Implementing the Teach Computing Curriculum

Session 3: Making it fit to your school

National Centre for Computing Education

Session 3: Making it fit to your school

By the end of this session you will be able to:

- understand potential challenges and barriers and how you can overcome them
- tailor the curriculum to meet the needs of your school
- provide support and challenge through the TCC

The Journey

- how long have you been delivering the Teach Computing Curriculum
- is the Curriculum being delivered in every year group? How do you know
- are there any barriers to the Curriculum being delivered effectively, do you have an allocated time for the subject to be delivered
- how confident are teachers in the delivery of the subject

Identifying the challenges..

What are some of the reasons the Teach Computing Curriculum may not be taught effectively in every classroom?

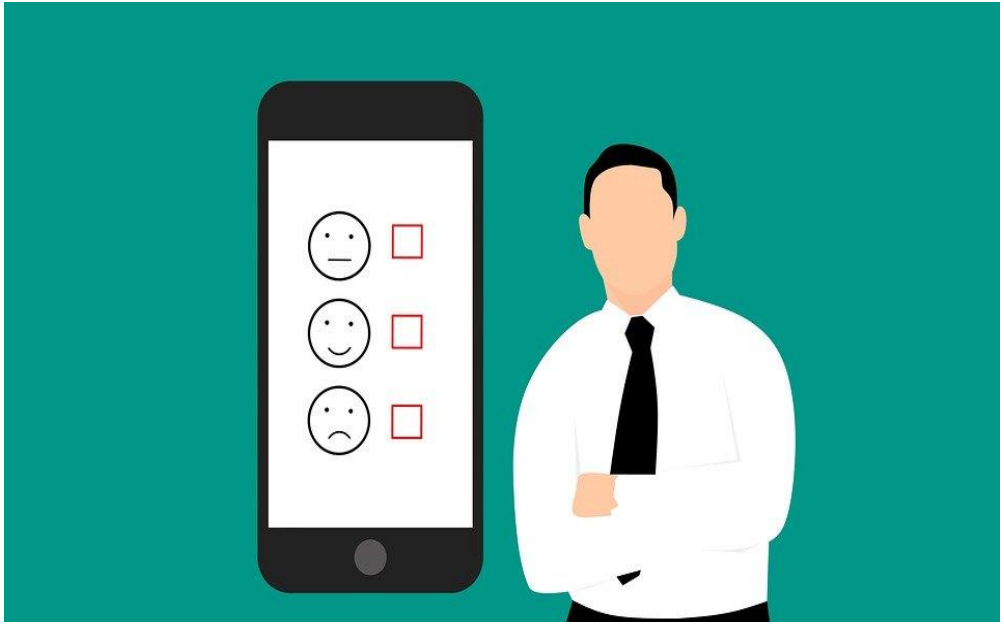
Use post it notes on your tables to list barriers/challenges

Barrier to learning: Staff Confidence

Do you know your confident/weaker members of staff? How?

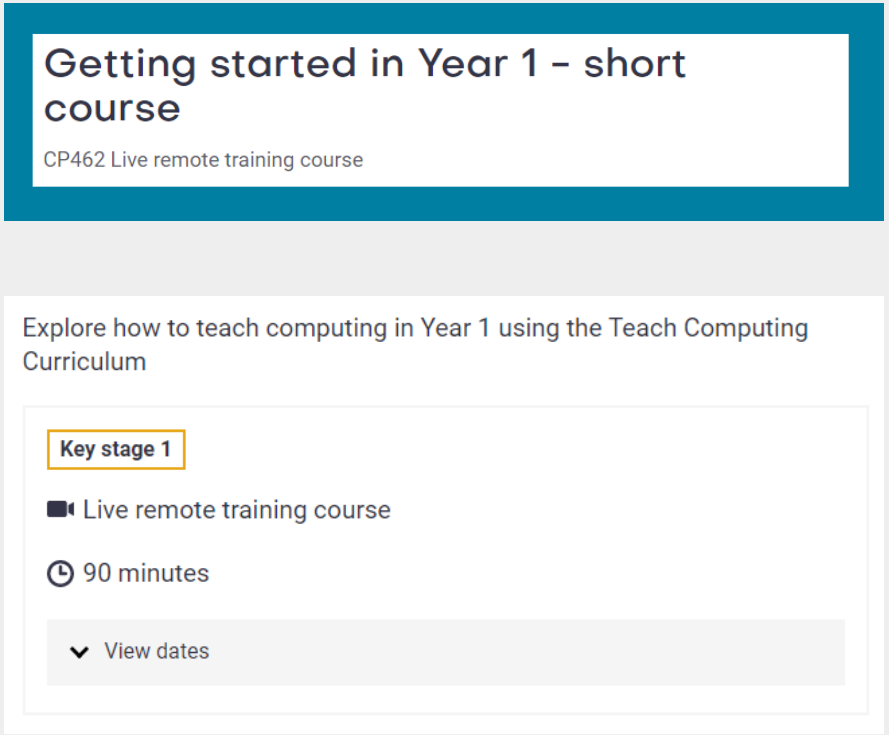
Have you completed staff audits? What have you used?

Have you completed any teacher voice?



Year group courses

Improve confidence of Teach Computing Curriculum lesson materials for the year that you teach.



Make it fit to your Curriculum Topics!

	Computing systems and networks[1]	Creating media	Programming A	Data and information	Creating media	Programming B
Year 1	Technology around us (1.1)*	Digital painting (1.2)	Moving a robot (1.3)	Grouping data (1.4)	Digital writing (1.5)	Programming animations (1.6)
Year 2	Information technology around us (2.1)	Digital photography (2.2)	Robot algorithms (2.3)	Pictograms (2.4)	Making music (2.5)	Programming quizzes (2.6)

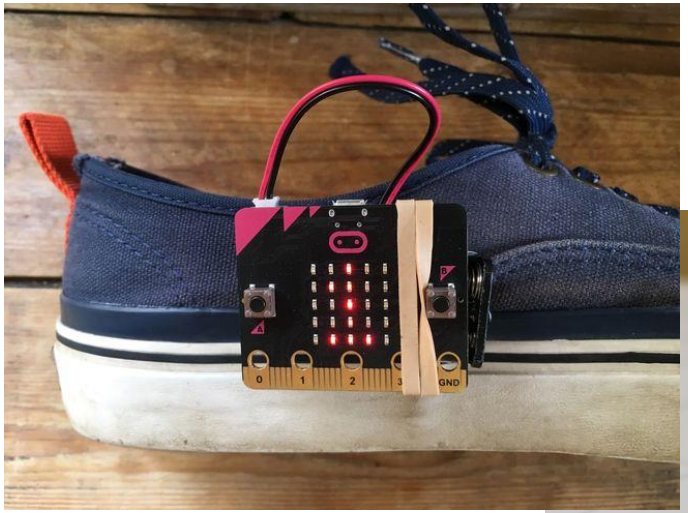
	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 3	Connecting computers (3.1)	Stop-frame animation (3.2)	Sequencing sounds (3.3)	Branching databases (3.4)	Desktop publishing (3.5)	Events and actions in programs (3.6)
Year 4	The internet (4.1)	Audio production (4.2)	Repetition in shapes (4.3)	Data logging (4.4)	Photo editing (4.5)	Repetition in games (4.6)
Year 5	Sharing information (5.1)	Video production (5.2)	Selection in physical computing (5.3)	Flat-file databases (5.4)	Vector drawing (5.5)	Selection in quizzes (5.6)
Year 6	Internet communication (6.1)	Webpage creation (6.2)	Variables in games (6.3)	Introduction to spreadsheets (6.4)	3D modelling (6.5)	Sensing (6.6)

Cross Curricular Opportunities and Links..

Computer Science

How computers and computer systems work & how they are designed and programmed

Foundations



Cross Curricular Opportunities and Links..

Information Technology

Looking at technology in our lives and the purposeful use of existing programs to develop products and solutions

Applications

Year 1- Painting, Writing and Grouping Data

Year 2- Photographs, Music and Pictograms

Year 3- Animation, Publishing and Branching databases

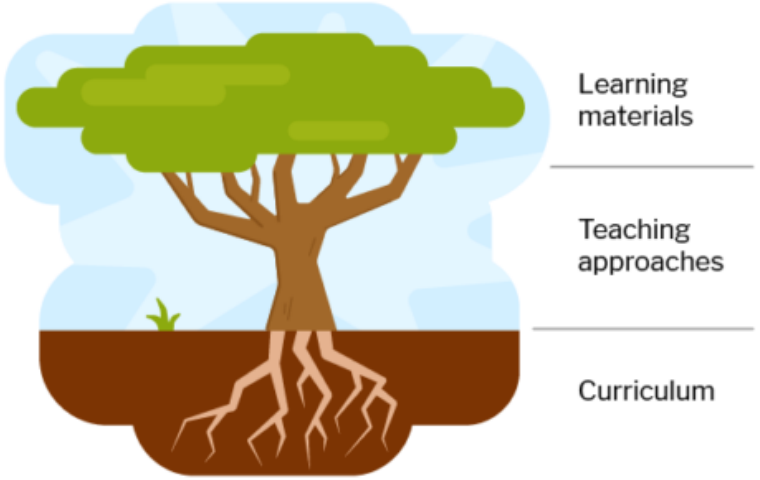
Year 4- Podcasting, Photo editing and Data Logging

Year 5- Vector drawing, video editing and databases

Year 6- 3D modelling, web page creation and spreadsheets

A Culturally Responsive Curriculum

- equitable and authentic learning experiences
- take into account the characteristics of all learners and their social environments



How could you modify the context of a unit to make it relevant to a diverse range of learners?

Examples:

Using a Wider Context: Year 3 The Internet- looking at the issue of fake news

Show diverse role models: Year 4 Audio Editing – produce a podcast on own choice of topic

Consider different perspectives: Year 6 Web Page Creation- design a web page with consideration of the user journey and needs

Making it work for your pupils!

The Curriculum is a starting point for your school!

Differentiation**:** ***Explorer Task*** and ***Scaffolding Opportunity***

Let’s look at the pedagogy behind the curriculum to help support this..

How we teach Computing?

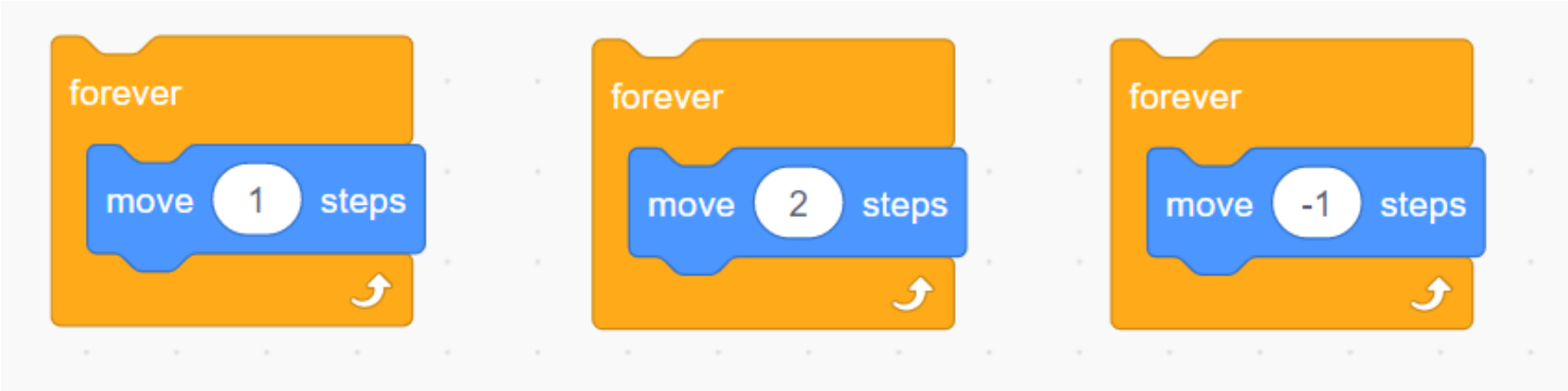
The work of the NCCE is underpinned by our 12 principles of Computing Pedagogy

Make Concrete!

Bring abstract concepts to life with **real-world, contextual examples** and a focus on interdependencies with **other curriculum subjects**:

- unplugged activities
- proposing analogies
- storytelling around concepts
- examples of the concepts in pupils' lives.

Make Concrete: Code



Read and Explore Code First!

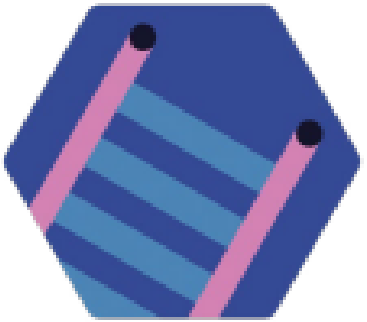
When teaching programming, **focus first on code ‘reading’ activities**, before code writing.

Research has shown that being able to read, trace, and explain code augments pupils’ ability to write code.

Read and Explore Code First!

Structure Lessons!

Use supportive frameworks when planning lessons, such as **PRIMM** (Predict, Run, Investigate, Modify, Make) and **Use-Modify-Create**. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.



Predict:

- focus on the function of the code
- encourage discussion
- work in pairs or threes

Run:

- provide students with working code to run
- learners check against predictions

Investigate:

- use a variety of activities, such as tracing, annotating, and questioning
- encourage students to discuss and work with the code in pairs or small groups

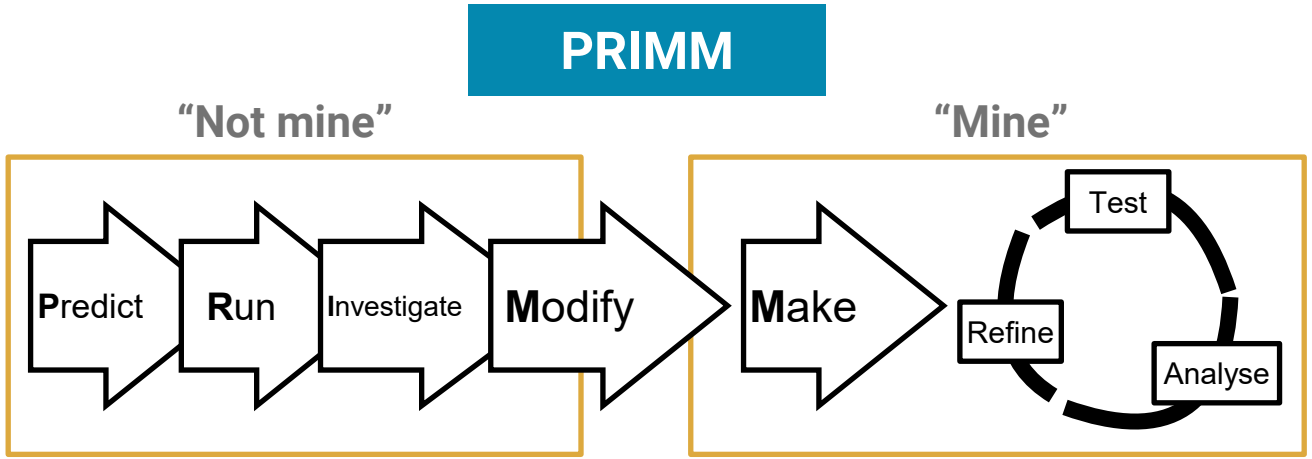
Modify:

- modify code in small steps to add new functionality
- apply what has been learnt about the structure of the code
- gradual increase in difficulty

Make:

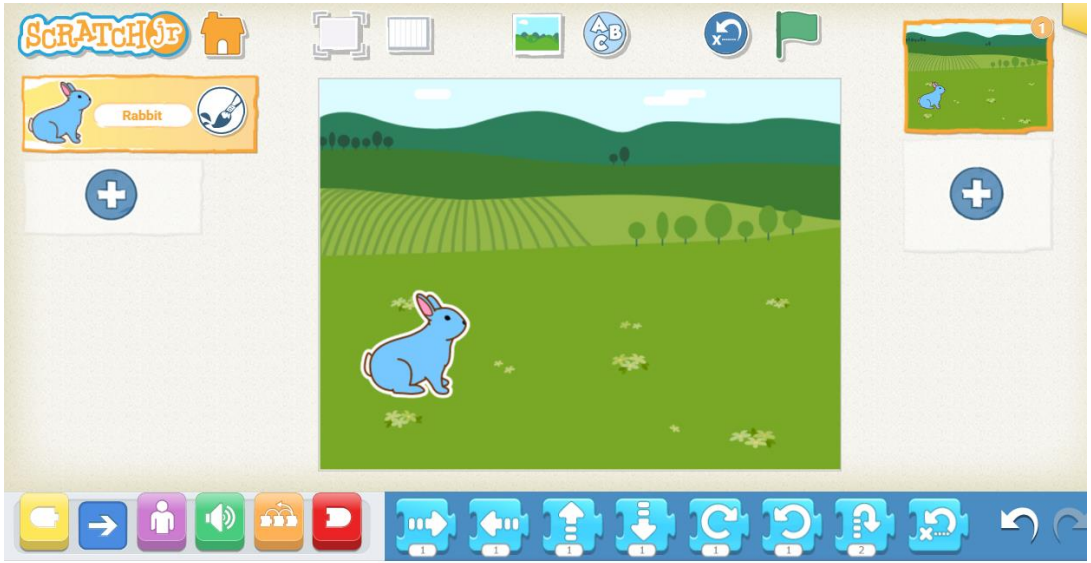
- create a new program
- practise the programming skills that have been learnt
- can be a design task or an open task

Structuring Lessons: PRIMM



SCRATCH

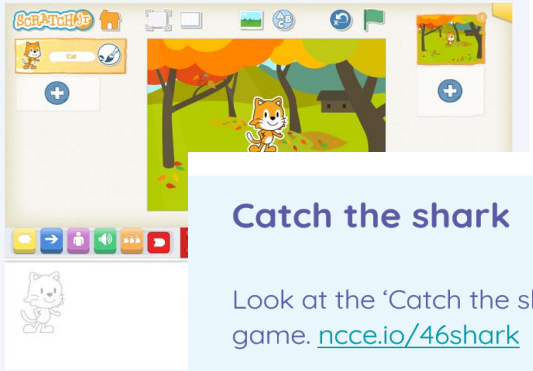
Predict: Scratch Junior



What will the rabbit do?

Teach Computing Curriculum: Investigate and Run

Can you spot the difference?



What difference will you see when you run the progran

Activity 1

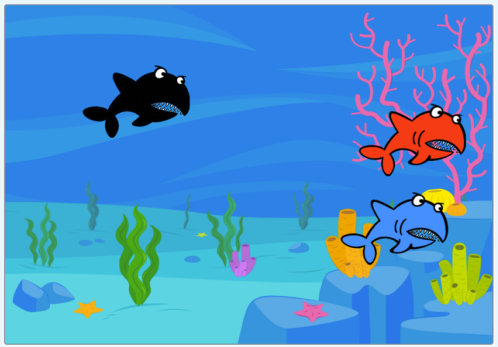
Catch the shark

Look at the 'Catch the shark' Scratch game. ncce.io/46shark

What has been included in this game? Think about:

- Backgrounds
- Sprites
- Other game features

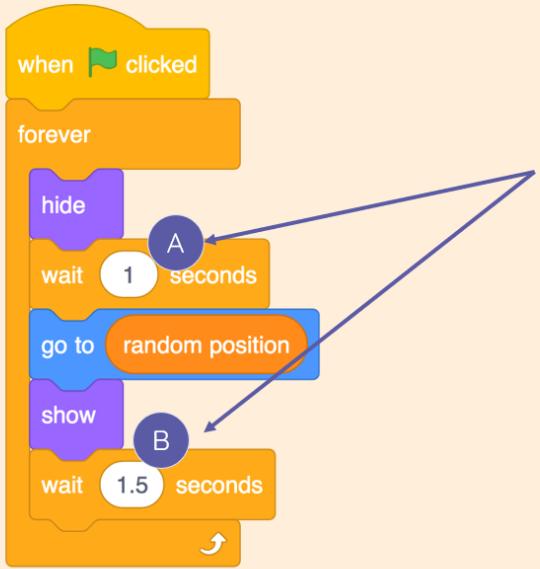
Hint: Try clicking on the sharks



Teach Computing Curriculum: Modify

Activity 2

Modifying the code for the black shark



- What will happen when you change the wait time at A or B?
- Test your predictions by changing the values in the **wait** blocks.
- What difference does it make to the game?

Foster Program Comprehension!

Use a variety of activities to consolidate knowledge and understanding of the function and structure of program s, including debugging, tracing, and **Parson ’s Problems**.

Parsons Problems

Parson's Problems bridge the gap between reading and writing code.

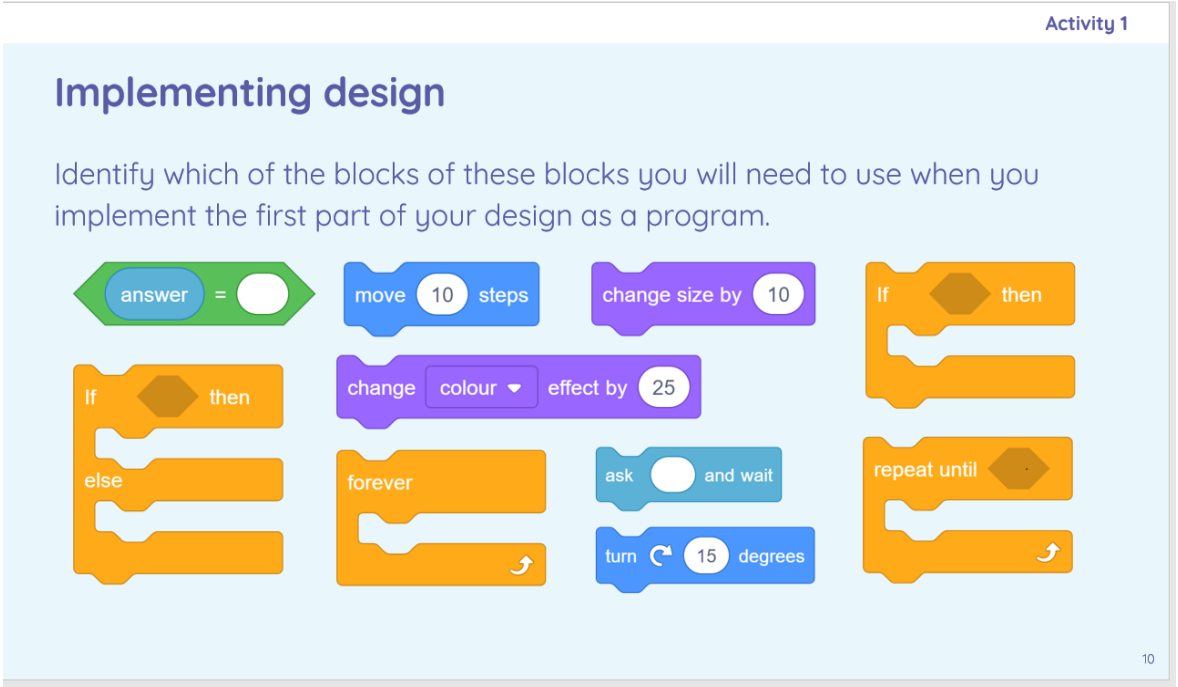
Code is presented to pupils jumbled up and the task is to put the instructions into the correct order so that the whole code performs a predefined task.

Selection- Quiz
https://scratch.mit.edu/projects/284587737

Dino Quiz Algorithm
Start
Ask question
If answer same as 4
**   say correct**
Else
**   say sorry wrong**

Parsons Problems

This example could be used to support learners with additional needs in the Year 5: Programming B- Selection in Quizzes, by creating a Parsons Problem for the activity.

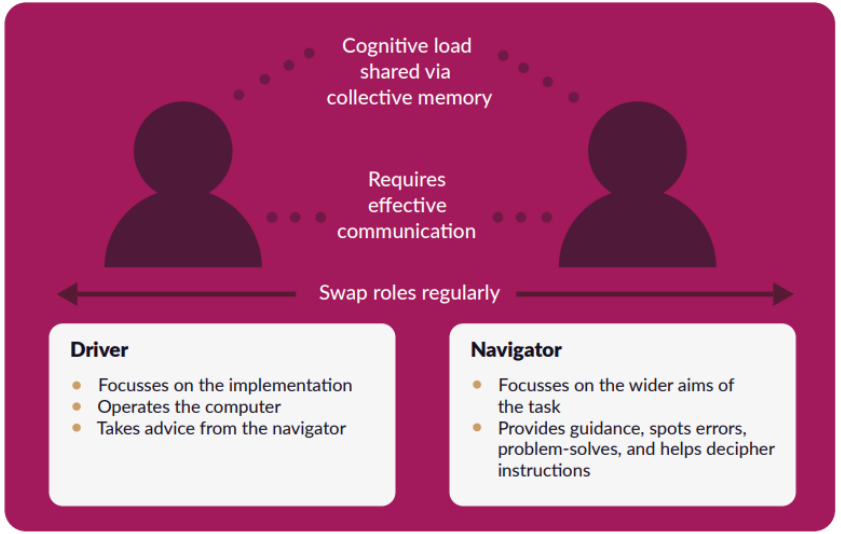


Work Together!

Encourage collaboration, specifically using **pair programming** and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

Work Together- Paired Programming

Which group of learners might this support? How?

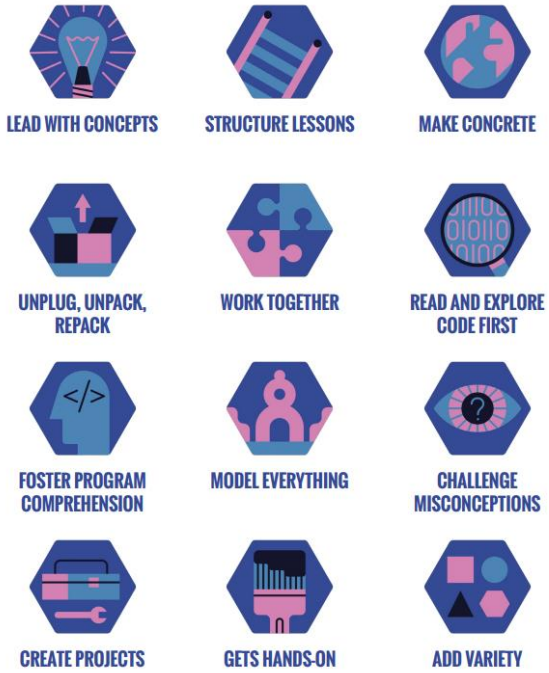


Making it work for your pupils!

Think of 3 pupils in your class:

- a disengaged pupil
- a pupils with SEND,
- a greater depth pupil

How could the ideas discussed support those learners?



Session 3: Making it fit to your school

By the end of this session you will be able to:

- understand potential challenges and barriers and how you can overcome them
- tailor the curriculum to meet the needs of your school
- provide support and challenge through the TCC

Any Questions?