PDST

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

LEAVING CERTIFICATE COMPUTER SCIENCE

National Workshop 3

© PDST 2018

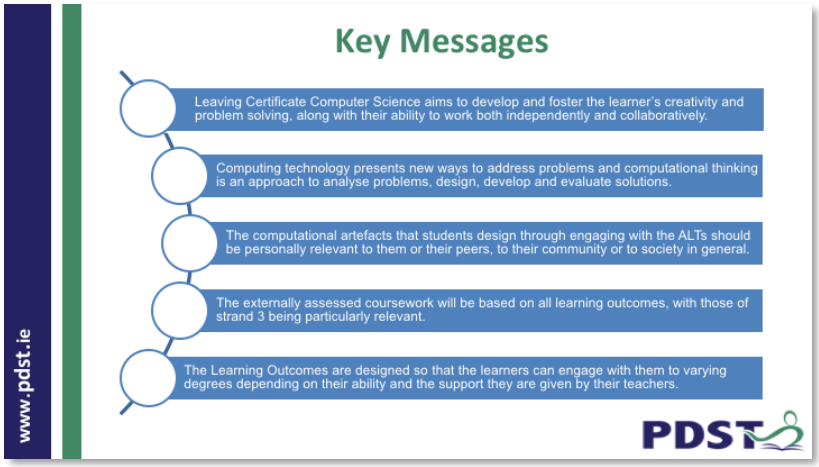
This work is made available under the terms of the Creative Commons Attribution Share Alike 3.0 Licence http://creativecommons.org/licenses/by-sa/3.0/ie/. You may use and re-use this material (not including images and logos) free of charge in any format or medium, under the terms of the Creative Commons Attribution Share Alike Licence.

Please cite as: PDST, *Leaving Certificate Computer Science, National Workshop 3*, Dublin, 2018

Table of Contents

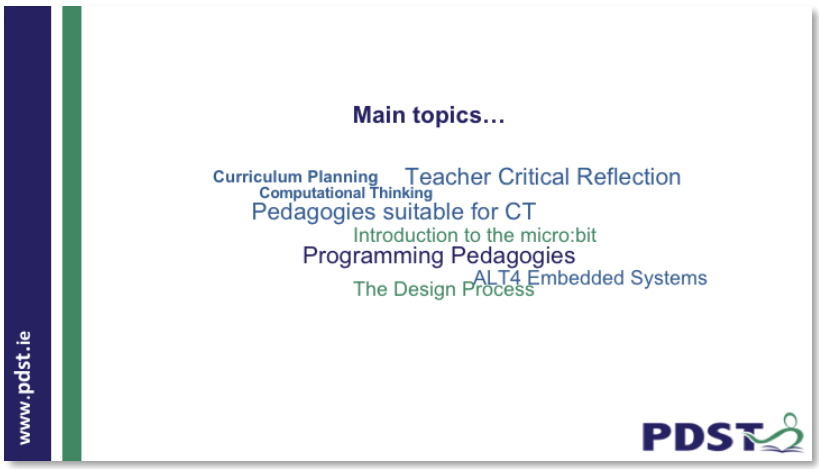
Session 1

Data Analytics and ALT2

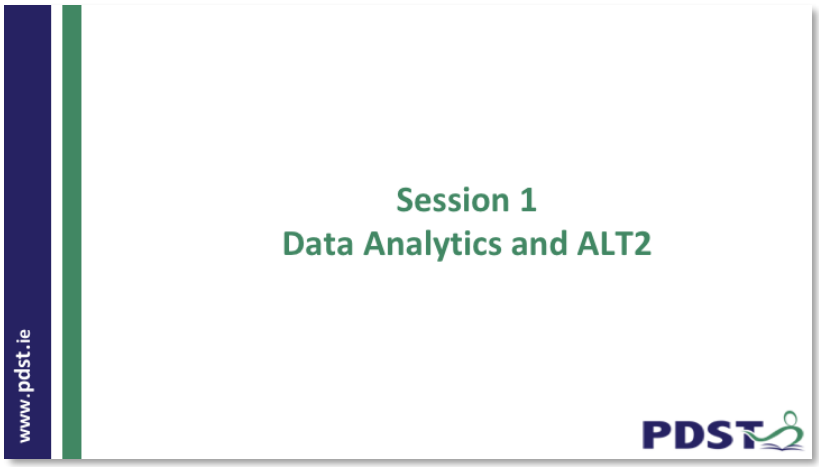


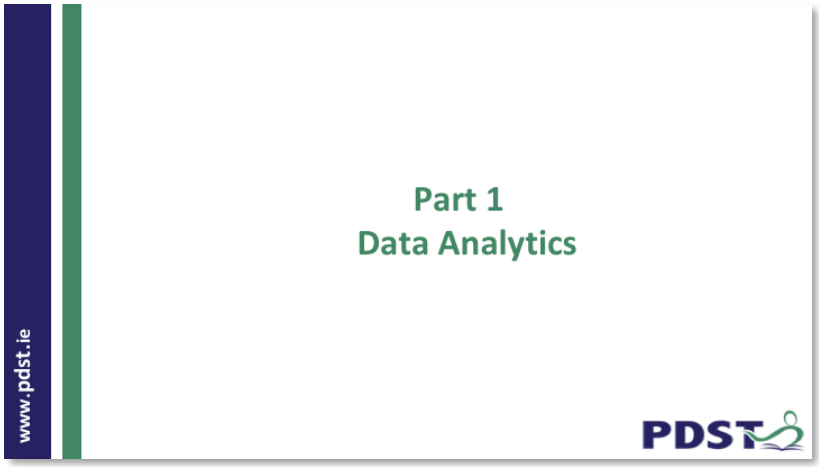
National Workshop 2

Quick Recap



NOTES:





What words do you associate with Data Science / Data Analytics?

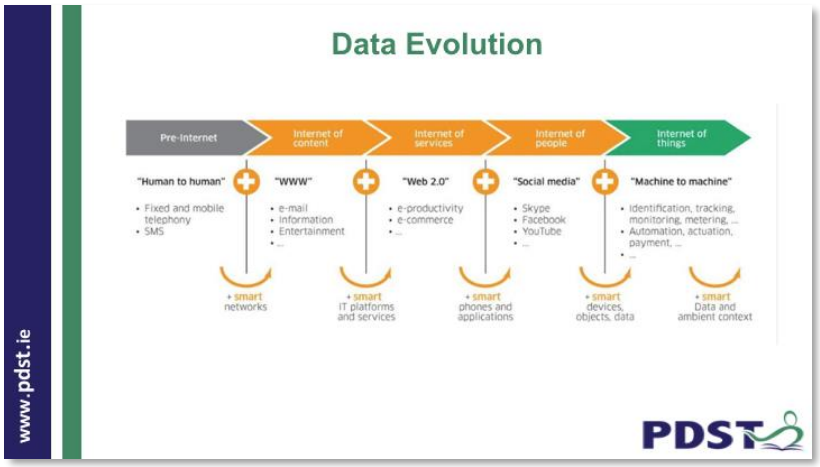
Activity 1 - Initial thoughts on terminology

Reflect on the Menti/Wordcloud activity and use the space provided to record your thoughts.

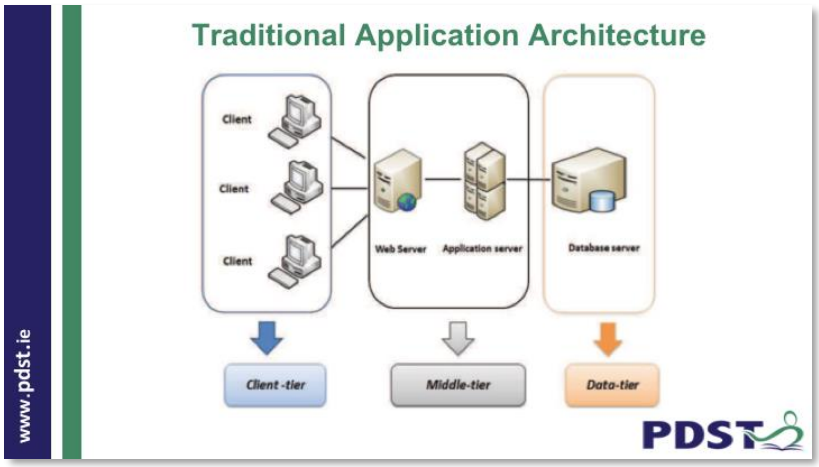
What words do you associate with Data Science/Data Analytics?

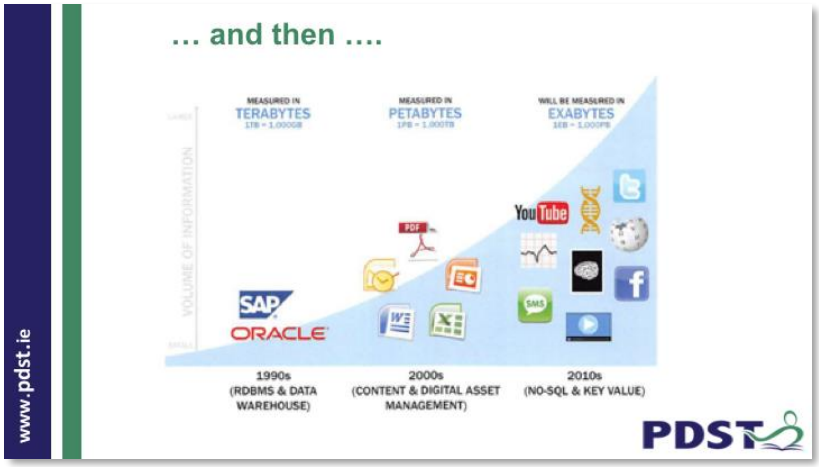
Other questions to consider ….
What is data?
Are data and information different, and if so how?
Where does data come from and what types/formats are data stored in?
What are the characteristics of good data? What about good information?
What is meant by the term Big Data?
How are Data Mining and Machine Learning related?

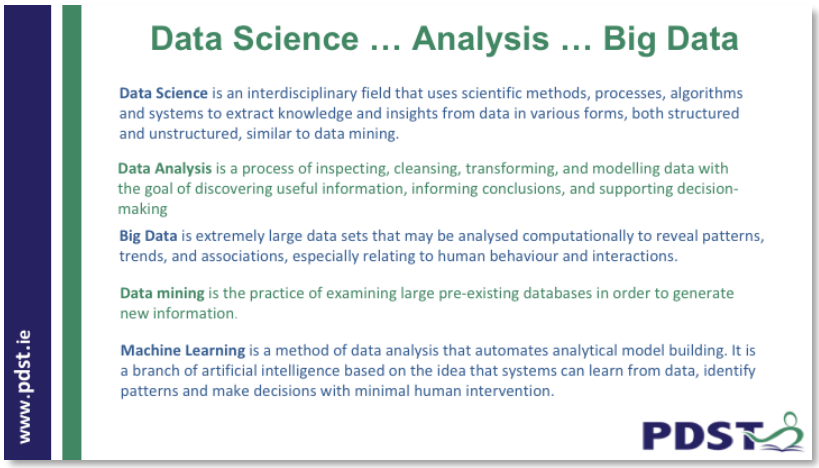
https://www.youtube.com/watch?v=eVSfJhssXUA



NOTES:


Traditional Application Architecture
Client
Client
Client
Web Server
Application server
Database server
Client -tier
Middle-tier
Data-tier
www.pdst.ie
PDST


… and then ….
MEASURED IN
TERABYTES
MEASURED IN
PETABYTES
WILL BE MEASURED IN
EXABYTES
VOLUME OF INFORMATION
SAP
ORACLE
1990s
(RDBMS & DATA WAREHOUSE)
2000s
(CONTENT & DIGITAL ASSET MANAGEMENT)
2010s
(NO-SQL & KEY VALUE)
www.pdst.ie
PDST


Data Science … Analysis … Big Data
Data Science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured, similar to data mining.
Data Analysis is a process of inspecting, cleansing, transforming, and modelling data with the goal of discovering useful information, informing conclusions, and supporting decision-making
Big Data is extremely large data sets that may be analysed computationally to reveal patterns, trends, and associations, especially relating to human behaviour and interactions.
Data mining is the practice of examining large pre-existing databases in order to generate new information.
Machine Learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.
www.pdst.ie
PDST

NOTES:

www.pdst.ie

Part 2
Applied Learning Task 2 (ALT2)

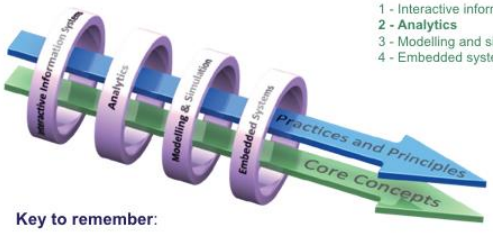
PDST

www.pdst.ie

LCCS Interwoven

The four applied learning tasks explore the four following contexts:

1 - Interactive information systems
2 - Analytics
3 - Modelling and simulation
4 - Embedded systems.



Key to remember:

Explore and teach the LOs through the lens of ALTs.

PDST

www.pdst.ie

'Students work in teams to carry out four applied learning tasks over the duration of the course.'

'Each of which results in the creation of a real or virtual computational artefact.'

'These artefacts should relate to the students' lives and interests.'

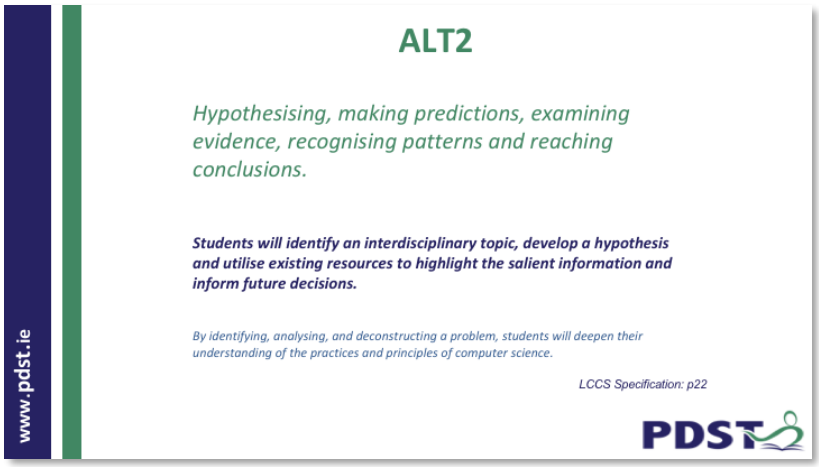
'Where possible, the artefacts should be beneficial to the community and society in general.'

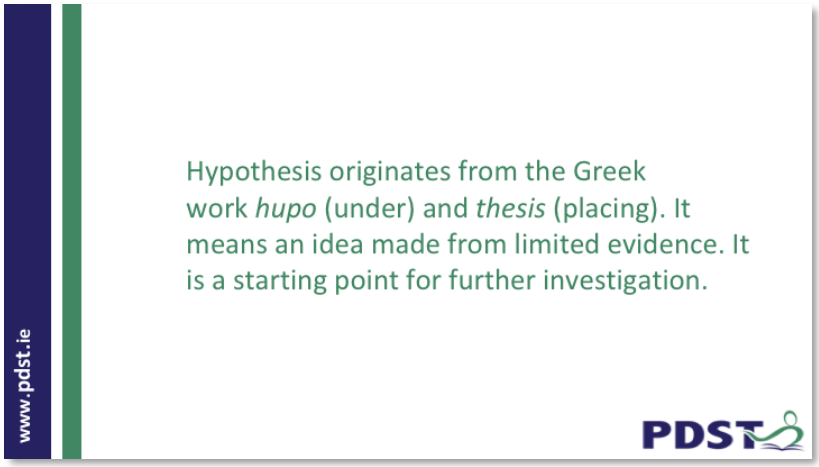
'Examples of computational artefacts include programs, games, web pages, simulations, visualisations, digital animations, robotic systems, and apps.'

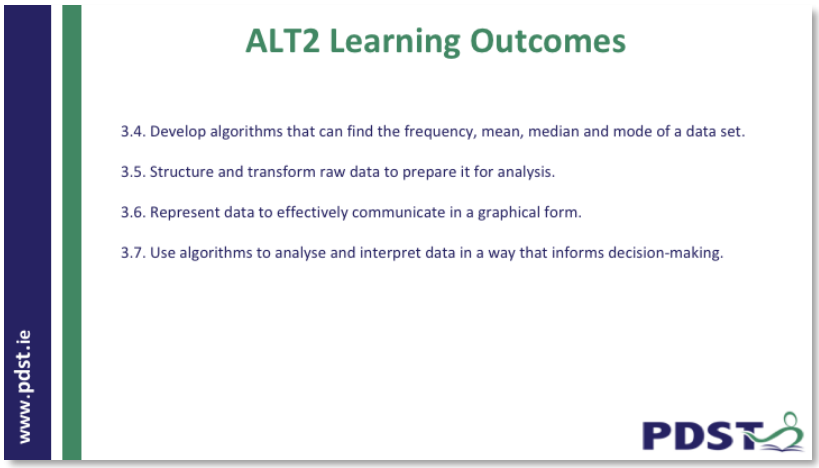
LCCS Specification: p15

PDST

NOTES:


ALT2
Hypothesising, making predictions, examining evidence, recognising patterns and reaching conclusions.
Students will identify an interdisciplinary topic, develop a hypothesis and utilise existing resources to highlight the salient information and inform future decisions.
By identifying, analysing, and deconstructing a problem, students will deepen their understanding of the practices and principles of computer science.
LCCS Specification: p22
www.pdst.ie
PDST

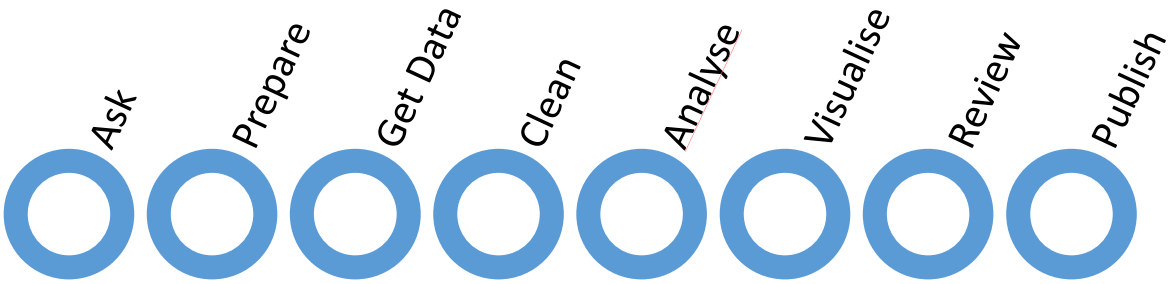

Hypothesis originates from the Greek work hupo (under) and thesis (placing). It means an idea made from limited evidence. It is a starting point for further investigation.
www.pdst.ie
PDST


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.
www.pdst.ie
PDST

NOTES:

Make up your own hypotheses and record them here …..

Data Science Arc



ASK – The Question that starts the journey

PREPARE – Sketch out, think through ideas to organise work.

GET DATA – Collect, enter, reuse or repurpose.

CLEAN – Format, layout, organise.

ANALYSE – Format, layout, organise, sort, filter, summarize, triangulate.

VISUALISE – Format charts, tables, add logos, branding, colours.

REVIEW – Gather feedback, find errors, check interpretations.

PUBLISH – Secure and share within or outside the team.

Microsoft

Hacking STEM

Activity 2: Supporting ALT2

Discuss the following questions in pairs.

What prior knowledge will students have that is relevant to ALT2?

What may challenge students in dealing with ALT2?

From what you have seen and heard in this session so far, what approach could you take to introduce ALT2 to your students?

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

Sample Resources

Demonstration 1

Use the prompts below to record your thoughts *as the demonstration is being presented*

How is this demonstration/example extending my thinking in relation to ALT2?

What questions do I have in relation to the demonstration?

What ideas has this demonstration given me that I could use to support my own students?

Sample 1

1. Browse to the link below and download one of the sample ALT2 resources.
https://nccacurriculum.azurewebsites.net/Senior-cycle/Senior-Cycle-Subjects/Computer-Science/Support-Material-for-Teaching-and-Learning/2-ALT-Resources/ALT2-Support

2. Unpack the resource and run it from your device

Checklist

	Tick (Yes/No)
I know where the samples are	
I know how to download and browse each example	
I understand what I need to do to run the sample locally (in my own local environment)	
I can run the sample locally (in my own local environment)	

3. Browse through the resource and use the space provided to record your notes about the sample. Explain how the sample fits the Data Science Arc. The prompts may be helpful.

What is the name of the sample? What hypothesis is the sample designed to test? What data source(s) does the sample use? Where does the data come from? What is the output and how is it interpreted in the sample?

Describe how the algorithm works and identify what programming constructs a student would need to know in order to understand it.

How do you support students to engage with this process? Consider the Learning Outcomes that could be referenced.

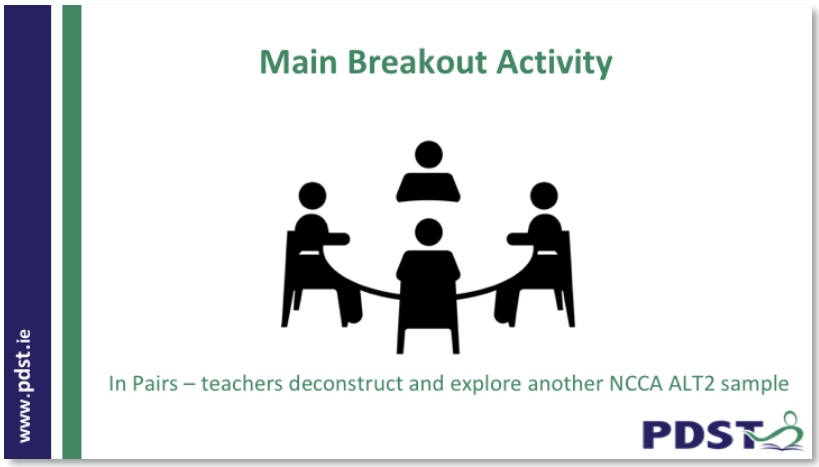
How might this example be modified/enhanced?

What new ideas has this example given me that I could try with my own students?

Demonstration 2 (temperature processing)

Use the prompts below to record your thoughts *as the demonstration is being presented*

How is this demonstration/example extending my thinking in relation to ALT2? What ideas has this demonstration given me that I could use to support my own students? What hypotheses might students devise that relate to this example.


Main Breakout Activity
www.pdst.ie
In Pairs – teachers deconstruct and explore another NCCA ALT2 sample
PDST


What did you do?
How did you do it?
How would you support students to engage in a similar process
What has challenged your thinking?
Which LOs did you use?
Problems?
www.pdst.ie
Presentation & Debrief
PDST

NOTES:

Session 2

ALT2 Development and Curriculum Planning

www.pdst.ie

Session 2

ALT2 – Project Design and Development
Curriculum & Assessment Planning

PDST

www.pdst.ie

Main Group Activity

Groups will begin to develop a new ALT2

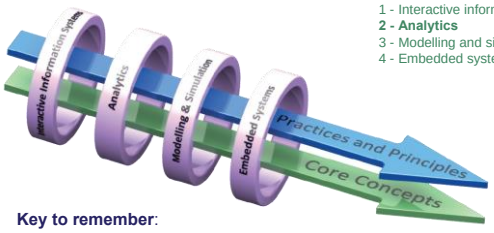
PDST

www.pdst.ie

LCCS Interwoven

The four applied learning tasks explore the four following contexts:

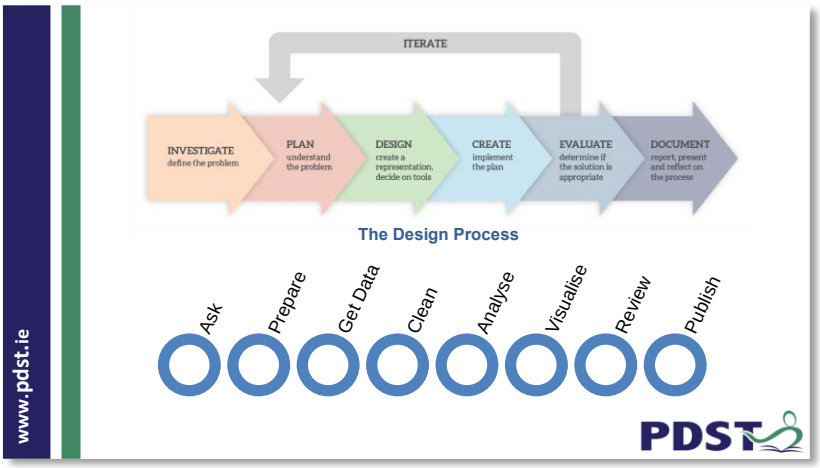
1 - Interactive information systems
2 - Analytics
3 - Modelling and simulation
4 - Embedded systems.

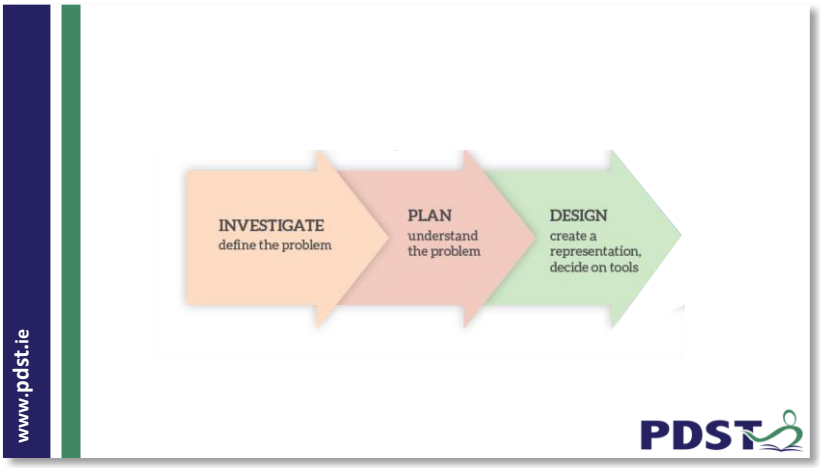


Key to remember:

Explore and teach the LOs through the lens of ALTs.

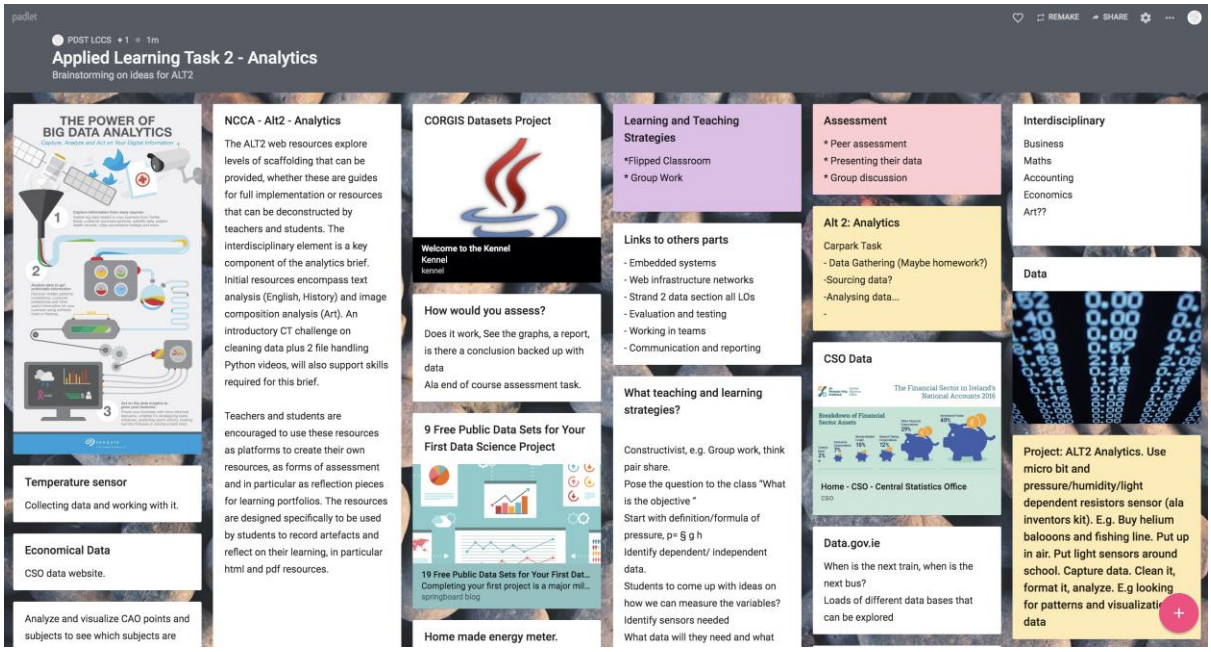
PDST


ITERATE
INVESTIGATE
define the problem
PLAN
understand the problem
DESIGN
create a representation, decide on tools
CREATE
implement the plan
EVALUATE
determine if the solution is appropriate
DOCUMENT
report, present and reflect on the process
The Design Process
Ask
Prepare
Get Data
Clean
Analyse
Visualise
Review
Publish
www.pdst.ie
PDST


INVESTIGATE
define the problem
PLAN
understand the problem
DESIGN
create a representation, decide on tools
www.pdst.ie
PDST


1
INVESTIGATE
define the problem
www.pdst.ie
PDST

ALT 2 Brainstorm on Padlet



padlet

https:// pdstlccs.padlet.org/cpd/6s9qwbnzl8rm

NOTES:

Additional Resources - Data Sets

Find the full list of data sets at:
https://docs.google.com/spreadsheets/d/1vR9tFCchkV6SLbCrponOhho45Nr2Zqk6xKgZ17KuCBU/edit?usp=sharing

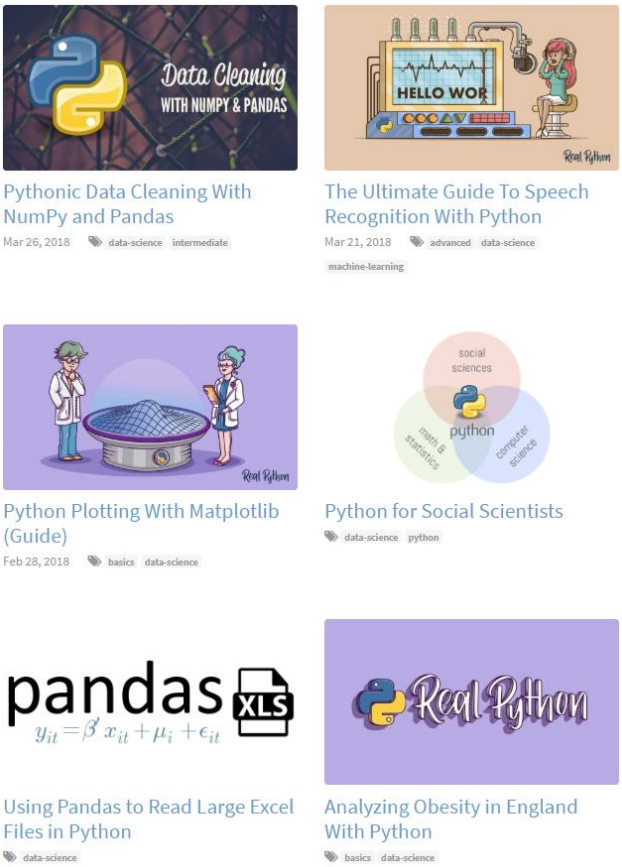
NOTES:

Additional Resources - Useful Tutorials

Microsoft

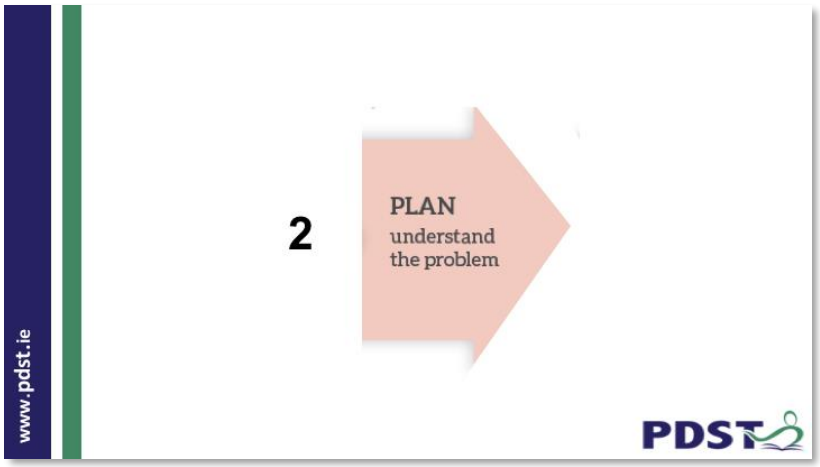
Hacking STEM

https://www.microsoft.com/en-us/education/education-workshop/comparing-speeds.aspx



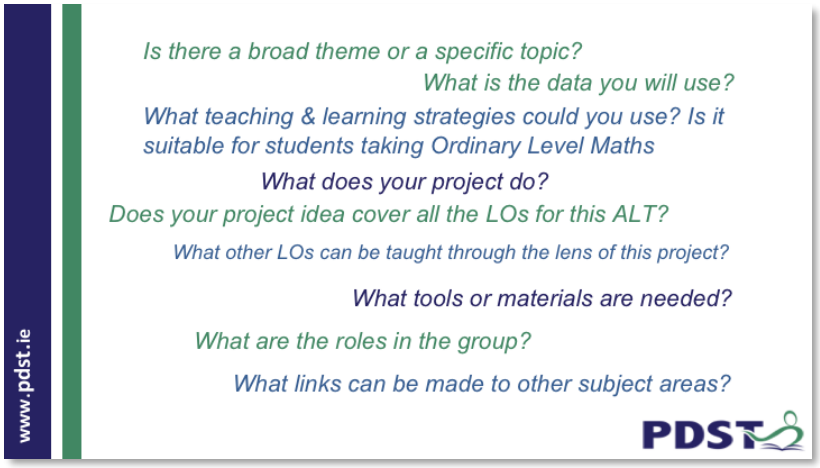
https://realpython.com/tutorials/data-science/

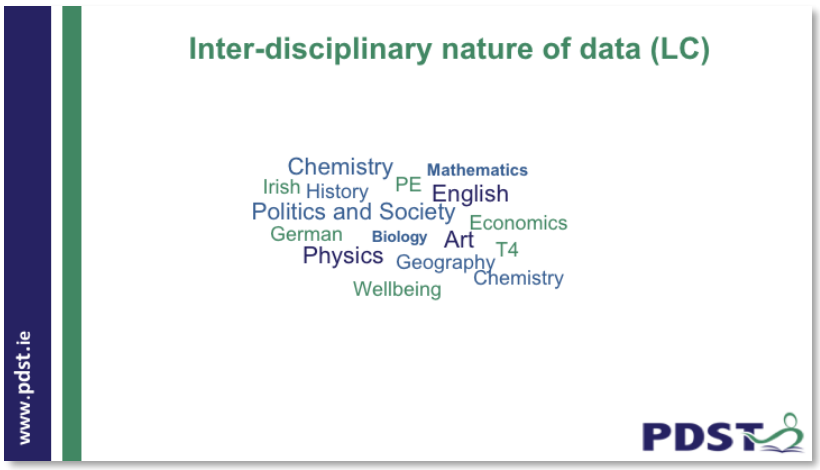
NOTES:



Pick one of the suggestions for ALT2

Dissect the idea





www.pdst.ie

3 DESIGN
create a representation, decide on tools

PDST

www.pdst.ie

https://blog.overflow.io/8-tips-for-creating-better-user-flows-e46eb0d2a2c6

PDST

www.pdst.ie

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

PDST

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

www.pdst.ie

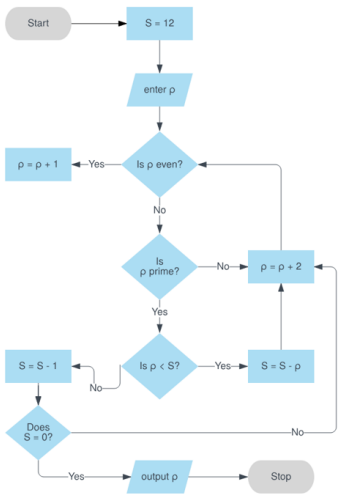
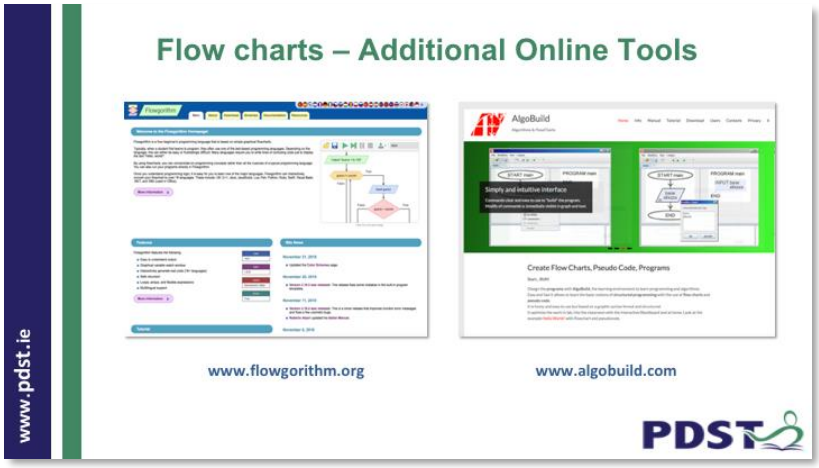
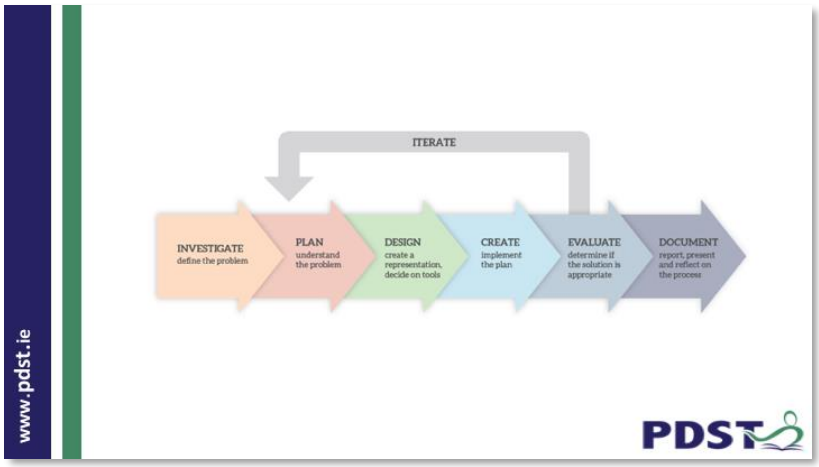


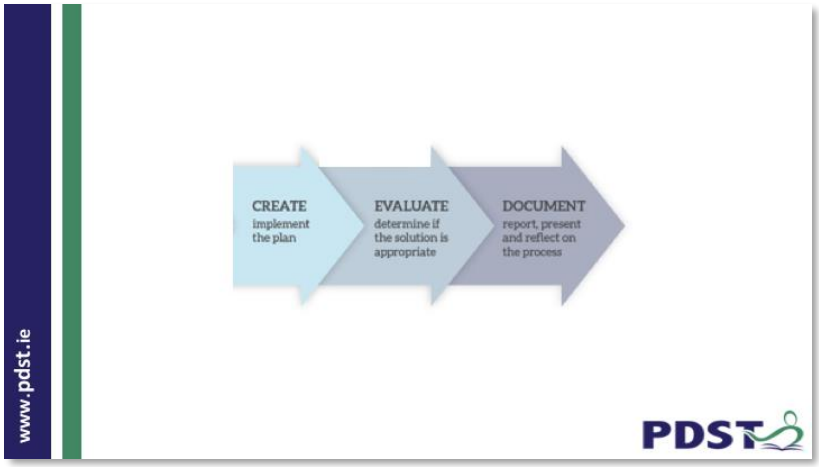
Image sourced from Lucid Software



www.flowgorithm.org www.algobuild.com

NOTES:

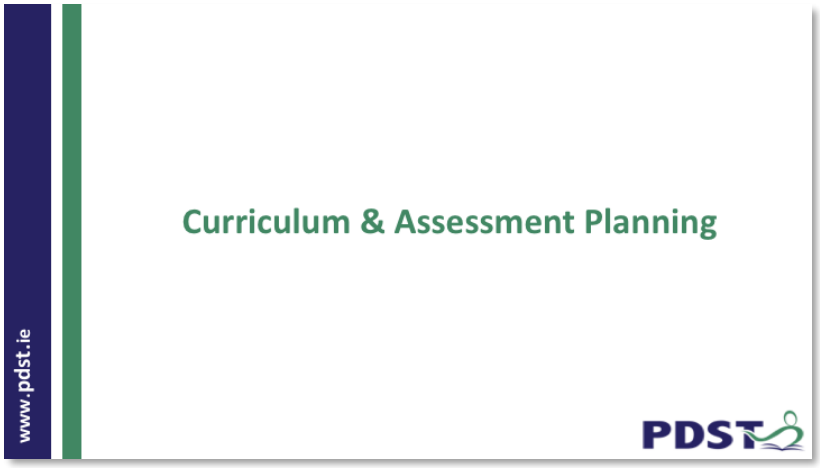

ITERATE
INVESTIGATE
define the problem
PLAN
understand the problem
DESIGN
create a representation, decide on tools
CREATE
implement the plan
EVALUATE
determine if the solution is appropriate
DOCUMENT
report, present and reflect on the process
www.pdst.ie
PDST

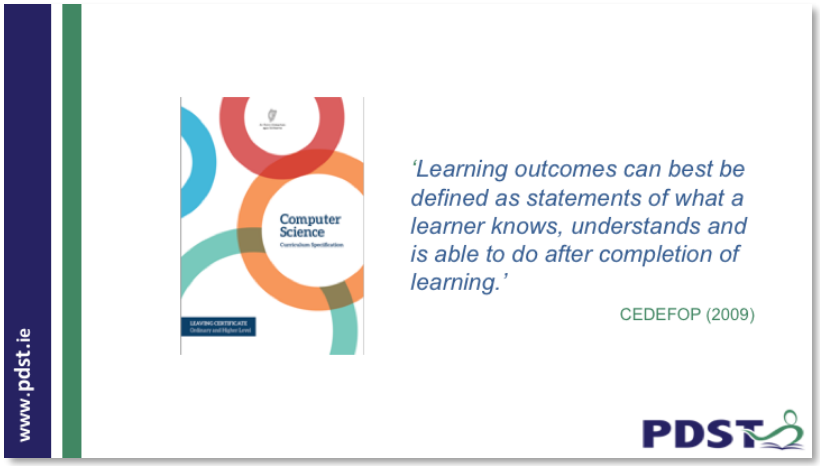

CREATE
implement the plan
EVALUATE
determine if the solution is appropriate
DOCUMENT
report, present and reflect on the process
www.pdst.ie
PDST

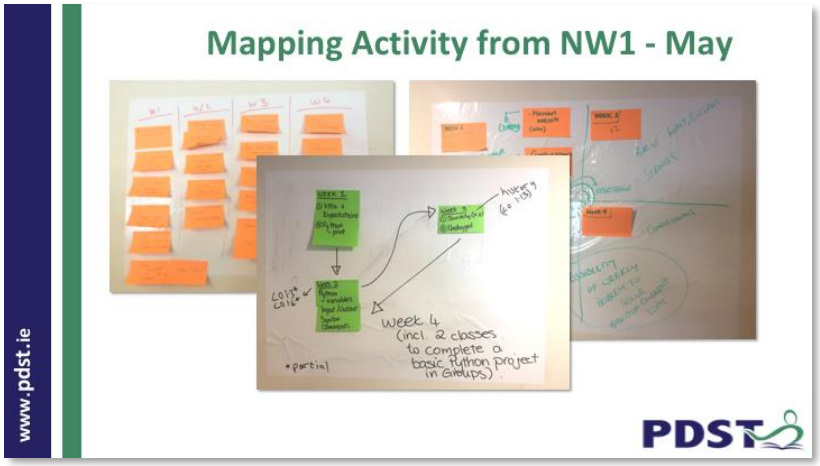

What did you do?
How did you do it?
How would you support students to engage in a similar process
Roles & Group Dynamics
What has challenged your thinking?
Making Links
Problems
Which LOs did you use?
How did you engage with the Design Process?
Presentation & Debrief
www.pdst.ie
PDST

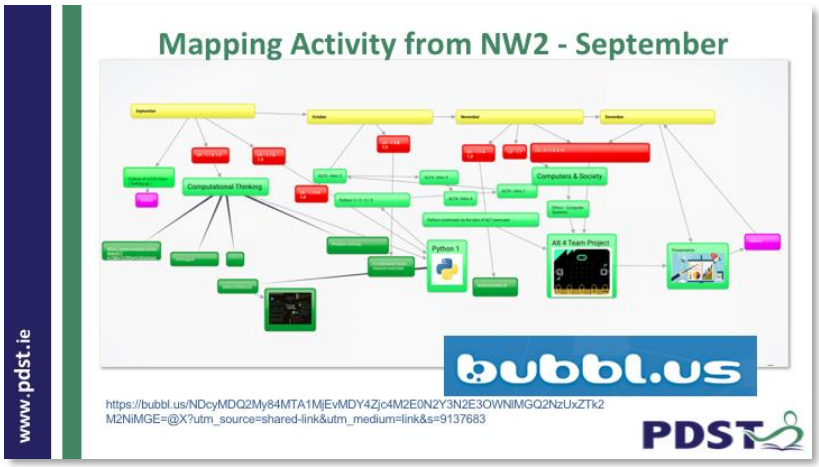
NOTES:

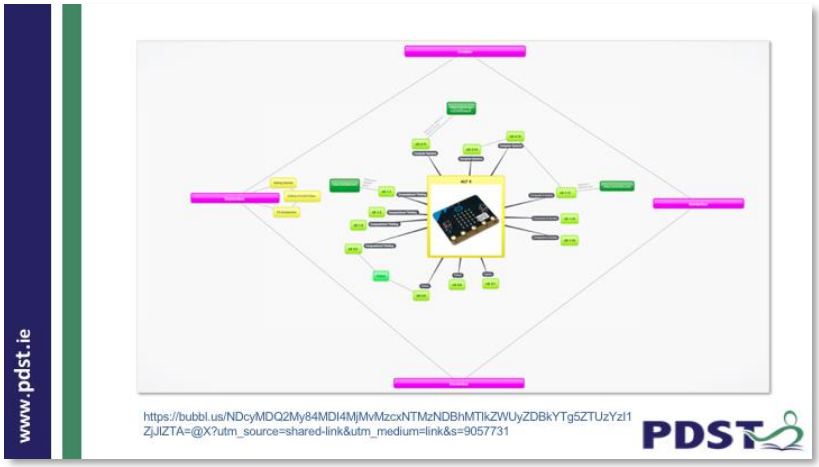
Curriculum and Assessment Planning

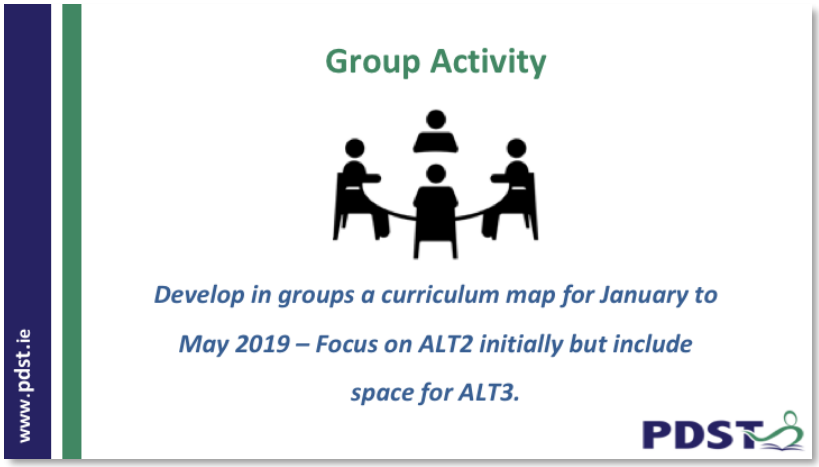




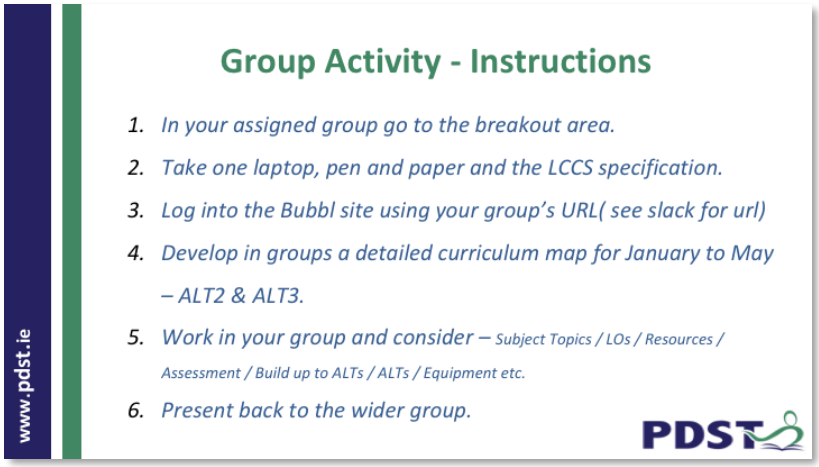


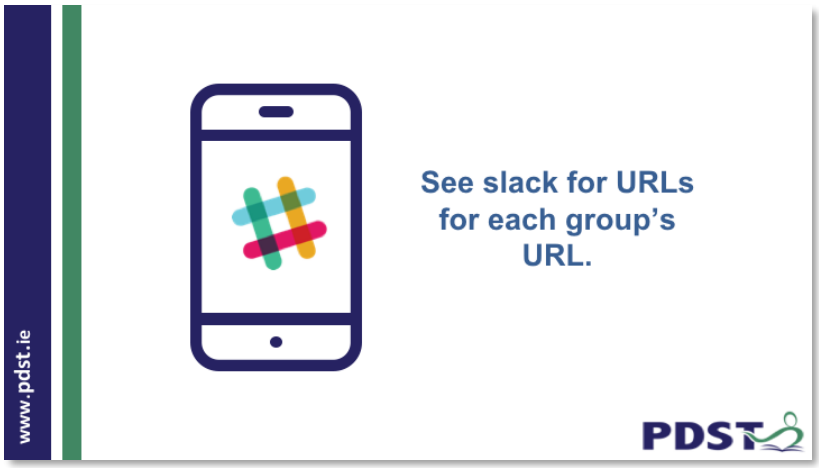

Mapping Activity from NW2 - September
www.pdst.ie
bubbl.us
https://bubbl.us/NDcyMDQ2My84MTA1MjEvMDY4Zjc4M2E0N2Y3N2E3OWNlMGQ2NzUxZTk2M2NiMGE=@X?utm_source=shared-link&utm_medium=link&s=9137683
PDST

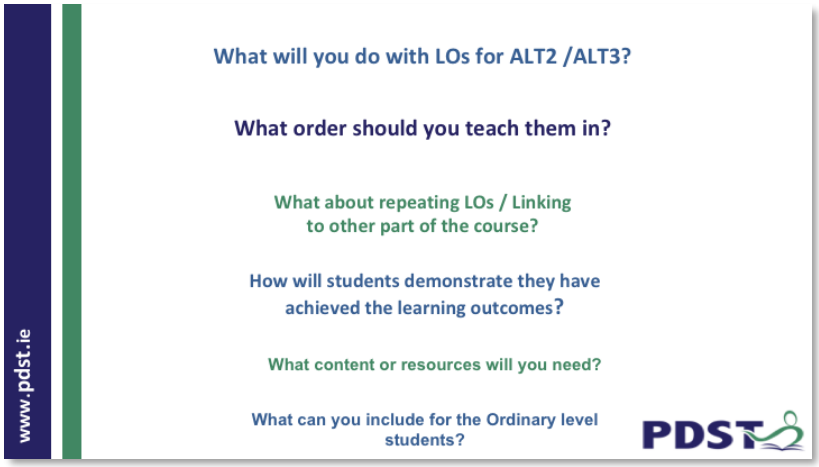

www.pdst.ie
https://bubbl.us/NDcyMDQ2My84MDI4MjMvMzcxNTMzNDBhMTlkZWUyZDBkYTg5ZTUzYzI1ZjJlZTA=@X?utm_source=shared-link&utm_medium=link&s=9057731
PDST

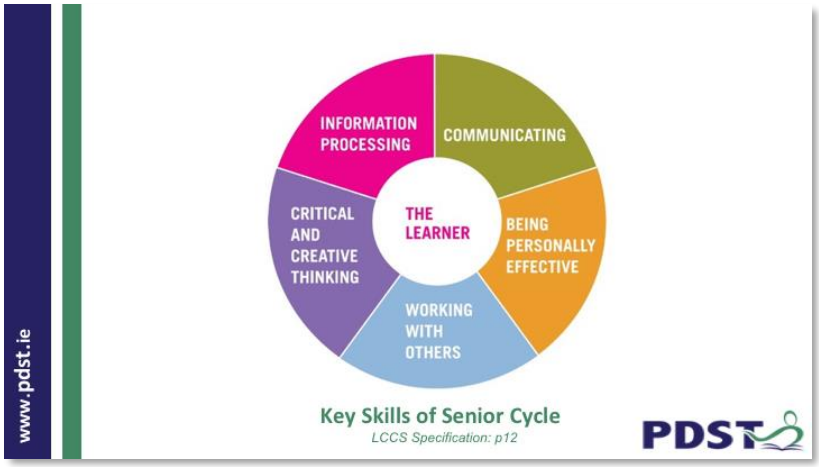

Group Activity
www.pdst.ie
Develop in groups a curriculum map for January to May 2019 – Focus on ALT2 initially but include space for ALT3.
PDST


www.pdst.ie
Key Message to remember:
Explore and teach the LOs through the lens of ALTs.
There are numerous ways to achieve this.
PDST


www.pdst.ie
Group Activity - Instructions
1. In your assigned group go to the breakout area.
2. Take one laptop, pen and paper and the LCCS specification.
3. Log into the Bubbl site using your group's URL(see slack for url)
4. Develop in groups a detailed curriculum map for January to May – ALT2 & ALT3.
5. Work in your group and consider – Subject Topics / LOs / Resources / Assessment / Build up to ALTs / ALTs / Equipment etc.
6. Present back to the wider group.
PDST


www.pdst.ie
See slack for URLs for each group's URL.
PDST


What will you do with LOs for ALT2 /ALT3?
What order should you teach them in?
What about repeating LOs / Linking to other part of the course?
How will students demonstrate they have achieved the learning outcomes?
What content or resources will you need?
What can you include for the Ordinary level students?
www.pdst.ie
PDST


INFORMATION PROCESSING
COMMUNICATING
CRITICAL AND CREATIVE THINKING
THE LEARNER
BEING PERSONALLY EFFECTIVE
WORKING WITH OTHERS
Key Skills of Senior Cycle
LCCS Specification: p12
www.pdst.ie
PDST


What has worked well for you?
What has challenged your thinking?
Why did you make these decisions?
Where are the links to other parts of the course
Which LOs did you use?
Where do you want to be in September 2019 in terms of the course?
Presentation & Debrief
www.pdst.ie
PDST

NOTES:

Following your engagement with the development process for ALT2 you might find it useful to consider the following question.

What additional supports might your own students need as they engage with this process?

ADDITIONAL NOTES:

Session 3

Computational Thinking

www.pdst.ie

Session 3
Computational Thinking

PDST

www.pdst.ie

Key Messages

- Leaving Certificate Computer Science aims to develop and foster the learner's creativity and problem solving, along with their ability to work both independently and collaboratively.
- Computing technology presents new ways to address problems and computational thinking is an approach to analyse problems, design, develop and evaluate solutions
- The computational artefacts that students design through engaging with the ALTs should be personally relevant to them or their peers, to their community or to society in general.
- The externally assessed coursework will be based on all learning outcomes, with those of strand 3 being particularly relevant.
- The Learning Outcomes are designed so that the learners can engage with them to varying degrees depending on their ability and the support they are given by their teachers.

PDST

www.pdst.ie

How did you TC?

$x^2 - y^2 = 41$

MATHS

SISTERS

4 CHALICES

PDST

Introduction

- Computational Thinking is a major element of LCCS.
  Strand 1: Practices and principles, LOs 1.1 – 1.10
  Strand 2: Core Concepts, i.e. Abstraction, Algorithms, Data, and Evaluation and Testing

- Without a clear impression of what computational thinking is and what it involves teachers may naturally struggle to deal with it fairly and consistently.

- The purpose of this session to offer teachers space to build up their own understanding of computational thinking and how best to mediate it in the classroom. In order to do this we will we will explore two viewpoints on computational thinking i.e. those of Wing[1] and Denning[2].

- There is actually relatively little literature on computational thinking and what little there is can be particularly confusing especially when it comes to seeking a definition. Part of the reason for the confusion arises from contradictory views and publications. A classic example is that of the opinions of Wing compared to those of Denning.

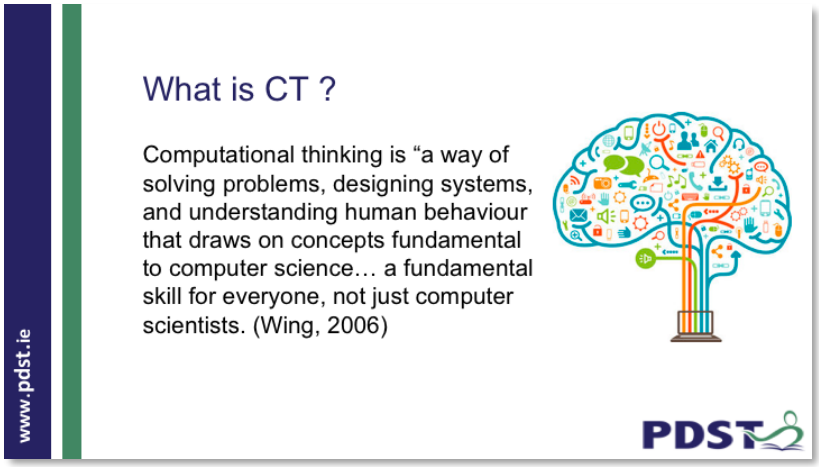
- Recall from National Workshop 2 we presented six computational concepts – algorithmic thinking, abstraction, decomposition, generalising and patterns, evaluation, and logic.

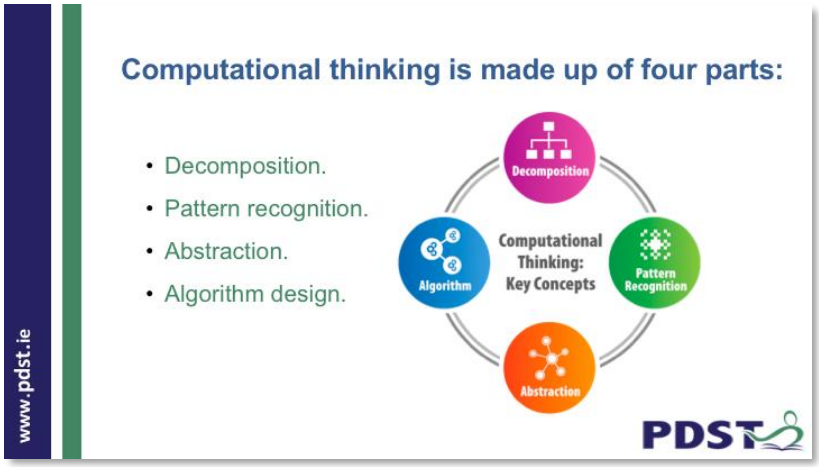
- We also presented a number of tried and trusted unplugged pedagogies namely, storytelling, kinaesthetic, magic, puzzles and role play. The respective activities we presented for these pedagogies were built around the following: The Diving Bell and Butterfly book, Binary Cards, Card Tricks, Knights Tour and finally a role play in which the instructions of a short Python program were acted out by teachers.

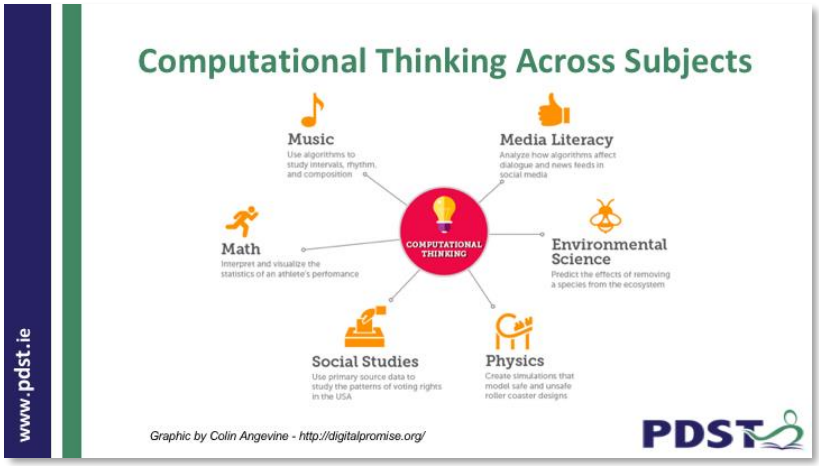
- We will start with a reflection of how four pillars of computational thinking were used by teachers and students in problems they have worked through, such as the Chalice problem and the question of who has more sisters, boys or girls.

[1] Wing, J., Communications of the ACM, Vol 49, 3 (March 2006), 33-35 Computational Thinking http://www.cs.cmu.edu/~./15110-s13/Wing06-ct.pdf
[2] Denning, P., Communications of the ACM, Vol 60, 6 (June 2017), 33-39 Remaining Trouble Spots with Computational Thinking http://denninginstitute.com/pjd/PUBS/CACMcols/cacm-trouble-ct.pdf

Warm-up: How were **Computational Pillars** (Decomposition, Pattern Recognition, Abstraction and Algorithms used in the “Brothers and Sisters” problem and the problem of the difference of 2 squares).







Some unresolved questions to consider …

What is computational thinking?

How can it be assessed?

Is it good for everyone?

Group Activity 1 Wing vs. Denning

In groups discuss the following *three* questions. Use the space provided to record your notes.

Question 1. What is computational thinking?

Wing:
Denning:

Question 2. How Do We Measure Students' Computational Abilities?

Wing:
Denning:

Question 3. Is Computational Thinking Good for Everyone?

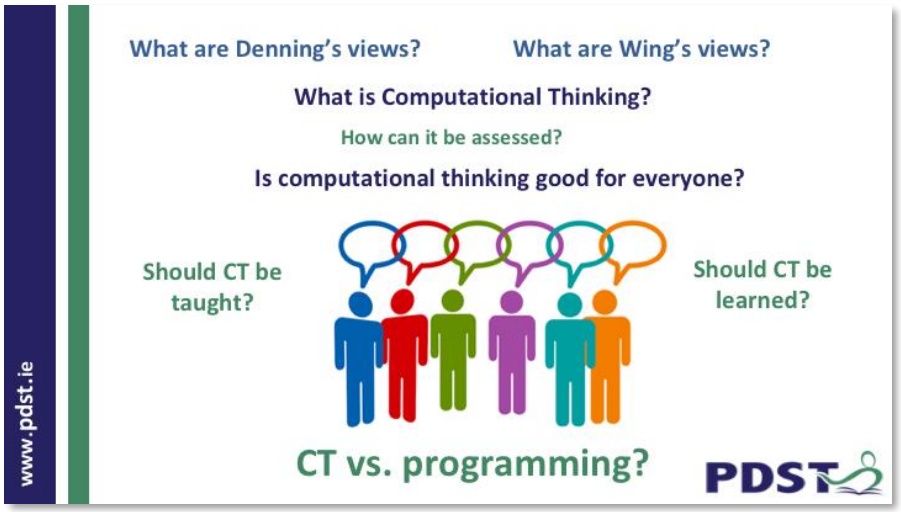
Wing:
Denning:

Activity 2 (Individual reflection on the preceding discussion).

Record your thoughts in the space provided - you may find the prompts helpful to guide your thinking

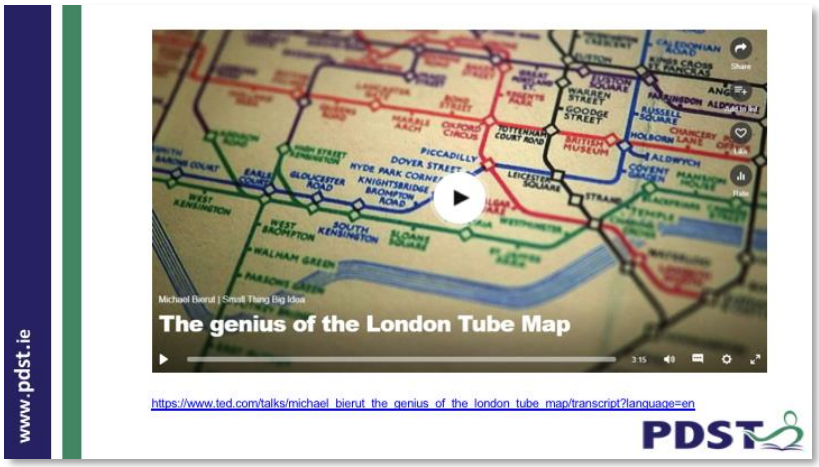
How does CT relate to programming?	*How will you assess CT?*
How does CT relate to other subjects?	*Do you think CT is best taught or learned?*

Activity 3 – Feedback/Debrief

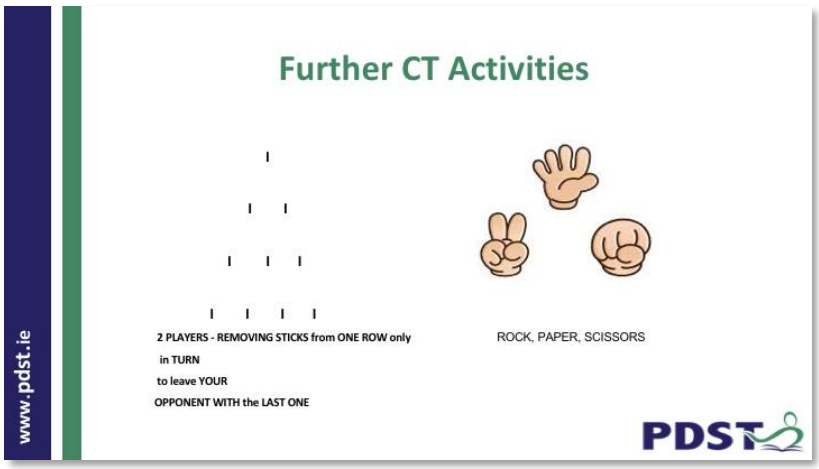


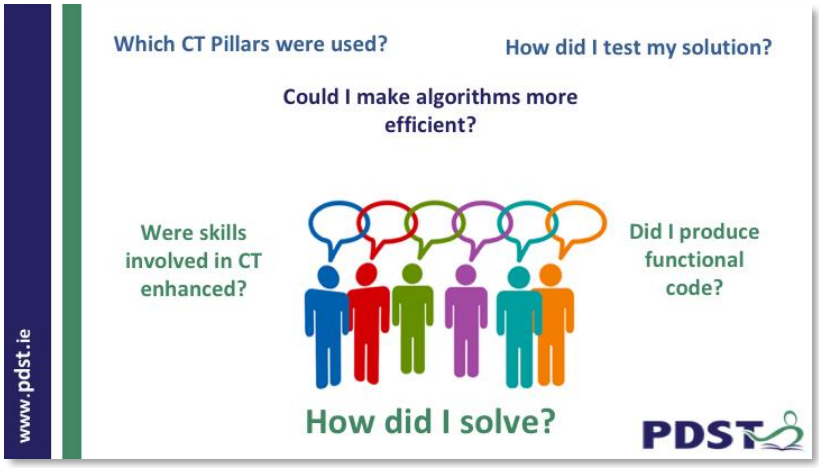
Use the space below to record the feedback from other groups. You may find the following prompts useful:

- What is challenging your thinking?
- Do you *agree* or *disagree* with the main points being put forward?



https://www.ted.com/talks/michael_bierut_the_genius_of_the_london_tube_map/transcript?language=en





Rock-Paper-Scissors Break-out Activity 1:

Write your plan for solving the Rock-Paper-Scissors problem.

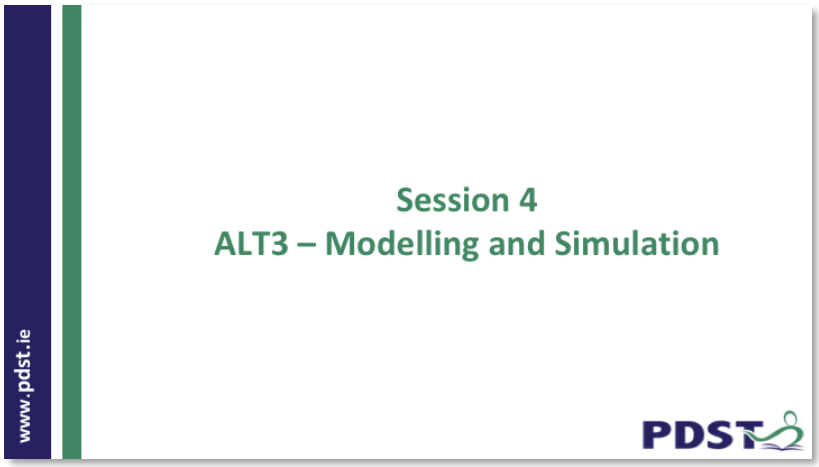
Stick Game - Break-out Activity 2:

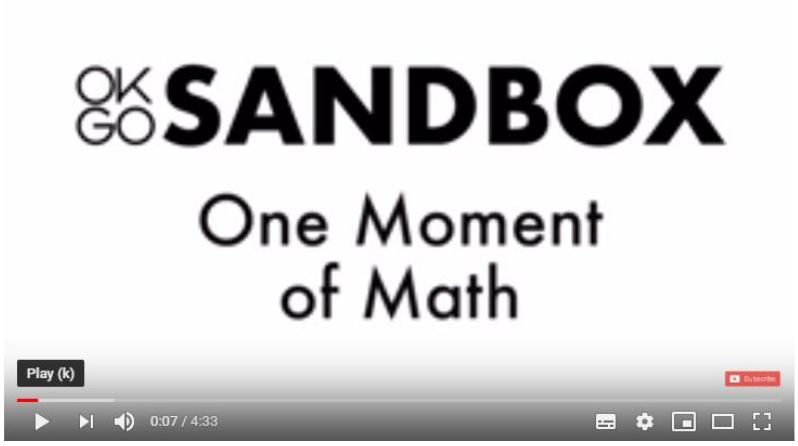
Prepare your plan for devising winning Strategies for the ‘Stick game’

ADDITIONAL NOTES:

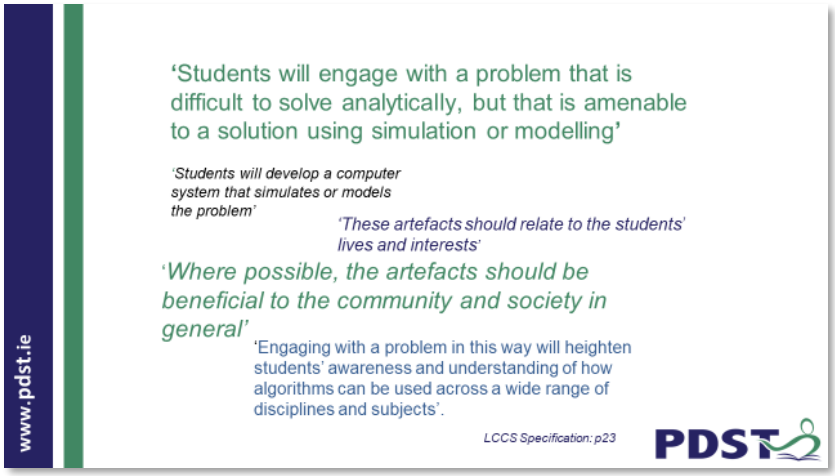
Session 4

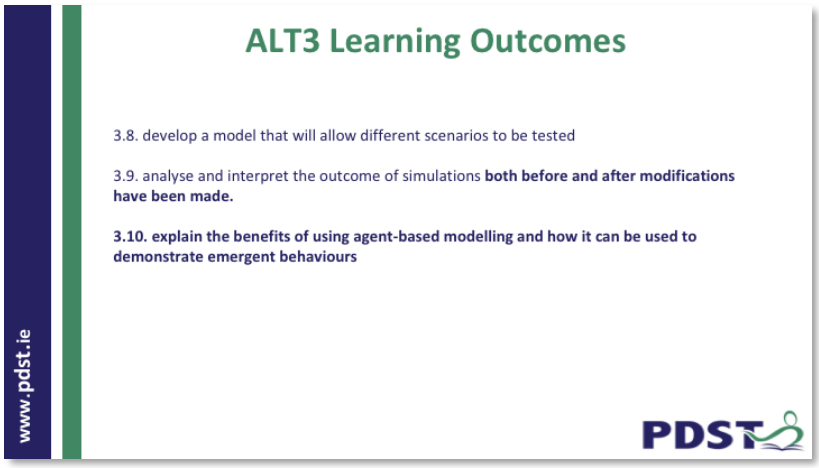
ALT3 Modelling and Simulation





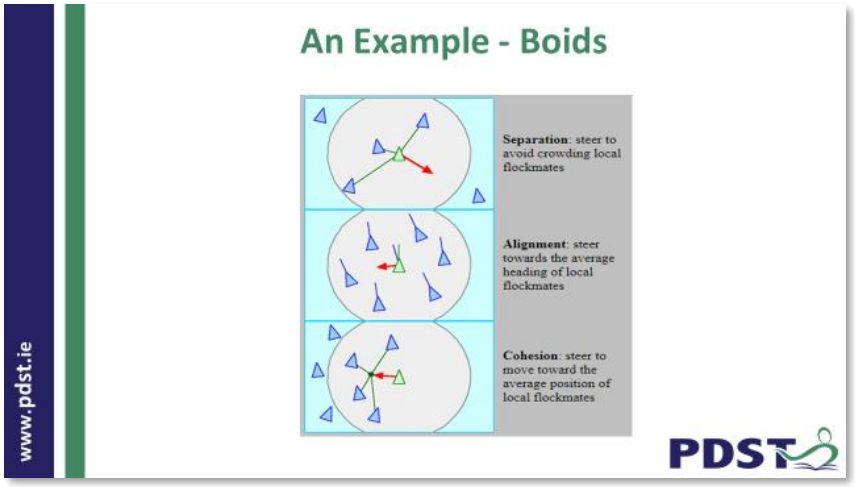
https://www.youtube.com/watch?v=DvElCxbimxc




ALT3 Learning Outcomes
3.8. develop a model that will allow different scenarios to be tested
3.9. analyse and interpret the outcome of simulations both before and after modifications have been made.
3.10. explain the benefits of using agent-based modelling and how it can be used to demonstrate emergent behaviours
www.pdst.ie
PDST

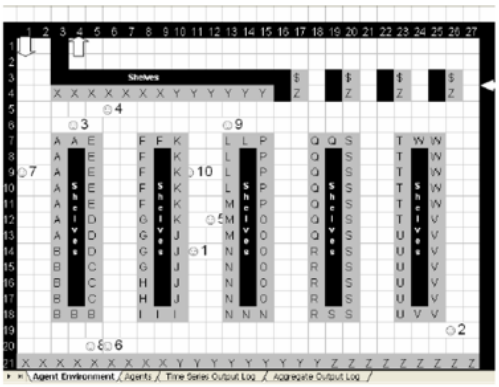
NOTES:

https://www.youtube.com/watch?v=M0iZ52kUOiQ



NOTES:

Example - Grocery Store



www.pdst.ie

PDST

Demonstration - Fireflies

https://ncase.me/fireflies/

www.pdst.ie

PDST

Individual Reflection

www.pdst.ie

PDST

Demonstration - Fireflies

"The focus on ALT3 to explore problems that are difficult to solve analytically, but that are amenable to a solution using simulation or modelling"

Source: **https://ncase.me/fireflies/**

Christmas lights gently floating in mid- air, fireflies always add a little bit of magic to the forests they live in.

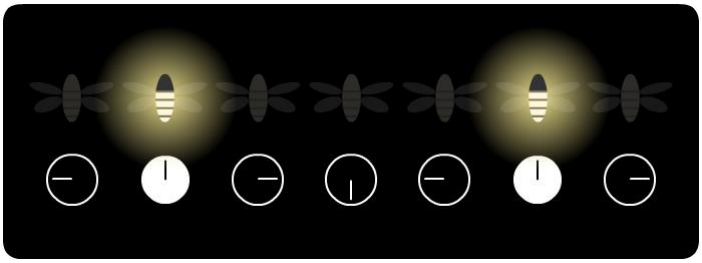
But some firefly species add even more magic. In Southeast Asia, if you go out to the riverbank deep in the night, you'll be treated to this stunning lightshow – courtesy of the Thailand firefly:

A whole mangrove forest, lighting up all at once, plunging into darkness, then lighting up all again – in near-perfect synchrony. How do thousands of fireflies coordinate with each other? Who is the conductor of this silent symphony?

This was a mystery for nearly a century, and in 1992, a team of US scientists set out to solve it. Sadly, but not surprisingly, the press mocked them –"Govt. Blows Your Tax $$ to Study Fireflies in Borneo: Not a Bright Idea!" read one tabloid. Undeterred, the scientists pressed on, cameras and calculators in hand.

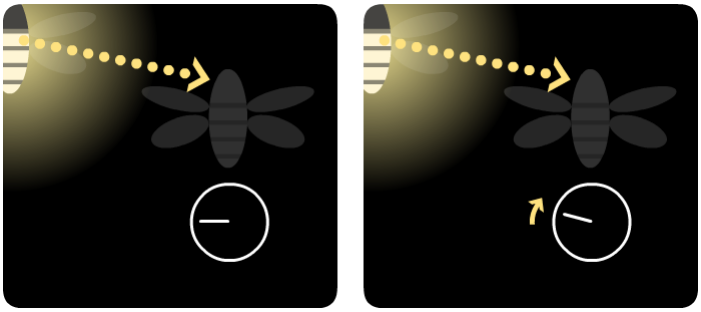
So, how do thousands of fireflies pick a leader to follow, a maestro to keep their flashing in time? The answer, the scientists found, is simple: They don't.

Each firefly has its own individual internal clock, and every time the clock "strikes twelve", it flashes.



And instead of synchronizing their clocks to a central leader, each firefly does the following:

Step 1: when you see a nearby firefly flash, nudge your clock a little bit forward.



Step 2: that's it.

Can such a small, short-ranged interaction really cause an entire forest-ful of fireflies to flash in synchronous harmony? Only one way to find out...

Nudge thy neighbor:
(click to turn on→) ON/OFF

Wait. Watch carefully. What do you see?

At first, not much. The fireflies still seem to be flashing without pattern. But after a while, you may notice small “patches” of fireflies firing together – but these individual “patches” are still out of sync.

After a bit longer, you start seeing fireflies performing a “wave”, like excited fans in the stands during a baseball game.

Finally, all but a few fireflies are flashing together – and eventually, even these laggards join in the collective dance. What's more amazing? If you disrupt these fireflies by causing some chaos...

Activity – play with the demo.

Change some of the parameters. Log your observations.

Firefly observations

Reflection

From what you now know about modelling and simulation what might a pedagogic framework for ALT3 look like?

Models for variations in temperature over a 24 hour period

Temperature variations over a 24 hour period can be modelled using some simple maths and some basic data.

Below, for example, is a graph of the variations in temperature over a 24 hour period monitored by a weather station in Iowa in the USA.
(https://www.globe.gov/explore-science/scientists-blog/archived-posts/sciblog/index.html_p=95.html)

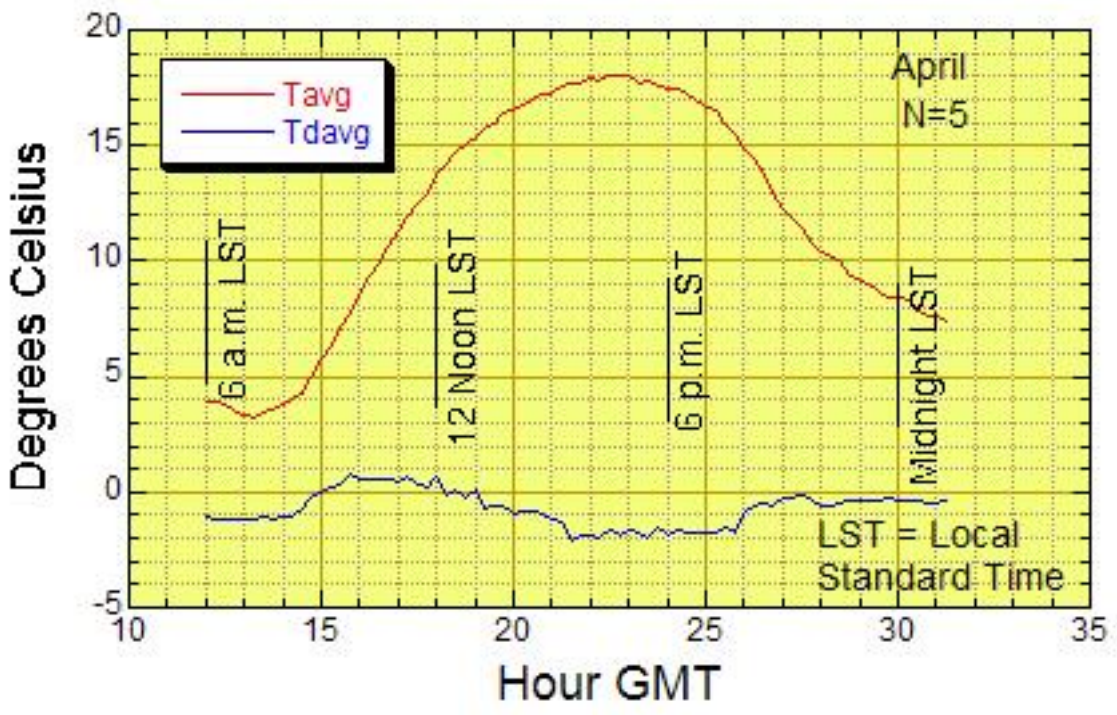


Figure 1. Air temperature Tavg (red) and dew point Tdavg (blue) at a site in Black Hawk County, Iowa. Height: 1.5 m above the surface. The data are averages of five days with clear skies in April, 2004

The shape of the data is approximately in the form of a sinusoidal wave or a sine wave. The maximum and minimum values over the 24 hour period will depend upon location, climate, cloud formations and a host of variables.

As part of the CSinP Temperature ALTs (Computer Science in Practice Applied Learning Tasks around temperature measurements) on measuring temperature and displaying it live on a website, this section provides a model for variations in temperature, which can be adjusted or modified for any particular location.

The basic model used will be very similar to the model presented to students in the LC HL 2015 exam in Maths. This model, as shown below, was to model the variation in the length of a day over a one-year time period.

Question 9 **(50 marks)**

The approximate length of the day in Galway, measured in hours from sunrise to sunset, may be calculated using the function

$$f(t) = 12{\cdot}25 + 4{\cdot}75\sin\left(\frac{2\pi}{365}t\right),$$

where t is the number of days after March 21st and $\left(\frac{2\pi}{365}t\right)$ is expressed in radians.

(a) Find the length of the day in Galway on June 5th (76 days after March 21st). Give your answer in hours and minutes, correct to the nearest minute.

The model we will use will be similar in form, and modelled using Python. Some key parameters in the model will need to be adjusted in order to model the local data more closely. In some cases, the model itself could be changed to reflect the data points measured by the CSinP Temperature ALTs.

The key parameters are :

BT Baseline Temperature for the day

ID Maximum Increase – Decrease in temperature over the course of the day

NDP Number of Data Points used to plot the variations in temperature measured in a day.

The modelled temperature is modelled for any hour in the day (MTH). The hour of the day is represented by *h* and the calculation for sine is in radians. The model below assumes hour 0 begins at the baseline temperature (BT).

$$\mathrm{MTH} = \mathrm{BT} + \mathrm{ID} * sin(\frac{2\pi * h}{NDP})$$

https://repl.it/@pbehan/plotting-data-from-firbase

A graph from the model is shown below in Figure 2. The model simulates the graph from the Iowa station, and uses data drawn from that graph to guide the key parameters.

(In addition, it is also possible to use measured data to generate the model. There are various websites which can perform this task.)

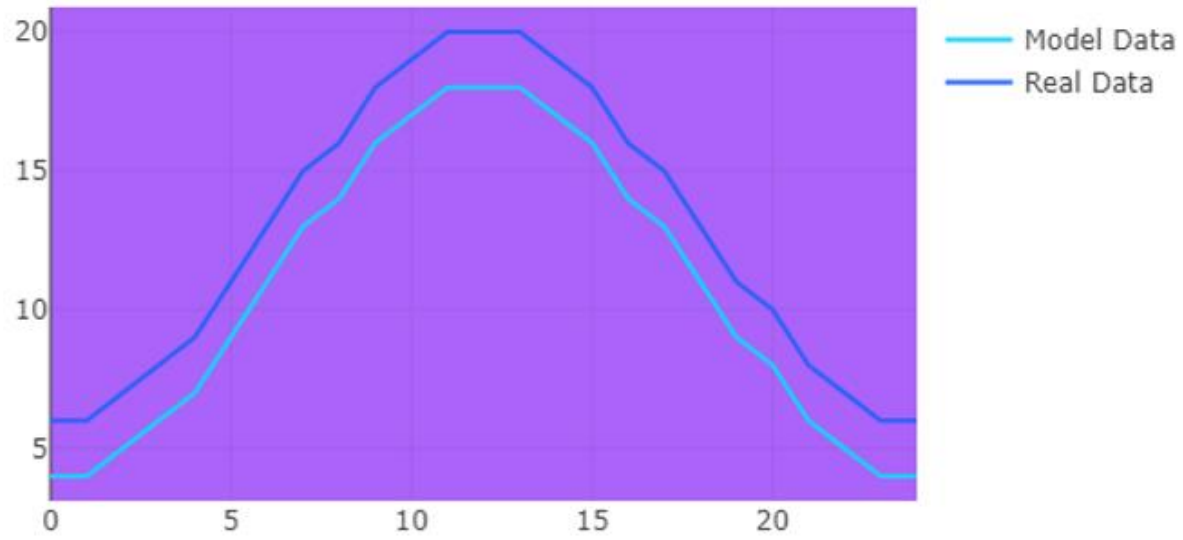


Figure 2. A plot from the html/JS program showing a model of the Iowa data above. This data was modelled with a lag to simulate a start time of 6am approximately (i.e. at hour zero).

The BT was set at 11 degrees, the ID at 7 degrees, and NDP to 25.

NOTES:

Reflection

Having completed the tutorial use the space provided below to log your thoughts.

What has this example taught me in relation to ALT3?

What questions do I have in relation to the example?

Look at some past LC papers and see if you can identify questions that would be suitable to base an ALT3 project on.

(Browse to: https://www.examinations.ie/exammaterialarchive/)

Session 5

ALT3 Project Design and Development

www.pdst.ie

Session 5

ALT3 – Project Design and Development

PDST

www.pdst.ie

'Students will engage with a problem that is difficult to solve analytically, but that is amenable to a solution using simulation or modelling'

'Students will develop a computer system that simulates or models the problem'

'These artefacts should relate to the students' lives and interests'

'Where possible, the artefacts should be beneficial to the community and society in general'

'Engaging with a problem in this way will heighten students' awareness and understanding of how algorithms can be used across a wide range of disciplines and subjects'.

LCCS Specification: p23

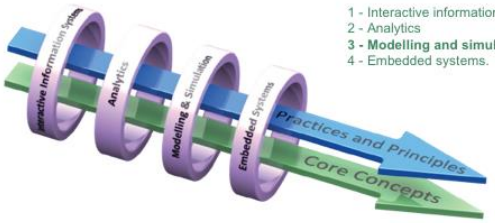
PDST

www.pdst.ie

LCCS Interwoven

The four applied learning tasks explore the four following contexts:

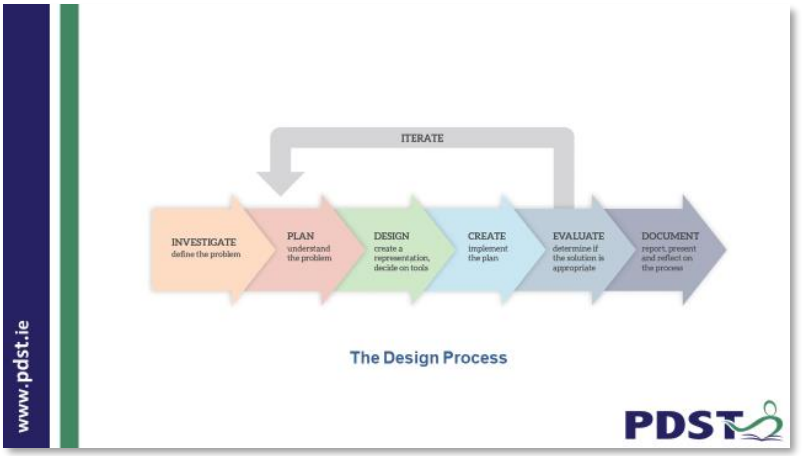
1 - Interactive information systems
2 - Analytics
3 - Modelling and simulation
4 - Embedded systems.



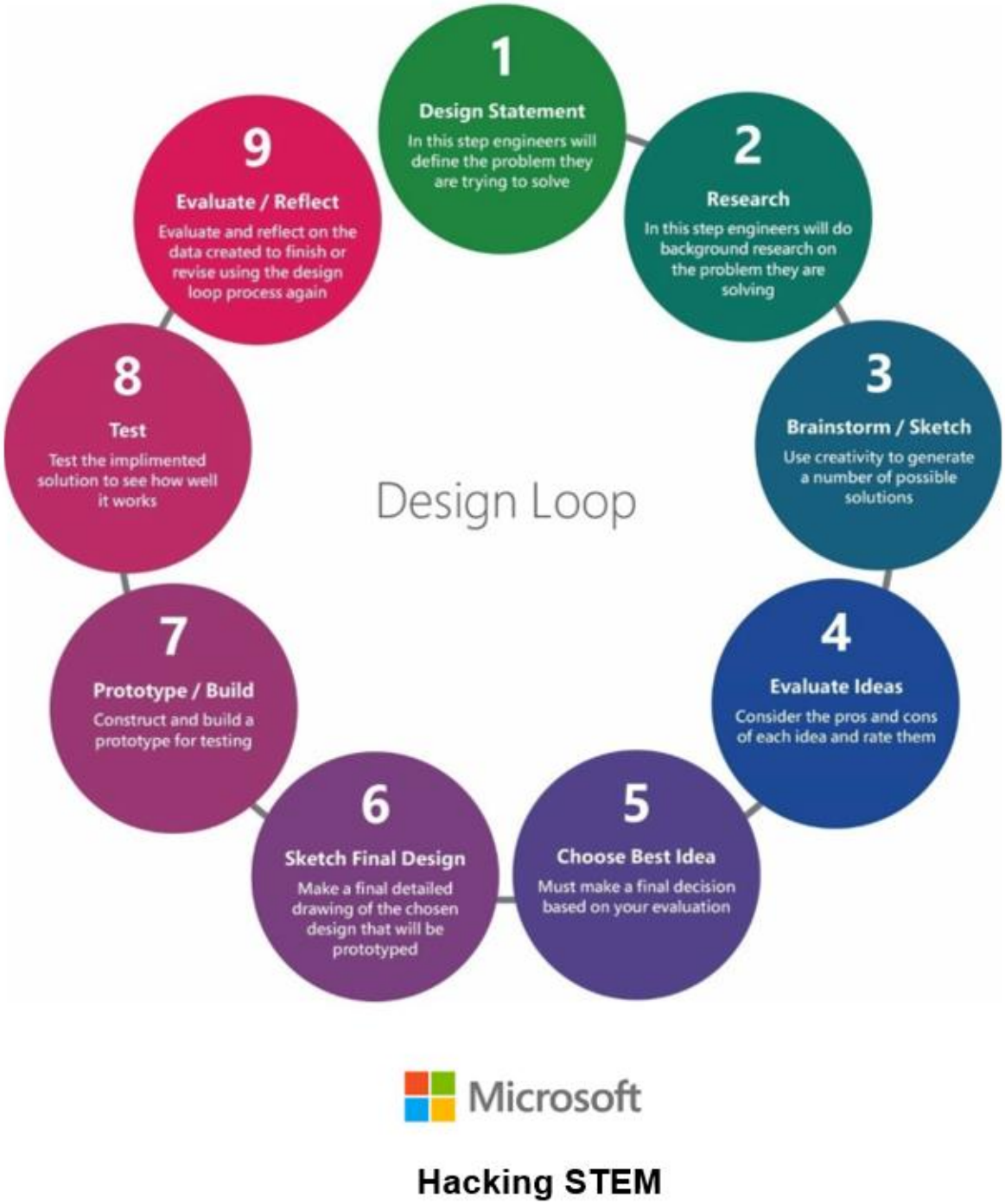
PDST

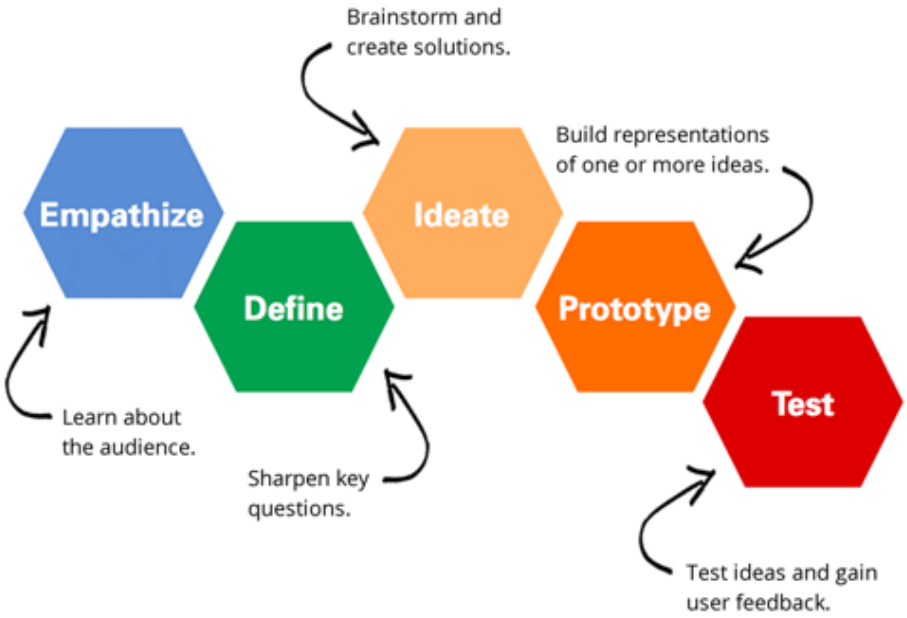

www.pdst.ie
Key Message to remember:
Explore and teach the LOs through the lens of ALTs.
There are numerous ways to achieve this.
PDST


www.pdst.ie
Main Group Activity
Completing a Design Brief for ALT3
PDST


www.pdst.ie
ITERATE
INVESTIGATE
define the problem
PLAN
understand the problem
DESIGN
create a representation, decide on tools
CREATE
implement the plan
EVALUATE
determine if the solution is appropriate
DOCUMENT
report, present and reflect on the process
The Design Process
PDST

Alternative Design Processes




Brainstorm and create solutions.
Empathize
Define
Ideate
Prototype
Test
Build representations of one or more ideas.
Learn about the audience.
Sharpen key questions.
Test ideas and gain user feedback.
d.
HASSO PLATTNER
Institute of Design at Stanford


UNDERSTAND
EXPLORE
MATERIALIZE
IMPLEMENT
Put the vision into effect.
EMPATHIZE
Conduct research to develop an understanding of your users.
TEST
Return to your users for feedback.
DESIGN THINKING 101
NNGROUP.COM
DEFINE
Combine all your research and observe where your users' problems exist.
PROTOTYPE
Build real, tactile representations for a range of your ideas.
IDEATE
Generate a range of crazy, creative ideas.

NNGROUP.COM

Design Brief for ALT3

'Finding the best Brexit Routes'

See manual for the full design brief.

www.pdst.ie

PDST

'Finding the best Brexit Routes'

Driver of *Fred's Frozen Foods* has to Service the following towns from Dublin:

Enniskillen, Donegal, Letterkenny, Derry, Belfast and Dundalk.

But what's the most cost effective route?

...To Letterkenny?

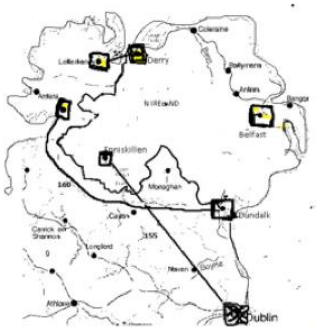
...To cover all the depots and arrive back in Dublin.

Model and Develop algorithms (see manual for full brief)

www.pdst.ie

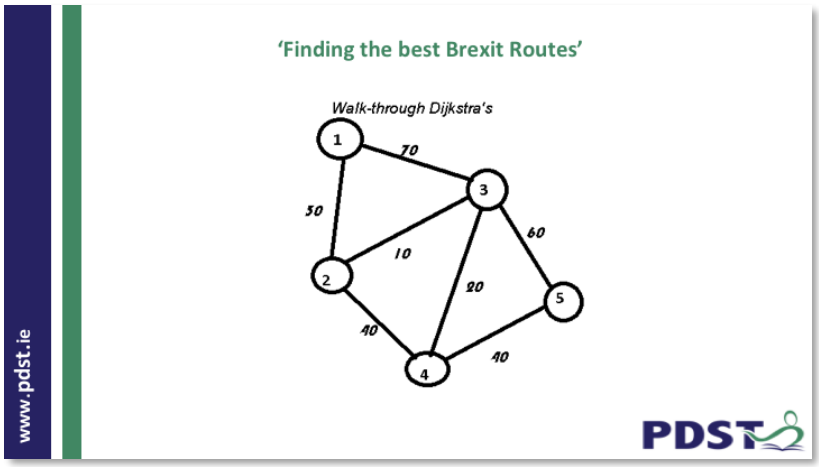
PDST

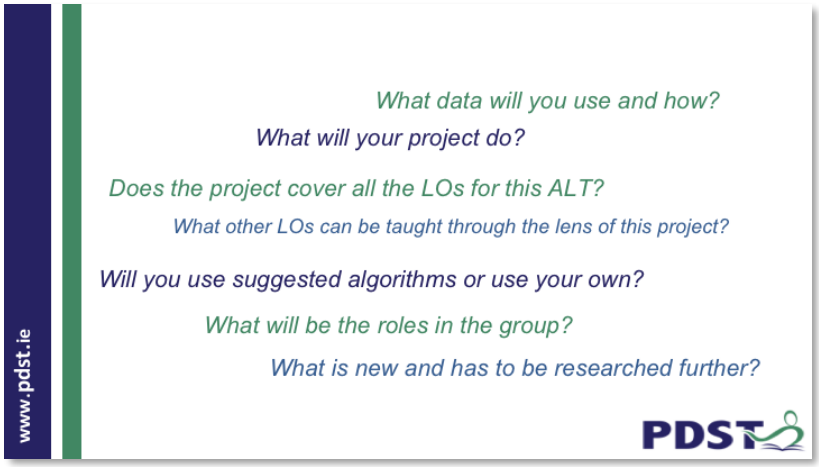
'Finding the best Brexit Routes'



www.pdst.ie

PDST


'Finding the best Brexit Routes'
Walk-through Dijkstra's
1
2
3
4
5
70
50
60
10
90
40
40
www.pdst.ie
PDST


What data will you use and how?
What will your project do?
Does the project cover all the LOs for this ALT?
What other LOs can be taught through the lens of this project?
Will you use suggested algorithms or use your own?
What will be the roles in the group?
What is new and has to be researched further?
www.pdst.ie
PDST

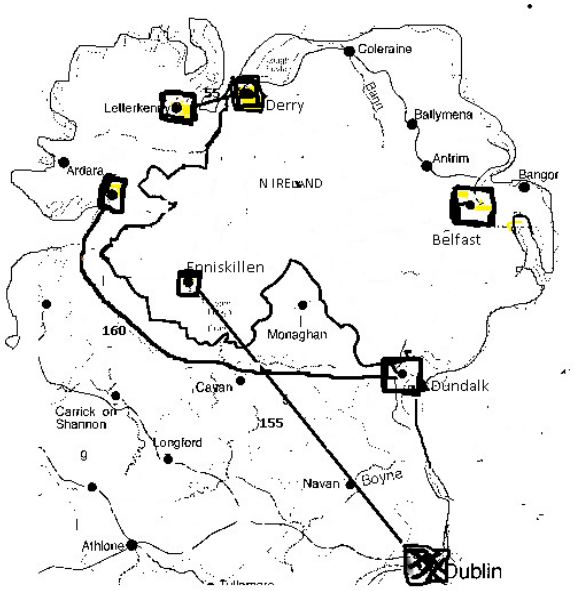

Presentation & Debrief
www.pdst.ie
PDST

Sample Brief ALT3: BREXIT Routes:

Introduction:

ALT3 involves a problem that is difficult to solve analytically but is amenable to a solution using modelling or simulation.

Consider the problem of a frozen foods company who has to deliver from Dublin to Enniskillen (E) Derry (Y) Belfast (B) Donegal (G) Letterkenny (L) Dundalk (K)



The problem can be approached by modelling the current situation and simulating various routes with a view to answering for example, what is the cheapest route to Letterkenny, or how can a driver deliver to all locations in a cost-effective way. The weightings given are not the distance but rather the cost of travelling between towns. They are a function of drivers' costs and costs associated with fuel, maintenance, tolls, customs charges, road surface etc. Dijkstra's Algorithm (shortest path) or Travelling Salesman algorithm may be used or adapted for a solution.

The cost of travelling between different routes are using the town codes above:

D-E €155 (as shown) G-L €55 L-E 95 Y-E €75 E-B €128

Y-K €125 K-B €130 D-K €130 D-E €155 G-K €160 B-Y €105

Aim: To Model the routes taken by goods transport drivers, to run simulations and to modify the model according to outside changes to benefit the .goods transport companies.
Date: 10 January 2019
Completion: 22 February 2019

Overview: Goods Transport Companies need to minimise costs when delivering to different locations.

Audience: Peers, Transport Companies.

Tasks / Marking Scheme:
Model the current situation to allow decision-making for companies transporting goods in deciding routes.(10%)

Modify / Adapt this model to allow for changes accompanying Brexit (or different types of Brexit)

1. Find the most cost-effective route to go from Dublin to Letterkenny.(15%)
2. Find the most cost-effective way to deliver goods to all the locations shown, when Enniskillen must be the first stop, and Dundalk the last. (15%)
3. Factor in to Task 2., a Brexit scenario which adds €120 to each border crossing – due to tolls, time and the need for extra documentation. (15%)
4. Document your solution and reflect on your work (15%) .
5. Present your solution as a group. (10%)
6. Ex-Impressio (20%)

Milestones / Learning Intentions:

- Familiarise yourself with some algorithms in this area – Dijkstra's / Travelling Salesman.
- Model the current situation.
- Simulate various situations
- Develop Algorithms and write code in Python for these.

Groupwork Roles:
Groups of 4/5 'randomly' selected. Roles to be clear and set at initial meeting. Team members work together to common goals.

Resources: Spreadsheet programs, Python, *Plotly*, Calculator, Pen and paper, etc..

Format of Presentation: Group presentation at completion, with peer questioning / discussion.

Success Criteria: Clear documentation, Effective Modelling, Use of efficient algorithms, functional and clear code.

Feedback: Self and peer assessment and feedback. Teacher feedback. All feedback to take into account the complexity of the problem.

NOTES:

Other ALT3 Ideas:

1. Extension to 'Brexit Routes':

In one of the scenarios in 'Brexit Routes', the most efficient route was to drive from Donegal to Dundalk avoiding passing into the North a second time. There are more that 5,000 trucks on the road in Ireland at any time, and, over time more and more may take this route. This could add to the 'weighting' on this road, due to congestion and delays. Drivers may then revert to go through the North. This could result in a dynamic similar to the 'predator-prey' model, and students could test various scenarios.

2. Brief: How to Take a Penalty

Introduction:
ALT3 involves a problem that is difficult to solve analytically but is amenable to a solution using modelling or simulation.

The Ireland women's hockey team got through to the World Cup final in 2018 via a penalty shoot-out. Similarly, important matches in men's soccer international tournaments have often been decided by penalties.

In the case of men's soccer, penalty-kicking techniques are practiced as potentially the whole team may have to be "on the spot" in a shoot-out. At this level, it amounts to a psychological battle between the goalkeeper and the penalty taker – can the goalkeeper guess which side the penalty will be taken.

Rob Eastaway, celebrity Mathematician, advocated that the taker should choose to go left or right randomly. However, it has been shown that it is very difficult for people to make choices randomly, and so the goalkeeper would have an advantage. If you ask people to say "left" or "right" at random 40 times, patterns generally emerge, and their list is usually easily distinguished from a true random list.

This ALT will focus on modelling a system to take a penalty, initially reducing the problem to two choices, left or right, by using a random generation program, and then to extend the problem to include more choices, and to "weight" these choices.

Aim: To Model the decision-making in Taking a Penalty in Soccer and to Modify and Extend this Model.

Date: 10 January 2019

Completion: 22 February 2019

Overview: Soccer players will often try to be somewhat "random" in choosing which way to take a penalty. In international events, it often appears that players are uneasy or confused about making decisions as often some of the best penalty-takers fluff their attempt. The purpose of this ALT is to model a robust system for taking penalties involving generating random choices and modifying this model.

Audience: Peers, soccer players and enthusiasts.

Tasks: Model the decision-making in taking a penalty by using a random generator to choose which way to take a penalty.
Modify / Adapt this model to suit the choices and preferences of the players ("agents").
Introduction of greater complexity, and therefore better outcomes can also be introduced by introducing a weighted system of choices.

Milestones / Learning Intentions:

1. Assess and model current situation.
2. Demonstrate, through experiment, people's lack of ability to choose randomly.
3. Develop and code a random generator.
4. Adapt and extend the model by increasing the agents' choices and preferences.

Groupwork Roles:
Groups of 4/5 'randomly' selected. Roles to be clear and set at initial meeting. Team members work together to common goals.

Resources: Spreadsheet programs, Python, *Plotly*, Calculator, Pen and paper, etc..

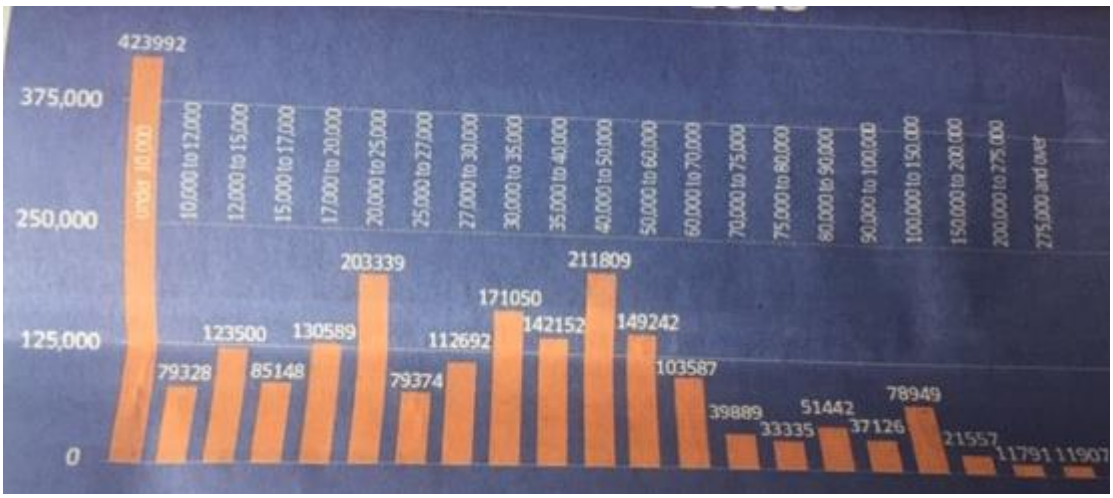
Format of Presentation:
Group presentation at completion, with peer questioning / discussion.

Feedback: Self and peer assessment and feedback. Teacher feedback. All feedback to take into account the complexity of the problem.

3. Brief - A 'Taxing' Problem which can't be Evaded:

Aim: To Provide New Models and Simulations for Solutions concerning the Irish PAYE system.
Date: 10 January 2019
Completion: 22 February 2019

Overview: At the moment, the marginal rate of income tax is charged at 40%. This is regardless of salary and possibly comes from a time when computers were unsophisticated, or even not used. Perhaps a better solution could be found, which would be of benefit to both the government and taxpayers. The spread of incomes in Ireland (Fig 2) is given below.



(Source : Sunday Business Post, April 2018)

Audience: Peers, taxpayers.

Tasks: Model the current situation of charging tax at the marginal rate, using the Computational Thinking pillars of decomposition, pattern recognition, abstraction and algorithm design. Record your work.

Develop new model(s) and use / adapt / develop these models in regard to some of the scenarios below. Analyse and draw conclusions on models which are optimal for government and some of the dummy "agents" involved.
Adapt model for use in one or more of the sample scenarios below, or work with different scenarios.

Choose your level of complexity from a simple problem, which has a high degree of certainty to one which is more complex and less certain.

The solution to complex problems usually involves at least one of:

- Non-trivial algorithms
- Use of sophisticated features of programming language
- Time-based simulation
- Complexity of non-programming part of the problem

Sample Scenarios

1. *The government may want to increase its tax take by say 4% to balance its books.*
2. *The government may want to increase its tax take by 4%, while not affecting those on incomes between €35,000 and €55,000, who may regard as the "squeezed middle".*
3. *Ensure that more money is put in to the pockets of people below circa €55,000, as this tends to give the economy a boost, as it helps the service and retail sectors.*
4. *The government may not want to help those in the range €35,000 to €55,000.*
5. *Model a scenario where there is an average increase in income of say 3%, or 5%*
6. *Model what you would consider to be a "fair" tax system.*

Milestones:

5. Assess and model current situation, finding out total tax take to the government overall and from various agents.
6. Devise new model.
7. Run various scenarios on new model and adapt and develop new models.

Groupwork Roles:

Groups of 4/5 randomly selected. Roles to be clear and set at initial meeting. Team members work together to common goals.

Resources: Spreadsheet programs, Python, *Plotly*, Calculator, Pen and paper, posters etc..

Format of Presentation:

Group presentation at completion, with peer questioning / discussion.

Feedback: Self and peer assessment and feedback. Teacher feedback. All feedback to take into account the complexity of the problem.

NOTES:

Appendices

Further Resources

Links to Tutorials, Videos and References

Dear Data

https://vimeo.com/166214790

The purpose of this video is simply to introduce the importance (potential/relevance) of data

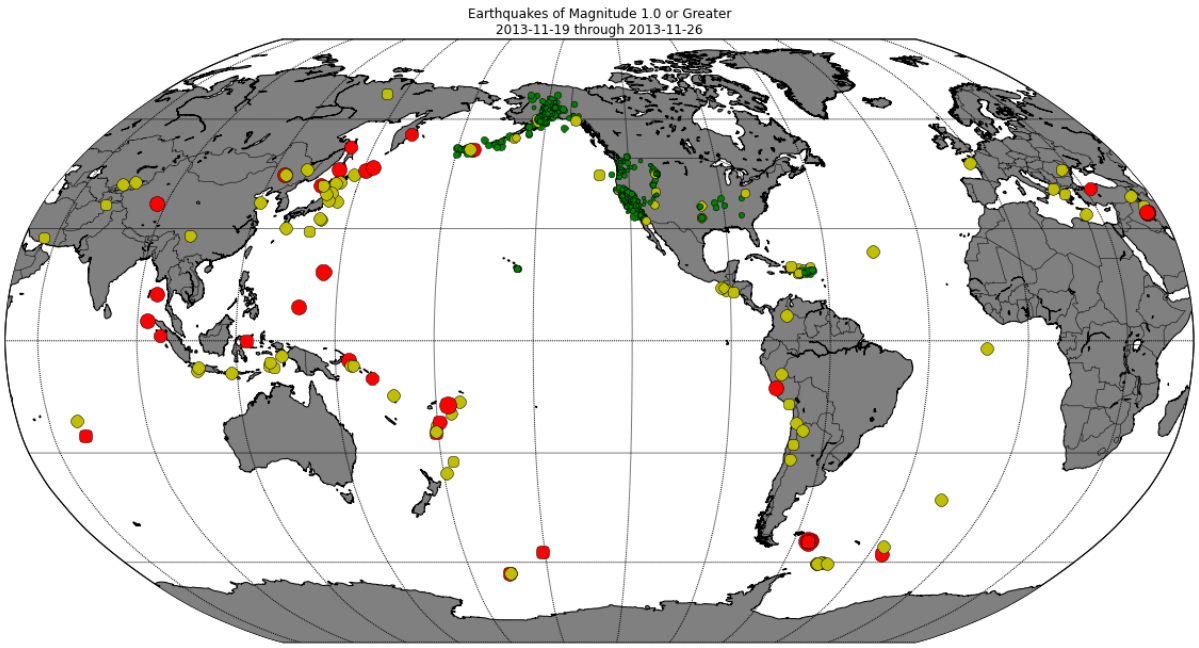
The video talks about ….

- Finding out about one another through data
- Indicator of personality and life(style) e.g. data on doors gives an indicator of the pace of a person's life
- Data is the starting point for questioning the world around us (as opposed to the end point i.e. as the answer to all our questions)

Visualization: Mapping Global Earthquake Activity¶

This project introduces the Basemap library, which can be used to create maps and plot geographical datasets.

The main goal of this project is to help you get comfortable making maps of geographical data. If you follow along with the tutorial, you'll end up making this map:



It takes fewer than 50 lines of code to generate this map from a raw dataset! This project will conclude with a list of datasets to explore, to help you find a project of your own to try.

Full tutorial can be accessed at http://introtopython.org/visualization_earthquakes.html

Analyzing Obesity in England With Python

I saw a sign at the gym yesterday that said, "Children are getting fatter every decade". Below that sign stood a graph that basically showed that in five years the average English child will weigh as much as a tractor.

I found this claim slightly unbelievable, so I decided to investigate…

Full tutorial can be accessed at https://realpython.com/analyzing-obesity-in-england-with-python/

Python vs Excel

Should I use Python or Excel?

This question is often asked by people just starting out in data analysis. While Python may be popular with the programming community, Excel is much more prevalent in the wider world. Most officer managers, sales people, marketers, etc. use Excel - and there's nothing wrong with that. It's a great tool if you know how to use it well, and it has turned many non-technical people into expert analysts.

The answer to whether you should use Python or Excel is not an easy one to answer. But in the end, there is no either/or: Instead, you can use them together.

Excel is great for viewing data, performing basic analysis, and drawing simple graphs, but it really isn't suitable for cleaning up data (unless you are willing to dive into VBA). If you have a 500MB Excel file with missing data, dates in different formats, no headers, it will take you forever to clean it by hand. The same can be said if your data is spread across a dozen CSV files, which is fairly common.

Doing all that cleanup is trivial with Python and Pandas, a Python library for data analysis. Built on top of Numpy, Pandas makes high-level tasks easy, and you can write your results back to an Excel file, so you can continue to share the results of your analysis with non-programmers.

So while Excel isn't going away, Python is a great tool if you want clean data and perform higher-level data analysis.

3 Kinds of Data that Do More Harm than Good

Now is a great time to think about the data you have, and how you're using it.

Data is today's gold, representing huge potential value for businesses. By analyzing lots of data, you can discover patterns that can help you introduce efficiency, make smarter decisions, innovate products, and reduce operational liabilities like cost and risk. By using hidden data, you can transform your business or disrupt an industry by using available data to solve a painful problem.

Read the complete article at: https://www.iotforall.com/3-big-data-challenges/ (CloudFactory, Dec 2017)

The Leading Causes of Dirty Data

The world is producing data at exponential rates. By 2025, the global datasphere will include 10 times the amount of data generated last year alone, according to IDC's Data Age 2025. All of this data generation compounds an already common problem: "dirty data."

Read the complete article at: https://www.iotforall.com/leading-causes-dirty-data/ (CloudFactory, Dec 2017)

How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did

https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/

How Companies Learn Your Secrets

By CHARLES DUHIGGFEB. 16, 2012
https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=1&_r=1&hp

References for Computational Thinking

Bower, M., Wood, L. N., Lai, J. W., Howe, C., Lister, R., Mason, R., Highfield, K., & Veal, J. (2017). *Improving the Computational Thinking Pedagogical Capabilities of School Teachers.* Australian Journal of Teacher Education, 42(3). http://dx.doi.org/10.14221/ajte.2017v42n3.4

Denning, P., Communications of the ACM, Vol 60, 6 (June 2017), 33-39 *Remaining Trouble Spots with Computational Thinking* http://denninginstitute.com/pjd/PUBS/CACMcols/cacm-trouble-ct.pdf

Grover, Shuchi (Nov 18)
A Tale of Two CTs (and a Revised Timeline for Computational Thinking)
https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext#

Grover, Shuchi Grover (Oct 18)
What CS in schools can learn from Science and Mathematics
http://www.shuchigrover.com/what-cs-can-learn-from-science-and-math/

Grover, Shuchi (May 13)
Learning to Code Isn't Enough
https://www.edsurge.com/news/2013-05-28-opinion-learning-to-code-isn-t-enough

Grover & Pea
Computational Thinking in K−12 : A Review of the State of the Field
https://pdfs.semanticscholar.org/919e/03f61a5261fc8cc7753c6fd81e9073c24e11.pdf

Harel, Itid (May 16, Blog)
American schools are teaching our kids how to code all wrong
https://qz.com/691614/american-schools-are-teaching-our-kids-how-to-code-all-wrong/

Lee et. al., ACM Inroads 2011 March Vol. 2 No. 1, *Computational Thinking for Youth in Practice*

Lockwood, J. and Mooney, A., Dept of Computer Science, Maynooth University,
Computational Thinking in Education: Where does it fit? A systematic literary review

Martin F, Chair, CSTA
Rethinking Computational Thinking - presentation
https://docs.google.com/presentation/d/16FzQD0oUcbkNdLpbEC2GXdtWga6Xr-hq62ySiXC10XQ/present?slide=id.g3eb4e2e4be_0_119

Martin F, Chair,, Chair, CSTA
Rethinking Computational Thinking
http://advocate.csteachers.org/2018/02/17/rethinking-computational-thinking/

Selby Cynthia Collins, How Can the Teaching of Programming Be Used to Enhance Computational Thinking Skills
(Thesis for the degree of Doctor of Philosophy), April 2014

Tedre, Matti; Denning, Peter J. (2016) The Long Quest for Computational Thinking. Proceedings of the 16th Koli Calling Conference on Computing Education Research , November 24-27, 2016, Koli, Finland: pp. 120-129.

Wing, J., Communications of the ACM, Vol 49, 3 (March 2006), 33-35 Computational Thinking
http://www.cs.cmu.edu/~./15110-s13/Wing06-ct.pdf

Twitter

https://twitter.com/guzdial/status/1041113715024822273
https://twitter.com/katyaskit/status/1010958287330611200

Books

The Power of Computational Thinking (Paul Curzon and Peter McOwan)

Computational Thinking: A Beginner's Guide to Problem-Solving and Programming (Karl Beecher)

Computer Science Education: Perspectives on Teaching and Learning in School (Edited by Sue Sentence, Erik Barendsen and Carsten Schulte)

Modelling and Simulation

WATCH!

https://www.youtube.com/watch?v=CPHjsWSzOY0&feature=youtu.be

Bouncing Ball Simulator

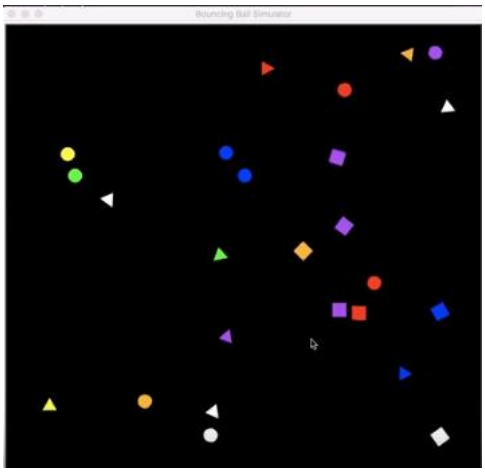
This four-part tutorial uses Python turtles to simulate bouncing balls – well worth a look

Part 1:
https://www.youtube.com/watch?v=HHQV3ifJopo

Part 2:
https://www.youtube.com/watch?v=ibdICVK0W3Q

Part 3:
https://www.youtube.com/watch?v=CZO_UzegLC0

Part 4:
https://www.youtube.com/watch?v=horBQxH0M5A



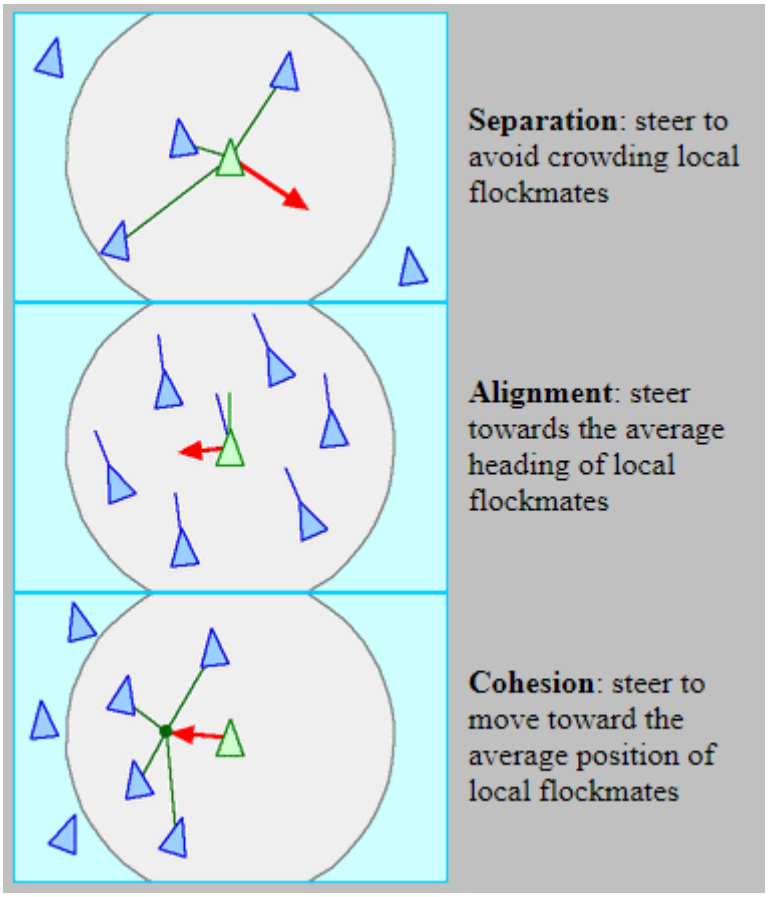
Boids[3]

In 1987 Craig Reynolds published "Flocks, herds and schools: A distributed behavioural model", which describes an agent-based model of herd behaviour. You can download his paper from https://thinkcomplex.com/boid.

Agents in this model are called "Boids", which is both a contraction of "bird-oid" and an accented pronunciation of "bird" (although Boids are also used to model fish and herding land animals).

Reynolds's own description of Boids can be found by browsing to his website http://www.red3d.com/cwr/boids/

The basic flocking model consists of three simple steering behaviours which describe how an individual boid manoeuvres based on the positions and velocities its nearby flock mates:



[3] This introduction is taken from http://greenteapress.com/complexity2/html/thinkcomplexity2011.html - another resource well worth checking out!

Game of Life

START BY WATCHING …..

https://www.youtube.com/watch?v=C2vgICfQawE

What is the Game of Life?

By Paul Callahan[4]

The Game of Life (or simply Life) is not a game in the conventional sense. There are no players, and no winning or losing. Once the "pieces" are placed in the starting position, the rules determine everything that happens later. Nevertheless, Life is full of surprises! In most cases, it is impossible to look at a starting position (or *pattern*) and see what will happen in the future. The only way to find out is to follow the rules of the game.

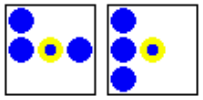
Rules of the Game of Life

Life is played on a grid of square cells--like a chess board but extending infinitely in every direction. A cell can be *live* or *dead*. A live cell is shown by putting a marker on its square. A dead cell is shown by leaving the square empty. Each cell in the grid has a neighbourhood consisting of the eight cells in every direction including diagonals.

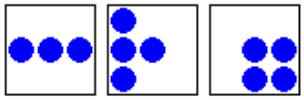
[4] http://www.math.com/students/wonders/life/life.html

To apply one step of the rules, we count the number of live neighbours for each cell. What happens next depends on this number.

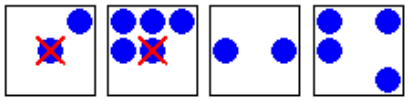
- A dead cell with exactly three live neighbors becomes a live cell (birth).



- A live cell with two or three live neighbors stays alive (survival).



- In all other cases, a cell dies or remains dead (overcrowding or loneliness).



Note: The number of live neighbours is always based on the cells ***before*** the rule was applied. In other words, we must first find all of the cells that change before changing any of them. Sounds like a job for a computer!

Background

Life was invented by the mathematician John Conway in 1970. He choose the rules carefully after trying many other possibilities, some of which caused the cells to die too fast and others which caused too many cells to be born. Life balances these tendencies, making it hard to tell whether a pattern will die out completely, form a stable population, or grow forever.

Life is just one example of a ***cellular automaton***, which is any system in which rules are applied to cells and their neighbours in a regular grid.

There has been much recent interest in cellular automata, a field of mathematical research. Life is one of the simplest cellular automata to have been studied, but many others have been invented, often to simulate systems in the real world.

In addition to the original rules, Life can be played on other kinds of grids with more complex patterns. There are rules for playing on hexagons arranged in a honeycomb pattern, and games where cells can have more than two states (imagine live cells with different colors).

Life is probably the most often programmed computer game in existence. There are many different variations and information on the web. (See the Paul Callahan's home page for more information.)

Why is Life So Interesting?

Life is one of the simplest examples of what is sometimes called "emergent complexity" or "self-organizing systems." This subject area has captured the attention of scientists and mathematicians in diverse fields. It is the study of how elaborate patterns and behaviours can emerge from very simple rules. It helps us understand, for example, how the petals on a rose or the stripes on a zebra can arise from a tissue of living cells growing together. It can even help us understand the diversity of life that has evolved on earth.

In Life, as in nature, we observe many fascinating phenomena. Nature, however, is complicated and we aren't sure of all the rules. The game of Life lets us observe a system where we know all the rules. Just like we can study simple animals (like worms) to discover things about more complex animals (like humans), people can study the game of Life to learn about patterns and behaviours in more complex systems.
The rules described above are all that's needed to discover anything there is to know about Life, and we'll see that this includes a great deal. Unlike most computer games, the rules themselves create the patterns, rather than programmers creating a complex set of game situations

How Complex Can Life Get?

A computer can be built inside the Life "universe". Space does not permit a detailed description, but you can find much more information in some of the references given at the bottom. Briefly, streams of gliders and spaceships can be used to send information just as electrical signals are used to send information in a physical computer. These streams of gliders can react in a way to perform all of the logical functions on which a modern computer is based. It would be very impractical to build a computer this way, but given a large enough Life pattern and enough time, we could run any program that runs on a computer. Several interesting special-purpose computers have been constructed as Life, including one that outputs the prime numbers.

A universal constructor can even be built. This is a pattern that can take a blueprint for some other Life pattern (or its own) and build that pattern. No one has built this yet, since it would be very large, but it has been shown to be possible. This means that Life patterns could exist

that reproduce themselves. They could even modify their blueprints just as living things combine and mutate their genes. Who can say what would develop in a large enough universe of reproducing Life patterns?

What is Life Good For?

Studying the patterns of Life can result in discoveries in other areas of math and science. The behaviour of cells or animals can be better understood using simple rules. Behaviour that seems intelligent, such as we see in ant colonies might just be simple rules that we don't understand yet. Take a look at this simulation of termites piling up woodchips. (click here) There are only 2 rules in this system, and yet, a seemingly "intelligent" pattern emerges. What does this say about the nature of intelligence?

Traffic problems might be solved by analysing them with the mathematical tools learned from these types of simulations. (Unjamming Traffic with Computers)

Computer viruses are also examples of cellular automata. Finding the cure for computer viruses could be hidden in the patterns of this simple game.
Human diseases might be cured if we could better understand why cells live and die.

Exploring the galaxies would be easier if machines could be invented that could build themselves. Imagine sending a probe to Mars that could build a copy of itself. Although this is theoretically possible, it hasn't been invented yet!

Is Life Alive?

Would living creatures evolve in a sufficiently large Life universe if we waited long enough? We can see that Life, simple as it is to describe, exhibits much of the complexity of our own universe. It is intriguing to ask what would happen in an infinitely large Life space seeded with random patterns. It seems that likely that complexity would emerge beyond what we can see when we watch Life on a computer. Even in our own universe, there is a huge difference between what we know about natural history and what we can observe on a human time frame.

On the other hand, Life has only two dimensions, unlike our own universe, and that is a severe limitation. There are other properties of Life -- the tendency to stabilize locally into oscillators -- that may make it an unlikely place for living things to develop. The answer to

this question remains unknown, but Life illustrates at a simplified level the kinds of evolutionary forces that we witness in our own universe.

Bibliography

Berlekamp, Conway, and Guy: *Winning Ways (for your Mathematical Pl*ays), Volume 2, (c)1982. ISBN 0-12-091152-3.

Gardner, Martin: *Wheels, Life, and Other Mathematical Amusements*, (c)1983. ISBN 0-7167-1589-9.

Poundstone, William: *The Recursive Universe*, (c)1985. ISBN 0-688-03975-8.

For more information browse to the following sites …

https://bitstorm.org/gameoflife/	A Game of Life simulator
https://www.youtube.com/watch?v=ouipbDkwHWA	Explains the rules of Game of Life
https://www.youtube.com/watch?v=R9Plq-D1gEk	John Conway - machines on Mars! Different iterations - 18 months of coffee breaks! – on a Go Board before computers (PDP-7)
https://www.youtube.com/watch?v=xOCe5HUObD4	John Conway – on his own life
http://golly.sourceforge.net/	Golly is an open source, cross-platform application for exploring Conway's Game of Life and many other types of cellular automata.

Hair simulation

Follow this fascinating tutorial from Khan Academy to find out how Pixar Animation managed to model and simulate Princess Merida's hair in the 2012 film, Brave.

The series starts at https://www.khanacademy.org/partner-content/pixar/simulation/hair-simulation-101/v/hair-simulation-intro

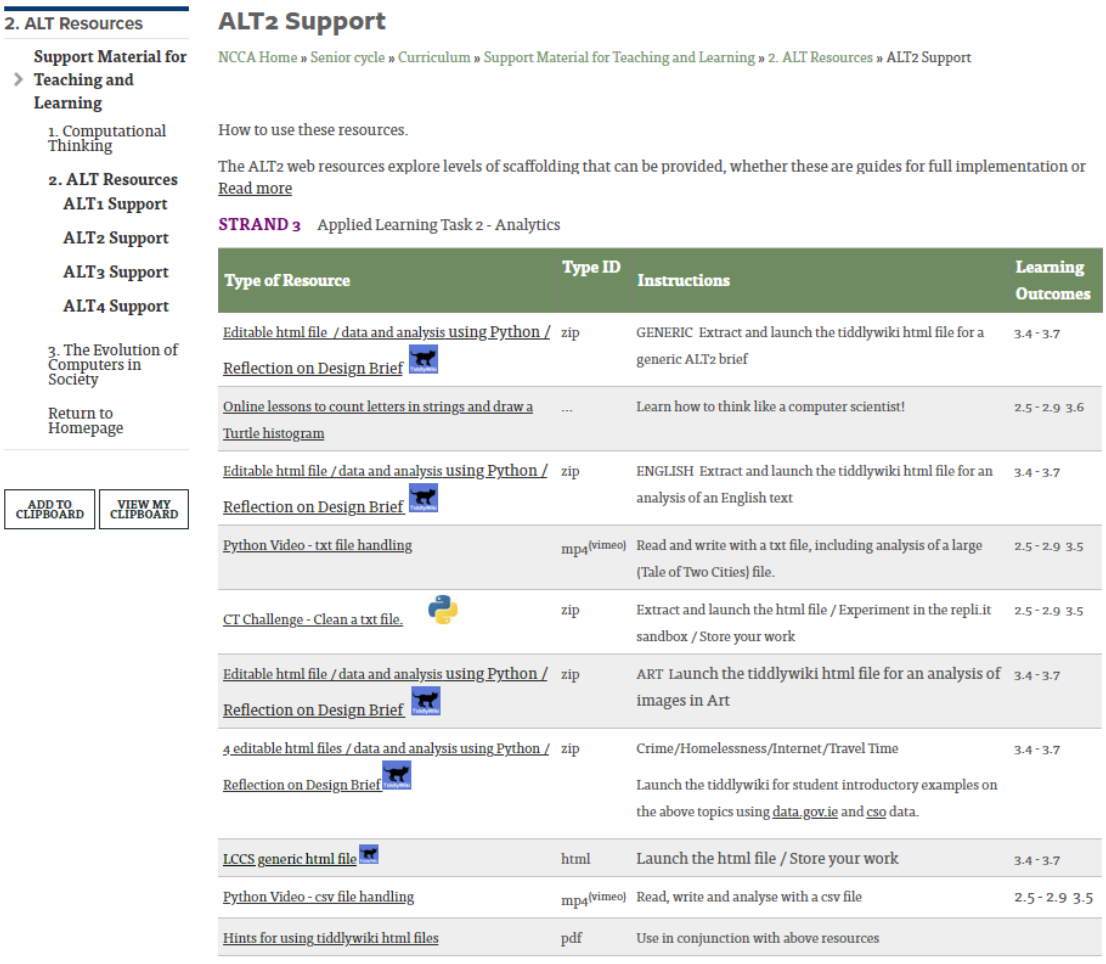
National Workshop 3

Tutorial Demonstrations for ALT2 and ALT3

Sample 1

The NCCA website contains a number of indicative samples designed to help teachers and students prepare for Applied Learning task 2. Browse to the link below and take some time to read about the resources.

https://nccacurriculum.azurewebsites.net/Senior-cycle/Senior-Cycle-Subjects/Computer-Science/Support-Material-for-Teaching-and-Learning/2-ALT-Resources/ALT2-Support



2. ALT Resources

Support Material for Teaching and Learning

1. Computational Thinking

2. ALT Resources

ALT1 Support

ALT2 Support

ALT3 Support

ALT4 Support

3. The Evolution of Computers in Society

Return to Homepage

ADD TO CLIPBOARD | VIEW MY CLIPBOARD

ALT2 Support

NCCA Home » Senior cycle » Curriculum » Support Material for Teaching and Learning » 2. ALT Resources » ALT2 Support

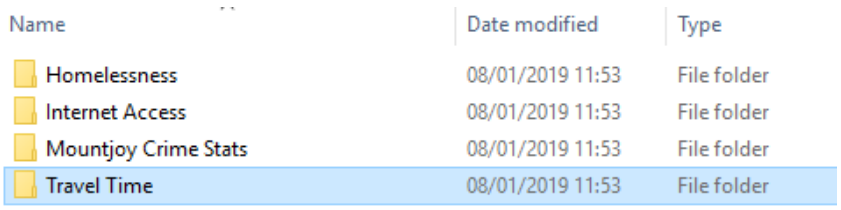
How to use these resources.

The ALT2 web resources explore levels of scaffolding that can be provided, whether these are guides for full implementation or Read more

STRAND 3 Applied Learning Task 2 - Analytics

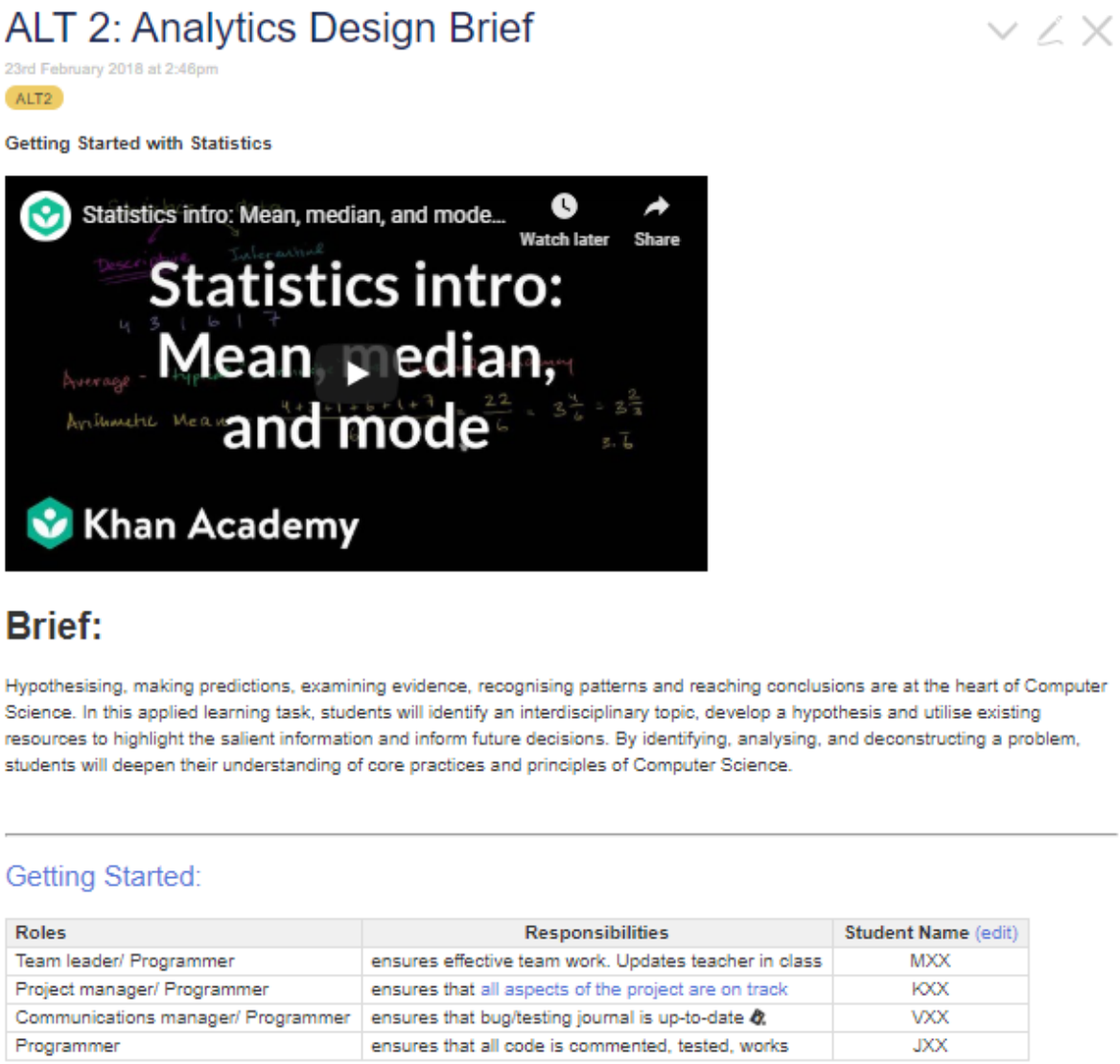
Type of Resource	Type ID	Instructions	Learning Outcomes
Editable html file / data and analysis using Python / Reflection on Design Brief	zip	GENERIC Extract and launch the tiddlywiki html file for a generic ALT2 brief	3.4 - 3.7
Online lessons to count letters in strings and draw a Turtle histogram	...	Learn how to think like a computer scientist!	2.5 - 2.9 3.6
Editable html file / data and analysis using Python / Reflection on Design Brief	zip	ENGLISH Extract and launch the tiddlywiki html file for an analysis of an English text	3.4 - 3.7
Python Video - txt file handling	mp4(vimeo)	Read and write with a txt file, including analysis of a large (Tale of Two Cities) file.	2.5 - 2.9 3.5
CT Challenge - Clean a txt file.	zip	Extract and launch the html file / Experiment in the repli.it sandbox / Store your work	2.5 - 2.9 3.5
Editable html file / data and analysis using Python / Reflection on Design Brief	zip	ART Launch the tiddlywiki html file for an analysis of images in Art	3.4 - 3.7
4 editable html files / data and analysis using Python / Reflection on Design Brief	zip	Crime/Homelessness/Internet/Travel Time Launch the tiddlywiki for student introductory examples on the above topics using data.gov.ie and cso data.	3.4 - 3.7
LCCS generic html file	html	Launch the html file / Store your work	3.4 - 3.7
Python Video - csv file handling	mp4(vimeo)	Read, write and analyse with a csv file	2.5 - 2.9 3.5
Hints for using tiddlywiki html files	pdf	Use in conjunction with above resources	

Download and unpack the resource on crime/homelessness/internet and travel time. Browse to the ‘Travel Time’ folder and launch the application.



Name	Date modified	Type
Homelessness	08/01/2019 11:53	File folder
Internet Access	08/01/2019 11:53	File folder
Mountjoy Crime Stats	08/01/2019 11:53	File folder
Travel Time	08/01/2019 11:53	File folder

You should see a screen like the one shown below:



Scroll down taking some time to read each section.

To run the sample copy and paste the data from the bottom of the page into a file and save it as **data.txt** (Use Notepad)

Then copy and paste the code (listing) from the page and save in in your favourite Python editor. The listing is provided on the next page for convenience,

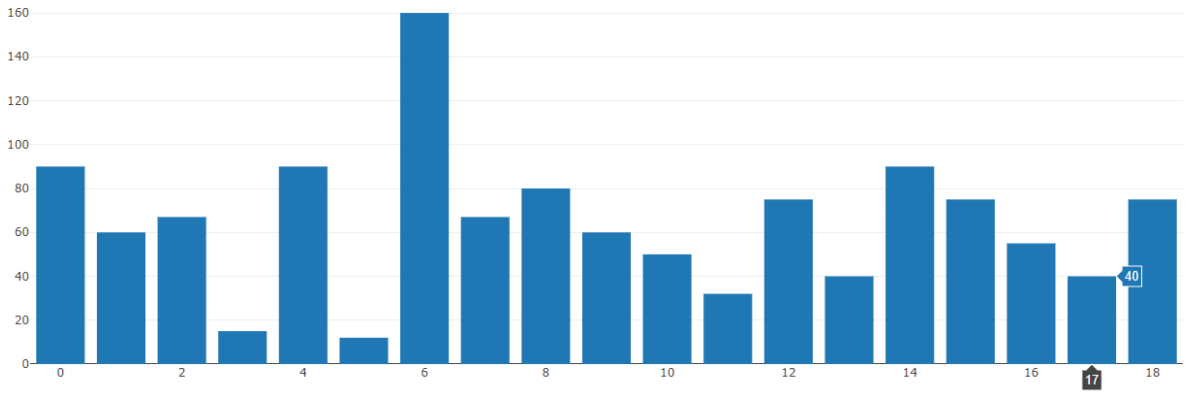
travel.py

```
import re
import statistics
import plotly.plotly
from plotly.graph_objs import Bar, Layout


file = open("data.txt","r")
#http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python

string = file.read()

file.close()

clean_string = re.sub(' minutes', '', string)

clean_string = re.sub(' ', '', clean_string)

string_array = clean_string.split('\n')

print(string_array)

import math


int_array = [int(i) for i in string_array]


mean_value = statistics.mean(int_array)
print (mean_value)

median_value =(statistics.median_grouped(int_array, 1))
print (median_value)

#mode value =(statistics.mode(int array))
#print (mode_value)

plotly.offline.plot({           #https://plot.ly/python/getting-started/#initialization-for-offline-plotting
    "data": [Bar(y=int_array)],
    "layout": Layout(title="Travel times")
})
```

In order to understand the code it may be necessary to research the following libraries

re	https://docs.python.org/3/library/re.html
statistics	https://docs.python.org/3/library/statistics.html
plotly	https://plot.ly/python/

Run the program and you should see the following output



Sample 2

Purpose

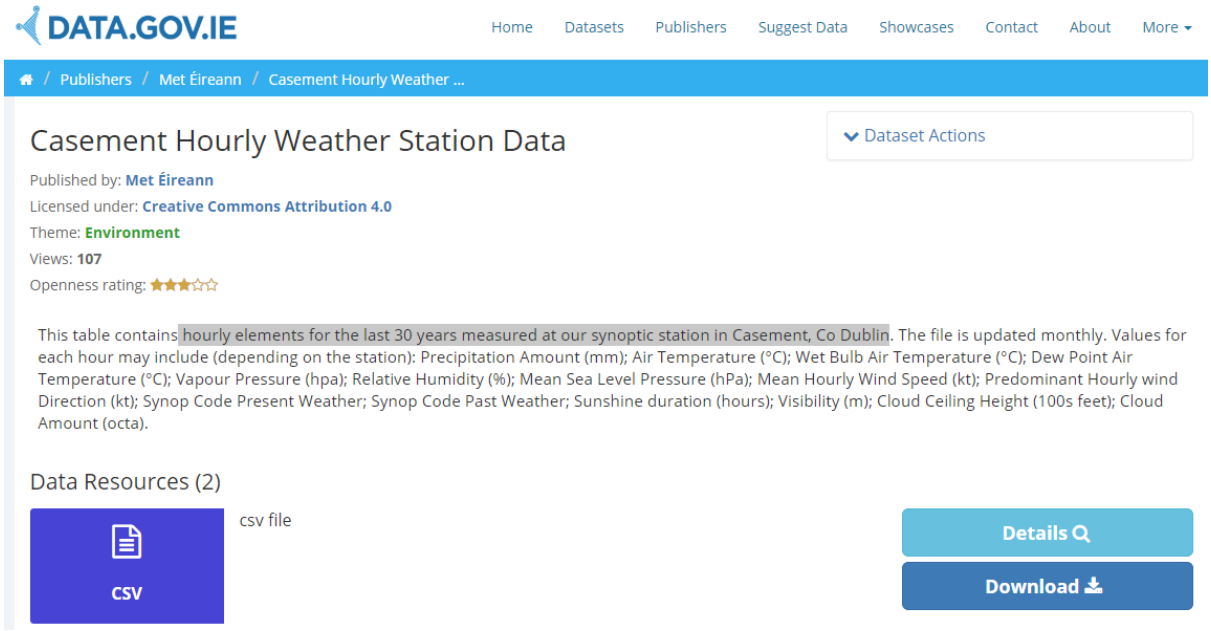
To explore and analyse a CSV file which contains hourly weather values (e.g. rainfall, temperature etc) for the last 30 years measured at Met Éireann's synoptic station in Casement, Co Dublin

Step 1

Download the data file.

Browse to https://data.gov.ie/dataset/casement-hourly-weather-station-data and click on the download button to download the CSV file.

IMPORTANT: Save the file as **weather data.csv**



Step 2

Browse the data file noting as many observations as you can.

As part of my analysis work I created the following two files by copying and pasting rows from the original **weather data.csv**

1. A file called **Jan 88.csv**. This file was made up of rows 25-768 inclusive. The file contained all the weather data for the 31 days of January i.e. from 01/01/1988 00:00 up to 31/01/1988 23:00
2. A file called **1988.csv**. This file was made up of rows 25-8808 inclusive. The file contained all the weather data for the 366 days of 1988 i.e. from 01/01/1988 00:00 up to 31/12/1988 23:00

I also created a breakdown/map of the entire file as shown here to the right.

This helped me gain a better understanding of the file's contents and how its contents is organised especially in terms of relationships between hours, days, months and years.

I noted how information such as day, month, year and hour were contained as part of the date string.

In particular, I noted that the month could be extracted from the date string by using the 4th and 5th characters.

My final observation was that the temperature values were decimal and would therefore need to be represented as floats in any Python program that processed them.

date	ind	rain	ind	temp	i
01/01/1988 00:00	0	0.1	0	6.6	
01/01/1988 01:00	0	0	0	6.6	
...					
01/01/1988 23:00	0	0	0	9.5	
02/01/1988 00:00	0	0	0	9.1	
...					
02/01/1988 23:00	0	0.1	0	6.5	
03/01/1988 00:00	2	0	0	6.6	
...					
31/01/1988 23:00	0	0	0	7.5	
01/02/1988 00:00	0	0	0	6.3	
...					
30/11/1988 23:00	0	0	0	7.2	
01/12/1988 00:00	2	0	0	6.8	
...					
31/12/1988 23:00	3	0	0	7.7	
01/01/1989 00:00	3	0	0	7.2	
...					
31/12/1989 23:00	0	0	0	9.2	
01/01/1990 00:00	0	0	0	9.2	
...					
31/12/1990 23:00	0	0	0	1.5	
01/01/1991 00:00	0	0	0	1.5	
...					
31/12/2017 23:00	0	0	0	3.6	
01/01/2018 00:00	0	0.1	0	4	
...					
30/11/2018 23:00	0	0	0	6.2	
01/12/2018 00:00	0	0	0	6	

Some observations I made when I opened the file initially were:

- Rows 1-3 contain header information
- Row 24 is a header row for the data that follows it. The meaning of each column header can be decoded from the key information contained in rows 6-22. The conclusion I draw from this is that the data following row 24 is highly structured.
- The 'temp' column grabs my attention simply because temperature it is the focus of the next demonstration which processes temperature generated by a micro:bit.
- The file contains over 270,000 rows of data – difficult to process manually and certainly more amenable to some automated processing and possibly requiring some form of condensing (by way of summary)
- The data values correspond to an element of weather at a given date/time.
- The format of every date value is '*dd/mm/yyyy hh:00*' where, *hh* is given is 24-hour format. I take 00:00 to mean midnight and 23:00 to mean 11pm.
- Hourly values are provided starting from 1st Jan 1988 up to 1st Dec 2018 (a span of almost 31 years – a total of 372 months)
- So there are 24 sets of values for every day. This means for a 31-day month there are 744 (i.e. 31*24) rows of data. There would be 8,760 rows of values for a normal 365-day year.
- By scrolling down to row number 1441 I note that values are given for 29th Feb and assume that all leap years must be accounted for

My next step is to decide upon some 'project' that makes use/sense of the data. I decide to focus on temperature and ponder whether the data contains any trends e.g. does it display seasonal variations, are temperatures increasing/decreasing?

There is an overwhelming amount of data so I decide to focus on monthly temperature as opposed to hourly values. In order to do this, I will need to write a program to generate a list of average monthly temperatures over the 31 year period. (The list will contain 372 values. Each value will be the average temperature for that month.)

The average temperature can be calculated by adding up every temperature value in a month and then dividing by the number of days in the month.

Step 3. Design and Implement. The pseudo-code for my initial algorithm is as follows:

*Open the file **weather data.csv***
Ignore the first 24 rows
For every row of data
    Extract the value of the month from the current row
    If the month is the same as it was previously
        Add the temperature to the running total of temperatures
    Else (the month has changed)
        Calculate the monthly average
        Append the average to the list of monthly averages
        Reset the running total of temperatures
Display the list of monthly averages

My initial program looked like this – take some time to study the code before keying it in and running it. The notes on the next page provide an explanation of the variables and logic.

```
1  import csv
2
3  # Extract the month from a date string e.g. '01/02/1988 05:00' will return 2
4  def getMM(dateStr):
5      return int(dateStr[3:5])
6
7
8  with open('weather data.csv') as csv_file:
9      csv_reader = csv.reader(csv_file, delimiter=',')
10     monthly_avg_temps = []
11     total_temp = 0
12     prev_month = 1
13     number_of_readings = 0
14
15     for row in csv_reader:
16
17         this_month = getMM(row[0])
18         if this_month == prev_month:
19             number_of_readings = number_of_readings + 1
20             total_temp = total_temp + float(row[4])
21         else:
22             # save the average temperature for the month
23             monthly_avg_temps.append(total_temp/number_of_readings)
24             # The month has changed so reset the total
25             total_temp = float(row[4])
26             # reset the number of readings to one
27             number_of_readings = 1
28
29         # save the value of this month
30         prev_month = this_month
31
32     # make sure to append in the last months average
33     monthly_avg_temps.append(total_temp/number_of_readings)
34
35 print("Monthly average temps: ", monthly_avg_temps)
```

Notes on the program

1. Since the weather data file is in CSV format I decided to research the use of the csv Python library for reading/writing CSV files. See https://realpython.com/python-csv/ for more information.

2. In order to simplify the implementation, I deleted rows 1-24 of the data file – this meant that the program could process every row of data unconditionally i.e. no need to check that the row number is greater than 24 on each loop iteration.

3. The significant variables and logic are explained as follows:

- `row` is a list which contains as its elements all the values in each individual row of the CSV file.
  `row[0]` is the first element of row and this contains the value in the date column.
  `row[4]` is the fifth element of row and this contains the value in the temperature column.

- The main processing of the program is carried out in the `for` loop on line 15 (the loop body spans lines 16-30). The contents of `row` are updated with the values contained in the next row of data from the CSV file on each iteration of the loop.

- `this_month` is assigned a value at the start of each loop iteration on line 17. The variable is assigned the value of the 'mm' component of the date column. The function `getMM` is used to extract the month from the date. (The function works by slicing the date from position 3 up to, but not including position 5 of the date. Remember indices are zero based)

- `prev_month`. This variable is critical to the logic of the program. It is used on line 18 to determine whether the current row is the first row of a new month or just another row of values in the same month as the previous row. The variable is initially set to 1 on line 12 and in updated on line 30 at the end of each loop iteration. Line 30 is the last line in the `for` loop body and therefore the last line before a new row is read (i.e. the last line before `row` gets a new value). This line saves the value of the current month before a new row is read.

- `monthly_avg_temps` is a list of monthly average temperatures. The purpose of the program is to populate this list. The empty list is created on line 10. As the averages are calculated they are appended to the list – line 23.

- `total_temp` is a running total. Initially set to 0 (line 11) the value is updated by adding the temperature value of the current row to the accumulated temperature for the month so far. This addition is performed on line 20. The actual temperature value is extracted using `row[4]` i.e. the 5th column value of the row. Since the row data is in string format it must be converted to a `float` so that it can be added. Every time the month changes (e.g. after reading 31*24 rows) the value of `total_temp` is reset to be the first temperature value for the new month. This is performed on line 25.

- `number_of_readings` is a simple counter variable. It is used to count the number of readings for each month. The variable is created on line 13 where its value is initialised to 0. Every time a new row is read for the same month the value of `number_of_readings` is increased by 1 (line 19). The value of is reset to 1 on the first row of every new month. The average temperature for the month is calculated by dividing the total temperature for the month (`total_temp`) by the `number_of_readings` for the month (line 23). This calculation is also carried out on line 33 to record the average temperature for the final month of the final year in the file.

After running the program a snapshot of the output looked like this:

```
>>> %Run 'weather v1.py'

  Monthly average temps:  [5.366935483870956, 4.605890804597696, 6.60349462365
  592, 8.480555555555572, 10.852956989247344, 14.296805555555586, 13.653091397
  849483, 14.0943548387097, 12.215972222222245, 10.24717741935487, 6.206249999
  999991, 8.050672043010742, 7.26895161290323, 5.952380952380947, 6.8700268817
  2043, 6.039027777777777, 11.810752688172053, 13.516944444444443, 17.49381720
  430108, 15.317607526881726, 13.079027777777794, 11.24180107526882, 7.1899999
```

At a glance I note that the initial output looks good. Taking the first value to be the monthly average temperature for Jan 1988 and the second to be the average for Feb 1988 etc. it makes sense that the averages for winter months are relatively low compared to summer months.

However, further testing was required in order to verify that the data was indeed correct.

Step 4.

The next logical step for me was to display the monthly averages graphically.

This would involve the use of the `plotly` library – see https://plot.ly/python/ for more information and https://images.plot.ly/plotly-documentation/images/python_cheat_sheet.pdf for a nice Python cheat sheet on `plotly`.

I added the following two import statements (lines 2 and 3) at the start of my program

```
1  import csv
2  import plotly as plt
3  import plotly.graph_objs as graf
```

My next step is to prepare the x and y data for `plotly`. The data in `monthly_avg_temps` would make up my y values so I needed to decide what to use for my x-values. The simple choice was to use a list of numbers that would correspond to the monthly temperature readings i.e. 0 would correspond to the average temperature calculated for Jan 1988, 1 for Feb 1988 and so on right up to 317 which would correspond to Dec 2108.

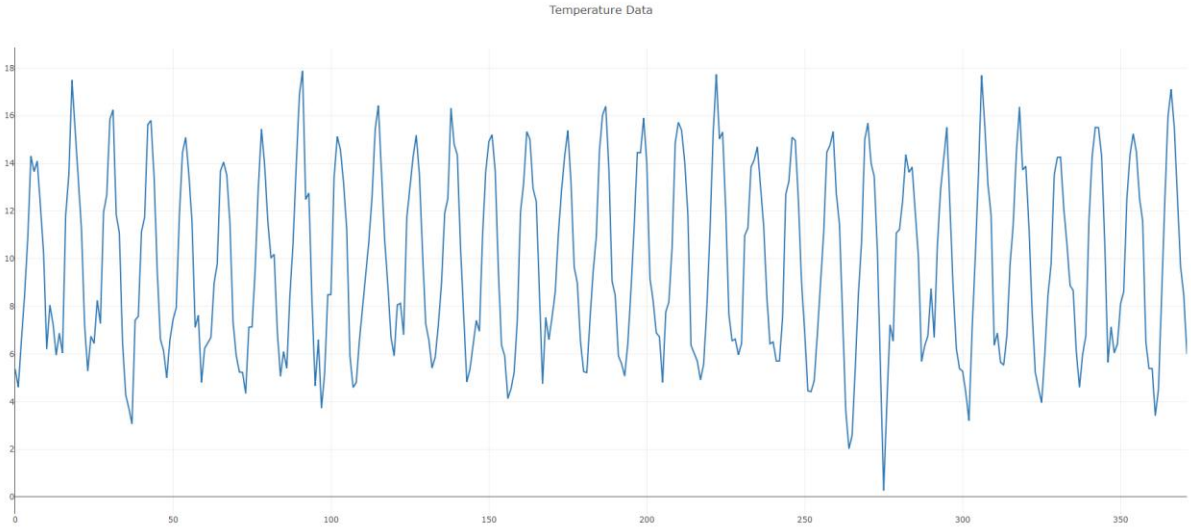
The code to create and populate this list (months) is inserted immediately after the for loop as follows:

```
37 # Create an populate a list of values for the x-axis
38 months = []
39 for i in range(len(monthly_avg_temps)):
40     months.append(i)
```

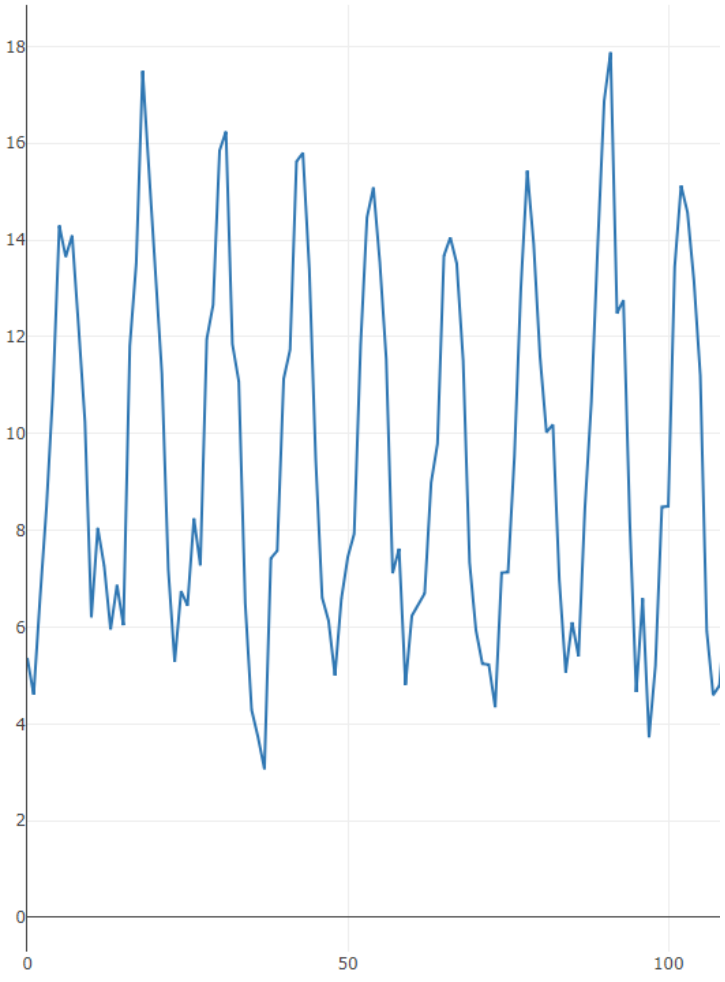
As a final step I just needed to write the code to display the graph.

```
42 #plot the data
43 plotData = graf.Scatter(name = "model1", x = months, y = monthly_avg_temps)
44 layoutSettings = graf.Layout(title = "Temperature Data")
45 plt.offline.plot({
46     "data" : [plotData],
47     "layout" : layoutSettings  })
```

When I ran my program the final output looked like this:



A smaller section of the above graph is shown here on the right to add clarity.



It is important to take time to interpret and evaluate the output.

Upon reflection there is scope to improve the values on the x-axis – perhaps they could be more meaningful e.g. start at 01/88, 02/88 and continue until 12/18?

Although both graphs clearly depict the cyclical nature of temperature – i.e. higher during summer months and lower during winter months an overall annual trend is not apparent.

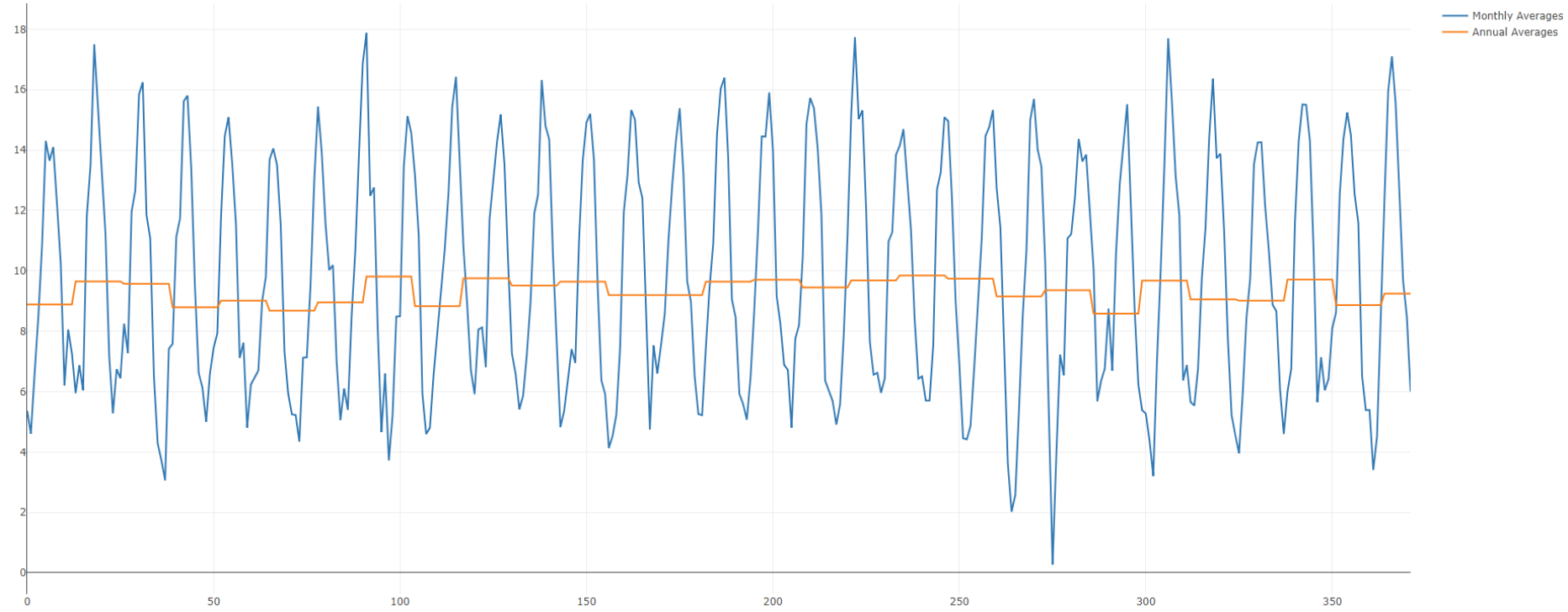
Step 5.

As a final enhancement I added code to calculate and display the annual average temperatures on the same graph.

The code listing is provided below without any explanation.

```
import csv
import plotly as plt
import plotly.graph_objs as graf

# Extract the month from a date string e.g. '01/02/1988 05:00' will return 2
def getMM(dateStr):
    return int(dateStr[3:5])


with open('weather data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    monthly_avg_temps = []
    total_temp = 0
    prev_month = 1
    number_of_readings = 0

    for row in csv_reader:

        this_month = getMM(row[0])
        if this_month == prev_month:
            number_of_readings = number_of_readings + 1
            total_temp = total_temp + float(row[4])
        else:
            # save the average temperature for the month
            monthly_avg_temps.append(total_temp/number_of_readings)
            # The month has changed so reset the total to be the first temp reading of the
next month
            total_temp = float(row[4])
            # reset the number of readings to one
            number_of_readings = 1

        # save the value of this month
        prev_month = this_month

    monthly_avg_temps.append(total_temp/number_of_readings)


# Create an populate a list of values for the x-axis
months = []
for i in range(len(monthly_avg_temps)):
    months.append(i)

# Compute annual averages
annual_avg_temps = []
total_temp = 0
for i in range(1, len(monthly_avg_temps)+1):
    if i%12 == 0:
        for j in range(13):
            annual_avg_temps.append(total_temp/12)
        total_temp = 0
    else:
        total_temp += monthly_avg_temps[i-1]


# plot both sets of data
plotData1 = graf.Scatter(name = "Monthly Averages", x = months, y = monthly_avg_temps)
plotData2 = graf.Scatter(name = "Annual Averages", x = months, y = annual_avg_temps)
plotData = [plotData1, plotData2]
layoutSettings = graf.Layout(title = "Temperature Data")
plt.offline.plot(plotData, filename='Hourly.html')
```

When run, the previous listing generates the following output:



The annualised average temperatures appear as an almost horizontal line across the centre of the graph.

It would appear that temperatures in Casement, Co. Dublin have remained fairly constant over the past 31 years.

What conclusions, if any, about climate change can be inferred from the above?

Sample 3[5]

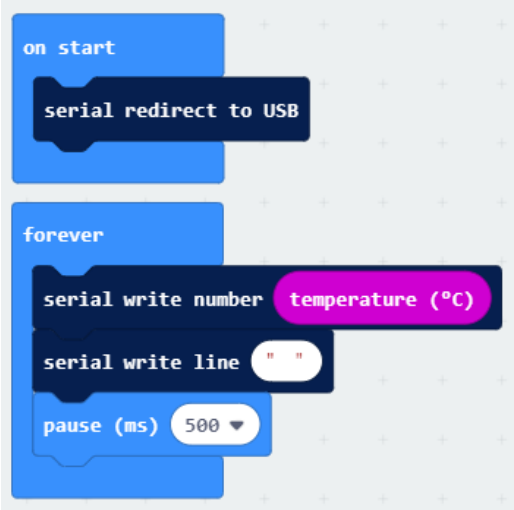
Purpose:

a) to program a micro:bit to generate and send temperature data to a serial port

b) to develop a Python program to receive (and process) the above temperature data

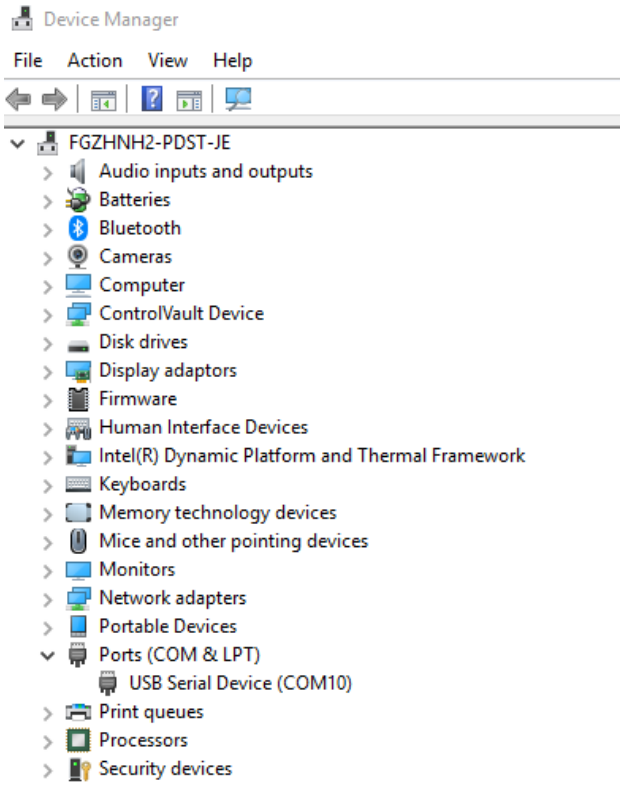
Part a)

1. Create the code blocks shown here to the right on https://makecode.microbit.org/



2. Connect a micro:bit to your device and flash (download) the code.

3. Use the device manager to discover the name of the port your micro:bit is connected to. In my case this was COM10 – see screenshot below.



[5] NCCA video resources forthcoming

Part b)

Key in, save and run the following Python code to your favourite editor. (You will need to make sure the packages `serial` and `pyserial` are installed.)

```
1  import serial
2
3  ser = serial.Serial()
4  ser.baudrate = 115200
5  ser.port = 'com10'
6
7  ser.open()
8
9  dataString = []
10
11 while True:
12     infoString = str(ser.readline())
13     print(infoString)
```

Note that the port name (i.e. com10) is hardcoded on line 5 – you may need to change this value to suit your own configuration.

When the above code is run the output generated will be similar to that shown here …

```
>>> %Run 'displayTemp v1.py'
 b'           \r\n'
 b'24                              \r\n'
 b'24                              \r\n'
 b'24                              \r\n'
 b'24                              \r\n'
 b'24                              \r\n'
 b'24                              \r\n'
```

The program will continue to display the live temperature data forever (`while True`) or until the program is interrupted by the user.

A new temperature value is generated every 500ms (i.e every half second) by the micro:bit.

Experiment! Wrap your hand around the micro:bit and you should see an increase in the value of the temperature.

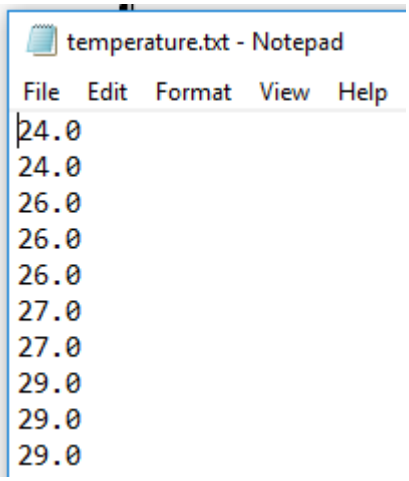
In this next version of the program the data is 'cleaned' and saved to a text file called `temperature.txt`

```
1  import serial
2
3  ser = serial.Serial()
4  ser.baudrate = 115200
5  ser.port = 'com10'
6
7  ser.open()
8
9  while True:
10     infoString = str(ser.readline())
11
12     if len(infoString) > 3 :
13         # extract the temperature data
14         cleanString = infoString[2:]
15         cleanString = cleanString.replace(" ", "")
16         cleanString = cleanString.replace("\\n'", "")
17         cleanString = cleanString.replace("\\r", "")
18         temperature = float(cleanString)
19
20         # write the temperature to a file
21         dataFile = open("temperature.txt", "a")
22         dataFile.write(str(temperature)+"\n")
23         dataFile.close()
```

Lines 14-18 extract the value of the temperature from the raw string generated by the micro:bit.

Lines 21-23 cause the individual temperature readings to be written on a separate line in the output file.

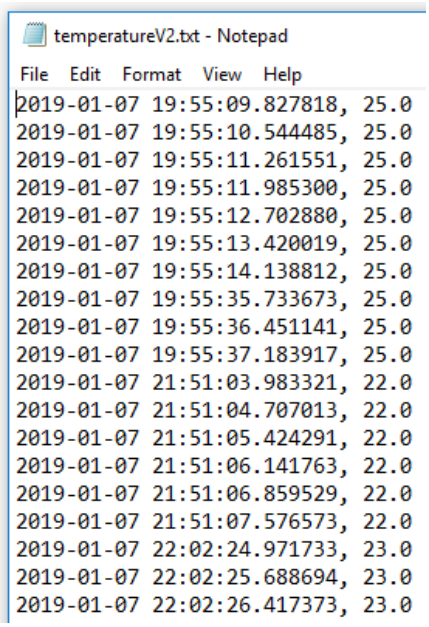
A sample of the resulting file is shown here to the right



As a slight refinement to the program the temperature data is timestamped (see lines 21 and 22 below)

```
1  import serial, datetime
2
3  ser = serial.Serial()
4  ser.baudrate = 115200
5  ser.port = 'com10'
6
7  ser.open()
8
9  while True:
10     infoString = str(ser.readline())
11
12     if len(infoString) > 3 :
13         # extract the temperature data
14         cleanString = infoString[2:]
15         cleanString = cleanString.replace(" ", "")
16         cleanString = cleanString.replace("\\n'", "")
17         cleanString = cleanString.replace("\\r", "")
18         temperature = float(cleanString)
19
20         # prepare the string to write
21         timeStamp = str(datetime.datetime.now())
22         dataString = timeStamp+ ", " +str(temperature)
23         print(dataString+u'\u00b0'+"C")
24
25         # write the string to a file
26         dataFile = open("temperatureV2.txt", "a")
27         dataFile.write(dataString+"\n")
28         dataFile.close()
```

A sample output file is shown here to the right



The program below demonstrates how the temperature data saved by the previous program can be plotted.

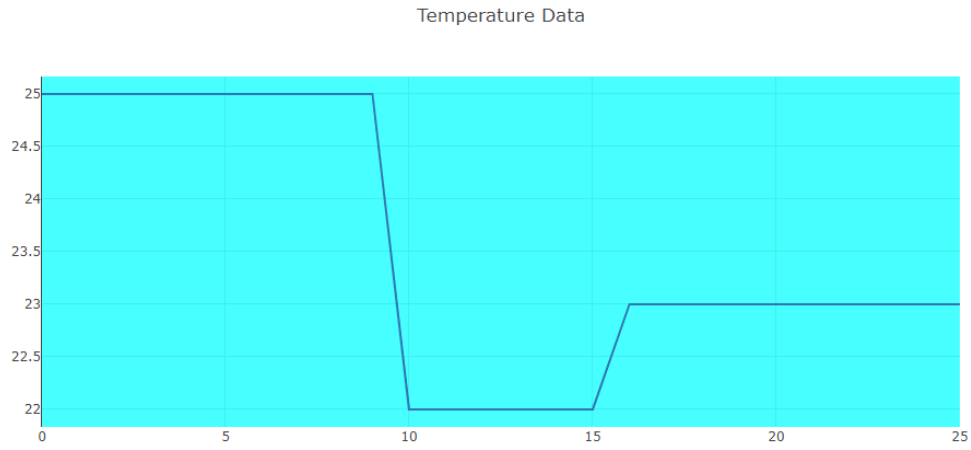
The program works by building up a list of temperature values to plot – these are the y-axis/dependent variables. (Lines 8-11)

Once the number of values is known a 'dummy' list of x-values is created – 1 per y-value. This is done on lines 14-16

Finally the lists are plotted – this is taken care of by lines 20-24

```
1  import plotly as plt
2  import plotly.graph_objs as graf
3
4  #open the temperature data file
5  dataFile = open("temperatureV2.txt", "r")
6
7  # Create a list for the temperature values (y-axis)
8  temperature_values = [] # a list of all the temperature values
9  for line in dataFile:
10     # extract the temperature from each line
11     temperature_values.append(line.split()[2])
12
13 # Create a list for x-axis
14 x_axis = []
15 for i in range(len(temperature_values)):
16     x_axis.append(i)
17 print(x_axis)
18
19 #plot the data
20 plotData = graf.Scatter(name = "model", x = x_axis, y = temperature_values)
21 layoutSettings = graf.Layout(title = "Temperature Data", plot_bgcolor = "cyan", height = 500, width = 1000)
22 plt.offline.plot({
23     "data" : [plotData],
24     "layout" : layoutSettings  })
```

When the above program is run using the data shown on the previous page the following graph is generated.



BLANK PAGE

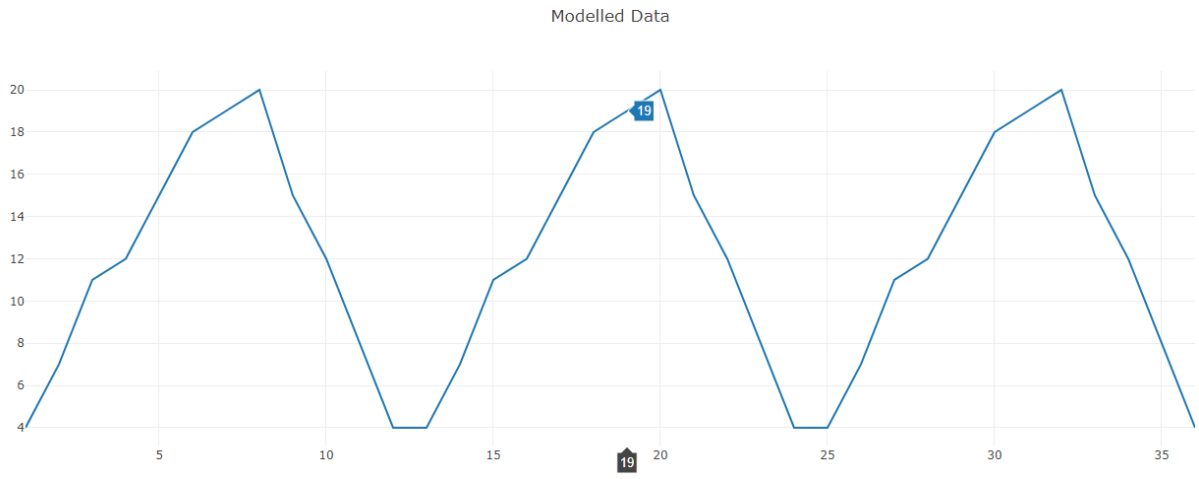
Sample 4

This sample is based around a resource which is yet to be published by the NCCA – refer to pages 71-74 of the workbook.

The idea of this sample is to develop a simple mathematical model based on temperature data. The source of the data is not too important – it could have been generated by a micro:bit temperature sensor or come from some official source for weather data. In the listing below the temperature data is just hardcoded into a list.

```
import plotly as plt
import plotly.graph_objs as graf

# Average daytime temperatures - Jan-Dec
temps = [4, 7, 11, 12, 15, 18, 19, 20, 15, 12, 8, 4]
temps *=3 # 3 years

months = [None] * 36
for i in range(36):
    months[i] = i+1

#plot the data
plotData = graf.Scatter(name = "model1", x = months, y = temps)
layoutSettings = graf.Layout(title = "Modelled Data")
plt.offline.plot({
    "data" : [plotData],
    "layout" : layoutSettings  })
```

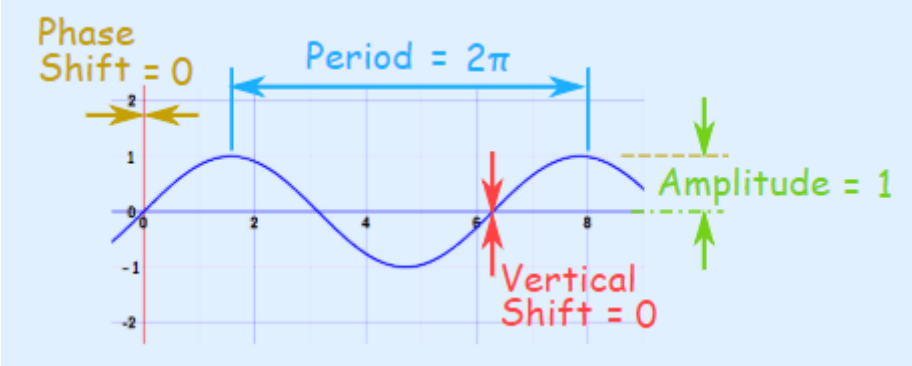
The program generates the following output – a graph of the temperature data repeated over three years (36 months)



The cyclical nature of the data is evident from the graph.

Cyclical data can be modelled mathematically using trigonometric (periodic) functions such as `sin` and `cos`. We wish to develop a `sin` model for the temperature data provided so that we can use it to predict future temperatures.

In order to develop this sample further it is first necessary to understand some terminology – amplitude, period and phase shift[6]. These terms are depicted in the following illustration[7] of the graph of $y = \sin x$



In particular, it is useful to know that in the equation:

$$a \sin b(x \pm c) \pm d$$

a is the amplitude,

$\frac{2\pi}{b}$ is the period,

c is the phase shift (a positive value means leftwards shift; a negative value means right)

d is the vertical shift (a positive value means upward shift; a negative value means downward)

See https://www.mathsisfun.com/algebra/amplitude-period-frequency-phase-shift.html for further examples.

[6] See https://www.khanacademy.org/math/algebra2/trig-functions/constructing-sinusoids-alg2/v/trig-function-equation for an explanation of how to formulate the equation of a sinusoidal function from a graph.

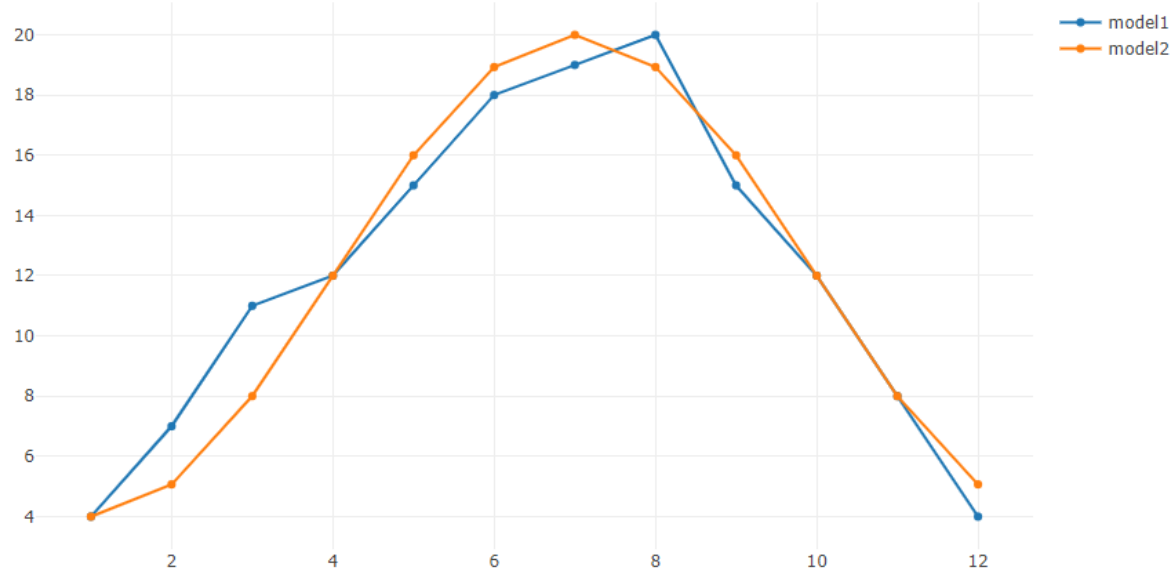
[7] Taken from https://www.mathsisfun.com/algebra/amplitude-period-frequency-phase-shift.html

In this next listing we plot the same temperature values (`temps`). This time however we also create a model based on these values. The model is then used to generate its own values which are stored in `temps2`.

```
1  import plotly as plt
2  import plotly.graph_objs as graf
3  import math
4
5  # Average daytime temperatures - Jan-Dec
6  temps = [4, 7, 11, 12, 15, 18, 19, 20, 15, 12, 8, 4] # y-values
7
8  # Generate the x-values
9  months = [None] * 12
10 for i in range(12):
11     months[i] = i+1
12
13 # Now generate the model
14 baseline = 12
15 amplitude = (max(temps) - min(temps))/2
16 period = (2*math.pi)/12
17 phase = (3*math.pi)/2 # sin is at its minimum at 3*pi/2
18
19 temps2 = []
20 for month in range(12) :
21     # sin model: a*sin(bx + c) + d
22     currTemp = amplitude * math.sin(period*month+phase) + baseline
23     currTemp = round(currTemp, 2)
24     temps2.append(str(currTemp))
25
26 #plot the data
27 plotData1 = graf.Scatter(name = "model1", x = months, y = temps)
28 plotData2 = graf.Scatter(name = "model2", x = months, y = temps2)
29 plotData = [plotData1, plotData2]
30 layoutSettings = graf.Layout(title = "Modelled Data v2")
31 plt.offline.plot(plotData, filename='bar-line.html')
```

Lines 27-31 plot both sets of values on the same graph. The resulting graph is shown on the next page.

Take care to examine lines 14-17. Note that the value 12 which is assigned to the variable basline on line 14 was discovered by ‘trial and error’. These lines set the values of the variables which are used to create the mathematical model. The model itself is used to generate data on line 22.



Exercises

1. Key in the code and experiment by changing the value of `baseline`.

2. Modify the code to use the 24 hourly temperature values recorded a Casement weather station on 01/01/1988. Start by hardcoding the values into the list and then refine your program to read the values from the CSV file.

3. Create a model based on the average monthly temperatures recorded a Casement weather station since 01/01/1988. Use your model to generate data and display this data on a graph along with the actual data.