PDST

Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

An Roinn Oideachais agus Scileanna
Department of Education and Skills

National Workshop 3

LEAVING CERTIFICATE
COMPUTER SCIENCE

Session Schedule

Section 1	Introduction to Algorithms
Section 2	Algorithms for mean, median and mode
Section 3	Algorithms/Libraries for ALT2

By the end of this session participants will have:

- reflected on the importance of and the ubiquitious nature of algorithms in today’s society.
- participitated in a coding activities relating to measures of central tendancy
- enhanched their knowledge of the use of Python libraries in relation to ALT2
- reflected on ideas to facilitate the effective learning of algorithms in their own classrooms and, in particular, in relation to ALT2

PDST

Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

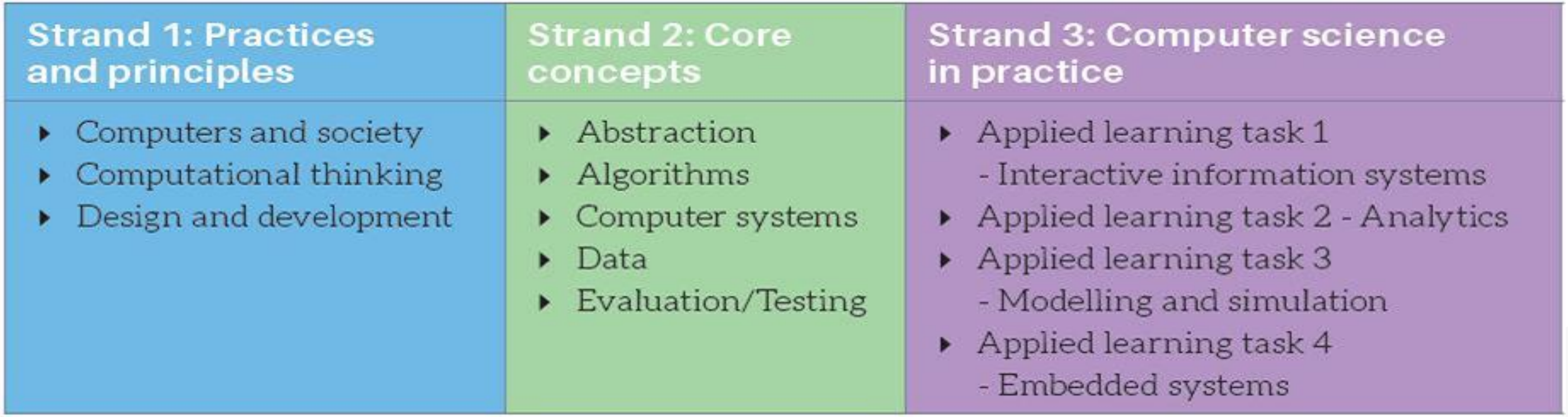
Section I

Introduction to Algorithms

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. “

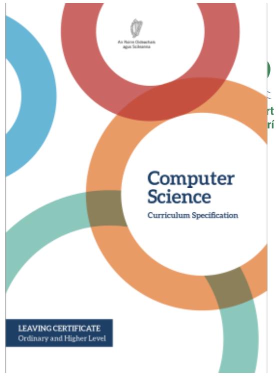
NCCA Curriculum specification, Page 1



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

NCCA Curriculum specification, Page 11

LCCS Learning Outcomes



Computer Science

Curriculum Specification

LEAVING CERTIFICATE

Ordinary and Higher Level

2.5 use pseudo code to outline the functionality of an algorithm

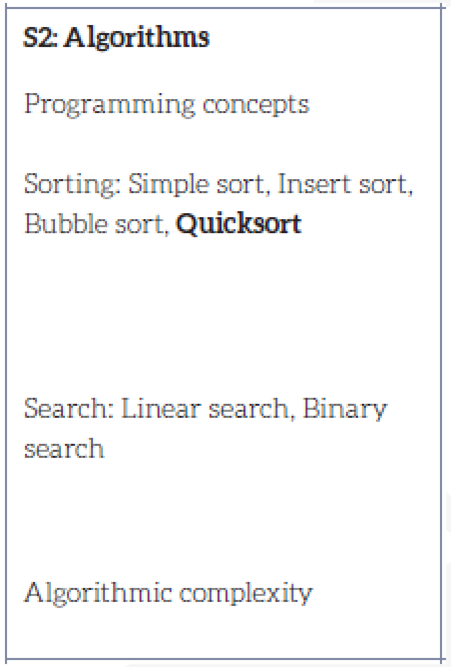
2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

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

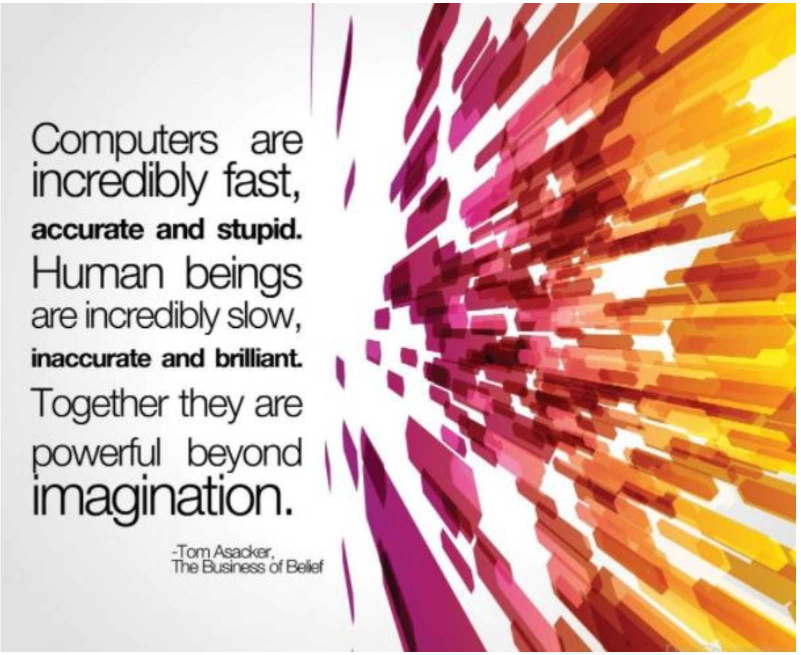
See also learning outcomes 1.6, 1.7 1.14, 1.22, 2.3, 3.4 and 3.7 … plus others

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.



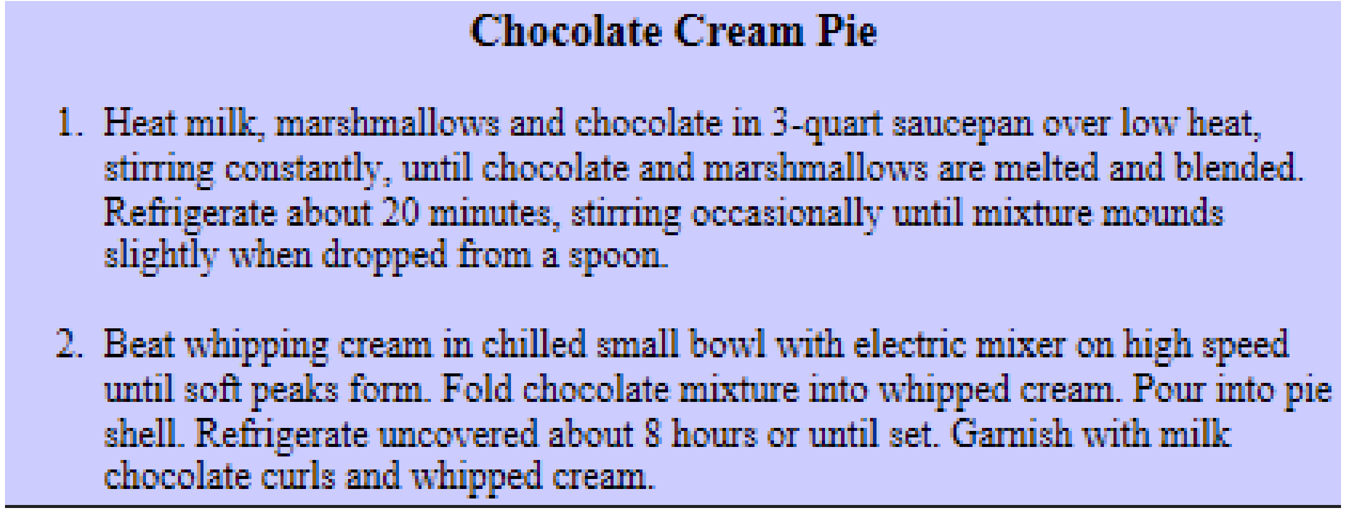
Computers are incredibly fast, **accurate and stupid.** Human beings are incredibly slow, **inaccurate and brilliant.** Together they are powerful beyond imagination.

-Tom Asacker,
The Business of Belief

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

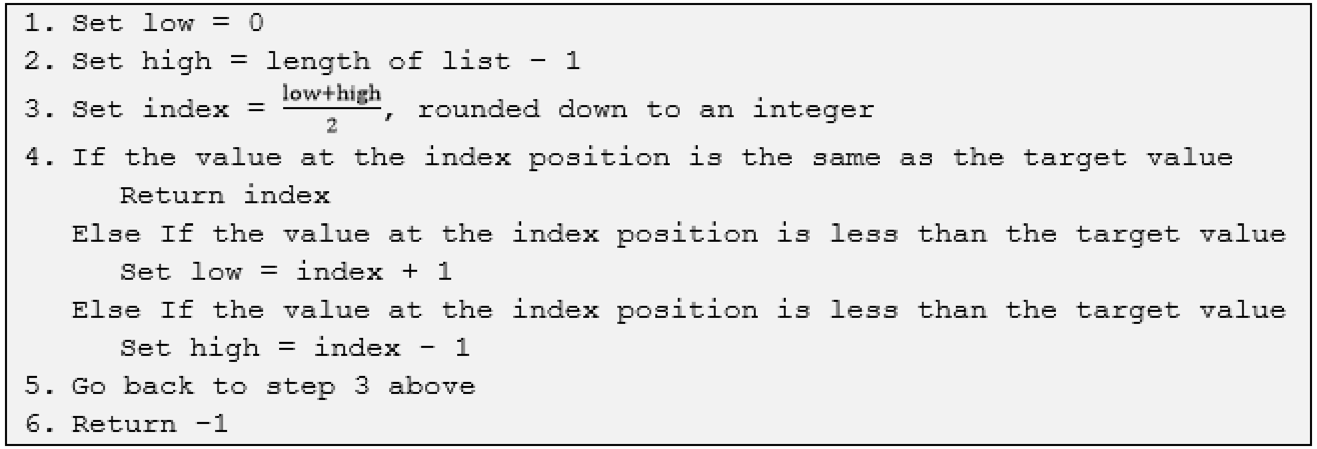
Some Examples …



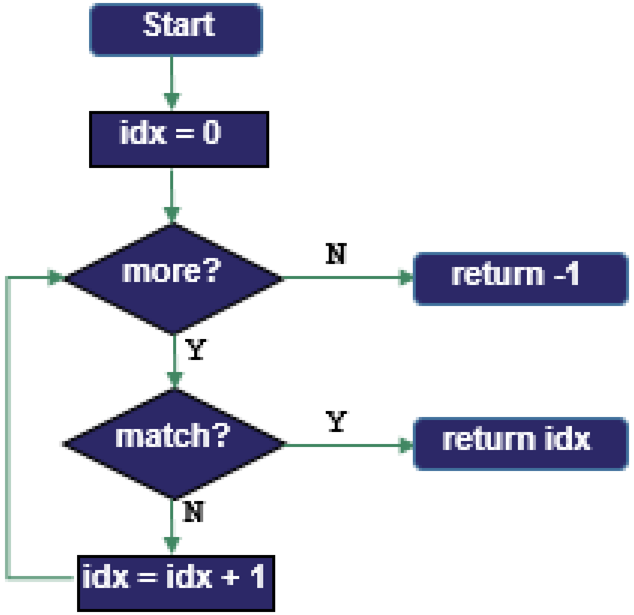
Chocolate Cream Pie

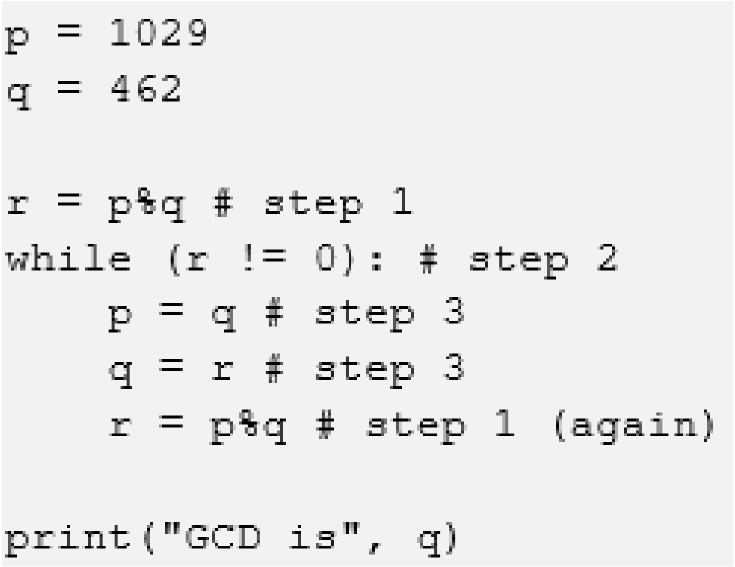
1. Heat milk, marshmallows and chocolate in 3-quart saucepan over low heat, stirring constantly, until chocolate and marshmallows are melted and blended. Refrigerate about 20 minutes, stirring occasionally until mixture mounds slightly when dropped from a spoon.

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.



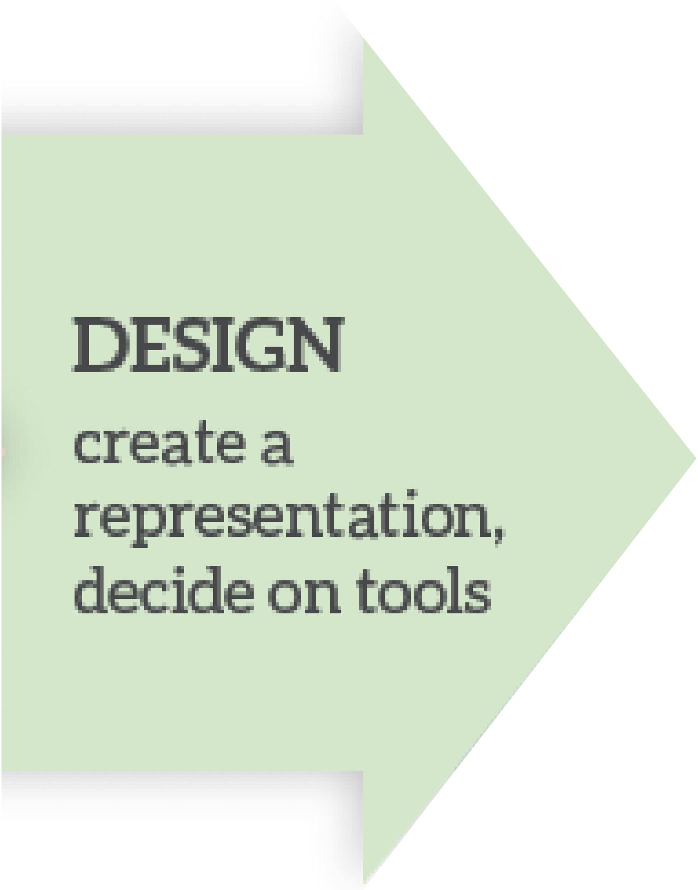
```
1. Set low = 0
2. Set high = length of list - 1
3. Set index = $\frac{low+high}{2}$, rounded down to an integer
4. If the value at the index position is the same as the target value
      Return index
   Else If the value at the index position is less than the target value
      Set low = index + 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)
```

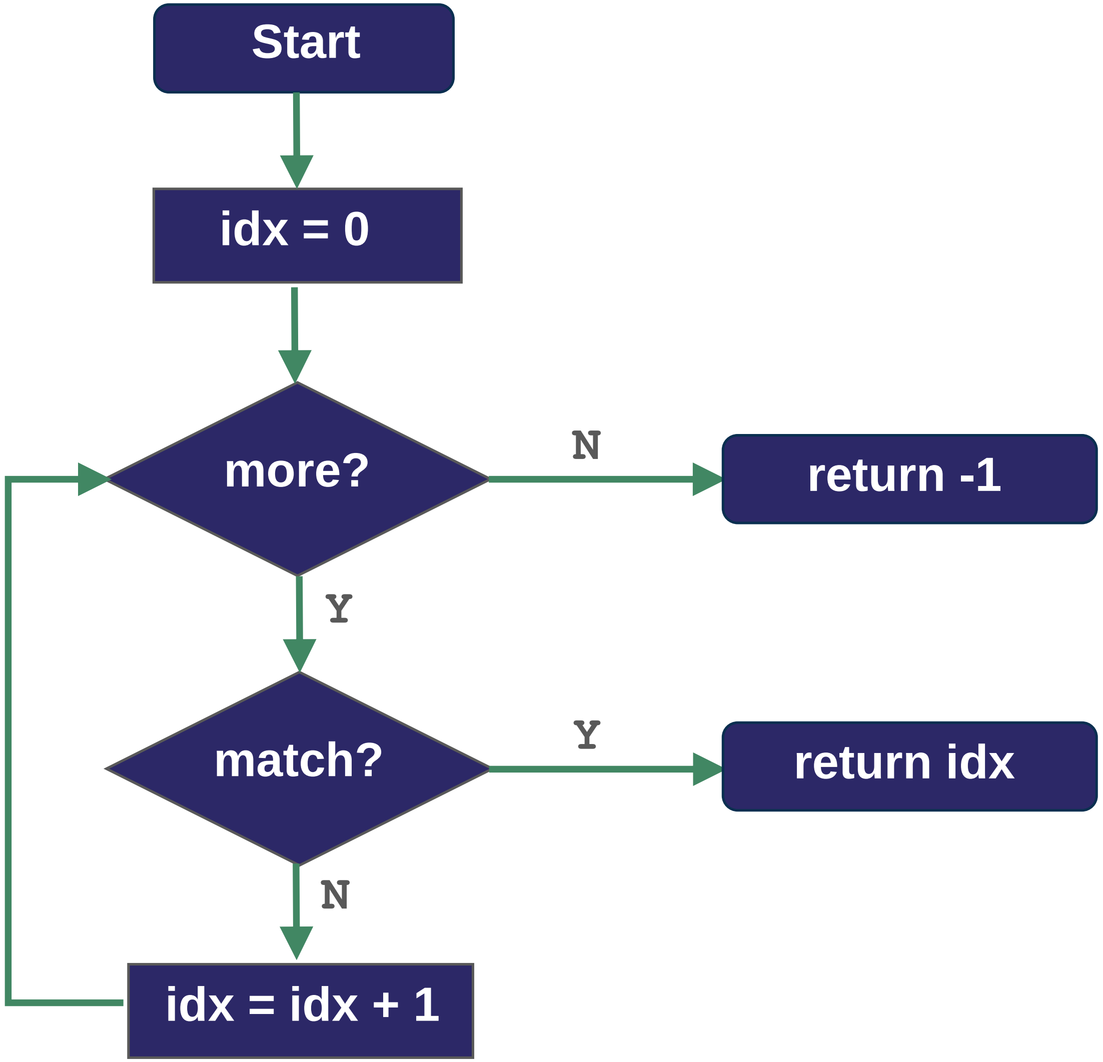
3

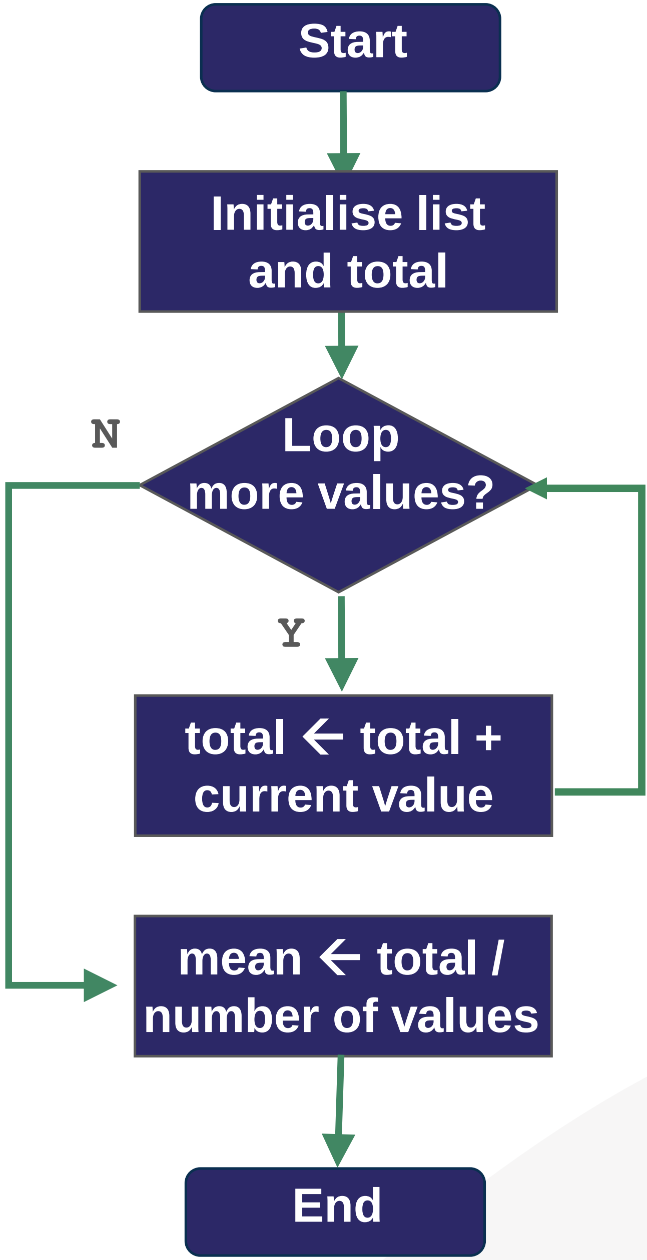


DESIGN

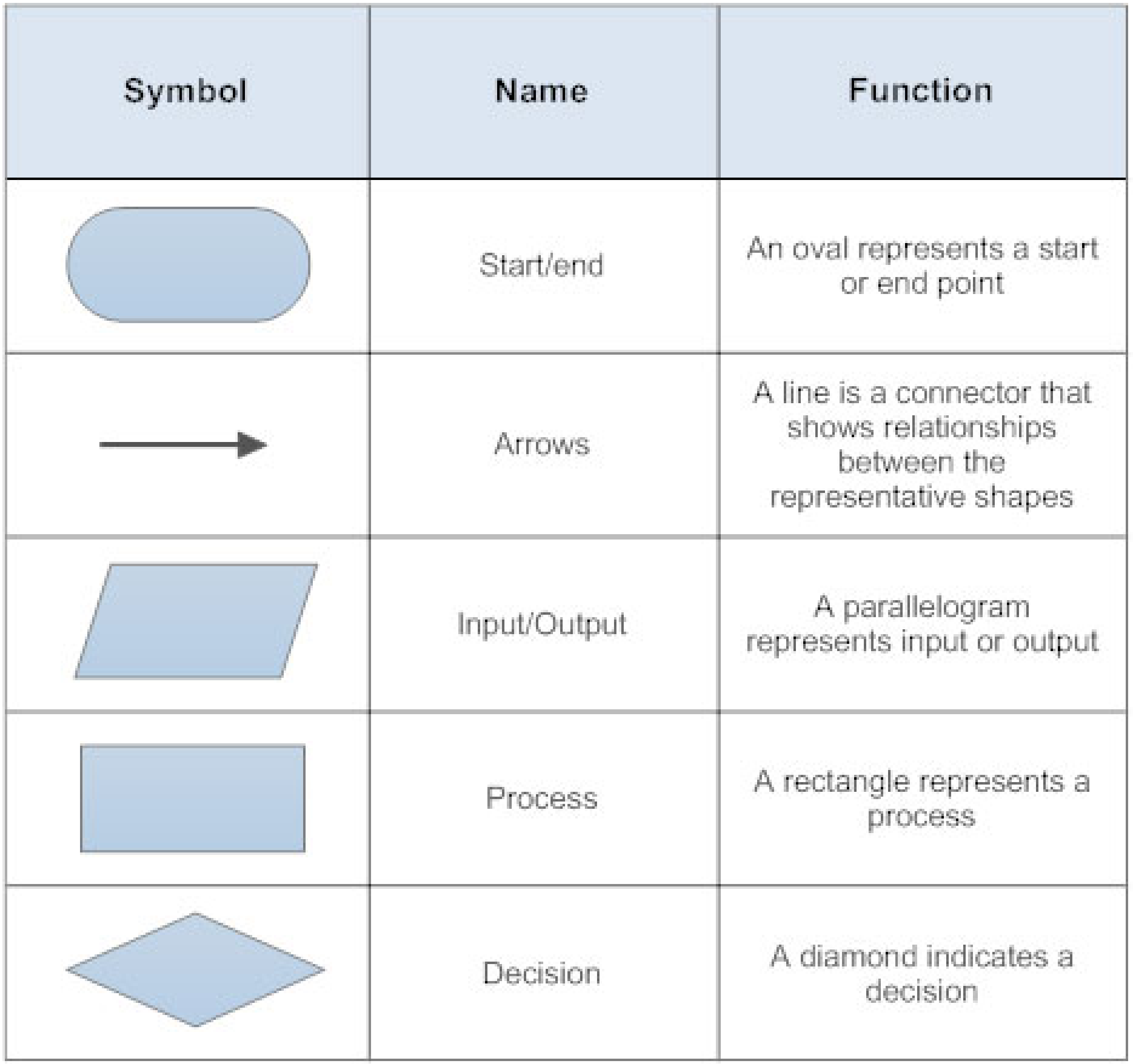
create a representation, decide on tools

Flow charts


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

Flow charts



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

Section II

Algorithms for Mean, Median, and Mode

Measures of Central Tendancy

https://www.khanacademy.org/math/cc-sixth-grade-math/cc-6th-data-statistics/mean-and-median/v/mean-median-and-mode

Recap of ALT2 Learning Outcomes

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 Tendancy

```
# 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

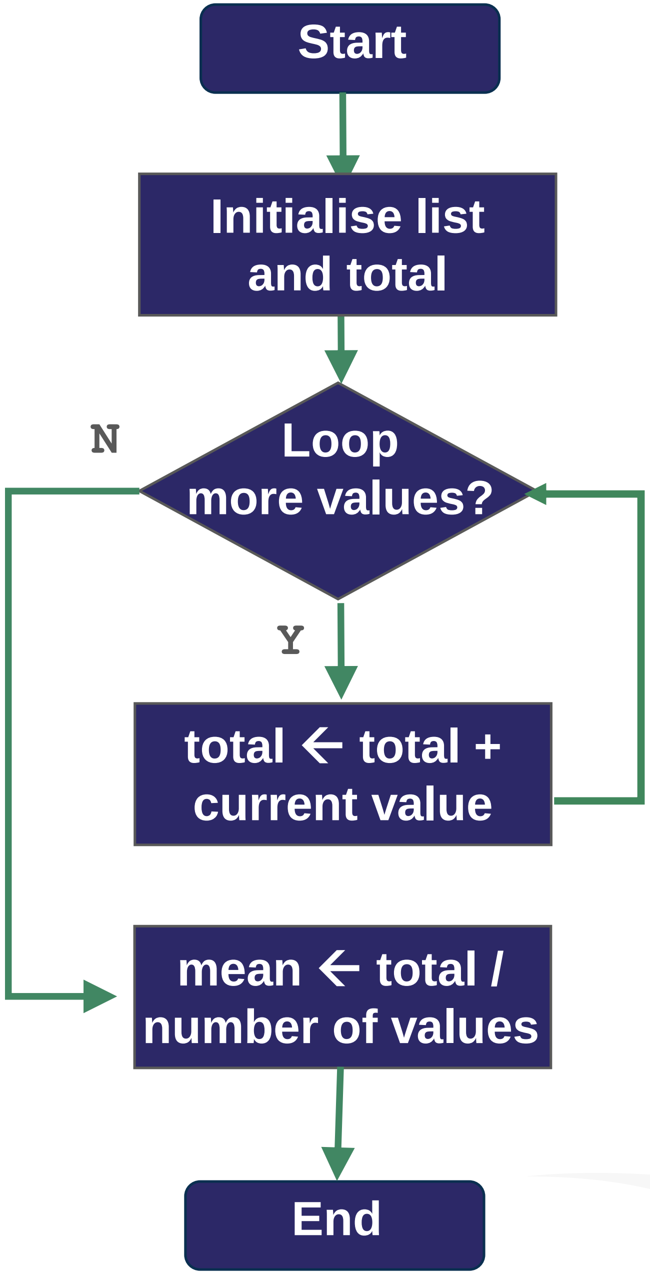
Divide the total by the number of values

95/5 → 19

Output: The mean

```
19
```

Flowchart for mean


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

```
# 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
print("The mean is ", my_mean)
```

Arithmetic Mean

Initialise the list

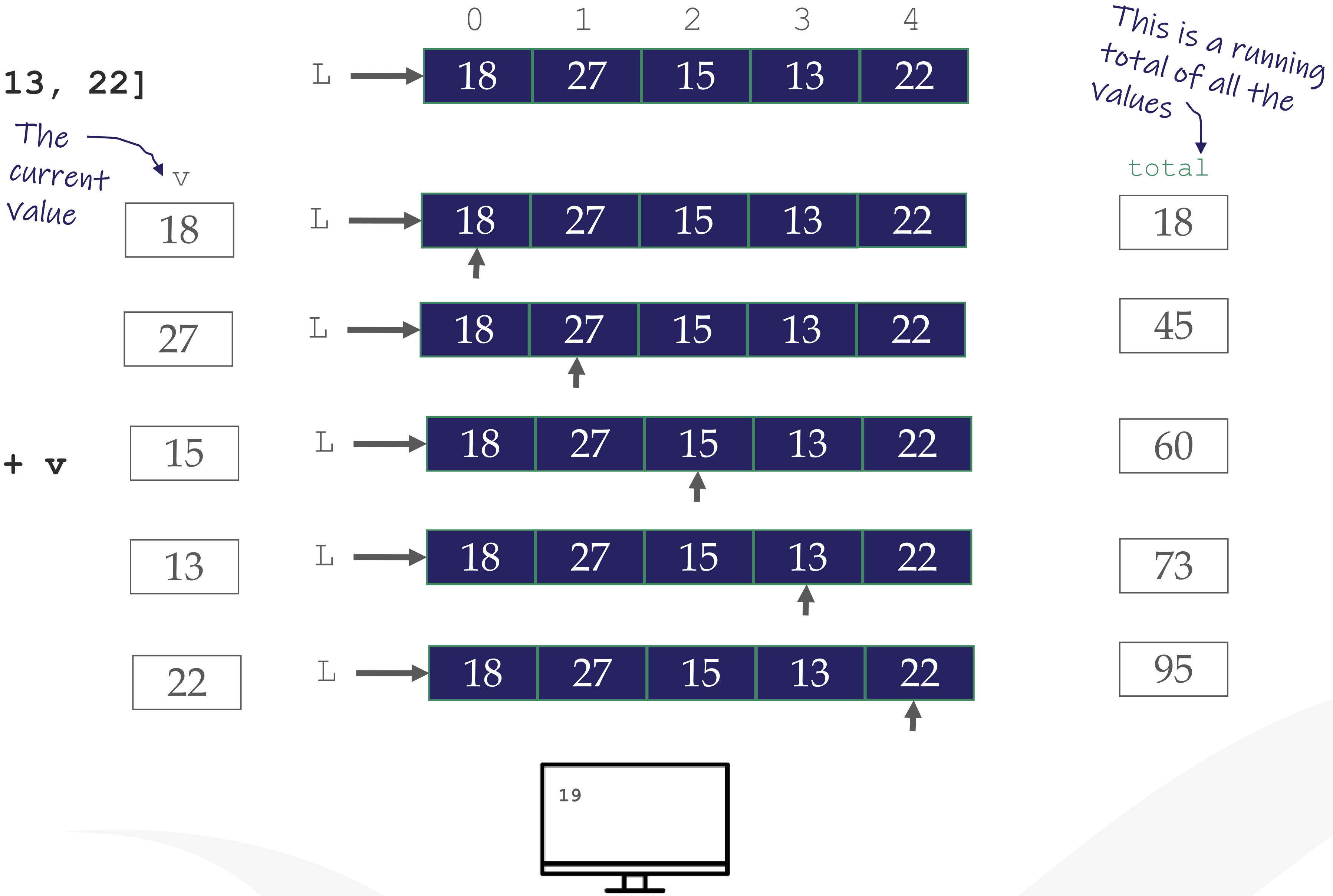
```
L = [18, 27, 15, 13, 22]
```

Compute a running total

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

Compute and dsiplay the mean

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



The current value

This is a running total of all the values

```
19
```

Median

Middle value in a sorted list

Input: A list of values

0	1	2	3	4
18	27	15	13	22

Step 1. Sort the list

0	1	2	3	4
13	15	18	22	27

Step 2. Find middle position

0	1	2	3	4
13	15	18	22	27

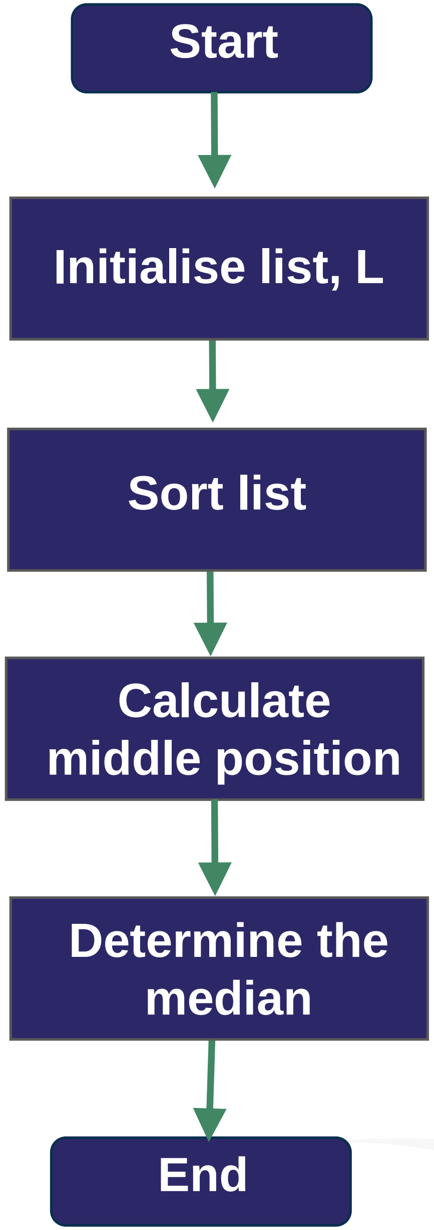
Step 3. Determine the median

0	1	2	3	4
13	15	18	22	27

Output: The median

```
18
```

Flowchart for Median



```python
# 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

Initialise the list

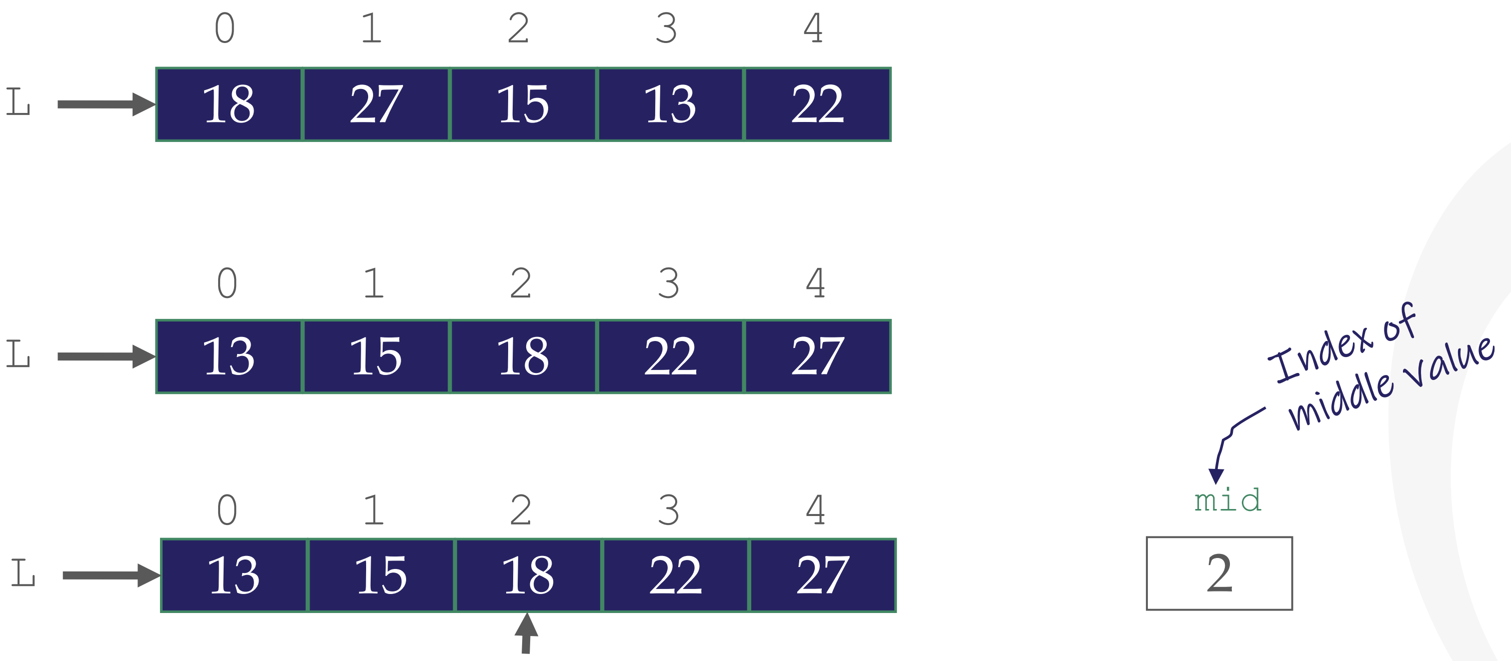
```
L = [18, 27, 15, 13, 22]
```

Sort the list

```
L.sort()
```

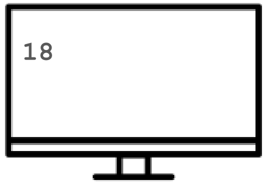
Calculate the position of the middle value

```
mid = 5//2
```



Display the middle value

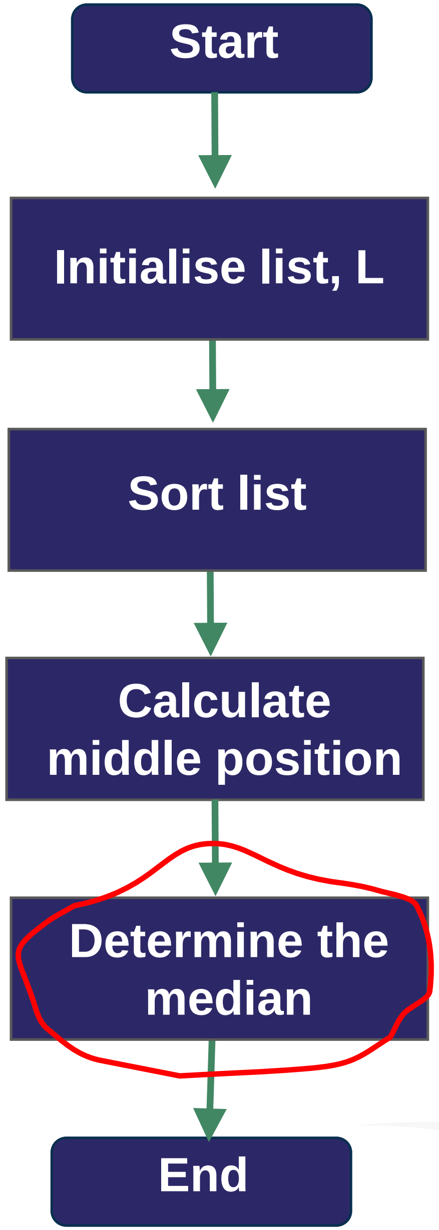
```
print(L[mid])
```



```
18
```

Question. What if there are an even number of values?

Median (dealing with an even number of values)



Start

Initialise list, L

Sort list

Calculate middle position

Determine the median

End

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

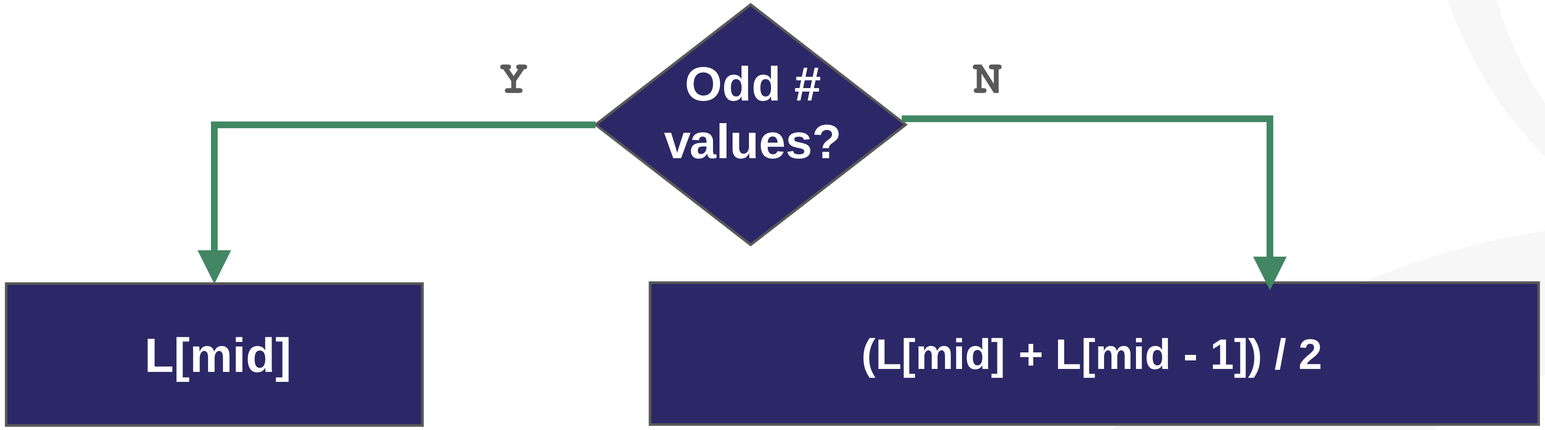
0	1	2	3	4
13	15	18	22	27

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



Odd # values?

Y

N

L[mid]

(L[mid] + L[mid - 1]) / 2

Mode

The most frequently occurring value

Input: A list of values

18	16	17	18	19	18	17

Output: The mode

```
18
```

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

18	16	17	18	19	18	17

Mode

The most frequently occurring value

Input: A list of values

18	16	17	18	19	18	17

Step 1. Create a list of unique values

18	16	17	19

18 occurs 3 times

16 occurs once

There is a correspondence between the values in the two lists

Step 2. Create a list of frequencies

3	1	2	1

The two lists tell us the frequency of each value

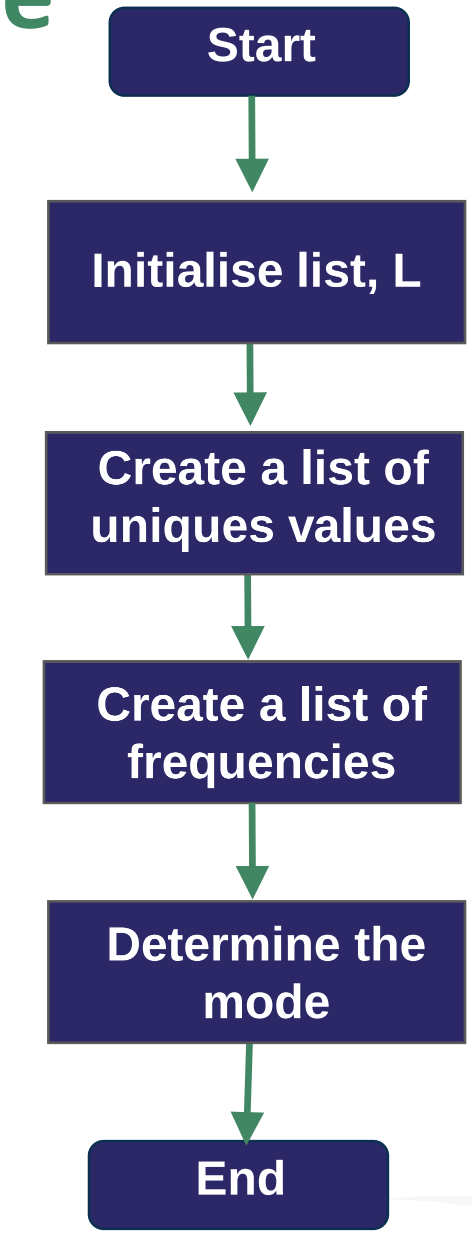
Step 3. Determine the mode

The value that corresponds to the highest frequency

Output: The mode

```
18
```

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)
```

Activity

Measures of Central Tendancy

Group Activity / Breakout

Section III

Python Libraries for ALT2

(A quick introduction)

statistics
matplotlib
re
pandas

Measures of Central Tendancy

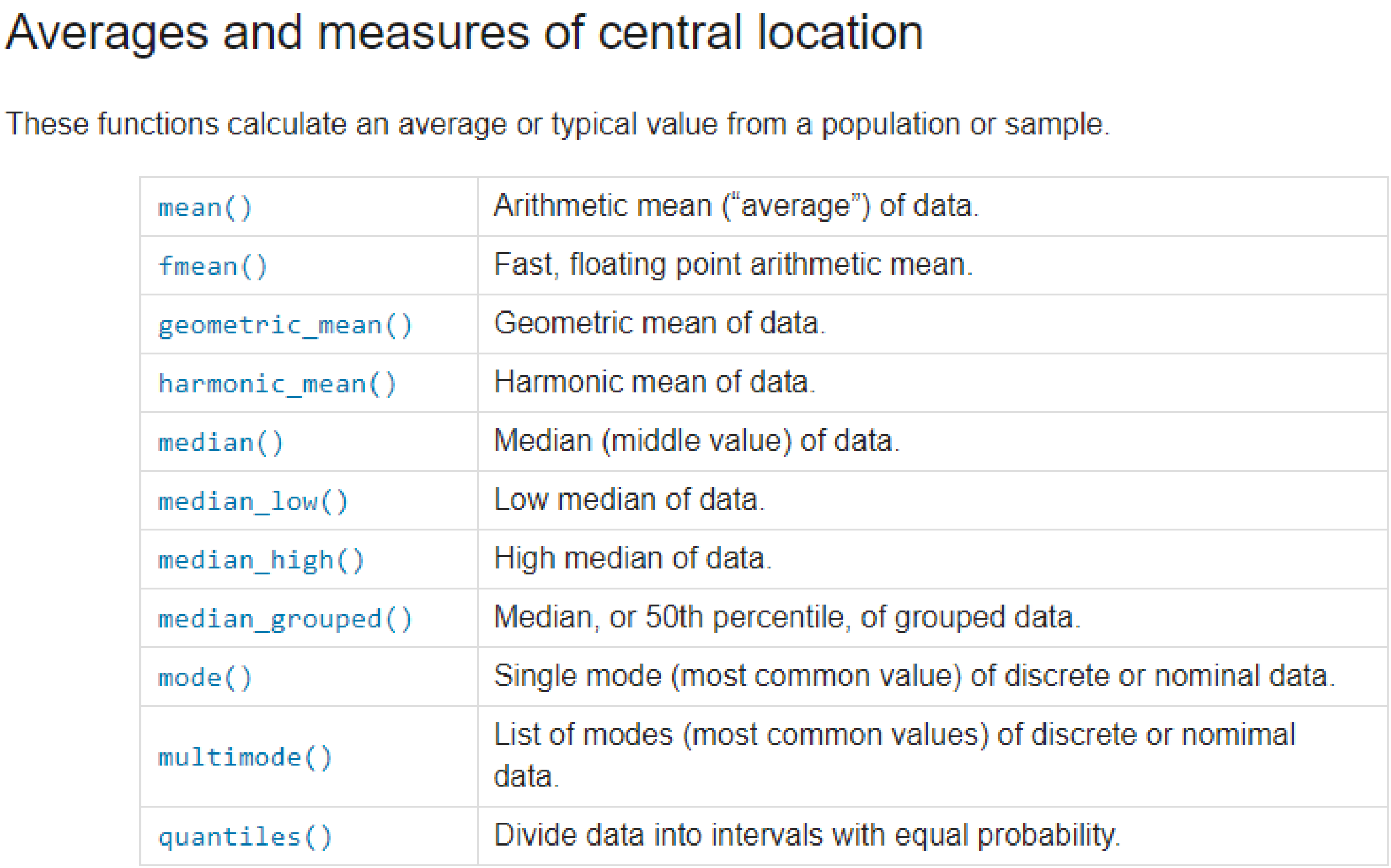
```
# A simple program to calculate and display averages
from statistics import *

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

# Compute the 3 averages
arithmetic_mean = mean(values)
median_value = median(values)
modal_value = 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
>>>
```

Measures of Central Tendancy

Check out the online documentation



Averages and measures of central location

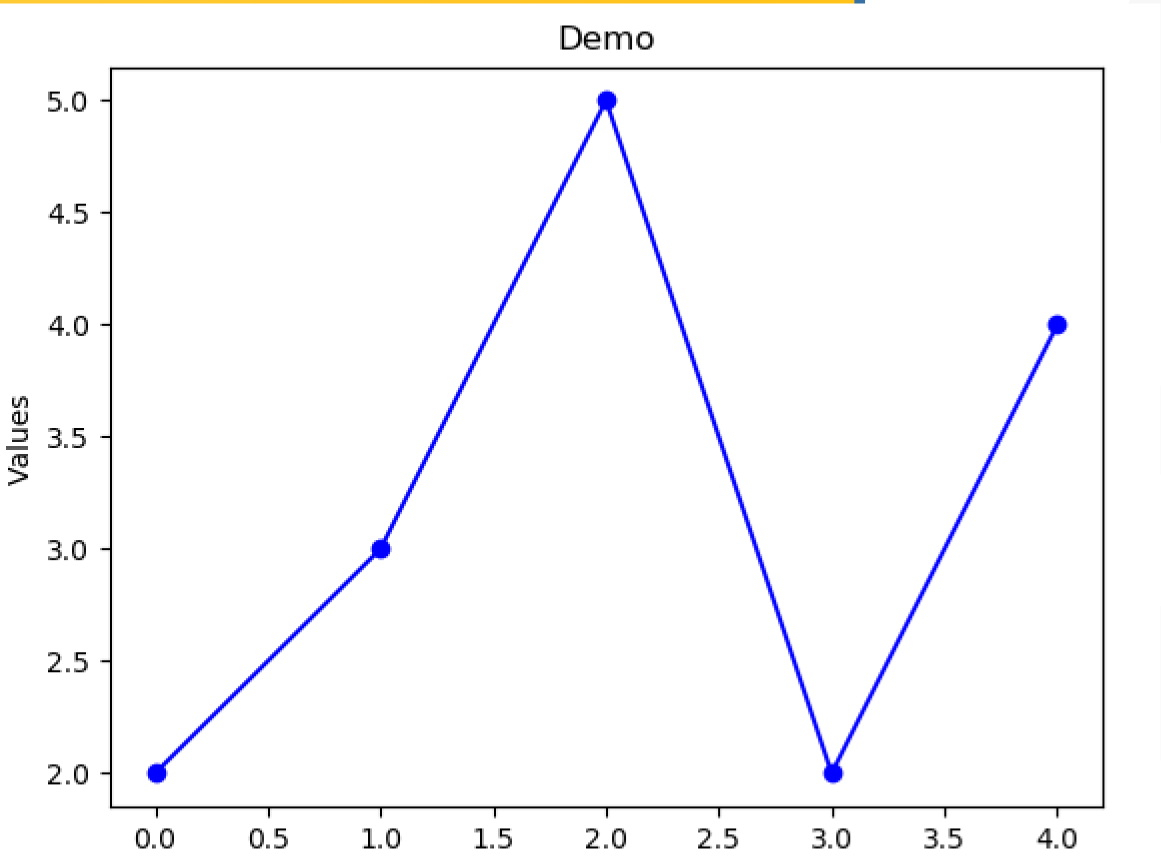
These functions calculate an average or typical value from a population or sample.

`mean()`	Arithmetic mean (“average”) of data.
`fmean()`	Fast, floating point arithmetic mean.
`geometric_mean()`	Geometric mean of data.
`harmonic_mean()`	Harmonic mean of data.
`median()`	Median (middle value) of data.
`median_low()`	Low median of data.
`median_high()`	High median of data.
`median_grouped()`	Median, or 50th percentile, of grouped data.
`mode()`	Single mode (most common value) of discrete or nominal data.
`multimode()`	List of modes (most common values) of discrete or nomimal data.
`quantiles()`	Divide data into intervals with equal probability.

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

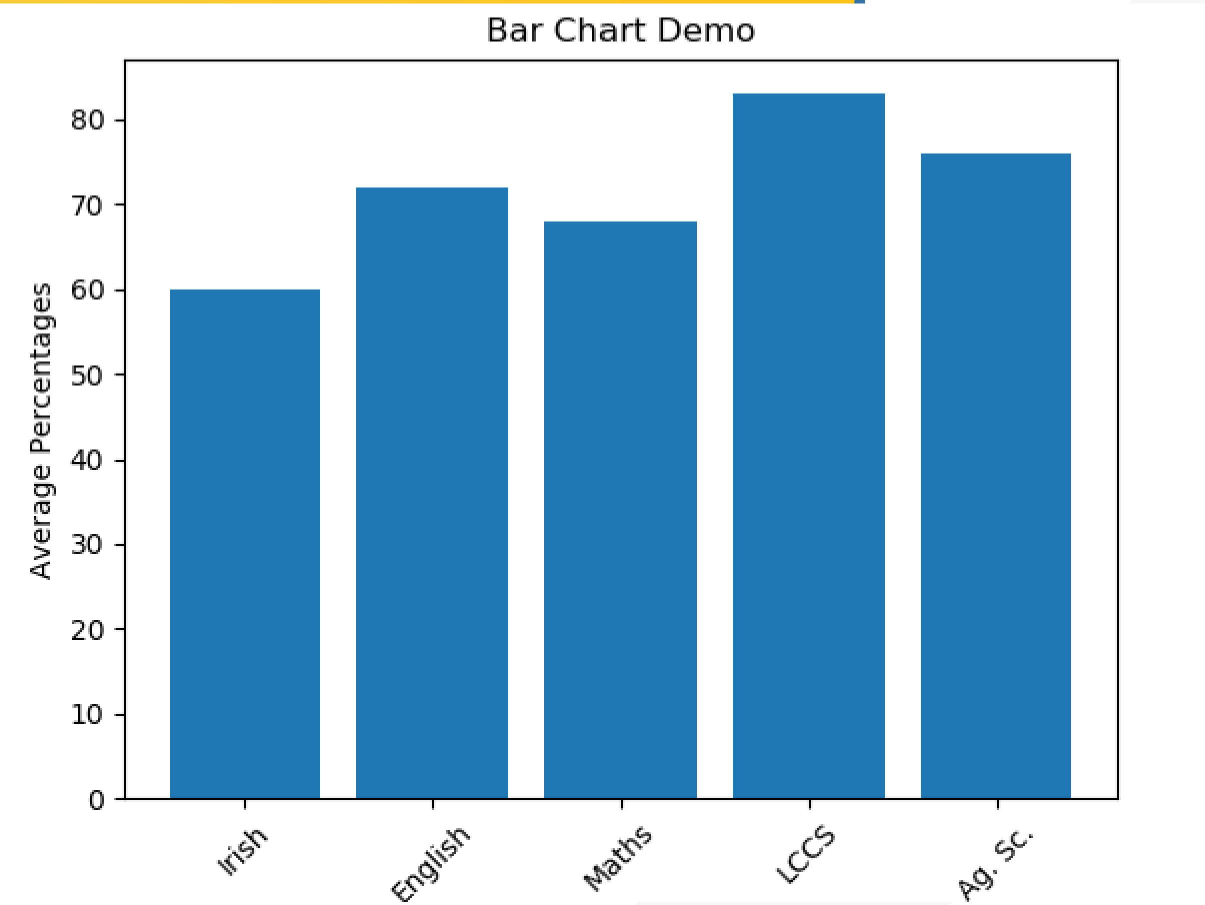
Demonstration of matplotlib

```
# A simple program to demonstrate use of matplotlib
from matplotlib import pyplot as plt

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

# Intervals for the x-axis
x_axis = [0, 1, 2, 3, 4]

plt.plot(x_axis, values, color='blue', lin

plt.title("Demo") # graph title
plt.ylabel("Values") # label the y-axis
plt.show() # Display the plot
```



Demonstration of `matplotlib`

```
# A simple program to demonstrate use of matplotlib
from matplotlib import pyplot as plt

# Initialise a list of subjects
subjects = ['Irish', 'English', 'Maths', 'LCCS', 'Ag. Sc.']

percentages = [60, 72, 68, 83, 76] # Avera

# Plot a bar chart
plt.bar(subjects, percentages)

plt.title("Bar Chart Demo") # graph title
plt.ylabel("Average Percentages") # label
# put the names of the subjects on the x-a
plt.xticks(range(len(subjects)), subjects,

plt.show() # Display the plot
```



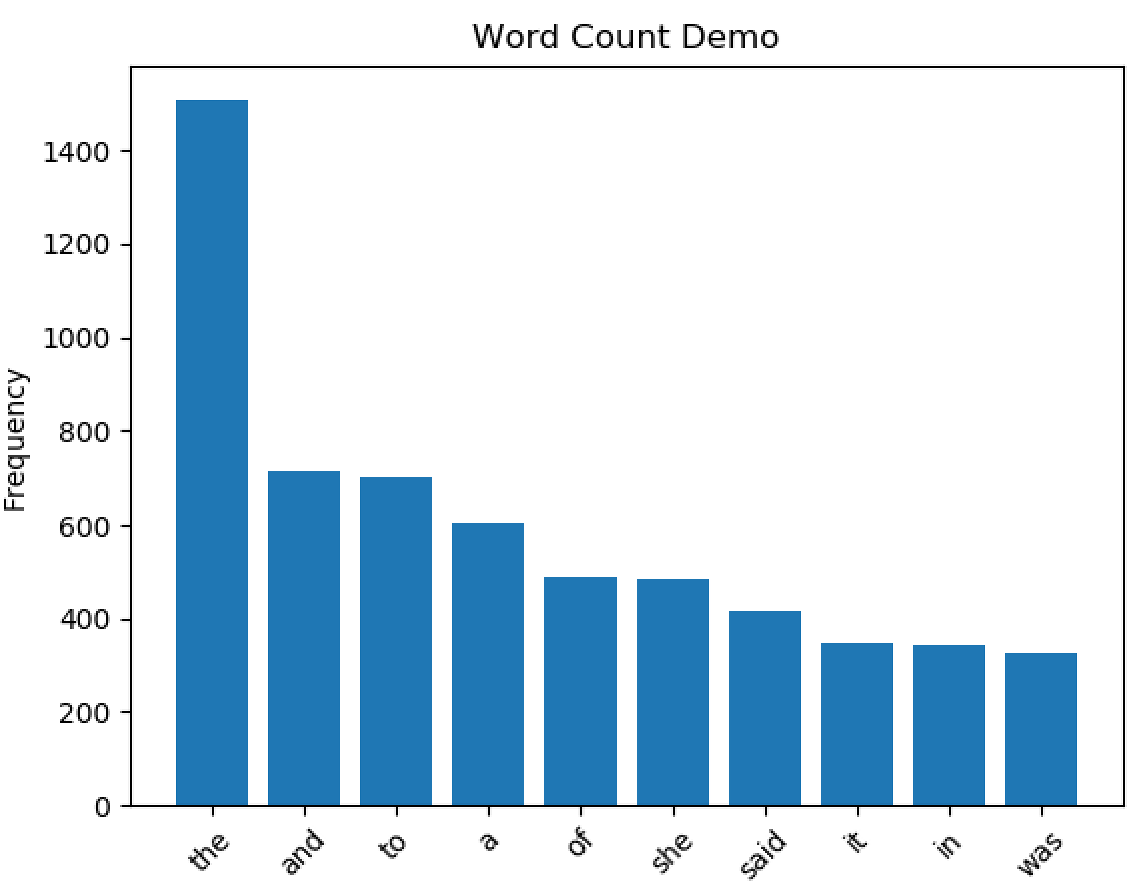
Text Analysis – word frequency

```
# A program to visualise the most common words in a file
from matplotlib import pyplot as plt
from collections import Counter

# IMPORTANT: Make sure book.txt exists in runtime directory
bookFile = open("book.txt","r") # Open the file
text = bookFile.read() # read the file
bookFile.close() # close the file
text_list = text.split() # create a list

# use counter to return the most common words
# format is .... [('the', 1507), ('and', 714), etc
most_common_words = Counter(text_list).most_common(10)

words = [] # an empty list of words
word_count = [] # an empty list of counts

# Build up the lists
for word, count in most_common_words:
    words.append(word) # append the word to the words list
    word_count.append(count)

# Now create and display the chart ….
```

Text Analysis – word frequency

… continued from previous slide

```
# Now create and display the chart ….

# Create the chart
plt.bar(words, word_count)
plt.title("Word Count Demo") # graph title
plt.ylabel("Frequency") # label the y-axis
# put the words on the x-axis
plt.xticks(range(len(words)), words, rotat
plt.show() # display the chart
```



Regular Expressions

A language that enables us to look for patterns in strings

```
import re

text1 = "THERE are 99 RED balloons"
print(re.sub('[0-9]', '', text1)) # remove digits
print(re.sub('[A-Z]', '', text1)) # remove uppercase
print(re.sub('[A-Z0-9]', '', text1)) # remove uppercase and digits
print(re.sub('[^a-z]', '', text1)) # leave lowercase
print(re.sub('[^a-zA-Z ]', '', text1)) # leave letters and spaces
print(re.sub('[^a-zA-Z0-9]', ' ', text1)) # leave letters and digits
print(re.sub(r'\b\w{1,4}\b', '', text1)) # remove words of length 1-3

text1 = "$%**$%joe*&$%^&"
print(re.sub('[^a-zA-Z0-9]', '', text1))
```

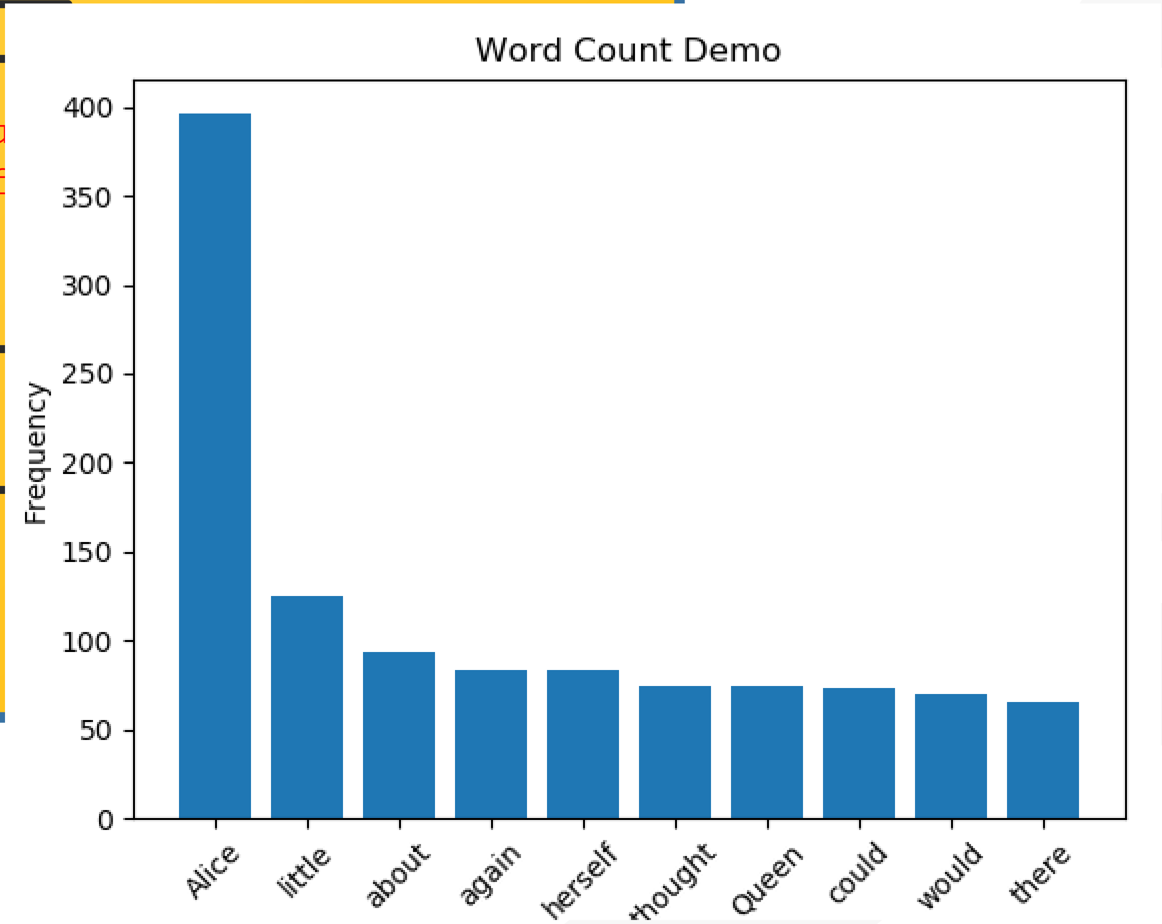
Output

```
THERE are  RED balloons
 are 99  balloons
 are   balloons
areballoons
THERE are  RED balloons
THERE are 99 RED balloons
THERE    balloons


joe
```

Text Analysis – word frequency

Eliminate words of three letters or less … use Regular Expressions

```
# A program to visualise the most common words in a file
from matplotlib import pyplot as plt
from collections import Counter
import re

# IMPORTANT: Make sure book.txt exists in ru
bookFile = open("book.txt","r") # Open the f
text = bookFile.read() # read the file
bookFile.close() # close the file

text = re.sub('[^a-zA-Z0-9 \n]', ' ', text)
text = re.sub(r'\b\w{1,4}\b', '', text)

text_list = text.split() # create a list

# Continue as before …
```

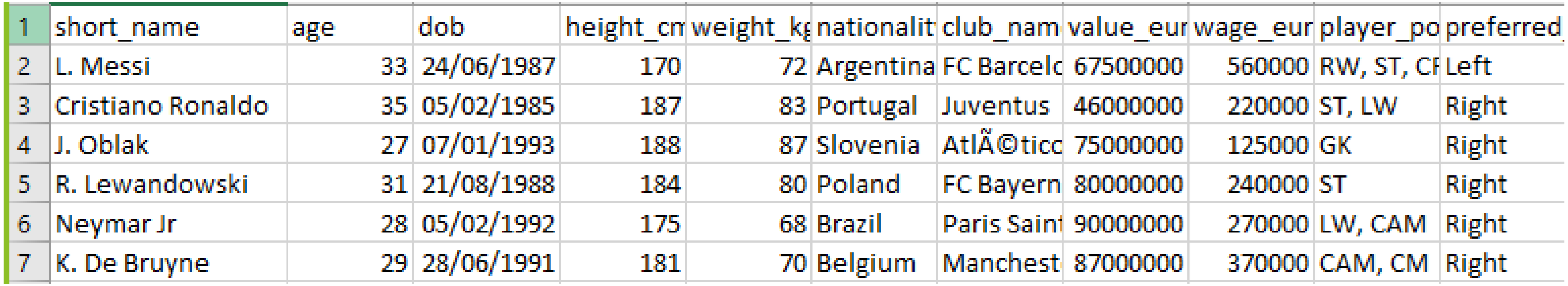


Import the `re` library

Use the `sub` method

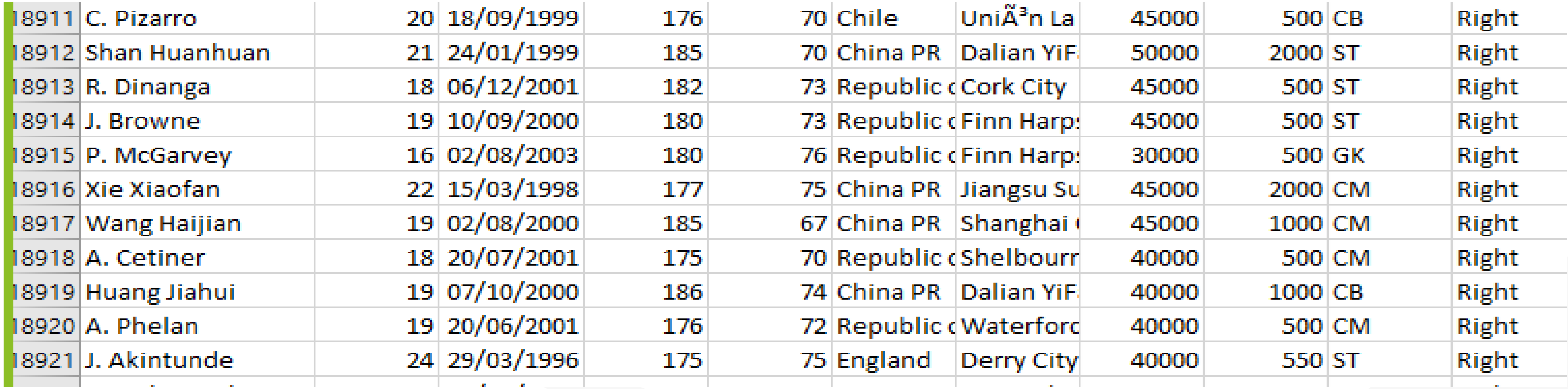
Pandas

Useful for very large files … this file was sourced on Kaggle



1	short_name	age	dob	height_cm	weight_kg	nationalit	club_nam	value_eur	wage_eur	player_po	preferred_
2	L. Messi	33	24/06/1987	170	72	Argentina	FC Barcelc	67500000	560000	RW, ST, CF	Left
3	Cristiano Ronaldo	35	05/02/1985	187	83	Portugal	Juventus	46000000	220000	ST, LW	Right
4	J. Oblak	27	07/01/1993	188	87	Slovenia	AtlÃ©ticc	75000000	125000	GK	Right
5	R. Lewandowski	31	21/08/1988	184	80	Poland	FC Bayern	80000000	240000	ST	Right
6	Neymar Jr	28	05/02/1992	175	68	Brazil	Paris Saint	90000000	270000	LW, CAM	Right
7	K. De Bruyne	29	28/06/1991	181	70	Belgium	Manchest	87000000	370000	CAM, CM	Right

… …



18911	C. Pizarro	20	18/09/1999	176	70	Chile	UniÃ³n La	45000	500	CB	Right
18912	Shan Huanhuan	21	24/01/1999	185	70	China PR	Dalian YiF	50000	2000	ST	Right
18913	R. Dinanga	18	06/12/2001	182	73	Republic c	Cork City	45000	500	ST	Right
18914	J. Browne	19	10/09/2000	180	73	Republic c	Finn Harp:	45000	500	ST	Right
18915	P. McGarvey	16	02/08/2003	180	76	Republic c	Finn Harp:	30000	500	GK	Right
18916	Xie Xiaofan	22	15/03/1998	177	75	China PR	Jiangsu Su	45000	2000	CM	Right
18917	Wang Haijian	19	02/08/2000	185	67	China PR	Shanghai (	45000	1000	CM	Right
18918	A. Cetiner	18	20/07/2001	175	70	Republic c	Shelbourr	40000	500	CM	Right
18919	Huang Jiahui	19	07/10/2000	186	74	China PR	Dalian YiF	40000	1000	CB	Right
18920	A. Phelan	19	20/06/2001	176	72	Republic c	Waterforc	40000	500	CM	Right
18921	J. Akintunde	24	29/03/1996	175	75	England	Derry City	40000	550	ST	Right

Let’s explore the player’s value

Pandas

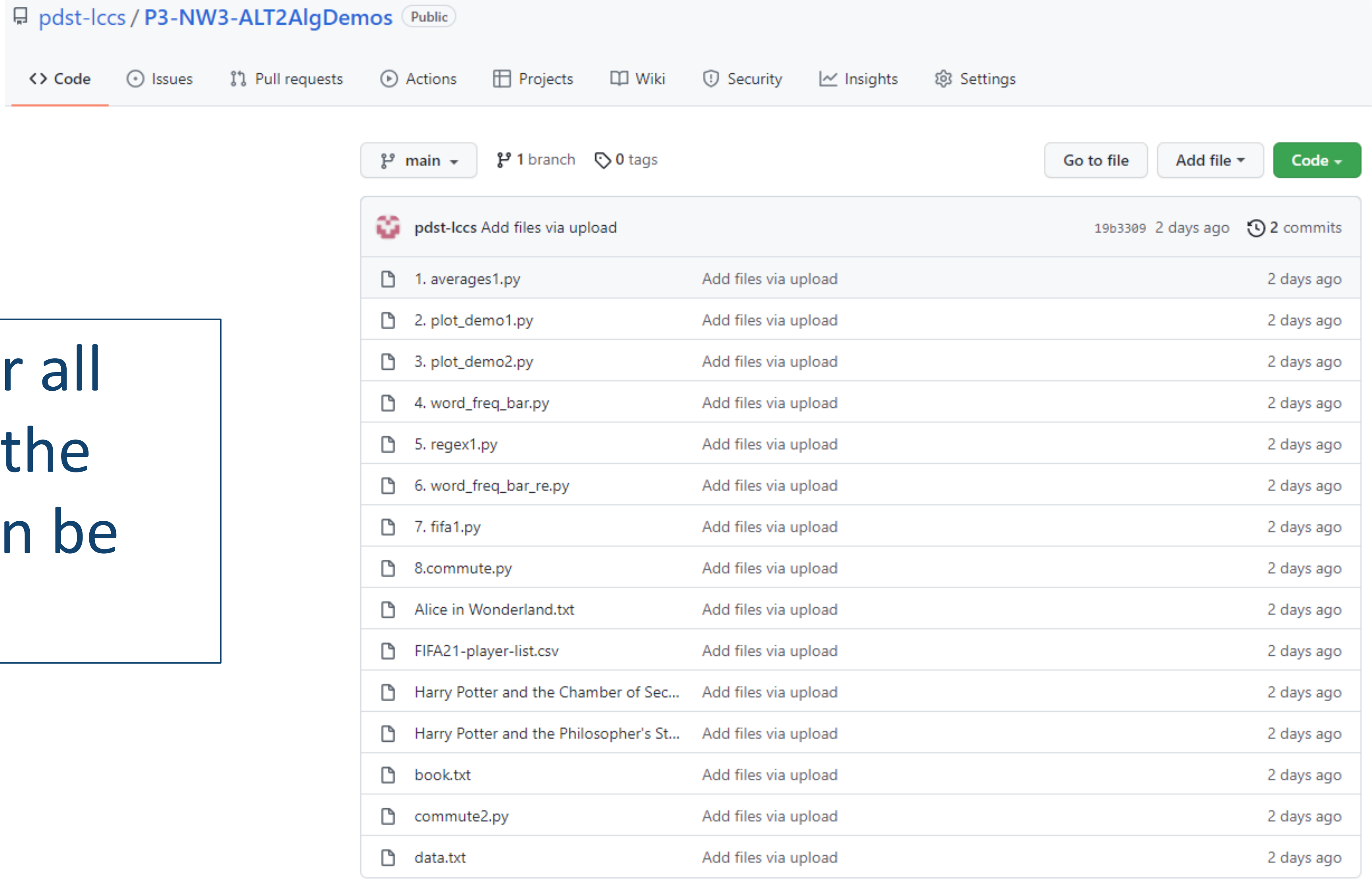
```python
# Using pandas - recommended for larger files
import statistics
import pandas

# Read the entire CSV file into a pandas DataFrame
df = pandas.read_csv('FIFA21-player-list.csv')

# Filter out the column, value_eur
player_values = df['value_eur']

# Compute and display the mean
mean_value = round(statistics.mean(player_values), 2)
print("Mean Value:", mean_value)

# Compute and display the median
median_value = statistics.median(player_values)
print("Median Value:", median_value)

# Compute and display the min and max values
print("Min: €%f, Max: €%f" %(min(player_values),max(player_values)))
```

Output looks like this:

```
Mean Value: 2224813.29
Median Value: 650000.0
Min: €0.000000, Max: €105500000.000000
```

GitHub

The source code for all the files shown on the preceding slides can be found on GitHub



pdst-lccs Add files via upload 19b3309 2 days ago 2 commits

1. averages1.py	Add files via upload	2 days ago
2. plot_demo1.py	Add files via upload	2 days ago
3. plot_demo2.py	Add files via upload	2 days ago
4. word_freq_bar.py	Add files via upload	2 days ago
5. regex1.py	Add files via upload	2 days ago
6. word_freq_bar_re.py	Add files via upload	2 days ago
7. fifa1.py	Add files via upload	2 days ago
8.commute.py	Add files via upload	2 days ago
Alice in Wonderland.txt	Add files via upload	2 days ago
FIFA21-player-list.csv	Add files via upload	2 days ago
Harry Potter and the Chamber of Sec...	Add files via upload	2 days ago
Harry Potter and the Philosopher's St...	Add files via upload	2 days ago
book.txt	Add files via upload	2 days ago
commute2.py	Add files via upload	2 days ago
data.txt	Add files via upload	2 days ago

https://github.com/pdst-lccs/P3-NW3-ALT2AlgDemos

Section V

Final Reflection – NCCA Sample ALT2

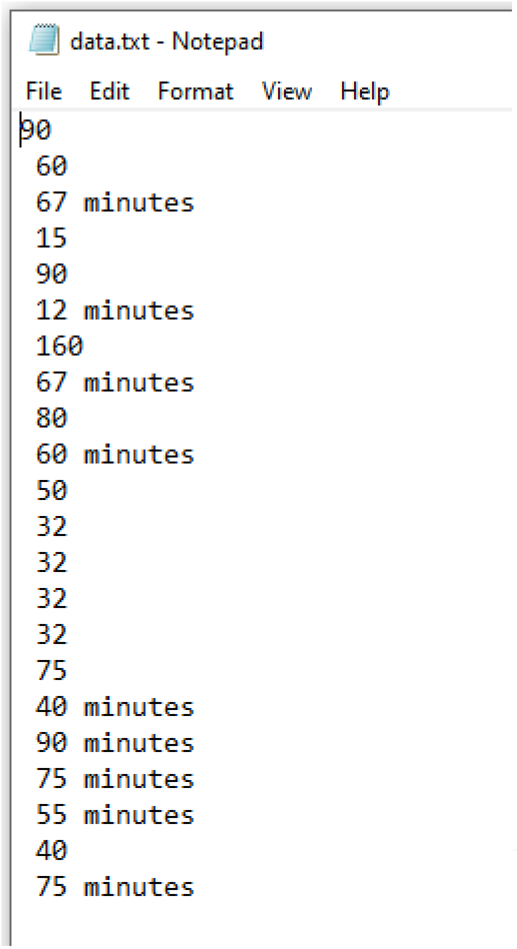
ALT2 Samples

NCCA
An Chomhairle Náisiúnta Curaclaim agus Measúnachta
National Council for Curriculum and Assessment

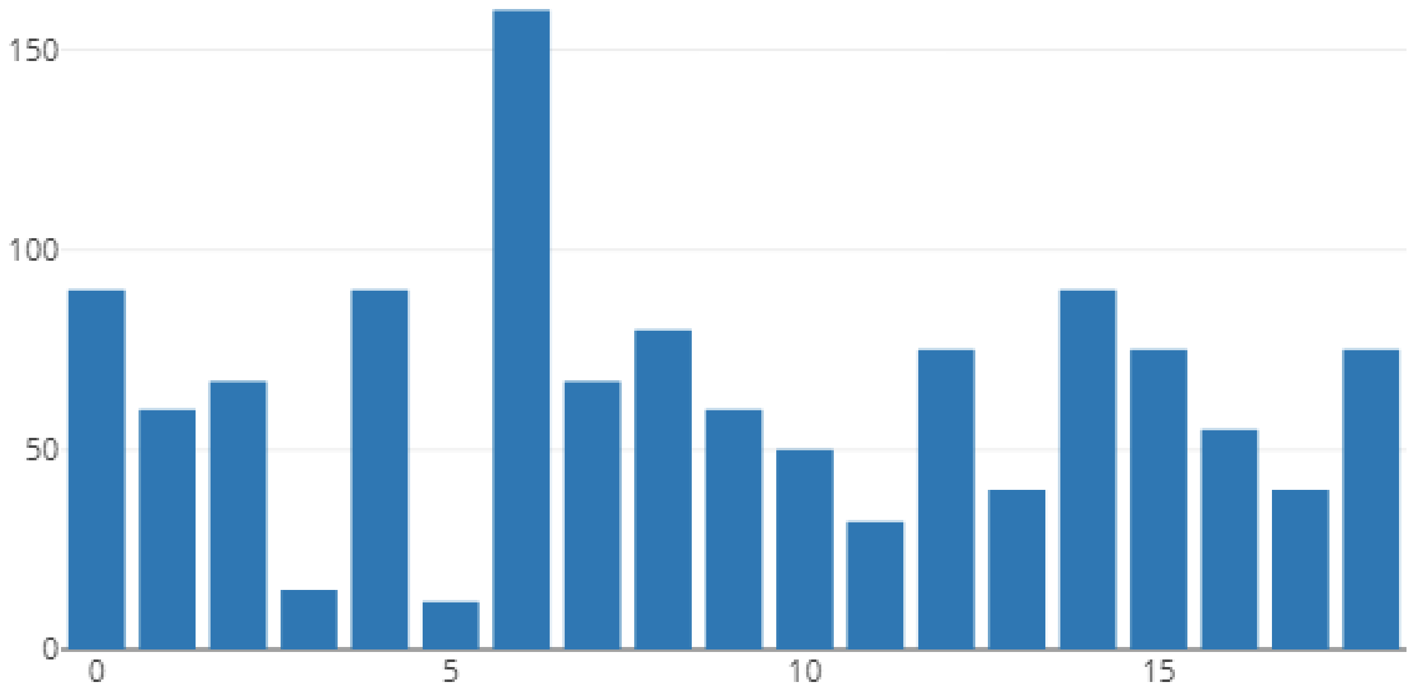
https://www.curriculumonline.ie/Senior-cycle/Senior-Cycle-Subjects/Computer-Science/CS-Support-for-Teaching-and-Learning/Support-Material-for-Teaching-and-Learning/2-ALT-Resources/ALT2-Support/

Commute Times

“Our topic is travel times, our data source are the other groups working and our hypothesis is that the average travel time will be 50 minutes and no one will have traveled for longer than 2 hours.”



```
90
 60
 67 minutes
 15
 90
 12 minutes
 160
 67 minutes
 80
 60 minutes
 50
 32
 32
 32
 32
 75
 40 minutes
 90 minutes
 75 minutes
 55 minutes
 40
 75 minutes
```



```
# Sample ALT2 - Commute times
import statistics
import re
import plotly.plotly
from plotly.graph_objs import Bar, Layout

# Open and read the data file
file = open("data.txt","r")
string = file.read()
file.close()

# Scrub the data
clean_string = re.sub(' minutes', '', string)
clean_string = re.sub(' ', '', clean_string)
string_array = clean_string.split('\n')

# Convert all the strings to integers
int_array = [int(i) for i in string_array]

# Determine and display the averages
mean_value = statistics.mean(int_array)
median_value = statistics.median_grouped(int_array, 1)
mode_value = statistics.mode(int_array)
print("Mean: %.2f, Median %d, Mode %d" %(mean_value, median_value, mode_value))

plotly.offline.plot({"data": [Bar(y=int_array)],
  "layout": Layout(title="word count")
})
```

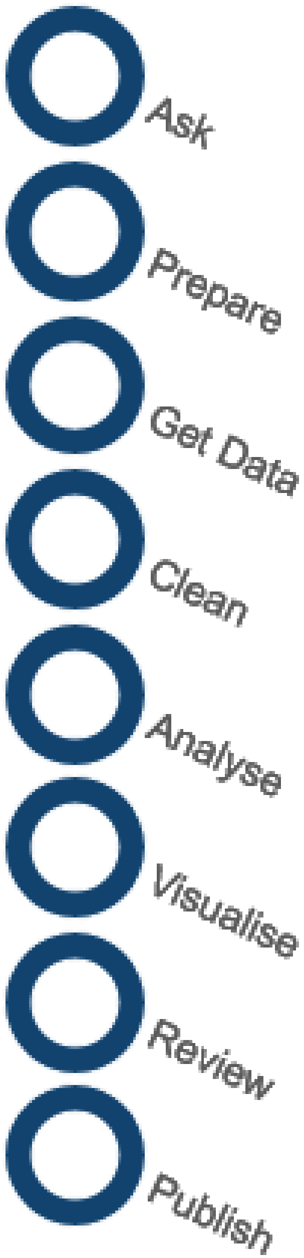
Final Reflection

1. What prior programming knowledge/skills would students need to have in order to engage with ALT2?

2. What will students enjoy most about ALT2? What might challenge them most?

3. How might the Data Science Arc be used to support student’s engagement with ALT2?

4. What next step(s) will you take to prepare your students for ALT2 and support their progress?



An Roinn Oideachais
Department of Education