



Computing
Scheme of Work

CRASH COURSE



Coding Unit for Children in Year 2

Who haven't used 2Code in Year 1



Year Group: 2
Number of
Lessons: 6

From **2**simple



Contents

Year 2 Crash Course – Medium Term Plan.....	4
Challenges.....	5
Free coding.....	5
Program Design.....	5
Levels of Scaffolded coding tasks.....	6
Lesson 1 - Introduction to coding.....	7
Aims.....	7
Success criteria.....	7
Resources.....	7
Activities.....	7
Lesson 2 - Introduction to Backgrounds and Characters.....	9
Aim.....	9
Success criteria.....	9
Resources.....	9
Activities.....	9
Lesson 3 - Introduction to Collision Detection.....	11
Aim.....	11
Success criteria.....	11
Resources.....	11
Activities.....	11
Lesson 4 – Using Repeat and Timer Commands.....	13
Aims.....	13
Success criteria.....	13
Resources.....	13
Activities.....	13
Lesson 5 – Debugging.....	16
Aims.....	16
Success criteria.....	16
Resources.....	16
Activities.....	16
Lesson 6 – Exploring the Possible Actions of Objects.....	18
Aims.....	18



Purple Mash Computing Scheme of Work – Y2 Crash Course – Contents

Success criteria.....	18
Resources.....	18
Activities.....	18
Appendix I: Other features of 2Code	21
Real Code Mode:.....	21
Sharing:	21
Assessment Guidance.....	22



Year 2 Crash Course – Medium Term Plan

Lesson	Aims	Success Criteria
1	Introduction to coding. Introduction to block coding on screen.	<ul style="list-style-type: none"> Children can explain what is meant by coding. Children can explain what a block of code is. Children can read through combined blocks of code. Children know that for the computer to make something happen, it needs to follow clear instructions.
2	Introduction to backgrounds and characters. Making a character move left and right.	<ul style="list-style-type: none"> Children can use Design Mode to have control over how my game looks. Children can write a program that controls how a character moves. Children can explain what is happening and write down/ talk through my code.
3	Introduction to Collision Detection.	<ul style="list-style-type: none"> Children can write a program where objects can stop moving and a sound is played when the objects collide.
4	To use Repeat and Timer commands.	<ul style="list-style-type: none"> Children can explain how to use the following terms in a computer program: Command, Repeat, Input, Output, Event, Collision Detection and Timer. Children can create a computer program including at least four of the above new coding vocabulary terms.
5	Debugging.	<ul style="list-style-type: none"> Children can explain what debug (debugging) means. Children can explain what they did so that their computer program did not work. Children can debug simple programs.
6	To explore the possible actions of different types of objects.	<ul style="list-style-type: none"> Children can create a computer program using different objects. Children can predict what the objects in classmates' programs will do, based on my knowledge of the objects' limitations, e.g. a turtle can only move in specific ways. Children can explain how they know that certain objects can only move in certain ways.

Differentiation

Children will need to be able to drag and drop in order to move code blocks around. If children have not had much practice with this then there are several example activities within the Activities section of 2DIY that help children to practise these skills in preparation: [2DIY activities to practise drag and drop](#). Within each category of activity, look for the example file then press the Play button. If the children have not used Purple Mash before, spend some time showing them how to log in and how to get to 2Code.

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'.



Purple Mash Computing Scheme of Work – Y2 Crash Course – Medium Term Plan

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

The crash-course aims to prepare children for using the Computing Scheme of Work Coding unit in year 3. To enhance children's ability to code and understand the process of coding and design, children should have had as many of the following experiences as possible:

Challenges

When using the guided activities, children should have attempted the challenges at the end of the guided lessons in 2Code and come up with solutions to these either individually or using shared coding as a group or class.

Free coding

Children will benefit from spending some time using:

- Y1-2 Free code Chimp (or Free code scenes)
- Y3-4 Free code Gibbon
- Y5-6 Free code Gorilla

To create their own programs.

Program Design

To master coding skills, children need to have the opportunity to explore program design and put computational thinking into practice. The crash course suggests some designing before coding in the plans. Children could do this through:

- Storyboarding their ideas for programs. For example, creating a storyboard when planning a program that will retell part of a story.
- Creating annotated diagrams. For example, creating an annotated diagram to plan a journey animation that tells the story of an historical event they have been studying.
- Creating a timeline of events in the program. For example, creating a game program against the computer, what are all the actions needed from the objects?

During the design process, children should be encouraged to clarify:

- the characters (objects and their properties)
- what they will do (actions and events)
- what order things will happen (the algorithm)
- rate their confidence at being able to code the different parts of their design and either refine the design or review possible solutions as a class or group.



Lesson 1 - Introduction to coding

Aims

- To understand what coding means in computing.
- To create unambiguous instructions like those required by a computer.
- To introduce 2Code.
- To use the 2Code program to create a simple program.

Success criteria

- Children can explain what coding means.
- Children know that for the computer to make something happen, it needs to follow clear instructions.
- Children can explain what a block of code is.
- Children can read through combined blocks of code.

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 don't lose this page.

- Code block cards. Children will need to use a few copies of each picture to create code away from the computer.
- Children will be looking at activities from the Chimp guided lessons. These can be found on the [main 2Code page](#).
- (Optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.



Activities


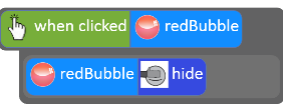
1. Explain to the children that we are looking at coding. Ask them if they know what coding is. Discuss briefly that it is the way that computer programmers input instructions into computers to create programs. Can they give any examples of computer programs that they have used?
2. Start off by doing some activities where the children must follow or give clear instructions.
3. Choose two children; one is a robot and the other is a coder. The coder needs to direct the robot to walk from one place in the classroom to another. How can they give the instructions so that the robot does not crash into objects in the way? Repeat a few times in different locations.
4. Tell the children that you are now going to be the robot and they are the coders. Stand by the whiteboard and ask the children to give you clear instructions, one step at a time, for drawing a basic picture of a house.
5. Once you have a set of clear instructions, discuss the way that coding languages use symbols rather than whole sentences. Can they come up with their own symbols for the instructions for drawing a house? For example, holding their fingers in a triangle shape for the roof. These symbols could be written on the whiteboard to create a program. This can become more complex, depending upon the understanding and interest level of the children. Some children will enjoy working out how their 'code' could be adapted to



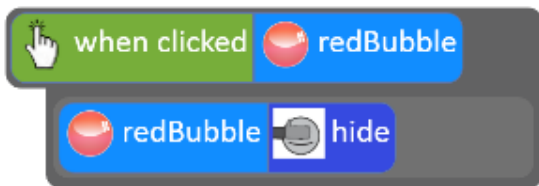
Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 1

draw a bungalow or a block of flats. Some children will want to be precise about the placement of the shapes, e.g. a symbol to show putting the roof on top of the house.

- 6. Show the children the printed code block cards. Explain that these are examples of code used on a computer. Can they suggest ways to combine the cards to make instructions?
- 7. Show children the 'Fun with Fish' lesson in the 2Code Chimp lesson area. Briefly show the Design View (button in top right-hand corner) to show what the code will be controlling. Work through challenges 1, 2 and 3 with the children watching the videos and reading through the blocks of code. Emphasise the need to give the computer clear instructions for moving the fish. Point out that each action is on a different line.
- 8. Direct the children to the Bubbles activity and ask them to complete the steps on their own computers. Show them the 'hint' button to use in case they are really stuck. To use a hint, click on the  at the top right of the screen to open the instructions then click on the . Children should finish step 3.

Step 1	Step 2	Step 3	Step 4
		Debugging challenge	Make your own bubbles

- 9. Bring the class back together and complete step 4 as a class coding activity. Children can suggest actions for the bubbles and work out how to write the code for them. During this class coding, make the point that when some blocks of code have indented lines of code within them for example:



The 'red bubble hide' is slightly indented. This is because it is the 'output' (what we want to happen). Show what would happen if you put the codes directly under each other: the red bubble would hide regardless of whether it was clicked if the 'red bubble –

hide' line is not inside the click **event**.



Lesson 2 - Introduction to Backgrounds and Characters

Aim

- Children will use Design Mode to add and change backgrounds and characters. They will use the Properties table to change the look of the objects.

Success criteria



- Children can make a background using Design Mode.
- Children can add characters using Design Mode.
- Children can use the drop-down menu to change backgrounds and characters.

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 don't lose this page.

- [Challenge cards](#). You will need to print copies of the images from these to distribute to the class.
- [Code block cards](#). These can be displayed somewhere to aid children in their planning.
- Free Code Chimp; this can be found on the [main 2Code page](#).

Activities

1. Show the children **Free Code Chimp**. This is different to the Chimp lessons because we have the freedom to create our own games and we can pick how we want our games to look.
2. Discuss any computer games that the children have played. On the interactive whiteboard (IWB), show images of some popular age-appropriate computer games.
3. Explain that the children are going to be programmers and that their job is to create their own simple program. We need to think about how our games will look. Refer again to the pictures of popular games. Just like pages in a book, a game needs to have a background. What else do games need? Discuss the use of characters in games and the purpose of some games (e.g. to collect things, save other characters, get onto another level or into another world).
4. Remind the children about Design View and the Code Mode in 2Code that they saw last week. On Free Code the design is blank because we must choose the background and characters.
5. Explain that if we want to add a background we need to go into design view and then press the  Background button.
6. Show children how to choose different backgrounds (by double-clicking on the question mark in the properties here ). Show how the Get Image and Paint Image buttons work.



Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 2

7. Show children how to add a character to their background by dragging it across. Show children how to change the character image by double clicking on the character and using the drop-down menu.
8. Distribute the challenge cards and explain that these are ideas for a program. They should use the printed images to design their program by annotating them with what each character will do e.g. 'when clicked, move up'. You might want to display the code block cards for pupils to refer to.
9. Once children have a design for their program and feel confident that they can code what they have designed, they should go to their devices and open up Free Code Chimp and try to recreate their design.
10. Children can save their screen by pressing the Save button (the blue floppy disk) at the top of the screen and saving into My Work. If they have not used Purple Mash to save before then they will need time to practise this.



Lesson 3 - Introduction to Collision Detection

Aim

- To use Collision Detection to make objects stop moving when they collide and play a sound when this happens.

Success criteria

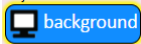

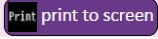

- Children can make objects stop when they collide.
- Children can program a sound to play when objects collide.

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 don't lose this page.

- [2Code Collision Detection video](#). NB This is an extract of another video, so it appears to start abruptly.
- Free Code Chimp; this can be found on the [main 2Code page](#).

Activities

- Show two real-life objects (either real people moving towards each other or objects in your hand). Are the objects moving in the same direction? What will happen when the objects meet? Explain the meaning of the word 'collide'. Explain that we are doing the same in our game; when two objects collide, we want them to stop and we want to hear a sound on impact.
- Open the Free Code Chimp area and ask a child to add a background and two characters using Design Mode.
- Switch to Code Mode and ask the children to look at the various coding options on the left-hand side of the screen. Which one could we use to make something happen when the objects collide?
- Watch and discuss the Collision Detection video. Recreate a collision between two objects as a class. Why can't we put the Collision Detection command in first? Explain how we must build the code first; make some code so that the characters are moving towards each other then can we add the Collision Detection.
- Explore some of the possible things that could happen when two objects collide:
 - One or other or both stop
 - One or other or both hide (e.g. a spaceman moving towards a rocket: upon collision, the spaceman hides, making it look as if he went into the rocket).
 - One or other or both move differently (change direction)
 - Background image could change (drag the  code block into the collision detection block)
 - A sound could play (drag the  code block into the collision detection block)
 - A message could print to the screen or an alert box could appear (drag the  or  code blocks into the collision detection block)
 - Some children could make a combination of actions happen upon collision e.g. a sound and an action.



Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 3

6. Children should decide upon something that they would like their objects to do. This could be in the form of a written design or by discussing their plans. They should then work on their devices to build up the code and add a Collision Detection command following their design.
7. For more practice of this concept see Chimp lessons 'Guard the Castle' and 'Princess and the Frog' which also use Collision Detection.



Lesson 4 – Using Repeat and Timer Commands

Aims

- To understand the meaning of repeat in 2Code.
- To understand how a timer works in 2Code.

Success criteria

- Children can create a computer program including a repeat and a timer.

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 don't lose this page.

- [Repeat and Timer example](#) code.
- A Beebot or other floor robot if available.
- Blank printable storyboards for designing.

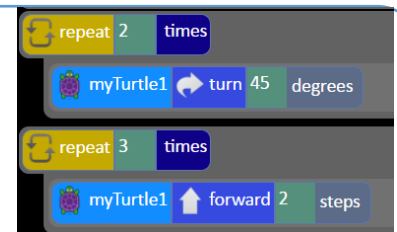
Activities


1. Ask children what they think 'repeat' means. You could relate this term to the use of Bee bots or other floor robots if the children have experience of these. How can you make a Beebot move in a square shape? Write the instructions without using repeat then refine the instructions to use repeat.
2. Look at the [Repeat and Timer example](#). Can the children see the repeat command used to make the turtle move?

First, the turtle repeats the action of turning 45 degrees, twice.

There is actually a shorter way that this can be coded without using 'repeat'; can the children work out what it is? (turn 90°)

Then, it repeats the action of going forwards two steps, three times.



3. Play the code. The turtle moves very quickly. You can slow it down (though it will stop the timer part of the program working properly). To see this, click on the  in the top menu bar before pressing play. Then in play mode look for the speed slider on the bottom right and set it to slow, then press the play button above the slider.





- Give children some time to try to make their own turtle move in a square pattern then bring the class back together again.
- Look at the code for the princess in the example, this uses timers to make the princess move in a certain

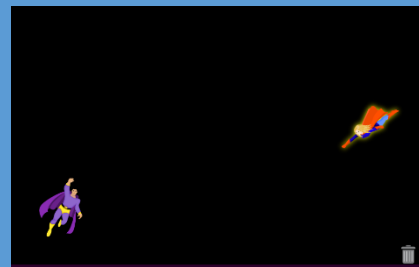
The princess moves up
 After 0.5 seconds, she moves down
 After another 0.5 seconds, she stops.
 (This is supposed to look like the princess jumping on the turtle).

```

myPrincess1 ↑ up
timer after 0.5 seconds
myPrincess1 ↓ down
timer after 0.5 seconds
myPrincess1 stop
    
```

- In the last part of the program, when the turtle and princess collide, there is a 'splat!' sound and the turtle hides (he has been squashed).
- Open free code Chimp on the board and recreate the following program using the timer. You could ask children to drag the blocks to demonstrate to the class.

Add two heroes and make their images different from each other



Make both heroes go up initially. Press play to show the children.

```

myHero1 ↑ up
myHero2 ↑ up
    
```

Add a timer making both heroes change direction after 2 seconds. Point out to the children that the timer command can be set to 'every', they need to check it is set correctly otherwise their code won't work as they expect it to. Play the code so children can see it working.

```

timer after 2 seconds
myHero2 ↓ down
myHero1 → right
    
```

Add collision detection so that a bang sound plays when the heroes collide and one of the hero images is set to 'OUCH' (you can find this in the splats section of the clipart picker).

```

when myHero1 collides with myHero2
  sound bang 1 times
  myHero2 image set to ouch
    
```

Play the code so children can see it working.

Add another timer that then sets the hero image back after 2 seconds.

```

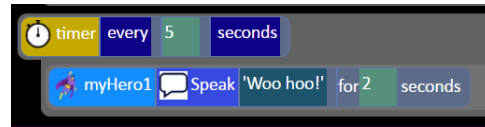
when myHero1 collides with myHero2
  sound bang 1 times
  myHero2 image set to ouch
  timer after 2 seconds
  myHero2 image set to myHero2
    
```

Play the code so children can see it working.



Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 4

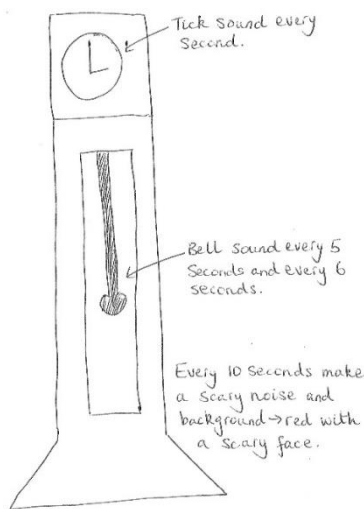
Add a final timer that makes the other hero say 'Woo Hoo!' every 5 seconds.



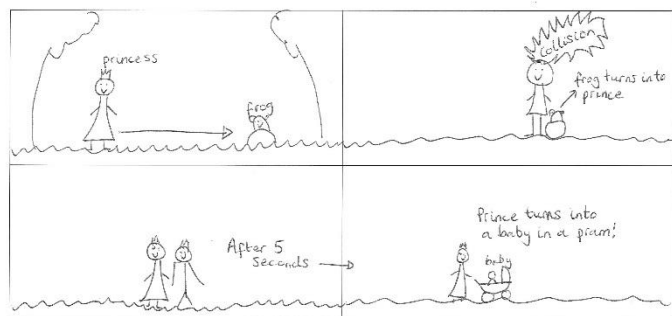
- In the Chimp guided activities, the use of the timer is introduced in a video in Step 5 of 'Princess and the Frog', and the 'Tick Tock Clock' and 'Magician' activities also use a timer. You can look at these with the children if they need more guidance or ideas or children could work on these activities.
- Children should have the chance to design a simple program using repeat or timers and create their design as a program. Here are examples of designs that you could use to explain how to design:

A note about design: 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.

Task: To make a ticking clock with sound effects using a timer



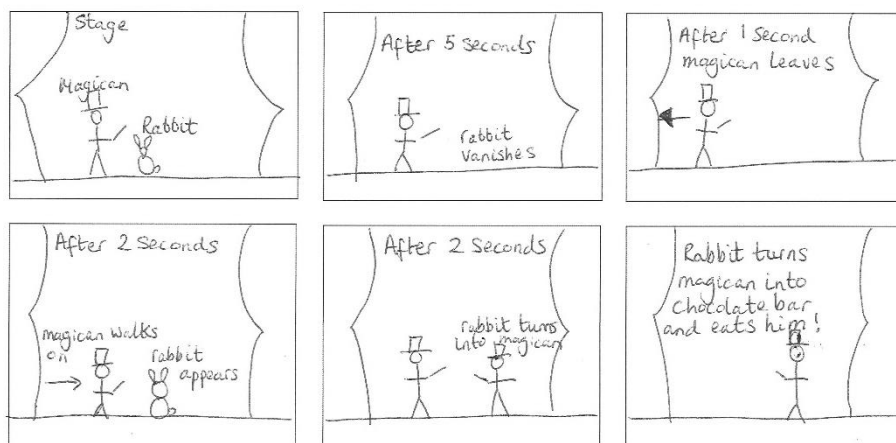
Task: Make an animated story using a princess and a frog. Use collision detection.



For more fantastic resources visit us at www.purplemash.com or follow us on Social Media Twitter: @2simplepurplemash and Facebook: <https://www.facebook.com/2simpleuk/>



Task: Make an animated story about a magician and a rabbit. Use timers.





Lesson 5 – Debugging

Aims

- To know what debugging means in computing.
- To intentionally break a program and then debug it.
- To debug other simple programs.

Success criteria

- Children can explain what debug (debugging) means.
- Children can explain what they did so that their computer program did not work.
- Children can debug simple programs.

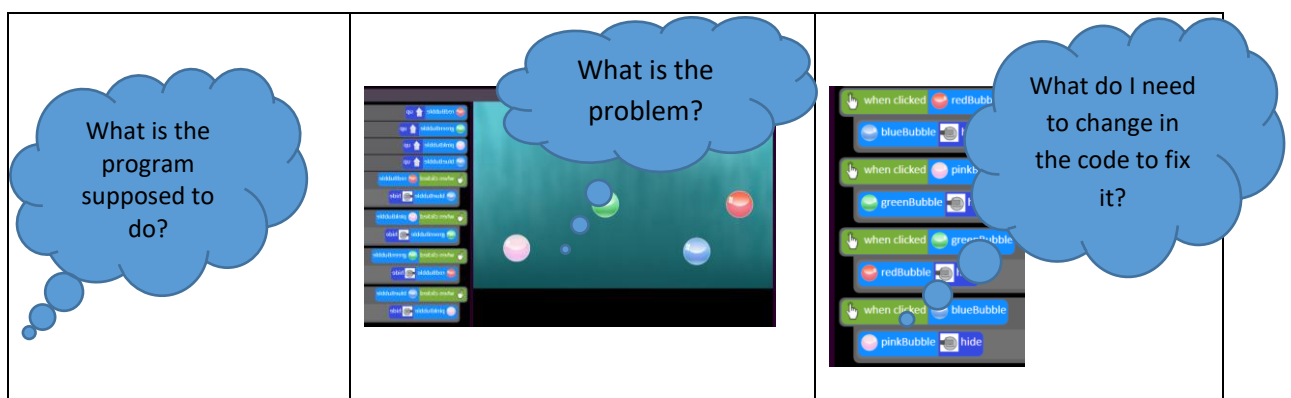
Resources

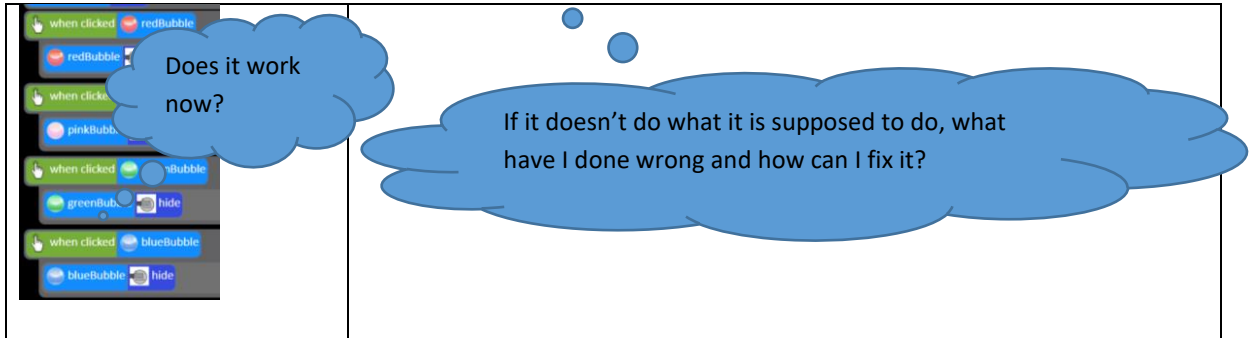
Unless otherwise stated, all resources can be found on the [main unit 2.1 page](#). From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Vocabulary flash cards.
- [Debug Challenges Chimp](#) these can be found on the [main 2Code page](#). Set them as a 2do for your class.

Activities

1. Today we are learning another new piece of vocabulary: debugging. Ask children if they know what debugging or to debug means? Explain that debugging means to fix code. Often, programmers make a mistake when they are coding, or they find that a program doesn't work, so they have to find and fix the mistakes to make the program work. The problems are called bugs and solving them is called debugging.
2. Before you start to debug a program, you need to think through some steps. Put the following on the board (there is a PDF of this table in the Downloads folder). Children could write/stick the steps into their 2Code workbooks.





3. Children should open [Debug Challenges Chimp](#) on their device and complete the challenges. They should save each challenge once they have completed it.
4. Once children have completed the Debug Challenges, bring them back together and open Free Code Chimp on the board. Explain that they will be using the program they created in the last lesson and that they will be 'breaking' it for their partner to debug.
5. Children should open one of their previously made programs and make a written record of what their program is supposed to do.
6. They should then break their code by changing a line of code. They should then write down what the problems are with the broken program.
7. They should then write an instruction for their partner, similar to the ones seen in the Debug Challenges, to help them debug their program.
8. Children should save their program in a class folder using their name in the title. Children open their partners' programs and try to debug the code using the steps they have in their workbooks/on the board.
9. Once the children have finished, they should check that the programs work according to the explanation that the initial children specified in their workbooks. If a program does not work according to the explanation, the code should be fixed until it works.



Lesson 6 – Exploring the Possible Actions of Objects

Aims

- To create programs using different kinds of objects whose behaviours are limited to specific actions.
- To predict what the objects will do in other programs, based on their knowledge of what the object is capable of.
- To discuss how logic helped them understand that they could only predict specific actions, as that is what the objects were limited to.

Success criteria

- Children can create a computer program using different objects.
- Children can predict what the objects in classmates' programs will do, based on my knowledge of the objects' limitations, e.g. a turtle can only move in specific ways.
- Children can explain how they know that certain objects can only move in certain ways

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 don't lose this page.

- [Logical Reasoning Example](#).

Activities

1. Today we will be creating our own programs using different kinds of objects. When used in programming, review what 'objects' are. The image below shows all the possible objects in Chimp, children will not have used all of these.


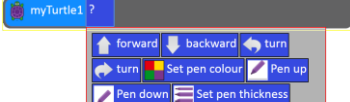
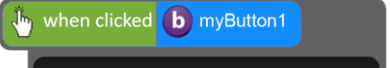


2. Remind children where objects can be found in Design Mode in Free Code Chimp.
3. Explain that in a computer program, objects can only do what we, the coders, tell them to do. Some objects can only do certain things.



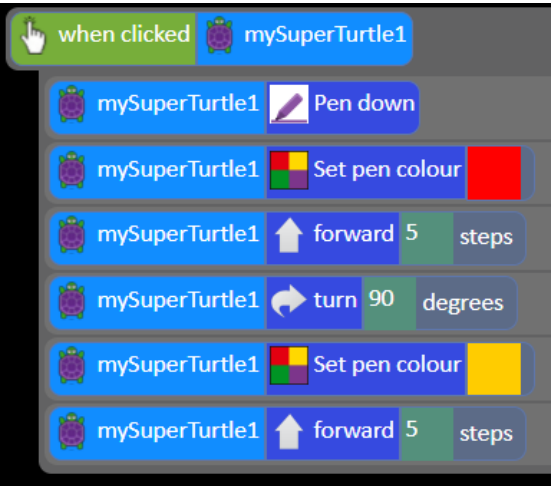
Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 6

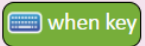
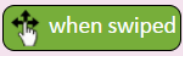
4. In this lesson we will only be using turtles and characters. In chimp freecode switch to design view and drag in a character, a turtle and a button. In code view, go through the different possible actions of the

<p><u>Characters</u></p> <p>Food, animals, princesses, heroes and emoticons can go up, down, left, right, hide and show.</p> 	<p><u>Turtles</u></p> <p>Can go forwards, backwards, turn clockwise and anticlockwise.</p> 	<p><u>Buttons</u></p> <p>Can be clicked.</p> 
--	---	--

objects.

5. Ask children to create a program in Free Code Chimp using one turtle and the When Clicked command. They should get the turtle to do three or four different things in sequence when clicked. For example:



6. All children should save their programs and come back together. Open one of the children's programs on the board in Design View and ask the children to predict what the turtle will do when they click on it. The turtle is limited to going forwards, backwards, turning left or turning right, so any of those answers is potentially correct. Go through two to four programs, getting the children to guess what the turtles might do, based on their logical knowledge of the turtles' limitations.
7. Explain to children that now they will be adding characters to their programs. They should add a character and then use the  command to attribute five actions to their character. Show on the IWB that characters can do more than turtles, so the children have more scope in terms of what to do with them. [Logical Reasoning Example](#).
8. Send children to devices to work on their programs. They should code a character to do something when the arrow keys and the spacebar are pressed. If children are using touchscreens, you will need to choose different keys or, alternatively use the  command and explain how to use this to the children.



Purple Mash Computing Scheme of Work – Y2 Crash Course – Lesson 6

9. Once children are done, they should save their programs and come back together at the whiteboard. Bring up a child's program on the board in Design View and ask them to predict what the character will do when it is clicked. Characters can go up, down, left, right or stop, so any of those answers are potentially correct. Repeat with two to four more programs, asking children to predict both turtles' and characters' actions, using their knowledge of the character limitations of movement to guide them.



Appendix I: Other features of 2Code

Real Code Mode:

On the more advanced lessons and Free Code modes it is possible to click the real code button and see the code in a simple subset of Javascript. The code can be edited in “real code” mode and clicking the “edit blocks” button will bring the user back to the usual graphical representation. If the user types code into real code window that is syntactically incorrect the real code window will flash red. Changing back to “edit blocks” will restore the code to the last state that was syntactically valid.

Sharing:

Once a 2Code program has been saved into Purple Mash, it can be shared by clicking on the globe icon in the toolbar. This will display a window with a hyperlink and an embed code. Clicking the hyperlink will launch the 2Code program in a web browser in fullscreen mode and the embed code can be used to embed a 2Code program into a blog or a website. A Purple Mash login is not required to run a shared 2Code program.



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	<p>Children have a basic understanding that coding involves writing instructions that a computer can follow.</p> <p>They are developing their understanding that these instructions must be precise and carefully structured through their work in Free Code Chimp making simple one and two step programs for example in the lesson 1 bubble program or making an object move when clicked on in (lesson 1).</p> <p>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 (lesson 1) ‘when the bubble is clicked it moves up’.</p> <p>With support, children can manipulate how their program looks using the 2Code design mode, by adding and changing backgrounds, characters (lesson 2), sounds (lesson 3) and objects. They can create a program that controls a character. They can make a character move when clicked but might not be able to plan how to make a character move when a different character (or the background) is clicked.</p> <p>Children are beginning to understand that they can correct unexpected outcomes by changing the code and they make attempts to identify the source of bugs.</p> <p>With support, children can explain the possible actions of objects including movement, clicking on them and collision. When looking at a simple program they can ‘read’ the code one line at a time but might not be able to envision the bigger picture of the overall effect of the program. Children will be able to suggest that an object might move when clicked but might not be able to suggest that an object might move when the background is clicked.</p>
Expected	<p>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 freecode Chimp. For example, in Lesson 1, step 16 they have been able to make a program that follows their algorithm e.g. ‘when the bubble is clicked it hides’. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code.</p> <p>In lesson 4, children used a planning format on paper before implementing on screen within 2Code as they recognise this is the best approach for designing a solution efficiently.</p> <p>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 actions to objects. They confidently include objects, actions, events and outputs successfully within their 2Code programs.</p>



Assessment Guidance

	<p>Children can talk through code which contains repeat and timer commands, explaining where they are positioned and what will happen (lesson 4).</p> <p>Children can predict program outcomes and attempt to debug (lesson 5). 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.</p> <p>Children can debug their own and other’s programs using design documentation to test against (Lesson 3).</p> <p>Most children will be able to save their files using a memorable file name e.g. their name and a simple title etc.</p>
Exceeding	<p>Children can explain and give examples that an algorithm is a set of instructions to complete a specific task.</p> <p>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.</p> <p>Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code.</p> <p>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.</p> <p>Children can identify and correct errors. 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.</p> <p>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, Lesson 6.</p>