Computing Scheme of Work Unit 2.1 –

Coding – New from 2021

Contents

Introduction

This unit consists of six lessons that assume children have completed Unit 1.7 in year 1. If children have not completed this unit, then you might wish to teach the Coding Catch-Up Unit instead. Children will be coding using the 2Code tool.

Key coding vocabulary is shown in **bold** within the lesson plans, use these new words in context to help children understand the meaning of them and build up their vocabulary of coding words.

The Chimp activities provide further practice of the concepts that the children will be learning and can be used as extension activities. More able children can be encouraged to explore other things that they can change in their programs and experiment with the options available, such as image and scale in 2Code.

Children will often be able to solve their own problems when they get stuck, either by reading through their code again or by asking their peers; this models the way that coding work is really done. More able children can be encouraged to support their peers, if necessary, helping them to understand but without doing the work for them.

Note: To force links within this document to open in a new tab, right-click on the link then select 'Open link in new tab'.

PRIMM

The coding lessons in these units are structured around the **PRIMM** approach. The whole approach may take place during a lesson or series of lessons.

Predict... what this code will do
Run... the code to check your prediction
Investigate... trace through the code to see if you were correct
Modify... the code to add detail, change actions/outcome
Make... a new program that uses the same ideas in a different way. Get creative!

Often lessons will start by looking at existing code, asking the children to 'read' it and make **P**redictions to what they think will happen when the code is run. You'll then **R**un the code and give them time to discuss what happens and relate it back to their predictions. You'll spend time with them **I**nvestigating the code, looking at how different parts work and helping them to understand how. Once children have an understanding of how the code works, they will be encouraged to **M**odify it - changing and adding code and re-running the program to view the impact of their changes. And once confident with this, they are encouraged to try and **M**ake their own program from scratch.

Levels of Scaffolded Coding Tasks

You can support children's learning and understanding by using different degrees of scaffolding when teaching children to code. The lessons provide many of these levels of scaffolding within them and using Free Code Chimp, Gibbon and Gorilla enables children to clarify their thinking and practise their skills. These are not progressive levels; children can benefit from all the levels of activities at whatever coding skill level they are:

Scaffolding	Task type	Examples of how to provide these opportunities
Most scaffolded ↓ Least scaffolded	Copying code	By giving children examples of code to copy.
	Targeted tasks	• Read and understand code • Remix code to achieve a particular outcome. • Debugging. • Use printed code snippets so that children can't run the code but must read it. • Include unplugged activities and 'explaining' tasks e.g. 'how do variables work?'
	Shared coding	• Sharing Challenge activities as a class or group on the whiteboard. • Complete guided activity challenges as a class. • After completing challenges; share methods to create a class version of the challenge. • Free coding as a class
	Guided exploration	• Exploring a limited repertoire of commands • Remixing code • Explore commands in free code before being taught what they do. • Use questioning to support children's learning. • PRIMM approach; Predict – Run – Investigate – Modify - Make
	Project design and code	**Projects (imitate, innovate, invent, remix)** There are different ways to scaffold learning in projects. This process can be applied to programming projects; • Using example projects e.g. the Guided 2Code activities. • Completing the challenges at the end of each guided activity. • Free code✓ • Create a project that imitates a high-quality exemplar. • Remixing ideas. • Independently creating a brand-new program.
	Tinkering	Use Free code Gorilla to access the full suite of 2Code objects and commands ✓ Use Free code to play and explore freely.

Adapted from work by Jane Waite - Computing at Schools https://www.computingatschool.org.uk/

In Literacy, some teachers follow a progression that scaffolds learning to write texts. At first children read lots of examples of the genre of text they are going to create. Then they create an ***imitation*** of an example text. Next, they create a variation of the text (***remix and innovate***). Finally, they get to ***inventing*** a brand-new version.

Medium-Term Plan

Lesson	Title	Success Criteria
1	Algorithms	• Children can explain that an **algorithm** is a set of instructions. • Children can describe the **algorithms** they created. • Children can explain that for the computer to make something happen, it needs to follow clear instructions.
2	Collision Detection	• Children can plan an **algorithm** that includes **collision detection**. • Children can create a program using **collision detection**. • Children read blocks of code and **predict** what will happen when it is **run**.
3	Using a Timer	• Children can create a program that uses a **timer-after command**. • Children can explain what the **timer-after command** does in their program. • Children can **predict** what will happen in a program that includes a **timer-after command**.
4	Different Object Types	• Children can create a computer program that includes different **objects** types. • Children can modify the **properties** of an **object**. • Children can use different **events** in their program to make **objects** move.
5	Buttons	• Children can create a computer program that includes a **button object**. • Children can explain what a **button** does in their program. • Children can modify the **properties** of a **button** to fit their program design.
6	'Smelly Code' Debugging	• Children can explain what **debug** (**debugging**) means. • Children can use a design document to start **debugging** a program. • Children can **debug** simple programs.

Lesson 1 - Algorithms

Aims

- To understand what an **algorithm** is.
- To create a computer program using an **algorithm**.

Success Criteria

- Children can explain that an **algorithm** is a set of instructions.
- Children can describe the **algorithms** they created.
- Children can explain that for the computer to make something happen, it needs to follow clear instructions.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Vocabulary Quiz Y2.
- Two identical sets of any construction toy
- Air Traffic Control Final Stage.
- (Optional) Vocabulary flash cards. The Teacher flash cards have been created so you can print them on A4 paper, cut them to size, fold them in half and glue them together. You can display and use these throughout coding lessons to support use of vocabulary.

Preparation

- Set Air Traffic Control Final Stage as a 2Do for your class.
- Build two models using two identical sets of any construction toy - one that follows the instructions and one that does not. An example would be using Lego Duplo to build a bird, one that follows these instructions and one that uses the same bricks but not the instructions. Download the instructions for your model so that you can display them on the board.
- (Optional) Print storyboard templates for program design.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Use the Y2 Coding Vocabulary Quiz on **slide 4** as a class to help refresh coding knowledge. It is set up so that you attempt all questions and then click the hand in button to check the answers. Run through the answers to the

	questions together. You could use the vocabulary cards to find the answers and display in the classroom.
Algorithms	Use **slides 5 and 6** to introduce todays lesson. Read and discuss the definition of an algorithm.
Activity 1: Lego Models	Display **slide 7**. Show the children the two models you built using identical construction toys - without displaying the instructions. Ask them which is correct. The answer you are looking for is that they are both correct; there is no such thing as 'correct' or 'incorrect' when building creatively. They might prefer one over the other, but both are correct.
	Display **slide 8**. Display the instructions on how to build the model and ask the questions on the slide. If you have enough building materials for the class, they could attempt to follow the **algorithm** to create the model themselves.
Making a Computer Program	With **slide 9**, discuss the process of making a computer program. Look at the plane taking off program. Children might remember this scene from Year 1. Tell them that this time you want them to concentrate on implementing this **algorithm.**
Air Traffic Control Final Stage	Display **slide 10**. Open up Air Traffic Control Final Stage for the class to see and revise the main elements of the code view.
	Display **slide 11**. Click on 'Design' and ask children to point out the **background** and **objects** in the design.
Algorithm -› Code	Display **slide 12.** Click on 'Exit Design' and ask children to help you add in code for the first step of the algorithm (**click event** to enable a plane to take off.). Click on the play button and **run** the program, can they remember why the **code** goes orange? It is when that bit of code **executes.**
Activity 2: Air Traffic Control	Display **slide 13**. Ask children to open Air Traffic Control Final Stage from their 2Dos and start adding the code to make the first 3 steps of the **algorithm** work.
Saving Your Code	With **slide 14**, remind children how to save their work and discuss why it is important to save their coding regularly so that they have a working version to go back to.
Air Traffic Control	When the majority of the class have programmed the **click events**, draw their attention back to **slide 15** and work together to program the last step of the **algorithm** - to make a **sound** play if the helicopter crashes into the yellow plane. When you have talked through the questions on the slide, as an extension ask: could we make a different sound play if the helicopter collides with the purple plane? (You would need two **collision detection** events – one for each plane)

Extension: Air Traffic Control	With **slide 16**, Challenge children to see if they can add **code** for the **collision detection** and complete making the program work. For the optional final step of the **algorithm** children will need to add 2 **actions** to their **collision detection** event. Children can now get creative by adding other planes and runways onto their design and program them. Note: Although you can add more landing strips into the design, you would not normally program them to do actions!
How Did You Get On?	Display **slide 17**. Ask children to work with a partner to read through each other's code and **predict** what will happen when they **run** the program.
Review Success Criteria	Display **slide 18**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 2 – Collision Detection

Aims

- To create a program using a given design.
- To understand the **collision detection** event.

Success Criteria

- Children can plan an **algorithm** that includes **collision detection**.
- Children can create a program that uses **collision detection**.
- Children can read blocks of code and **predict** what will happen when it is **run**.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Princess and the Frog. This is on the main 2Code Page (scroll down to the Chimp activities).
- Princess and Frog algorithm examples –
  - Storyboard Planner (blank).
  - Storyboard Example.
  - Algorithm and Scene Plan.
- Super Coder Poster.
- Blank paper for designing and for Super Coder strips.

Preparation

- Set Princess and the Frog as a 2Do for your class.
- Print and copy Storyboard Planner OR Algorithm and Scene Plan
- Cut blank paper into strips for Super Coder ideas.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Recap Events	Use **slide 4** to revise what happened in Air Traffic Control at the end of last lesson and introduce today's lesson.
Collision	Display **slide 5**. Show two real-life objects (either real people moving towards each other or objects in your hand) to help explain collision.

	Click though **slide 6** to support your explanation of collision detection.
	Watch the collision detection video on **slide 7.** NB This video is also on the unit main page if required.
Princess and Frog	Display **slide 8**. Work through stages 1 and 2 of Princess and Frog as a whole class. Watch the videos and demonstrate how to get back to the videos (click on the instruction) and unlock the hint. Remind children that unlocking the hint will lose a code star.
The Collision Detection Command	Use **optional slides 9-11** to further explain collision detection if needed.
Activity 1: Princess and Frog – Stage 1&2	Display **slide 12**. Ask children to start Princess and Frog from their 2Dos area and complete stages 1 and 2 independently.
Turn the Frog into a Prince	Display **slide 13**. Draw children's attention back to the screen to work through stage 3 together – the frog turns into a prince. Ask children: When do we want the frog to turn into the prince? How will we make sure he turns into a prince only if the princess collides with him? (You will need to add this code into the existing **collision detection event**) Demonstrate using the 'image' option in the **action** list (see below). Before clicking on play ask the children to **predict** what will happen when you **run** the program. *Here you could also demonstrate what happens if you put the 'frog image set to prince' **command** outside of the **collision detection** event.
Activity 2: Princess and Frog - Stage 3&4	With **slide 14**, ask children to independently work through stage 3 and the **debugging** challenge in stage 4.
Your Own Fairy Tale	Display **slide 15**. Draw children's attention back to the screen and watch the video for stage 5 together. Use the Storyboard Planner or the Algorithm and Scene Plan to plan what they what they want to happen in the story. Work together as a whole class to plan what the **algorithm** should be and write it on the plan. Ask children: What is the story? What do we want the princess and the frog to do? How will the **objects** move?

	What will happen if they collide? What will happen next? How could we use a **timer** to make something happen after an amount of time? How will the story end?
Activity 3: Princess and Frog - Stage 5	Display **slide 16**. Ask the children to complete stage 5 – the challenge stage – using code to implement the **algorithm** you have planned.
How Did You Get On?	With **slide 17**, share children's programs, celebrate achievements and encourage them to evaluate how they got on. Did their program work like the planned **algorithm**?
Super Coder	Display **slide 18** and discuss what children would do if there was no limit to their coding ability. (Children write ideas on strips of paper and stick up on your working wall – these could form the basis of future free code lessons).
Activity 4: Extension	**Slide 19** includes a variety of additional and extension ideas. Share children's work to a Display board (see Appendix 1 in Unit 2.1) and have a 'Coding Show' to share children's programs and celebrate achievements.
Review Success Criteria	Display **slide 20**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 3 – Using a Timer

Aims

- To understand that **algorithms** follow a **sequence**.
- To design an **algorithm** that follows a timed **sequence**.

Success Criteria

- Children can create a program that uses a **timer-after command**.
- Children can explain what the **timer-after command** does in their program.
- Children can **predict** what will happen in a program that includes a **timer-after command**.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

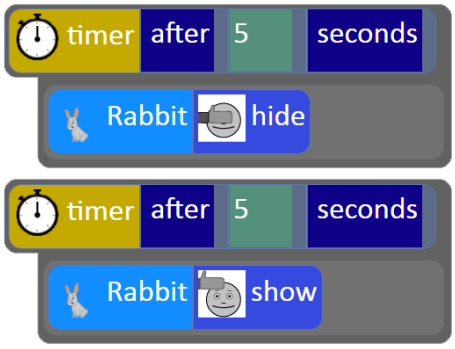
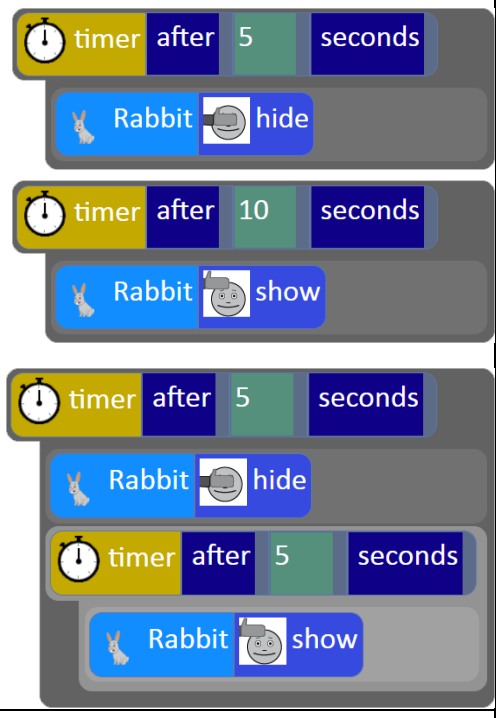
- Magician lesson This is on the main 2Code Page (scroll down to the Chimp activities).
- Storyboard Planner template

Preparation

- Set Magician lesson as a 2Do for your class.
- Print and copy the Storyboard Planner for the children. You might want to make this double sided.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Use **slide 4 and 5** to discuss the word 'timer'.
	Use **slide 6** to introduce the word 'sequence'.
Sequence of Actions	Display **slide 7** and follow the steps for this offline physical activity to introduce the concept of a **timer.**

	With **slide 8**, display the algorithm for the sequence. Ask the children to run the sequence and explain that this is called the output. Go through the questions on the slide.
	Display another sequence on **slide 9** and ask the children to 'run' it. Ask the questions on the slide and point out that in both sequences, each child did their action 5 sections after the last person.
Magician	Display **slide 10** open up the Magician guided lesson.
	Display **slide 11**. Work through Stage 1 as a class. This is a bit like using an **event** code block – it sets a **timer** and after the specified time the **object** (rabbit) will hide.
	Display **slide 12**. Work through stage 2 as a class. Add this incorrect code, **test** it and ask the children to help you **debug**: When you **run** the program both **timers** start at the same time (if you click on stop and play it again you could notice they both highlight orange at the same time) and the code to 'hide' and 'show' the rabbit **executes** at the same time – after 5 seconds.  Point out that the **timer** for the rabbit to 'show' needs to start **AFTER** the rabbit has hidden – like the first time when the class timer counted between children. You need to add the second **timer** *inside* the first **timer** OR work out that the rabbit 'shows' 10 seconds after the start (5 + 5) and alter the second **timer** to reflect that, so either of these solutions would work: 
Activity 1: Magician	Display **slide 13**. Ask children to start Magician from their 2Dos and work through stages 1-4 independently.

Your Own Magic Code	Display **slide 14** and introduce the stage 5 challenge. Look at the example storyboard plan and look at the different actions that are planned for the objects. Ask children: What do you want the characters to do in your **scene**? What order do you want the **actions** happen in? Explain that the children are going to design an **algorithm** for a sequence of **actions** that are different for the magician and rabbit **objects**. Children could either use the example **algorithm** as the plan for their code **OR** modify the **sequence** of **actions** to create their own **algorithm**.
Activity 2: Magician Challenge Stage	With **slide 15**, children to return to their 2Dos and complete stage 5. Remind children to save their work continuously as they build up their code.
How Did You Get On?	Display **slide 16**. Ask children to hand in their 2Dos and review their own code – does it do what they planned it to do? What do they need to do to ensure it does?
Activity 4: Extension	**Slide 17** includes an optional extension activity. In the Chimp activities, **'Jumping Monkey'** also use a timer. (Additional extension activity outlined below).
Review Success Criteria	Display **slide 18**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

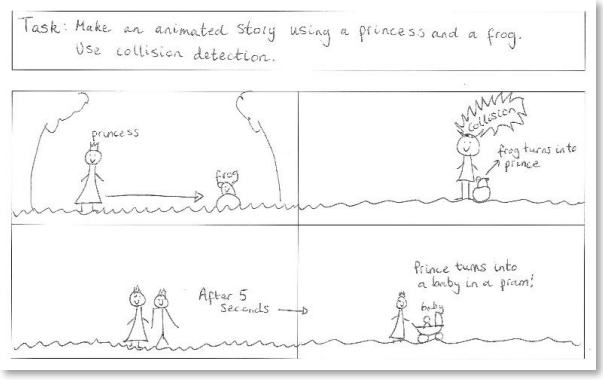
Additional Extension Activity:

Use Free Code Scenes along with planning templates such as the storyboards to plan a program that uses **timers**.

Encourage Children to think through their designs and annotate them including their confidence in coding what they have designed (red, amber, green), this will give you feedback on areas that Children need help with and help to ensure that Children create realistic designs and successful programs for their skill level.

Example plans are given below.

You could use Example Story program to demonstrate an example (Note: this example uses **key press events** which the children learn in lesson 4.)



Lesson 4 – Different Object Types

Aims

- To understand that different **objects** have different properties.
- To understand what different **events** do in code.

Success Criteria

- Children can create a computer program that includes different **object** types.
- Children can modify the **properties** of an **object**.
- Children can use different **events** in their program to make **objects** move.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Free Code Chimp (this is found on the main 2Code page).
- Snail Race.
- Turtle and Character.
- Road Scene.

Preparation

- Set Free Code Chimp as a 2Do.
- Set Road Scene as a 2Do for less confident children.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Objects and Actions	Display **slide 4** and introduce today's topic of objects and actions.
Snail Race	Display **slide 5.** Open Snail Race and work through stages 1-3 together as a class. Focus on the **actions** available for the snail **object** in stage 1 – have they seen these before? Up until now children have been programming **objects** to move left, right, up, down and stop – but this program works differently. Discuss with them how it is different – snails are a different type of **object** to ones they have used before and have different options for **actions**.
Different Actions	Look at the scene on **slide 6**. Ask children to predict what will happen when the program is run. Run the program Turtle and Character and see what happens.

Designing a Scene	Display **slide 7** and open Free Code Chimp in front of the class. Follow the instructions on the slide to set the scene.
Choosing Objects	Display **slide 8.** Look at the **object** types to choose from on the left and add a turtle and 3 other **objects** that would move (tell children that in this lesson we are using any object apart from the button – we are going to look at the button next lesson).
Changing Objects	Display **slide 9**. This is the first-time children have used Free Code Chimp in coding lessons so spend a bit of time browsing the clipart galleries – pointing out the categories and search option. Go through how to change the objects and also the size of the object. Recap how to move the objects around.
Activity 1: Create the Scene	With **slide 10**, challenge children to create a **scene** like this by setting the background and adding **objects** -they could choose different clipart so they all have different **objects** on their scenes. (You could set Road Scene as a 2Do for less confident children so they just have to add objects to the scene rather than create it).
Making Objects Move	Once the majority of children have made their scene draw their attention back to the board and click on 'Exit design' to start adding some code. Talk through **slide 11**, thinking about the **events** and the **when key**.
Actions for Objects	With **slide 12**, demonstrate how to add an action to the objects. Point out that the actions for a turtle and different to the other ones. When adding code for the turtle, in this lesson we are going to program it to move forwards. (Programming a turtle to turn involves some understanding of degrees of a turn – e.g. a quarter turn = 90 degrees, a half turn = 180 degrees, this is taught in Unit 3.1.) Ask children to predict what will happen, then run the code.
Adding to Your Code	Display **slide 13**. Challenge the children to add their own code to their programs. You could stop the program and return to the code, delete what you had in there and ask children to help you use timers to make the **objects** start at different times. Could they also add some food **objects** in to be eaten along the way? (This will involve using **collision detection**!) NB If children are coding on tablets, the **when key event** is not available. Instead, they could try using **when clicked** or **when swiped events**.
Activity 2: Program Your Scene	Display **slide 14**. Set the children back to their own designs to add code to make their **objects** move – challenge them to try using different **events** and add in **collision detection when** a key is pressed. NB If children are coding on tablets, the **when key event** is not available. Instead, they could try using **when clicked** or **when swiped events**.
How Did You Get On?	With **slide 15**, ask children save their designs. Share great examples with the class, discussing the code that has been used to make them work.
Review Success Criteria	Display **slide 16**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 5 – Buttons

Aims

- To create a program using a given **design**.
- To understand the function of **buttons** in a program.

Success Criteria

- Children can create a computer program that includes a **button object**.
- Children can explain what a **button** does in their program.
- Children can modify the **properties** of a **button** to fit their program design.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Free Code Chimp (this is found on the main 2Code page).

Preparation

- Set Free Code Chimp as a 2Do.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Buttons	Display **slide 4** to introduce today's topic of buttons.
Designing a Scene	With **slide 5**, explain that today children will be designing a computer program that includes a button. Follow the instructions on screen to demonstrate setting the scene. Choose a background with some green or sand or something an animal could walk along, or under the sea
Choosing Objects	Display **slide 6**. Add an animal **object** and a food **object**.
	Display **slide 7.** Recap how to change the object and the size, as well as how to move them around.
Add a Button	Display **slide 8**. Add a button to your scene.

Button Properties	Display **slide 9** and look at the button properties. Name the button and set the text for the button. You can also change the **text size**, **text colour** and **background** colour on the button.
Coding Your Button	Display **slide 10**. Look at the coding blocks and ask children to help you add code to make the animal move when the button is clicked on.
Adding More Buttons	Use **slides 11 and 12** to encourage children to consider adding more than 1 button and consider how they could be programmed in a way the develops the **scene**.
Activity 1: Create Your Program	Introduce the activity on **slide 13**. Children to start Free Code Chimp from their 2Dos and create a **scene** that includes at least one **button**, then add code to make it work.
How Did You Get On?	Display **slide 14**. Ask children to hand in their 2Dos and review their own code – how does their button work?
Review Success Criteria	Display **slide 15**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 6 – 'Smelly Code' Debugging

Aims

- To know what **debugging** means.
- To understand the need to **test** and **debug** a program repeatedly.
- To **debug** simple programs.

Success Criteria

- Children can explain what **debug** (**debugging**) means.
- Children can use a design document to start **debugging** a program.
- Children can **debug** simple programs.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 page. From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Debug Challenges Chimp – this is on the main 2Code page.
- Debugging Process.
- Smelly Code Worksheet.
- Smelly Code 1 & 2 2Code examples:
  - Smelly Code 1 - Monster Hero;
  - Smelly Code 2 – Car Crash.
- Smelly Code 1 Example Answers
- Smelly Code 2 Example Answers

Preparation

- Set Smelly Code 1 as a 2Do.
- Set Smelly Code 2 as a 2Do.
- Print and copy the Smelly Code Worksheet for each child or pair.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Display **slide 4** and discuss the key vocabulary for the lesson.
Debugging Process	Display **slide 5.** In the last two lessons, the children have been creating scenes and adding code to program them. Discuss the debugging process.

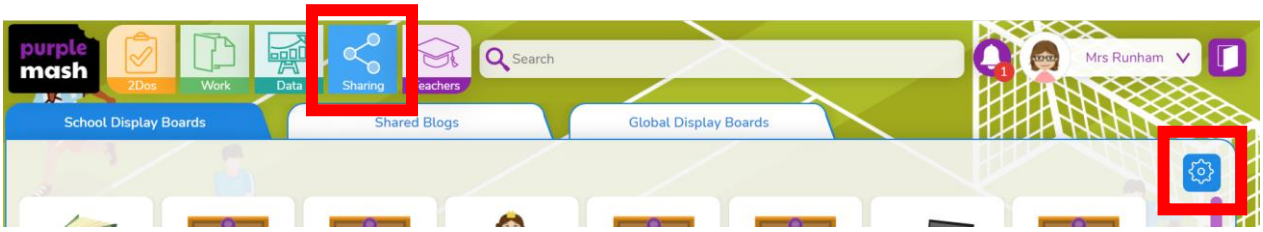
	Talk through the process shown on **slide 6**. Children could write/ stick the steps into workbooks if you use them.
Activity 1: Debugging Challenges	Display **slide 7**. Complete challenges 1 & 2 as a class, referring to the debugging process on slide 6 as you go.
	Slide 8 shows an optional challenge which uses an 'every' timer which children haven't learnt about yet. You need to click on the word 'after' and change it to 'every' - discuss what they think this does, how is it different to an 'After' timer?
Activity 2: Debugging Smelly Code	Display **slide 9**. Hand out the Smelly Code Worksheet either one per child or one per pair for partner work. These are also shown on **slides 10 & 11**. There are two examples of 'Smelly Code' that don't do what they should do. Challenge children to work with a partner to sniff out what is wrong with the code. Refer back to the Debugging Process (slide 6) which could be displayed on the board as they work through it. Ask the children to trace the code line by line and predict what will happen when the code is run. Will it do what it is supposed to do? Ask the children to circle the 'smelly' parts of the code (there are 3 bugs to find!) Ask them what they need to change to fix the code.
Activity 3: Fixing Smelly Code	Introduce the activity on **slide 12**. Tell children that you have set the two 'Smelly Code' programs as 2Dos. Ask them to open each one, **run** the code to **test** what it does, then stop the program, work out which parts of the code need fixing and **debug** the code until it works, then save the fixed file. Once the file has been fixed, they can 'sign-off' to say that all the bugs are fixed 'like real programmers do'. Review the examples as a class. How did they go about fixing the **bugs**? Review the Smelly Code 1 & 2 Example answers to check children have fixed it correctly.
How Did You Get On?	Display **slide 13**. Ask children to save their programs, then share great examples with the class, discussing the code that has been used to make them work and referring back to the **debugging** process.
Review Success Criteria	Display **slide 14**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Appendix 1: Display Boards

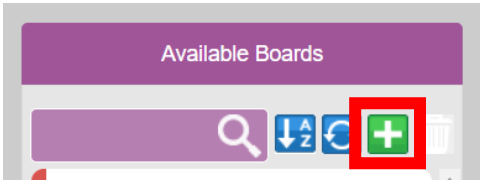
Create the Display Board

Creating the display board is usually something you do before the lesson.

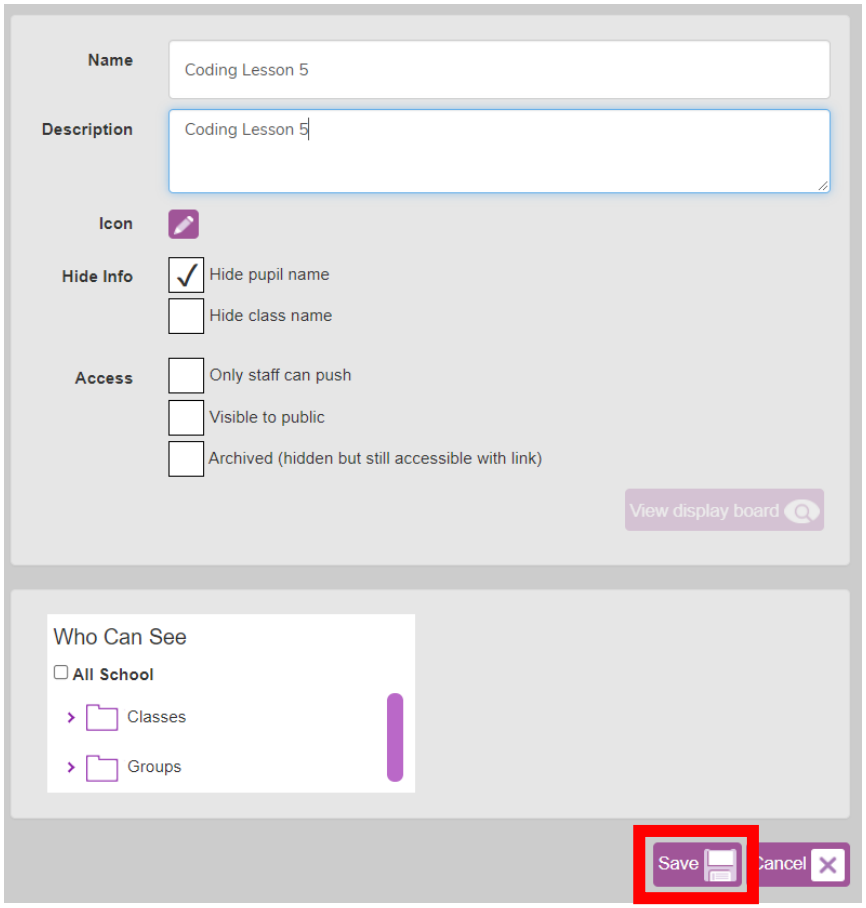
1. Click on the 'Sharing' button to find the Display Board tab, and then click on the settings cog:



2. Click on the '+' in the menu on the left:



3. Edit the settings (don't forget to add an icon by clicking on the), select the class and then click on 'Save':



4. Exit Display Board settings:

The Display Board will now be visible under the 'Sharing' button to all those you've selected to have access to it.

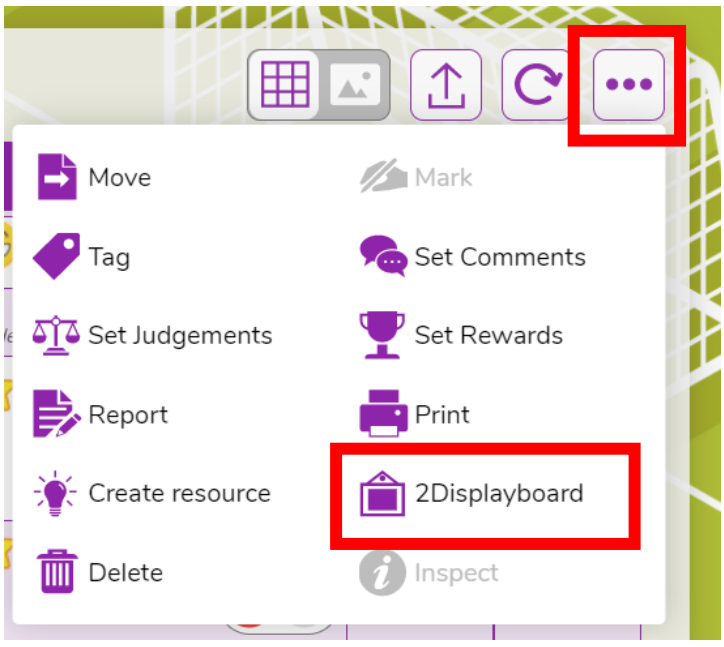
Adding work to a Display Board:

1. Click on 'View Folder' from the 2Do:

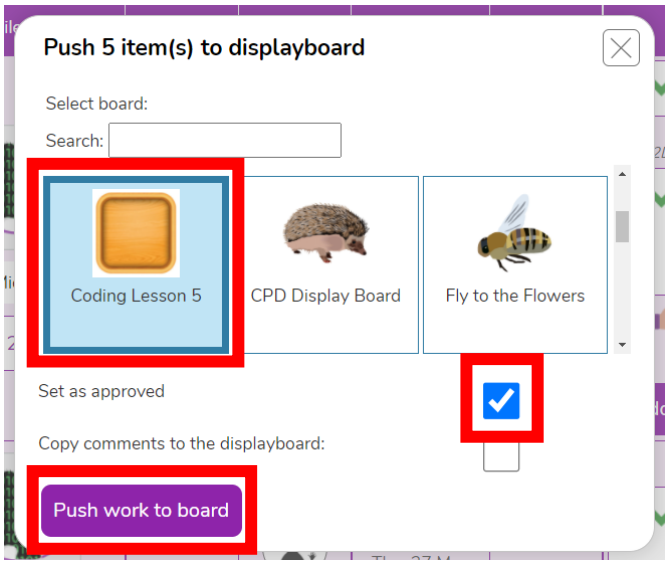
(or navigate to the work you want to share in the Work area).

2. Select the files you want to add to the display board or select all files in the folder using the tick at the top.

3. Click on the '...' menu button top right, then click on '2Displayboard':



4. Choose the display board you've made for the work, tick 'Set as approved' and 'Push work to board':

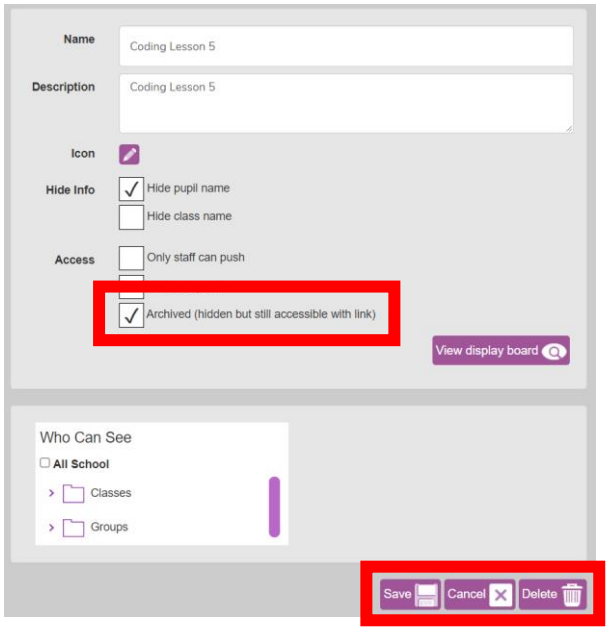


5. Click on 'Sharing' button and then on the display board, you should see the work you've added. It can be deleted by clicking on 'Edit' at the top of the board, then clicking work and then delete. This will remove it from the display board, it won't delete it from Purple Mash.

Deleting or Archiving a Display Board:

When you've finished the lesson you can return to the Display board settings and either delete it or archive it to stop it appearing under the 'Sharing' button.

1. Click on 'Sharing' and then on the settings cog.
2. Tick 'Archive', and then 'Save' OR 'Delete'
   Clicking on 'Delete' will delete the display board but the work will still be available in the work area, it doesn't not delete the files.



Assessment Guidance

The unit overview for year 2 contains details of national curricula mapped to the Purple Mash Units. The following information is an exemplar of what a child at an expected level would be able to demonstrate when completing this unit with additional exemplars to demonstrate how this would vary for a child with emerging or exceeding achievements.

Assessment Guidance	
Emerging	Children know that an algorithm is related to giving instructions. They can relate a simple one-step algorithm to the outcome of code in Free code Chimp. For example, in Lesson 1 they have been able to make a program that follows the algorithm e.g. 'when the helicopter is clicked it takes off'. With support, children can create a simple one step program that achieves a specific purpose. With support, children can identify and correct errors (Unit 2.1 Lesson 6). With support, children can identify the parts of an algorithm that control and initiate specific actions. Based on this, with support, children can predict what will happen in a program (Unit 2.1 Lesson 4).
Expected	Children can explain that an algorithm is a set of instructions to complete a task. They have turned algorithms of more than one step into code using free code Chimp. For example, in Lesson 4 and 5 they have been able to make a program that follows their algorithm e.g. 'when the animal is clicked it moves forward then turns right'. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code. (Unit 2.1 Lesson 5). Children use a planning format on paper before implementing on screen within 2Code as they recognise this is the best approach for designing a solution. They can use the Design Mode within 2Code to carefully see how their planned program will look and are able to switch into Code Mode to apply movements to objects (Unit 2.1. Lesson 4). They confidently include objects, actions, events and outputs successfully within their 2Code programs. Children can talk through code which contains a timer command, explaining where this command is positioned and what will happen (Unit 2.1. Lesson 3). Children can predict program outcomes and attempt to debug. For example, (Unit 2.1 Lesson 6). Children can identify the parts of a program that respond to specific events and initiate specific actions. Based on this, children can predict and describe, using a cause and effect sentence, what will happen in a program. (Unit 2.1 Lesson 6). Children can debug their own and other's programs using design documentation to test against (Unit 2.1 Lesson 6).

Assessment Guidance	
Exceeding	Children can explain and give examples that an algorithm is a set of instructions to complete a specific task. They can create complex and logical algorithms of several steps that accomplish the aim of the task that can be easily utilized to create executable code. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code (Unit 2.1 Lesson 5). Children can create more complex programs that utilize all the coding constructs that they have learnt about and extend their own learning by trying out different ways to code that achieve a specific purpose. Children can identify and correct errors. For example, (Unit 2.1 Lesson 6). An exceeding pupil will be able to apply their knowledge as a transferable skill across a range of debugging scenarios including making logical attempts to debug their own more complex code. Children can identify the parts of a program that respond to specific events and initiate specific actions. Based on this, children can adopt a systematic approach for predicting the behaviour of programs. Furthermore, using cause and affect language, Children can reason in detail about what will happen in a program. For example, (Unit 2.1 Lesson 5).