide

Tacú leis an bhFoghlaim Ghairmiúil i measc Ceannairí Scoile agus Múinteoirí

Supporting the Professional Learning of School Leaders and Teachers

# Session2: Algorithms I

LEAVING CERTIFICATE COMPUTER SCIENCE

# Overview of the session

| | |
|---|---|
| **Part 1** | Introduction to algorithms |
| **Part 2** | Algorithms for mean, median and mode |
| **Part 3** | Breakout activity (Python) |

# By the end of this session..

Participants will have…

…reflected on the importance of and the ubiquitous nature of algorithms in today's society

…participated in a coding activities relating to measures of central tendency

…reflected on ideas to facilitate the effective learning of algorithms in their own classrooms and, in particular, in relation to ALT2

Oide

Tacú leis an bhFoghlaim Ghairmiúil i measc Ceannairí Scoile agus Múinteoirí

Supporting the Professional Learning of School Leaders and Teachers

# Introduction to Algorithms

LEAVING CERTIFICATE COMPUTER SCIENCE

# Algorithms and the Specification

*"Computer science is the study of computers and algorithmic processes. Leaving Certificate Computer Science includes how programming and computational thinking can be applied to the solution of problems, and how computing technology impacts the world around us."*

NCCA Curriculum specification, Page 1

| Strand 1: Practices and principles | Strand 2: Core concepts | Strand 3: Computer science in practice |
|---|---|---|
| ▸ Computers and society<br>▸ Computational thinking<br>▸ Design and development | ▸ Abstraction<br>▸ Algorithms<br>▸ Computer systems<br>▸ Data<br>▸ Evaluation/Testing | ▸ Applied learning task 1 - Interactive information systems<br>▸ Applied learning task 2 - Analytics<br>▸ Applied learning task 3 - Modelling and simulation<br>▸ Applied learning task 4 - Embedded systems |

Page 11

# LCCS Learning Outcomes

2.5 use pseudo code to outline the functionality of an algorithm

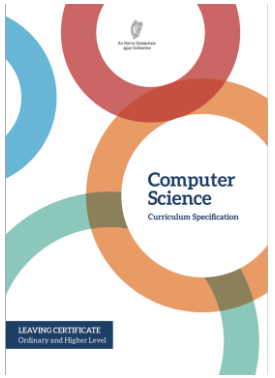
2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

2.7 implement algorithms using a programming language to solve a range of problems

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

2.9 assemble existing algorithms or create new ones that use functions (**including recursive**), procedures, and modules

**2.10 explain the common measures of algorithmic efficiency using any algorithms studied**




**S2: Algorithms**

Programming concepts

Sorting: Simple sort, Insert sort, Bubble sort, **Quicksort**

Search: Linear search, Binary search

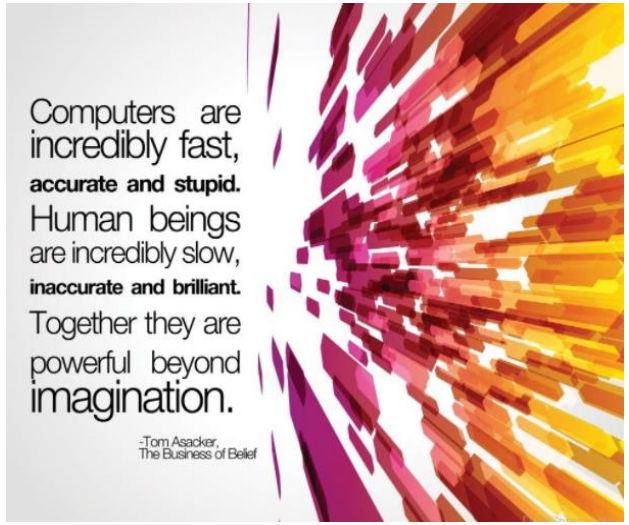
Algorithmic complexity

See also learning outcomes 1.6, 1.7 1.14, 1.22, 2.3, 3.4 and 3.7 … plus others

# What is an algorithm?

“A step-by-step procedure for solving a problem or accomplishing some end especially by a computer” Merriam-Webster

Because of their speed and reliability computers are an ideal tool for running algorithms.




Algorithms are:

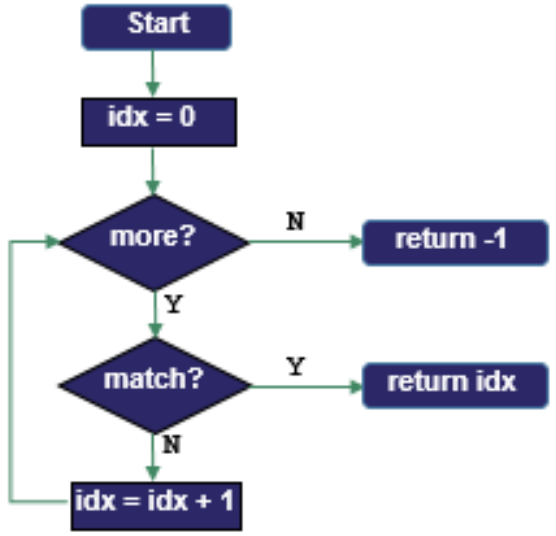
- a sequence of instructions
- a way of capturing intelligence
- general solutions to problems
- expressed in a variety of different ways
- Characterised by input, processing and output

# Some examples

**Chocolate Cream Pie**

1. Heat milk, marshmallows and chocolate in 3-quart saucepan over low heat, stirring constantly, until chocolate and marshmallows are melted and blended. Refrigerate about 20 minutes, stirring occasionally until mixture mounds slightly when dropped from a spoon.

2. Beat whipping cream in chilled small bowl with electric mixer on high speed until soft peaks form. Fold chocolate mixture into whipped cream. Pour into pie shell. Refrigerate uncovered about 8 hours or until set. Garnish with milk chocolate curls and whipped cream.

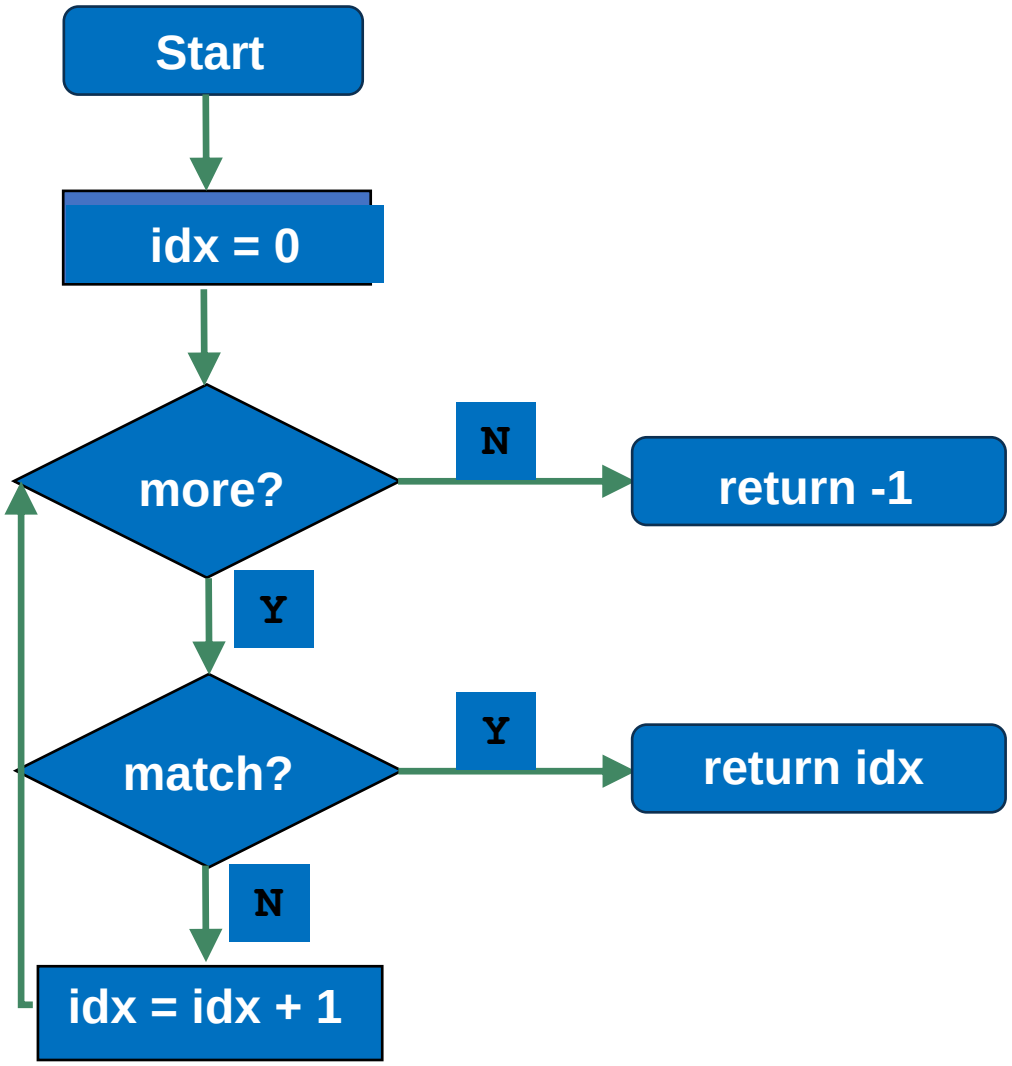


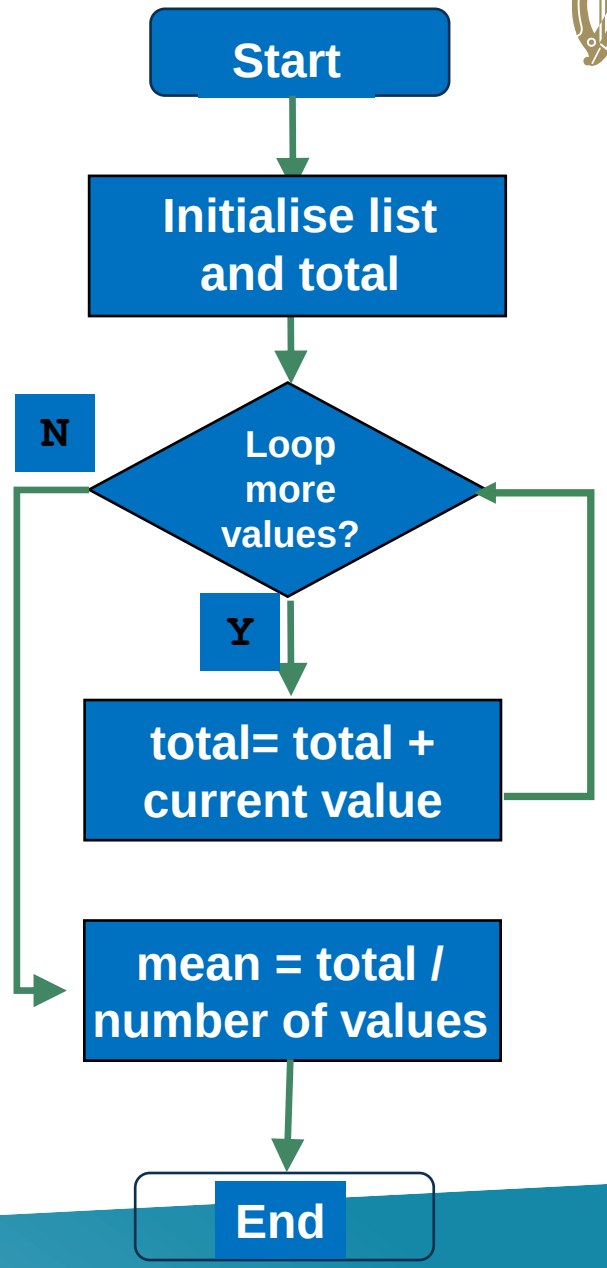

```
1. Set low = 0
2. Set high = length of list - 1
3. Set index = (low+high)/2, rounded down to an integer
4. If the value at the index position is the same as the target value
      Return index
   Else If the value at the index position is less than the target value
      Set low = index + 1
   Else If the value at the index position is less than the target value
      Set high = index - 1
5. Go back to step 3 above
6. Return -1
```

```
p = 1029
q = 462

r = p%q # step 1
while (r != 0): # step 2
    p = q # step 3
    q = r # step 3
    r = p%q # step 1 (again)

print("GCD is", q)
```


DESIGN
create a representation, decide on tools

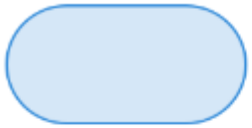
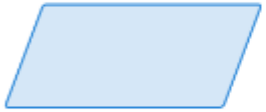
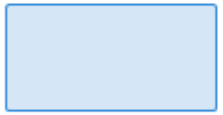
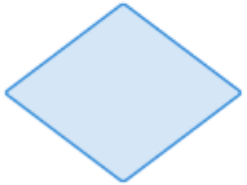
# Flowcharts


Start
idx = 0
more?
N
return -1
Y
match?
Y
return idx
N
idx = idx + 1


Start
Initialise list and total
N
Loop more values?
Y
total= total + current value
mean = total / number of values
End

# Quizlet (flowcharts)

https://quizlet.com/758767872/match

| Symbol | Name | Function |
|---|---|---|
|  | Start/End | An oval represents a start or end point |
|  | Arrows | A line is a connector that shows relationships between the representative shapes |
|  | Input/ Output | A parallelogram represents input or output |
|  | Process | A rectangle represents a process |
|  | Decision | A diamond indicates a decision |

Oide

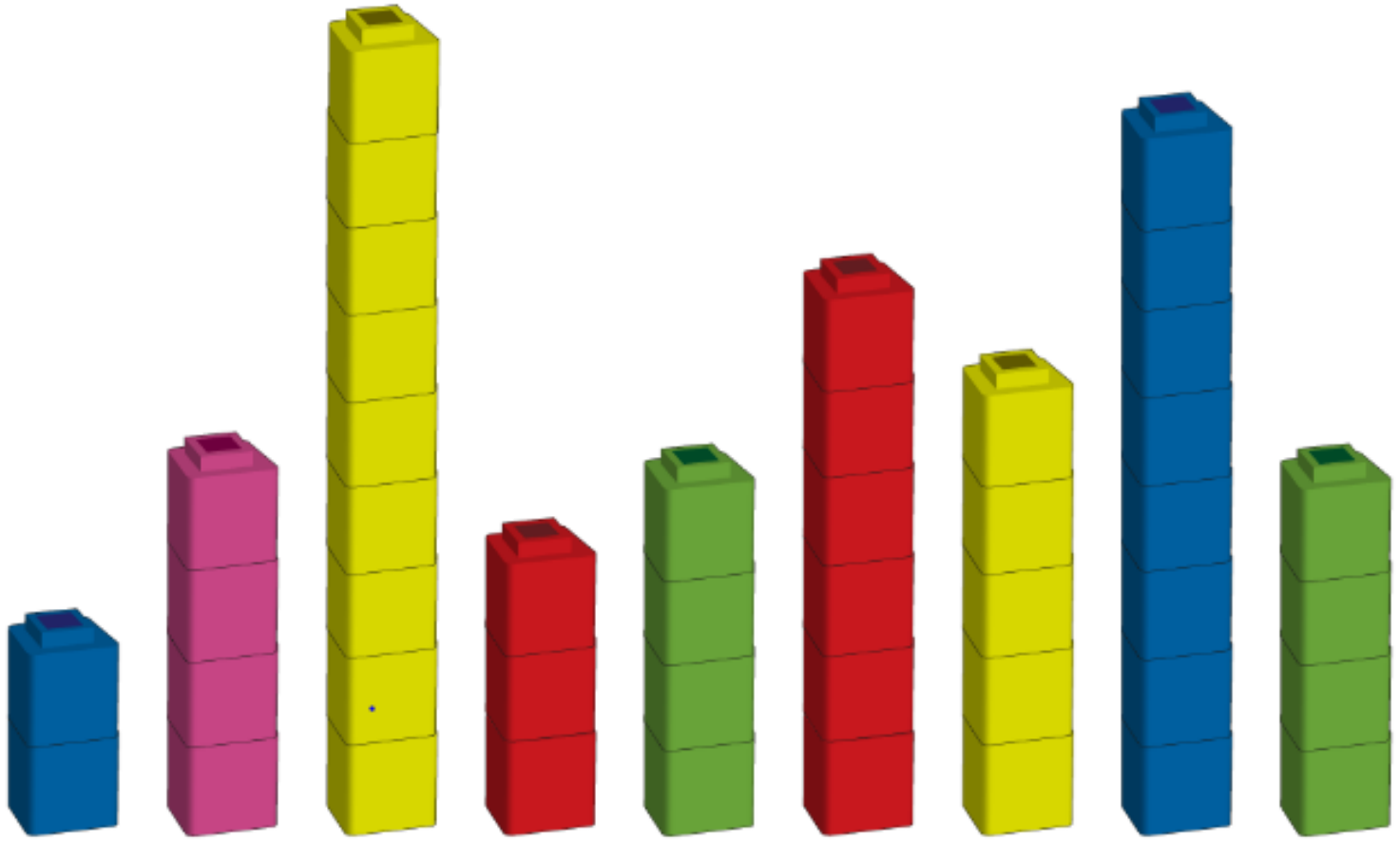
Tacú leis an bhFoghlaim Ghairmiúil i measc Ceannairí Scoile agus Múinteoirí

Supporting the Professional Learning of School Leaders and Teachers

# Algorithms for mean, median and mode

LEAVING CERTIFICATE COMPUTER SCIENCE

# Measures of Central Tendency



# A look ahead to ALT2

3.4. Develop algorithms that can find the frequency, mean, median and mode of a data set.

3.5. Structure and transform raw data to prepare it for analysis.

3.6. Represent data to effectively communicate in a graphical form.

3.7. Use algorithms to analyse and interpret data in a way that informs decision-making.

# Measures of Central Tendency

```
# A program to demonstrate the use of some statistics functions
import statistics

# Initialise a list of values
values = [2,3,5,2,4]

# Compute the 3 averages
arithmetic_mean = statistics.mean(values)
median_value = statistics.median(values)
modal_value = statistics.mode(values)

# Display the answers
print("The mean is ", arithmetic_mean)
print("The median and mode are %d and %d" %(median_value, modal_value))
```

When the program is run the output looks like this:

```
The mean is  3.2
The median and mode are 3 and 2
>>>
```

# Mean

## A representative value

Input: A list of values

| 18 | 27 | 15 | 13 | 22 |
|---|---|---|---|---|

Step 1. Add the values

| 18 | 27 | 15 | 13 | 22 |
|---|---|---|---|---|

0+18= 18

18+27 = 45

45+15 = 60

60+13 = 73

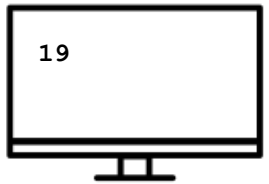
73+22= 95

Step 2. Calculate the mean

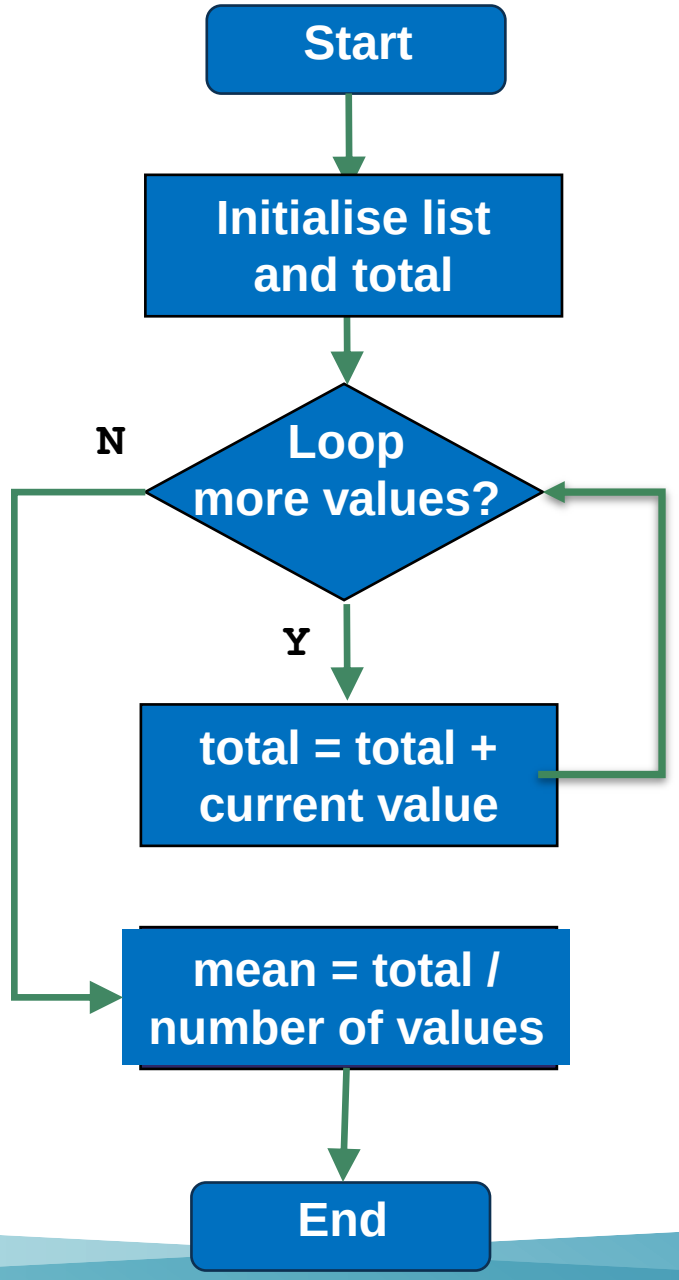
Divide the total by the number of values

95/5 = 19

Output: The mean



# Mean: Flowchart and code




```
# Program to find the mean of a list of values
# Version 1

# Calculate and return the mean of all the values
in L
def arithmetic_mean(L):

    # set the initial value of total to zero
    total = 0 # running total of values in L

    # Now loop over the list
    for v in L:
        total = total + v # running total

    # Divide by the total by the number of values
in L
    return total/5

# PYTHON STARTS EXECUTING FROM HERE ...
# Initialise a list of values
my_list = [18, 27, 15, 13, 22]
# Call the function
my_mean = arithmetic_mean(my_list)
# Display the answer
```

# Mean


## Initialise the list

```
L = [18, 27, 15, 13, 22]
```

## Compute a running total

```
total = 0
for v in L:
    total = total + v
```

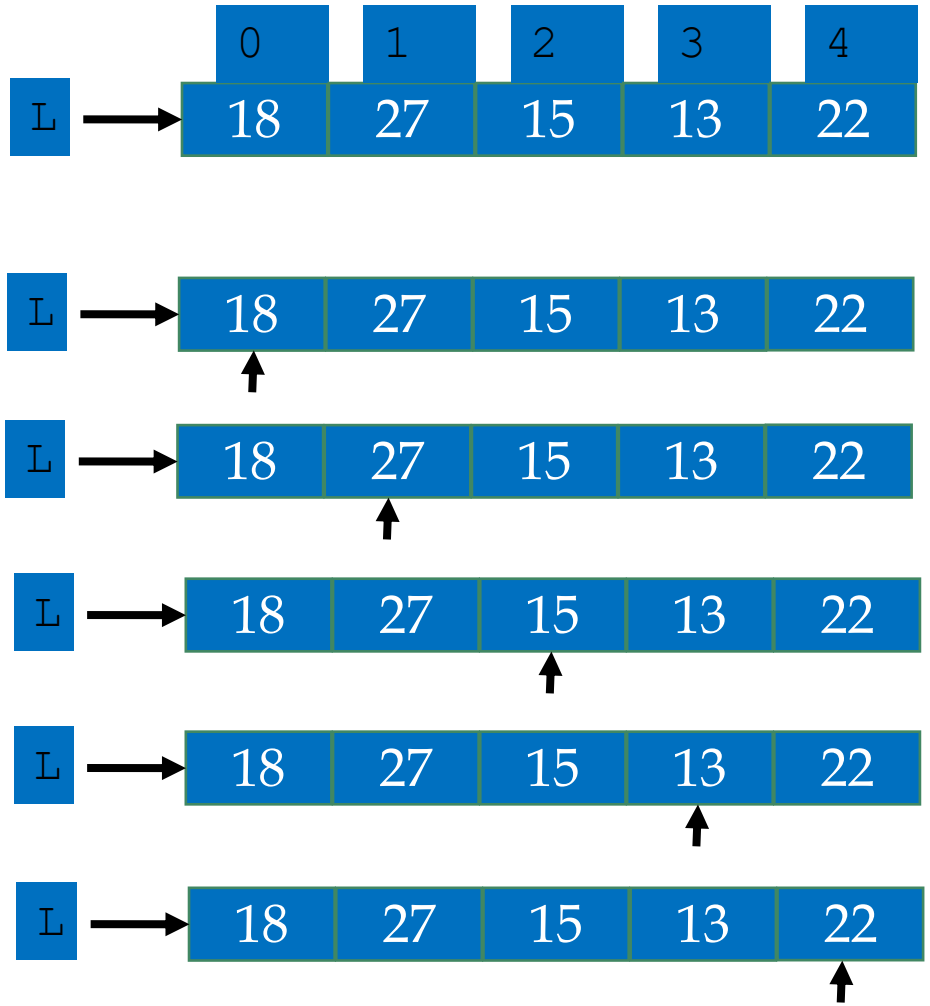
## Compute and display the mean

```
mean = total/5
print(mean)
```

The current value

v

| v |
|---|
| 18 |
| 27 |
| 15 |
| 13 |
| 22 |




This is a running total of all the values

| total |
|---|
| 18 |
| 45 |
| 60 |
| 73 |
| 95 |

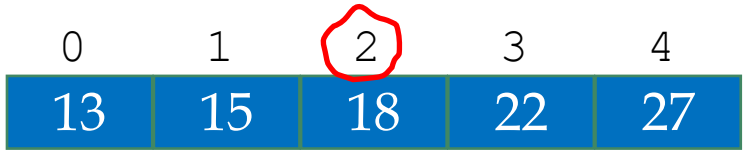
# Median

Middle value in a sorted list

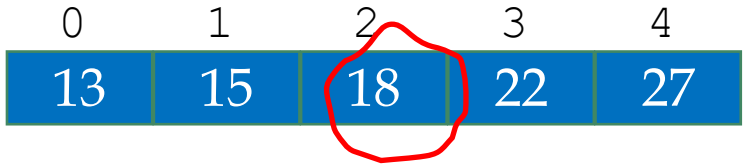
Input: A list of values


Step 1. Sort the list

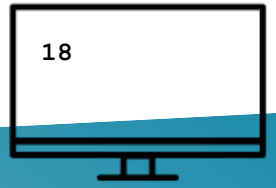

Step 2. Find middle position



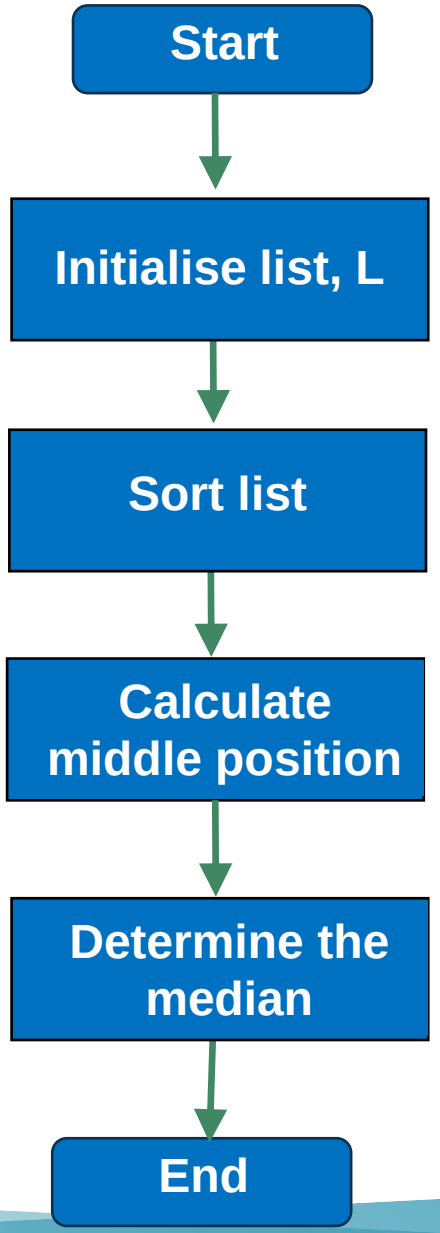

Step 3. Determine the median




Output: The median

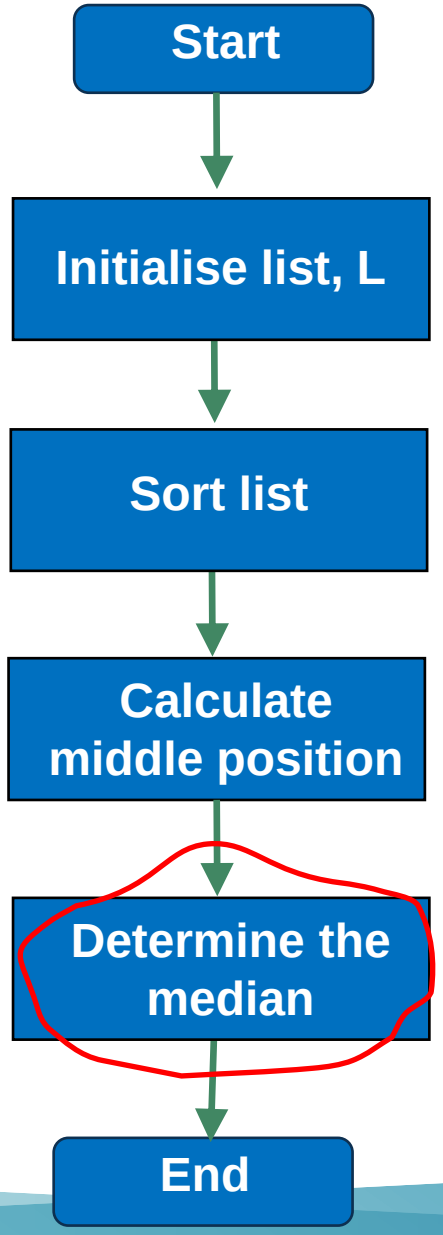


# Median




```
# A program to find the median of a list of values
# Version 1

L = [18, 27, 15, 13, 22]

# To find the median we need to sort the list
L.sort() # the values are sorted 'in place'

# The next step is to find the index of the middle value
num_values = len(L)
mid = num_values//2

median = L[mid] # the median is in the middle

# Display the result
print("The median value is: %.2f" %median)
```

# Median (dealing with an even number of values )




In a list with 5 values the median is at index 2.

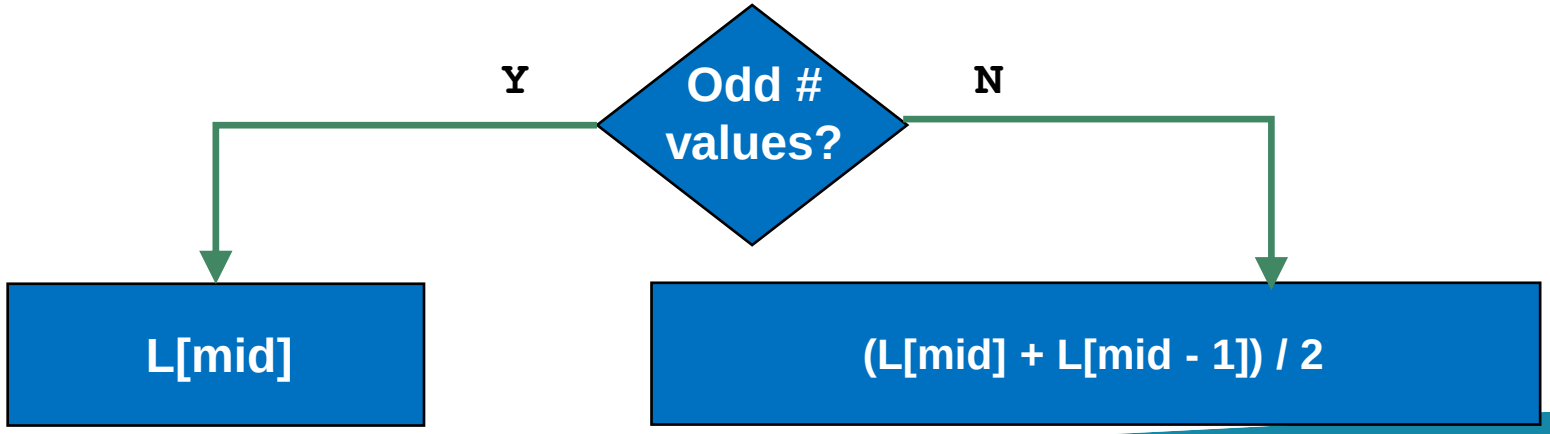
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 13 | 15 | 18 | 22 | 27 |

In a list with 4 values we need to use indices 1 and 2

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 13 | 15 | 18 | 22 |

The median is (15 + 18) / 2

The median is 16.5

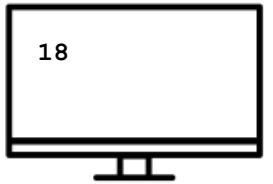


# Mode

The most frequently occurring value

Input: A list of values


Output: The mode



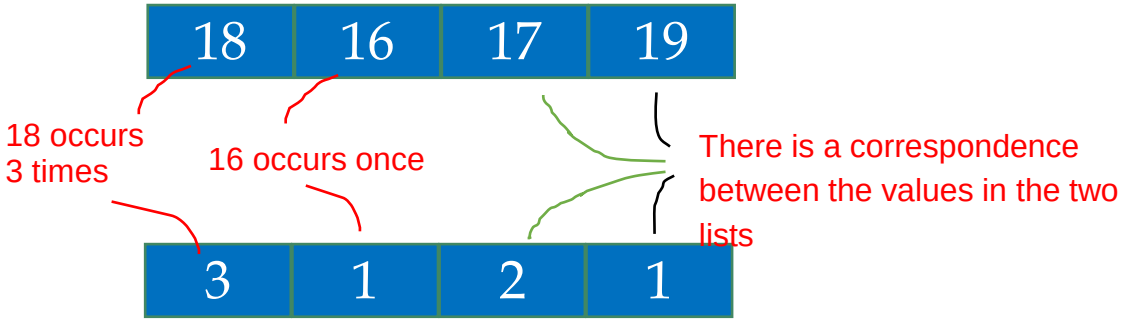

At a glance we can see the mode is 18 but how do we capture this algorithmically?

# Mode

The most frequently occurring value

Input: A list of values


Step 1. Create a list of unique values




Step 2. Create a list of frequencies

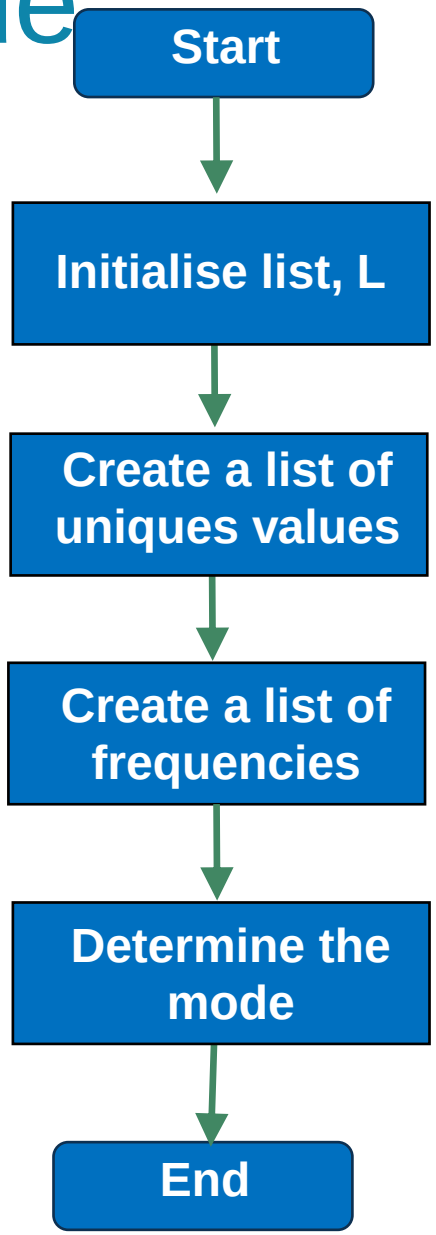
The two lists tell us the frequency of each value

Step 3. Determine the mode

**The value that corresponds to the highest frequency**

Output: The mode

# Mode




```
# A program to find the mode of a list of values
# Version 1

# Initialise a list of values
L = [18, 16, 17, 18, 19, 18, 17]

# Build up a list of unique values
unique_values = []
for value in L:
    if value not in unique_values:
        unique_values.append(value)

# Build up a list of frequencies
frequencies = []
for value in unique_values:
    frequency = L.count(value)
    frequencies.append(frequency)

# Find the mode
max_frequency = max(frequencies)
max_frequency_pos = frequencies.index(max_frequency)
mode = unique_values[max_frequency_pos]

print("Mode is", mode)
```

# Group Activity

Oide

An Roinn Oideachais
Department of Education