Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

LEAVING CERTIFICATE COMPUTER SCIENCE

National Workshop 2

Professional Learning Booklet

2023-2024

Contents

Key Messages

All learning outcomes (LOs) are interwoven.
The specification can be used in many different ways.

ALTs provide an opportunity to teach theoretical aspects of LCCS.

LCCS can be mediated through a constructivist pedagogical approach.

Group work is a key feature in the leaching, learning and assessment of LCCS.

Session 1: Number Systems

To convert between number systems (binary to decimal, decimal to hexadecimal, etc.) we first need to understand how to decompose a number.

Decomposition of a Decimal Number

Decomposition of a Binary Number

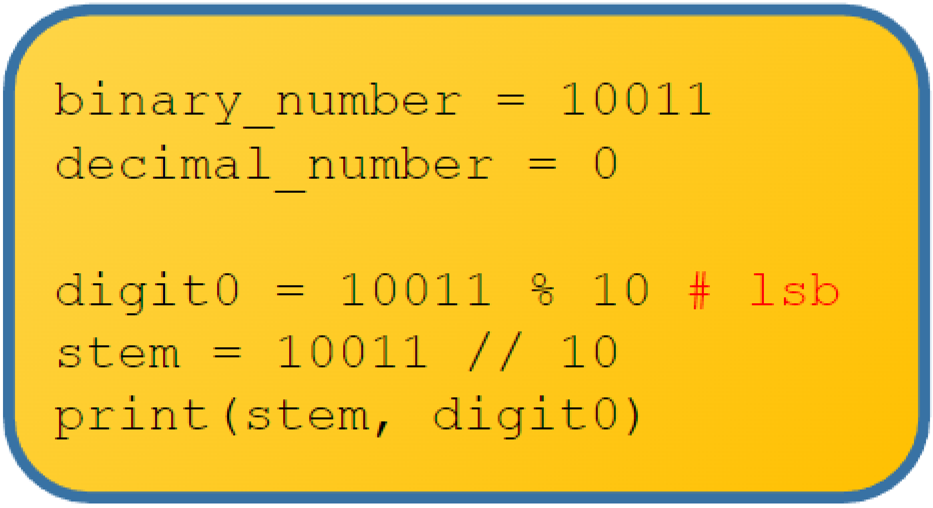
Converting Decimals to Binary

There are many ways to convert decimals to binary, today we look at a method that uses repeated division by 2.

Converting a Decimal Number to Binary

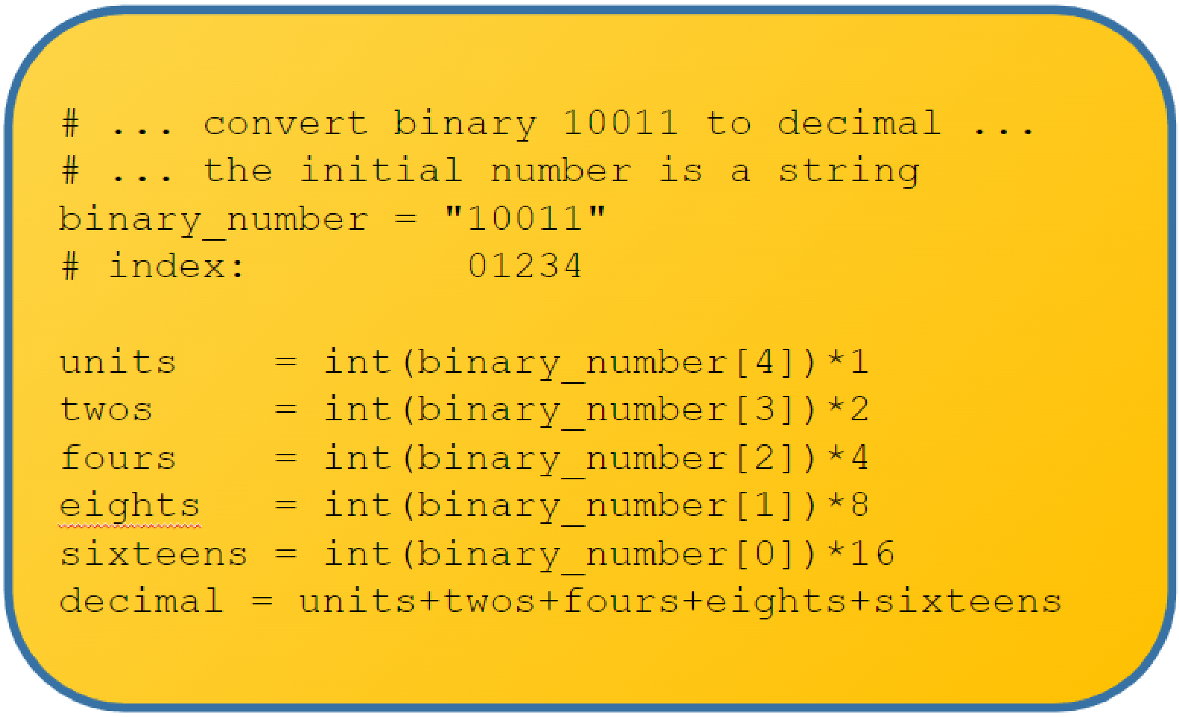
Breakout tasks

Develop the following code into a general solution to convert 10011 to decimal.



```
binary_number = 10011
decimal_number = 0

digit0 = 10011 % 10 # lsb
stem = 10011 // 10
print(stem, digit0)
```

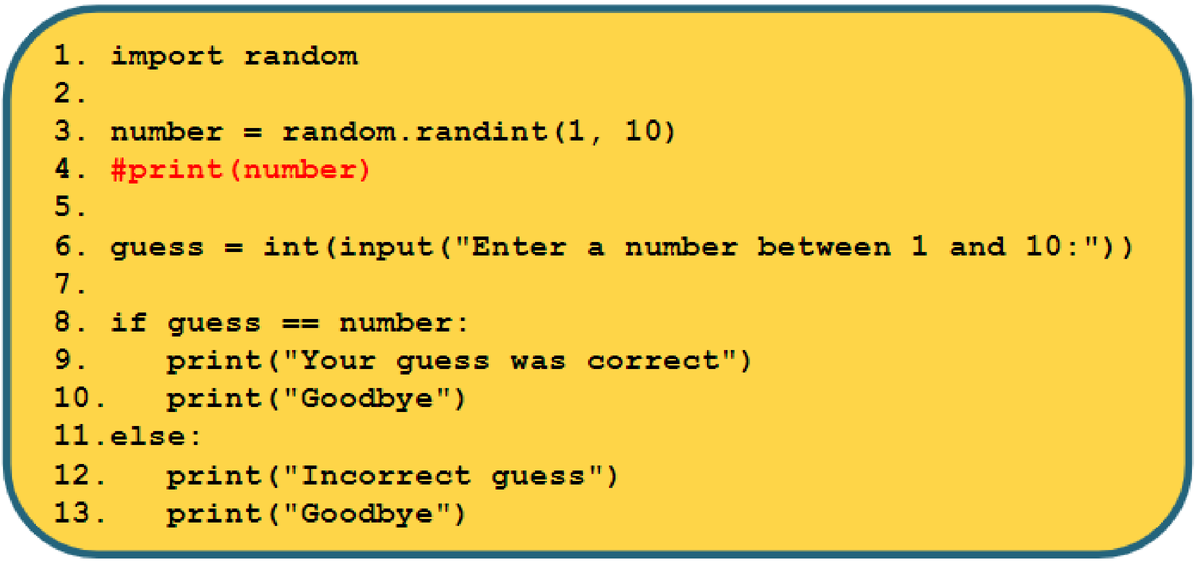
Develop the following code into a general solution to convert any binary number to decimal.



```
# ... convert binary 10011 to decimal ...
# ... the initial number is a string
binary_number = "10011"
# index:         01234

units    = int(binary_number[4])*1
twos     = int(binary_number[3])*2
fours    = int(binary_number[2])*4
eights   = int(binary_number[1])*8
sixteens = int(binary_number[0])*16
decimal = units+twos+fours+eights+sixteens
```

Session 2: PRIMM and Curriculum Planning

PRIMM



```
1. import random
2.
3. number = random.randint(1, 10)
4. #print(number)
5.
6. guess = int(input("Enter a number between 1 and 10:"))
7.
8. if guess == number:
9.     print("Your guess was correct")
10.    print("Goodbye")
11.else:
12.    print("Incorrect guess")
13.    print("Goodbye")
```

Predict: Discuss in pairs. What do you think the above program will do? Be precise. Be succinct.

Run: Download the program / Key it in. Execute the program. Test your prediction. Were you correct?

Investigate:

1. Uncomment line 4. What happens?
2. What is the purpose of line 4?
3. What would happen if you removed `int` from line 6?
4. Try changing == to != on line 8. What happens?
5. What if == was changed to = ?
6. What would happen if you don't enter an integer?
7. Try removing a bracket (anywhere). What happens?
8. Annotate each line of the program.

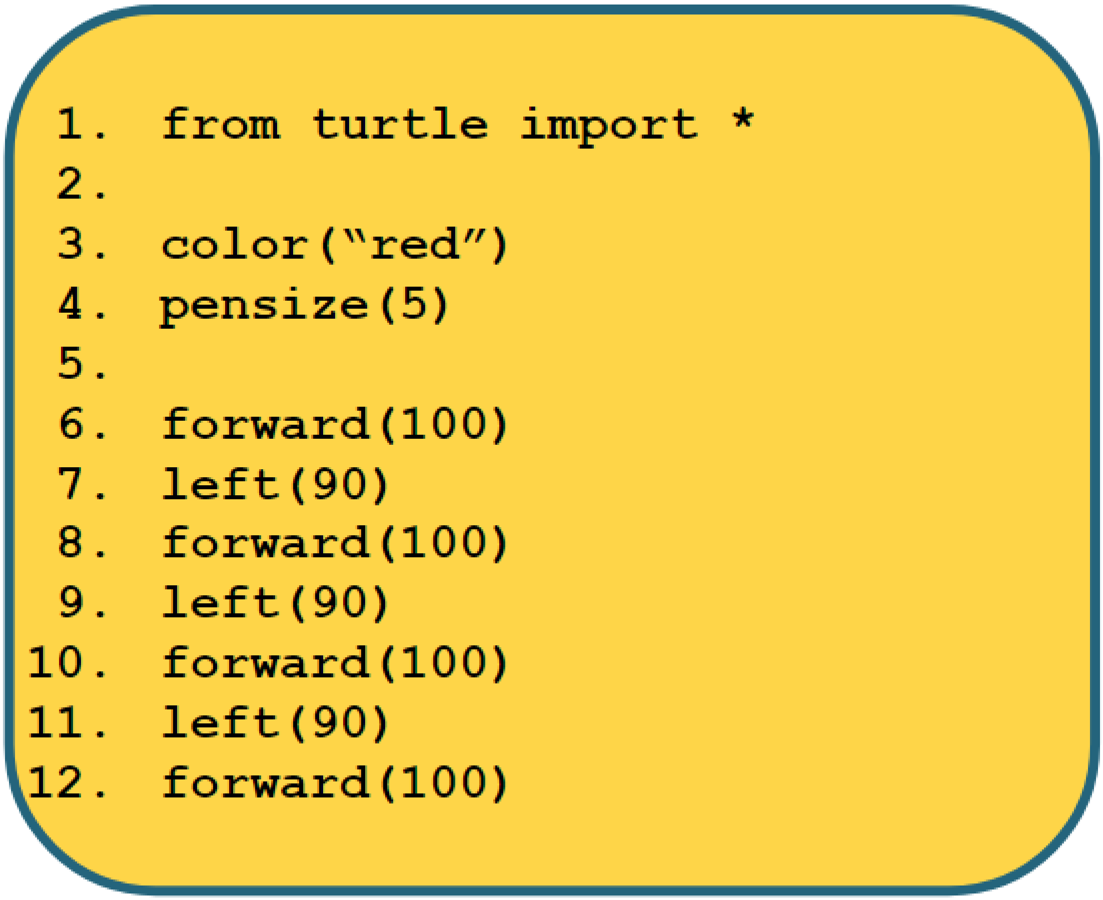
Modify:

1. Change the program so that it generates a number between 1 and 100? Can you be sure?
2. Change the program so that there is only one `print("Goodbye")` statement (without altering the logic)
3. Extend the program so that it tells the user if the number entered was *too high* or *too low*
4. Design an algorithm based on the program that would give the user 3 guesses
5. Get the computer to generate 4 numbers (lotto) OR ask the user how many numbers to generate?

Make:

Write a program that generates two numbers and prompts the user to enter their product

Task 1: Turtle Graphics



```
from turtle import *

color(“red”)
pensize(5)

forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
```

Investigate

Modify

Make

Task 2: Temperature Conversion

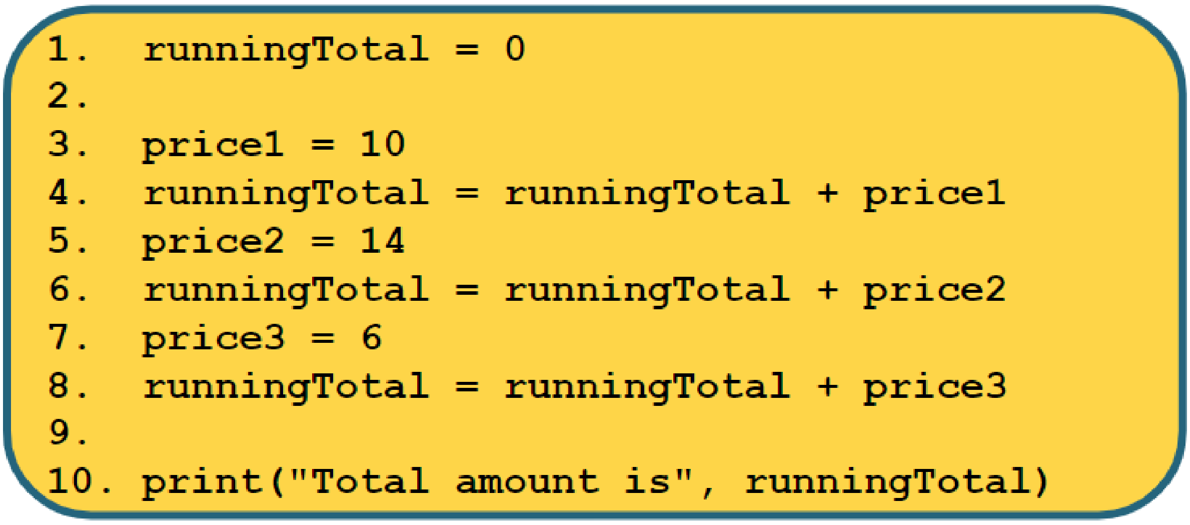
```
centigrade = float(input("Enter the Centigrade value: "))
fahrenheit = 9/5 * centigrade + 32
print(centigrade, "degrees C equals", fahrenheit, "degrees F")
```

Investigate

Modify

Make

Task 3: Running Total



```
runningTotal = 0

price1 = 10
runningTotal = runningTotal + price1
price2 = 14
runningTotal = runningTotal + price2
price3 = 6
runningTotal = runningTotal + price3

print("Total amount is", runningTotal)
```

Investigate

Modify

Make

Task 4: Average Height Calculator

```python
print("Average height calculator")
print("=========================")

h1 = int(input("Enter first height (cm): "))
h2 = int(input("Enter second height (cm): "))
h3 = int(input("Enter third height (cm): "))
h4 = int(input("Enter fourth height (cm): "))
h5 = int(input("Enter fifth height (cm): "))

avgHeigth = (h1+h2+h3+h4+h5)/5

print("The average height is ", avgHeigth, "cm")
```

Investigate

Modify

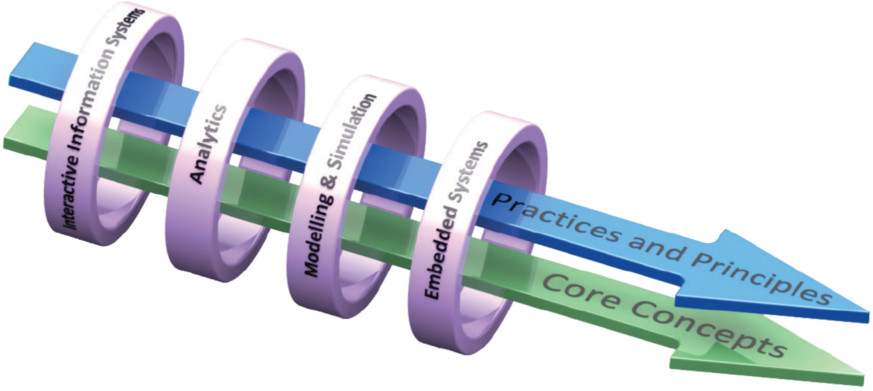
Make

Curriculum Planning

'Learning outcomes can best be defined as statements of what a learner knows, understands and is able to do after completion of learning.' CEDEFOP (2009)

How might you work with the learning outcomes?

What order might you teach them in?

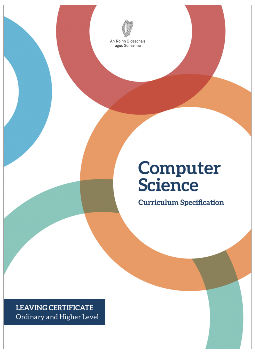


What about repeating LOs / linking to other parts of the course?

How might students demonstrate they have achieved the learning outcomes?

What content or resources might you need?

Group activity



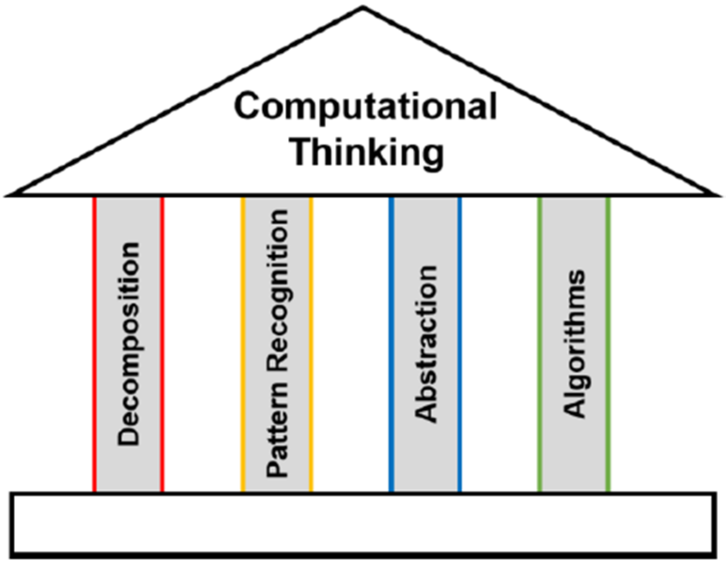
Use the LCCS specification, consider the following question:
How do you intend to approach LCCS in your classroom
(over the next 4 weeks/until mid-term/until Christmas)?

In your groups, consider:
Timeframe / Topics / LOs / Resources / Assessment /
Build up to ALTs / ALTs / Equipment etc.

Nominate:
A notetaker to summarise your group's work
A spokesperson to provide feedback

Key Message: Explore and teach the LOs through the lens of ALTs. There are several ways to achieve this.

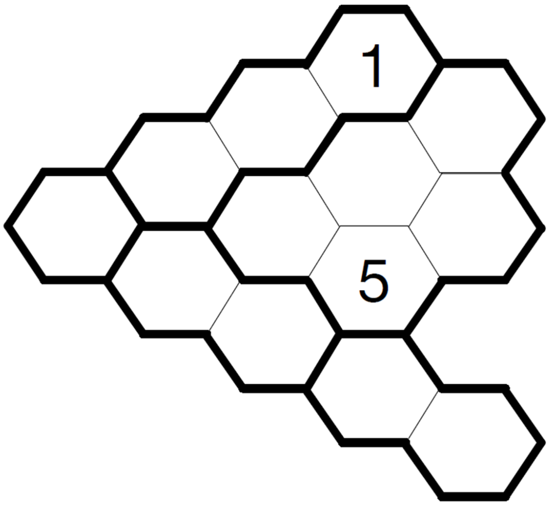
Session 3: Computational Thinking II



1. What is Computational Thinking?
2. Why is Computational Thinking important?

Applying computational thinking skills

Challenge: complete the cut hive puzzle below



Devise an algorithm to swap the white and black pieces. Pieces can move either by sliding into an adjacent empty square, or by jumping a single adjacent piece into the empty square immediately beyond. List the necessary steps.

○	○	○		●	●	●

●	●	●		○	○	○

Teaching Computational Thinking Concepts

Scenario 1: Storytelling

One pedagogy for teaching computational thinking concepts is storytelling. The aim of this part of the workshop is to demonstrate how storytelling can be used as a pedagogy to teach CT concepts. This resource is based on materials developed by Computing At Schools (CAS) UK and available from the Teaching London Computing website[1].

The task is to design a presentation on selected CT concepts based on the activity.

Introduction

Locked-in syndrome is a medical condition that can leave a person's intelligent mind locked inside a useless body, able to sense everything but unable to communicate.

As there is no cure for locked-in syndrome there isn't a lot that medics can do beyond making their patients comfortable. Or is there? (How could computational thinking skills be used?)

'The Diving Bell and the Butterfly' is the autobiography of Jean-Dominique Bauby, written after he woke up in a hospital bed totally paralysed. In the book, he describes life with locked-in syndrome.

Put yourself in his position after waking up in the hospital bed. How could you communicate? How could you write a whole book? You have only a helper with a pen and paper to write down your 'words'? All you can do is blink one eye. You can't move in any other way. That means you can't speak. Use the space provided to suggest a way to communicate?

[1] https://teachinglondoncomputing.org/free-workshops/computational-thinking-searching-to-speak/

[An initial idea might be to use morse code (short and long blinks) or map 1 blink for an A, 2 blinks for a B etc. The helper counts the blinks and writes down the letter. This is a core aspect of computational thinking called *algorithmic thinking*:]

Algorithmic thinking is the type of thinking that leads to algorithms. An *algorithm*[2] is a list of rules to follow in order to solve a problem.

A *protocol* is a special type of algorithm used to pass information between two people or computers. Because the algorithm in this example is used to pass information between Bauby and his helper, it is an example of a protocol. This algorithm has two parts – one for Bauby to follow and one for the helper.

Algorithms are a way of capturing intelligence and sharing it with others. Once developed, the necessary intelligence to solve a problem is encoded in the algorithm. The user(s) of the algorithm (in this case Bauby and the helper) need not have any understanding of what they are doing – they just need to be able to follow the instructions (e.g. count the blinks and write down the letter)

Key Point

Algorithmic thinking results in a general solution (in the form of steps) as opposed to a single answer. The power of algorithms is that they provide us with general solutions to problems.

[2] For an excellent introduction to algorithms watch the BBC4 documentary entitled *The Secret Rules of Modern Living: Algorithms* at https://www.youtube.com/watch?v=kiFfp-HAu64

Improving our initial algorithm

So far we have a way to map blinks to letters – what about spaces, what about punctuation marks, capital letters? Another important aspect of Computational Thinking (and algorithmic thinking) is *evaluation* i.e. testing (checking the details – what happens if the person blinks by mistake?) and thinking of ways to improve existing solutions.

Use the space provided to suggest improvements to turn blinks into letters involving the helper being the first part of the protocol.

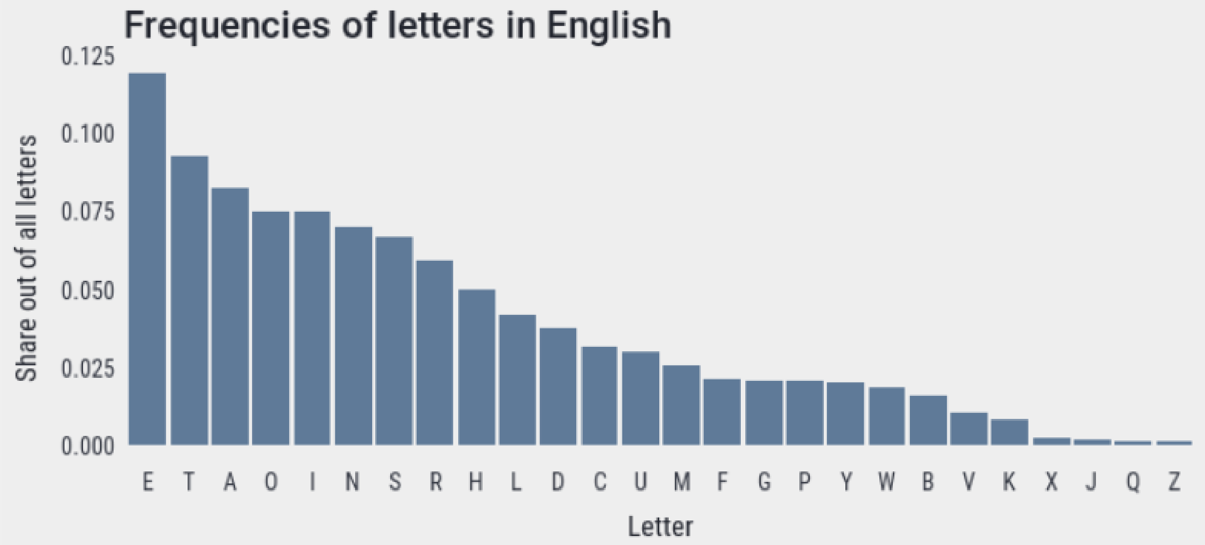
Examples might be

- Bauby blinks when the letter he is thinking of is read aloud by the helper (who starts at A each time). Try this with your neighbour. Communicate your initials. What are the best/worst case scenarios?
- Helper can guess the word (how can we use *pattern matching* to generalise this approach to invent predictive texting? Often problems turn out to be essentially the same as something you've already seen/solved in a different situation – pattern matching is the skill of spotting that a new situation is essentially the same as something you've seen before – essentially we are searching for a letter – this is the general form of the problem we are trying to solve).

Frequency Analysis

Frequency analysis has been used to crack secret codes throughout history. One example comes from the movie, 'The imitation game' in which a machine was programmed to decode words based on other words already known to exist in certain messages.

How might the frequency analysis of letters shown below be used to improve the algorithm in this story?



Aside: What CT skills do you think would be required in order to produce the above chart?

Using abstraction as a tool to evaluate our solution

Is the ‘new improved’ algorithm better? (What do we mean by better?) How fast is it?

Use the space below to compare the ‘speed’ of the original algorithm with the one based on frequency analysis.

Hints:

1. Use abstraction to ignore any unnecessary details
2. Consider the best and worst case scenarios

Describe how you used the concept of abstraction to help?

It is possible to reduce the worst case to five questions per letter - read on to see how!

Decomposition (Divide and Conquer)

Let's start with a guessing game.

Guess the number I am thinking of – between 1 and 100 - in 20 questions or less

A good strategy is to ask questions that rule out half the numbers each time.

e.g. Is it greater than 50? Yes

e.g. Is it greater than 75? No

e.g. Is it greater than 62? No

e.g. Is it greater than 56? Yes

etc

Analysis

Let's say the number I was thinking of was between 1 and 1,000,000. What would be the minimum number of questions to narrow it down to one number?

Half the numbers can be ruled out with each question. After one question, we are down to 500,000 numbers left, two questions 250,000, then 125,000, then about 64,000 possibilities (simplifying a little to make the numbers easier!), 32,000, 16,000, 8000, 4000, 2000, 1000… After 10 questions there are only 1000 numbers left out of the original million it could be. Keep going…500 left after another question, 250, 125, 64 (ish) 32, 16, 8, 4, 2 and on the 20th question there is only one possibility left. So with the right questions, in the worst case it takes only **twenty** questions to find the number.

Explain how the divide-and-conquer technique just described can be transferred to the context of this story i.e. to find one of the 26 letters of the alphabet.

Draw a decision tree showing questions to ask to get any letter of the alphabet in five questions or less.

Use the decision tree to design an encoding system for unique codes (made up of 1s and 0s) for each letter

Understanding People (as part of computational thinking)

Computational thinking is about solving problems for people. People therefore come first. You have to understand the problem you are solving from their point of view, before you dream up solutions. Otherwise your great technical solution could be useless. To be a great computer scientist, you have to understand people.

Perhaps we should have started with the person. Were we counting the right thing? As our measure of work – our 'abstraction' – we used the number of questions asked. That is the job of the helper and it may be tedious but it's not difficult. What if blinking was a great effort for Bauby. His solution involved him blinking only once per letter. Our divide-and-conquer algorithm requires him to blink five times. Multiply that by a whole book. We could have made it five times harder.

It could be that blinking is easy and our algorithm is better. We don't know the answer, because we didn't ask the question. We should have asked first. We should have started with the person. Furthermore, Bauby's original solution is easy for anyone to walk in and understand. Ours is more complex to follow and might need some explaining before the visitor understands and Bauby is not going to be the one to do the explaining. Thinking about people is important!

Final question

Could technology be used to implement the algorithm? If so, how and what should be taken into account?

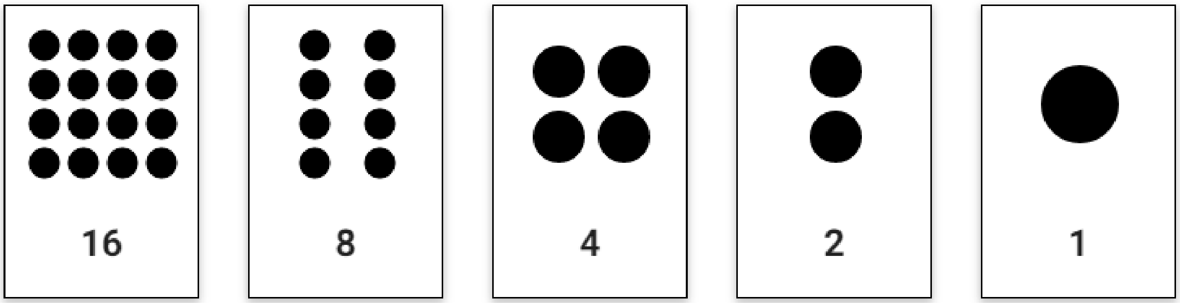
Reflection

Reflect on how the computational thinking concepts listed below were demonstrated in the story. Incorporate these concepts into your design.

Term	Example
Algorithm	
Abstraction	
Decomposition	
Pattern Recognition and Generalisation	
Evaluation	

Scenario 2: A Kinaesthetic Activity - Binary Numbers

By using five individual cards as illustrated below, devise a kinaesthetic classroom activity that would enable students to learn about the binary number system.



List some questions that could be asked during the activity.
e.g. What cards would need to be turned face down so that the (decimal) number 13 is displayed?

What other concepts could this activity be used to teach?
e.g. Other number systems?

Describe how the activity could be used to explore the computational thinking concept of **abstraction** at an introductory level, e.g. how do we represent binary digits?

Describe how the activity could be used to explore the computational thinking concept of **decomposition** at an introductory level, e.g. what are the individual pieces that make up the whole number?

Describe how the activity could be used to explore the computational thinking concept of **pattern recognition** at an introductory level, e.g. what happens to a binary number when we append a 0? A 1?

Describe how the activity could be used to explore the computational thinking concept of **logic** at an introductory level, e.g. what happens when we double a binary number

Describe how the activity could be used to explore the computational thinking concept of **algorithmic thinking** e.g. write a sequence of steps to convert from a binary number to a decimal number. What if each binary digit was being entered on the fly?

Describe how the activity could be used to explore the computational thinking concept of **evaluation,** e.g. how many different values can be represented by a given number of bits? Are there any limits?

Describe how the activity could be used to explore the computational thinking concept of **data representation,** e.g. how can letters (and other symbols) be represented in binary (and other number systems)?

What extensions to this activity do you think would be appropriate for LCCS? e.g. consider how a machine based on an imaginary binary currency (i.e. valid coins are 1, 2, 4, 8, and 16 units) might work. Give examples.

Scenario 3: Role Play

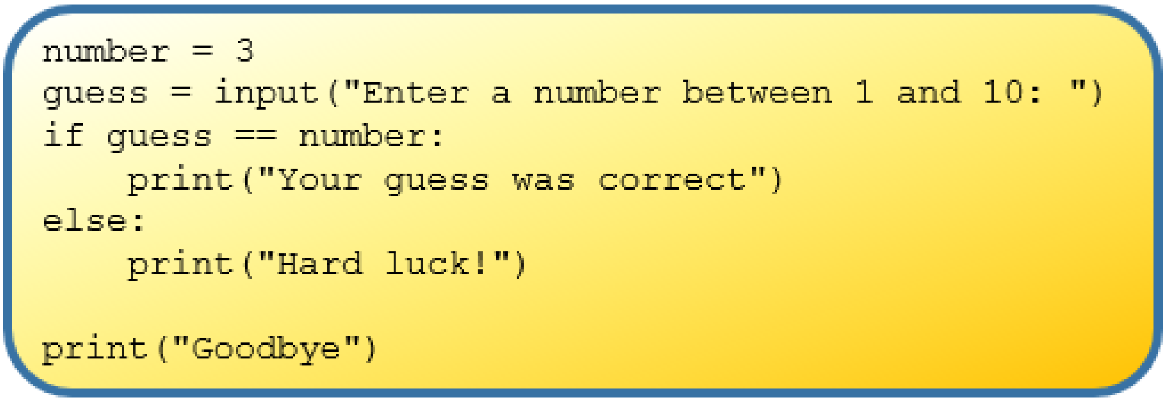
Browse to

https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-brain-in-a-bag-activity/ and watch the video entitled ‘brain in a bag’.

Describe how the computational thinking concepts of abstraction, decomposition and algorithmic thinking are addressed in the video.

Identify the any programming constructs addressed in the video.

Using ideas you have gleaned from the video create the program below[3] using people to represent instructions and the resources provided (string, baton etc).



```
number = 3
guess = input("Enter a number between 1 and 10: ")
if guess == number:
    print("Your guess was correct")
else:
    print("Hard luck!")

print("Goodbye")
```

Suggest ways in which this activity could be extended to re-inforce computational thinking concepts and programming constructs.

[3] From Python Skills Workshop, PDST, May 2018

Session 4: Introduction to ALT 4

3.11 use and control digital inputs and outputs within an embedded system

3.12 measure and store data returned from an analogue input

3.13 develop a program that utilises digital and analogue inputs

3.14 design automated applications using embedded systems

Think, Pair, Share

1. What are the uses of Embedded Systems?

2. What is the difference between digital and analogue data?

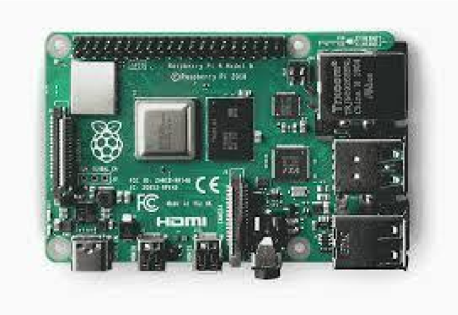
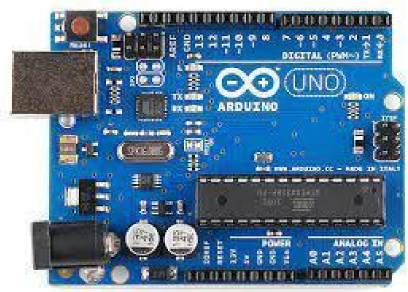
Use the space below to make notes of additional information relating to embedded systems.

Matching Exercise

Embedded System:	A compact integrated circuit (IC) that contains a processor core, memory, and input/output peripherals. It is specifically designed to execute control tasks and interact with other devices in an embedded system.
Microprocessor	A device that detects and measures physical quantities or environmental conditions, such as temperature, pressure, or light. It converts the measured parameter into an electrical signal that can be processed by an embedded system for further analysis or control.
Sensor	Refers to a computer system designed to perform specific functions within a larger mechanical or electronic system. It typically consists of a combination of hardware and software components, tailored to meet the requirements of the intended application
Microcontroller	Data or a signal used to represent data as discrete values typically expressed as binary digits (0s and 1s).
Digital	Refers to a representation or manipulation of data using continuous signals or values. These signals are typically used to measure or represent physical quantities that vary smoothly over a range of values, such as voltage or sound.
Analogue	A central processing unit (CPU) often referred to as the brain of a computer system. It is responsible for executing instructions and performing calculations. They are not inherently embedded systems but can be used in their design.

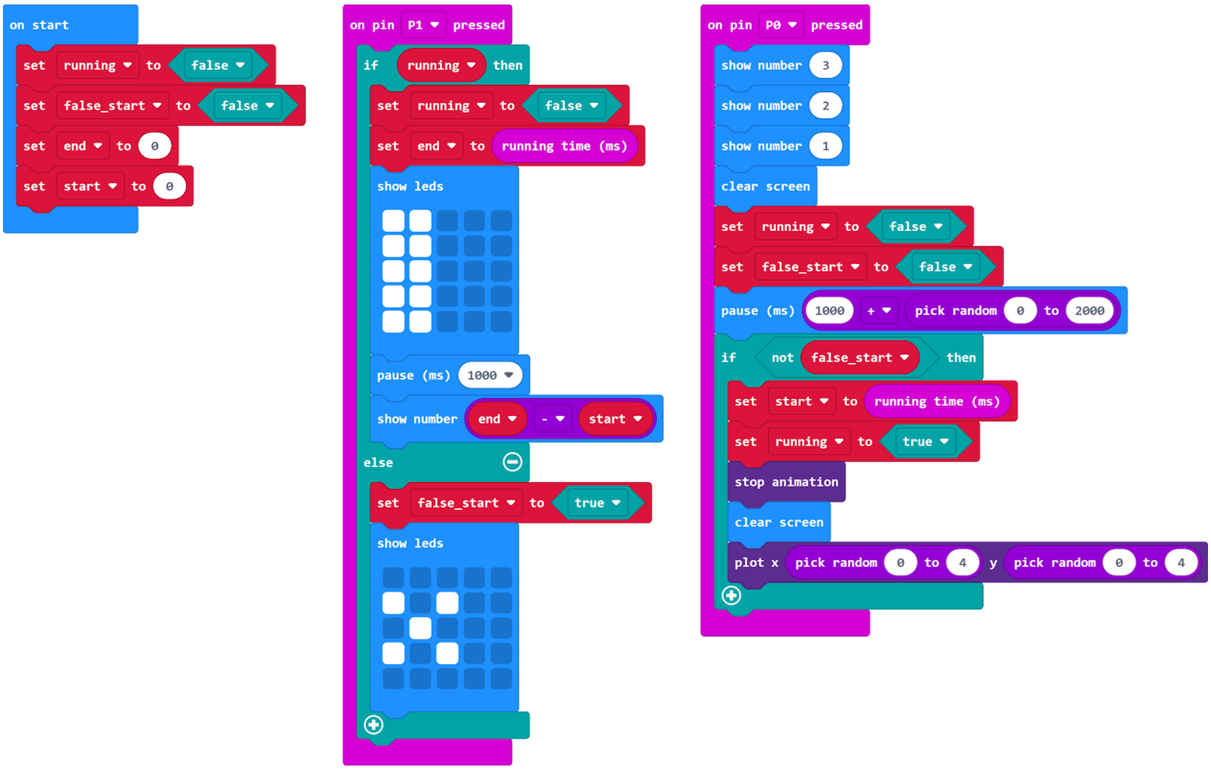
Embedded Systems

Raspberry Pi, Arduino, Micro: Bit

A **Raspberry Pi** is a low-cost, credit card-sized single-board computer that combines the processing power of a microprocessor and is capable of doing everything a desktop computer can do. It plugs into a computer monitor or TV, and uses a standard keyboard and mouse. Thonny is built in to Raspberry Pi OS. The **Raspberry Pi Pico** is a low-cost, high-performance microcontroller board, controlled via USB. It has an array of GPIO pins for connecting peripherals and is programmable via Thonny in micropython or circuitpython.	Raspberry Pi Foundation  https://www.raspberrypi.org/
An **Arduino** is an open-source platform used for building electronics projects. It consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. It is not a full-fledged computer like a Raspberry Pi. Programmed in C using the Arduino IDE	ARDUINO  https://www.arduino.cc/
A **Micro: bit** is a pocket-sized programmable device featuring an embedded microcontroller, sensors, and LEDs. It is an embedded system built around a microcontroller, designed for educational purposes.	 https://microbit.org/

The Micro:Bit

Code for Reaction Game Demonstration



```
on start
    set running to false
    set false_start to false
    set end to 0
    set start to 0
```

```
on pin P1 pressed
    if running then
        set running to false
        set end to running time (ms)
        show leds
            # # . . .
            # # . . .
            # # . . .
            # # . . .
            # # . . .
        pause (ms) 1000
        show number end - start
    else
        set false_start to true
        show leds
            . . . . .
            # . # . .
            . # . . .
            # . # . .
            . . . . .
```

```
on pin P0 pressed
    show number 3
    show number 2
    show number 1
    clear screen
    set running to false
    set false_start to false
    pause (ms) 1000 + pick random 0 to 2000
    if not false_start then
        set start to running time (ms)
        set running to true
        stop animation
        clear screen
        plot x pick random 0 to 4 y pick random 0 to 4
```

What learning outcomes are being experienced by this activity?

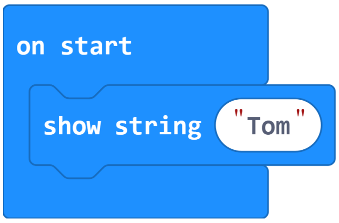
Identify the variables in the system. What are their names and their purpose?

Variable name	Purpose

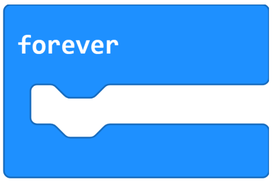
List some further ideas to modify or improve this game:

Example 1: Name Badge

Drag the **show string** block (Basic) into the program workspace and change the string so that it will display your own name.



```
on start
  show string "Tom"
```

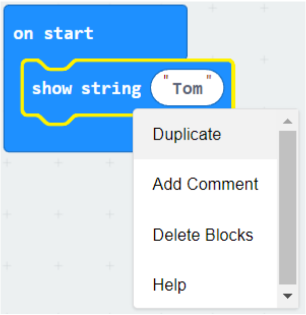


```
forever
```

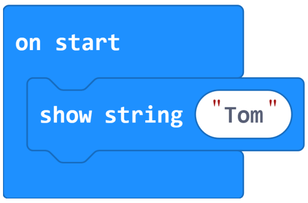
What if we wanted it to scroll your name continuously? Let's place a duplicate of the **show string** block inside the forever container.

Hover over the **show string** block and click duplicate.

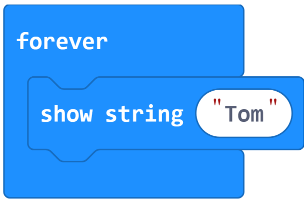
Snap the duplicated block to the forever container



```
on start
  show string "Tom"
```



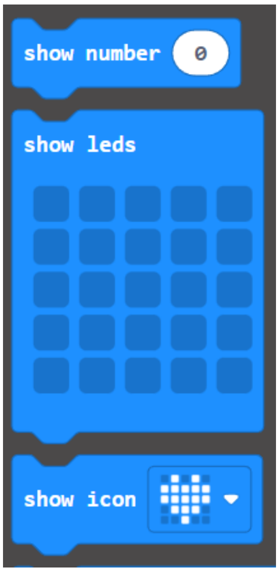
```
on start
  show string "Tom"
```



```
forever
  show string "Tom"
```

Is the **on start** block really necessary? Discard the **on start** block by dragging it off the workspace (don't forget you can select Ctrl-Z to undo). The final solution is shown.

Experiment with the following show commands



```
show number 0
show leds
show icon
```

What have you learned?

Devise some extension activities:

Example 2: Binary Number Converter

This would be a good example to use with students to follow a binary unplugged activity. (You will need to understand binary number representation for this example.)

The purpose of the program is to convert from binary to decimal.

As each binary digit is entered the decimal equivalent is calculated.

The abstractions are:

- button A represents a 1 and
- button B represents a 0

The table shown here on the right illustrates the first ten binary numbers and their decimal equivalents.

Binary	Decimal
0_2	0
1_2	1
10_2	2
11_2	3
100_2	4
101_2	5
110_2	6
111_2	7
1000_2	8
1001_2	9

Use your knowledge of decimal and binary numbers to fill in the binary values in the table below. The even numbers are on the left and the odd numbers are on the right.

Binary	Decimal	Binary	Decimal
	0		1
	2		3
	4		5
	6		7
	8		9
	10		11
	12		13
	14		15

Do you notice any patterns?

How does each new binary digit impact the magnitude of the decimal number? Only two possibilities exist for each digit; 0 or 1.

Adding a 0

Old Binary	Old Decimal	New Binary	New Decimal
1		10	
10		100	
11		110	

What happens each time a 0 is added?

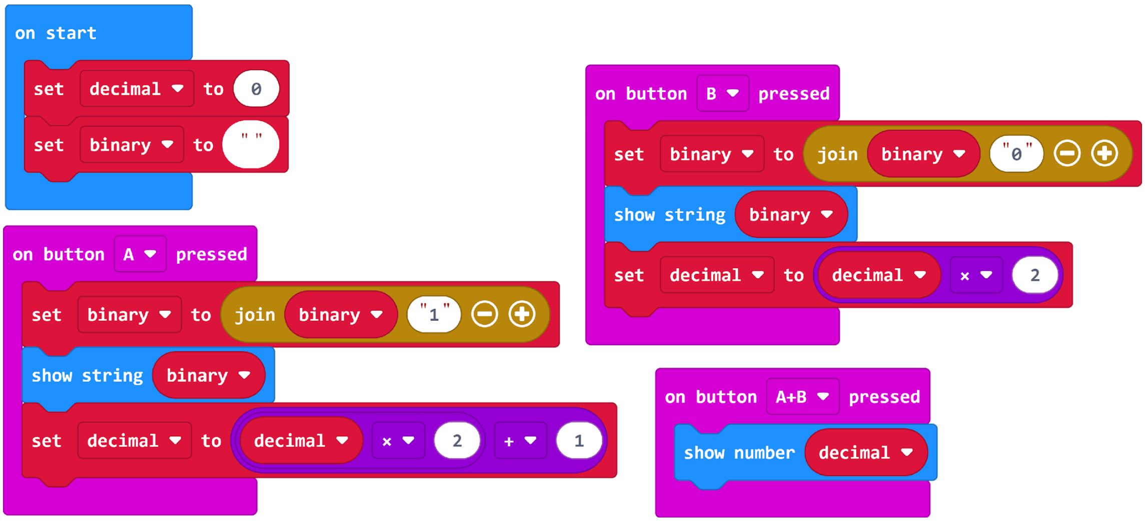
Adding a 1

Old Binary	Old Decimal	New Binary	New Decimal
1		11	
10		101	
11		111	

What happens each time a 1 is added?

It should be evident that each time a 0 is added the decimal value doubles. The effect of appending the digit 1 is to double the decimal value plus one.

The solution, shown here, exploits these two patterns:



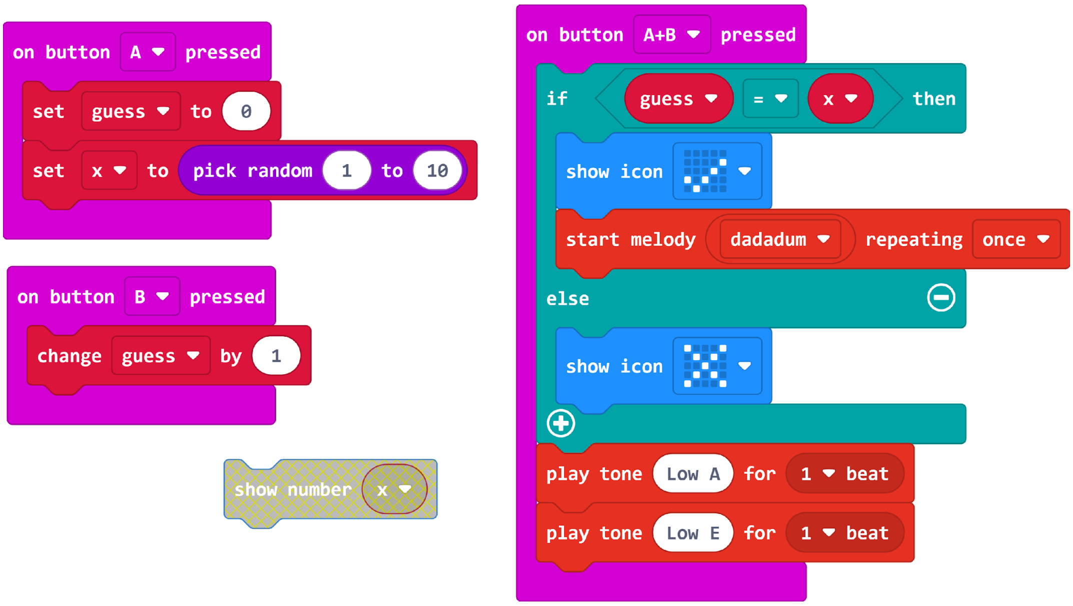
```
on start
    set decimal to 0
    set binary to " "

on button A pressed
    set binary to join binary "1"
    show string binary
    set decimal to decimal × 2 + 1

on button B pressed
    set binary to join binary "0"
    show string binary
    set decimal to decimal × 2

on button A+B pressed
    show number decimal
```

- Try it out yourself! Download the program and run it.
- Make some predictions – devise some use cases.
- Use the following questions to scaffold your learning.

Identify and explain the purpose of each variable. (Hint: Look for the set command.)
What are the program inputs and outputs?
What does the join command do? (Hint: use the Help facility.)
What are the limitations of the program? Can you suggest any improvements that could be made and if so how?

Example 3: Guess the Number

Study the program carefully and answer the questions that follow.



```
on button A pressed
    set guess to 0
    set x to pick random 1 to 10

on button B pressed
    change guess by 1

show number x

on button A+B pressed
    if guess = x then
        show icon
        start melody dadadum repeating once
    else
        show icon
    play tone Low A for 1 beat
    play tone Low E for 1 beat
```

What do you think would happen when button A is pressed? What about button B?

Predict what you think would happen when button A+B is pressed?

Download and run the program. Were your predictions correct?

The code block shows number x is *hatched out*. The hatch pattern indicates that the block will not be executed at runtime i.e. it is not part of the *runtime system*.

Snap the hatched block to a place which would make testing this program easier? Explain your choice.

Explain how the pick random command works
How does the if statement in this program work? Explain the relationship between the then and else part of an if statement? Are both parts always necessary?
What can you deduce from this program about the purpose of an if statement as a programming construct?
Outline three everyday examples of decisions that can result in only one of two values – true or false
Programmers need to recognise situations when to use an if statement. Based on your experience as an end-user can you identify any scenarios where software you have used would have needed an if statement in order to work?

Exercise

Looking back at the examples, identify as many specific examples of the following programming constructs/concepts used as you can. Record your answers on the next page.

Sequence – the flow of program execution from one line to the next.

Variables – an identifier used to reference a storage location for a value.

Assignment – a technique used to change the value of a variable.

Datatype – the type of data that represents a value e.g. number, string, Boolean.

Expression – code that evaluates to a value. (A value can be thought of as a literal value (e.g. 5, “Hello”, true) or an expression.).

Arithmetic Expression – code that evaluates to a numeric value (typically involving the use of arithmetic operators i.e. +, −. *, ÷, ** and sometimes used in conjunction with any of the *Math* commands e.g. remainder, min, max, absolute, pick random etc .

Boolean Expression – code that evaluates to either true or false (typically involving the use of simple relational operators i.e. =, ≠. >, ≥, <, ≤ and sometimes involving the use of compound relational operators (i.e. and, or, not).

Selection Statement *(aka decision)* – the use of if/if-else statement to select which block of code to execute.

Loop *(aka iteration/repetition)* – the use of forever/repeat/while/for to execute a block of code zero or more times.

Input – anywhere data is introduced to a program from an external source (e.g. user, sensor) through use of any of the *Input* commands such as, on button A/B/A+B pressed, on shake etc. or any of the *Input* variables e.g. temperature, acceleration, light level etc.

Output – anywhere a program supplies data to the outside world e.g. show number, show icon, show leds, show string, plot, unplot etc.

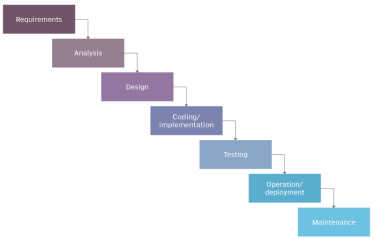
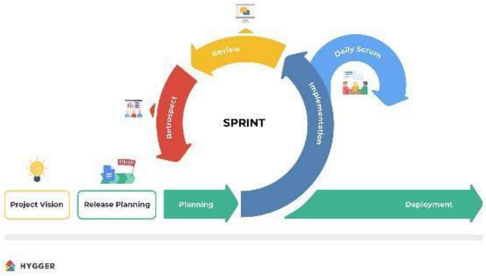
Example 1: Name Badge

Example 2: Binary Number Converter

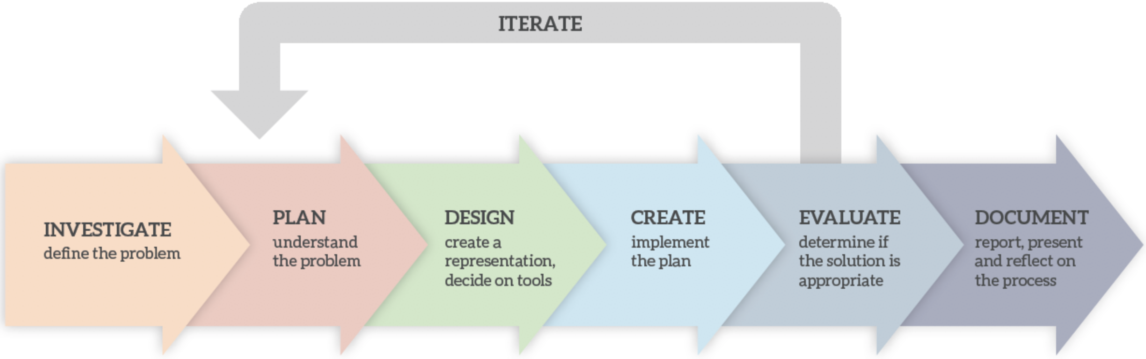
Example 3: Guess the Number

Design Methodologies

What are the key differences between agile and waterfall software development methodologies?

Waterfall	Agile
 Requirements Analysis Design Coding/ implementation Testing Operation/ deployment Maintenance	 Review Daily Scrum Retrospect Implementation SPRINT Project Vision Release Planning Planning Deployment HYGGER

How might setting shorter development cycles or milestones within the design process promote a more iterative and agile approach, allowing for regular reflection and adjustments? Consider how you might adapt the diagram.


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

Session 5: Investigating and Planning ALT 4

ALT 4: Investigate

What is an embedded system? Give examples from the world around us.

What are sensors? Digital inputs/outputs? Analogue inputs/outputs?

What are your hobbies/interests/passions? Can you think of example embedded systems that might support these?

What about other examples – for users other than yourself e.g. family members, friends, school, community organisation, society?

Investigate

ALT 4: Plan

In your assigned groups, evaluate your potential ideas for ALT 4
Choose one idea for further development - dissect the idea
You may use the following prompt questions to help you:

Is there a broad theme or a specific topic?
Who is the audience?
What teaching & learning strategies could you use?
What does your project do?
Does your project idea cover all the LOs for this ALT?
What other LOs can be taught through the lens of this project?
What tools or materials are needed?
What are the roles in the group?
What research or upskilling do you need to do?

Plan

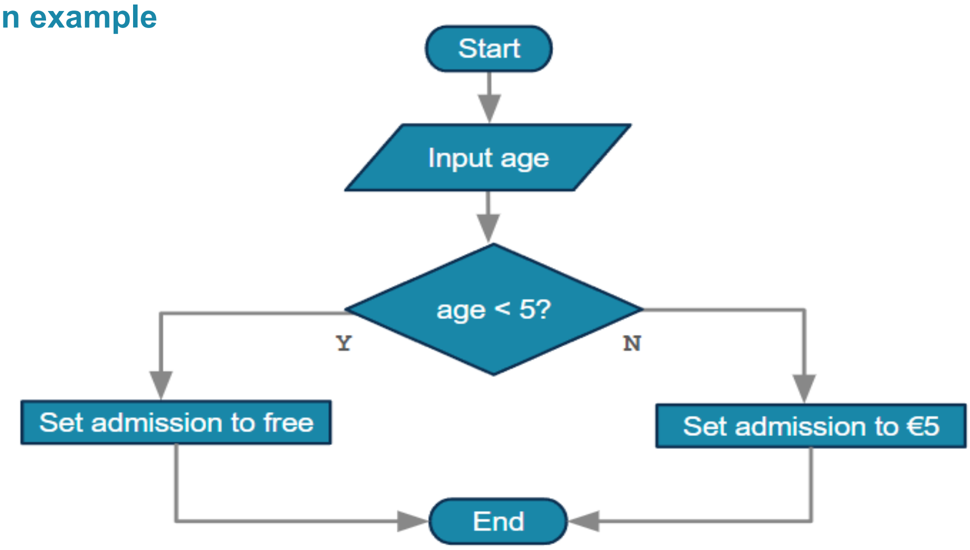
Session 6: Designing, Creating, Evaluating, and Documenting ALT4

ALT 4: Design

Flowcharts

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

Admission example



Pseudocode

- an artificial and informal language that helps programmers develop algorithms
- a plain language description of the steps in an algorithm
- allows us to express an algorithm without having to consider the specific syntax rules of a programming language

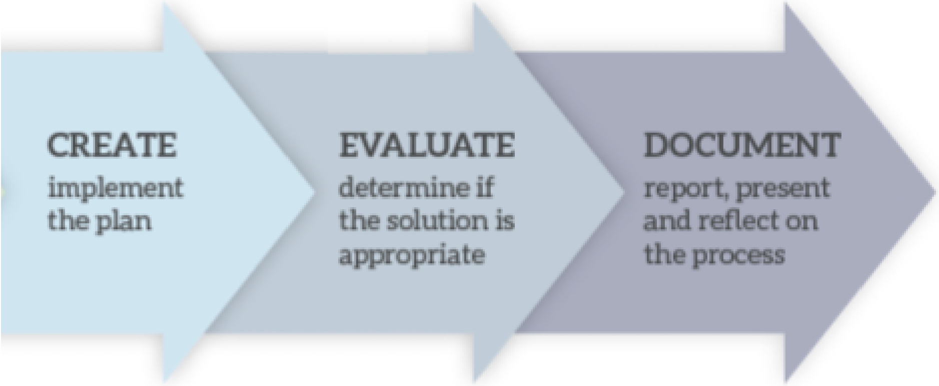
Golf example:

```
program start

check weather forecast

if rain predicted
    Stay home
else
    Go golfing
end if

program end
```

Task - develop a flowchart for the "*Should I play golf*?" example above

Design

ALT 4: Create



CREATE
implement the plan

EVALUATE
determine if the solution is appropriate

DOCUMENT
report, present and reflect on the process

Create

Evaluate

Document and Report

Notes

Notes