**Computing Medium Term Planning Cycle B**

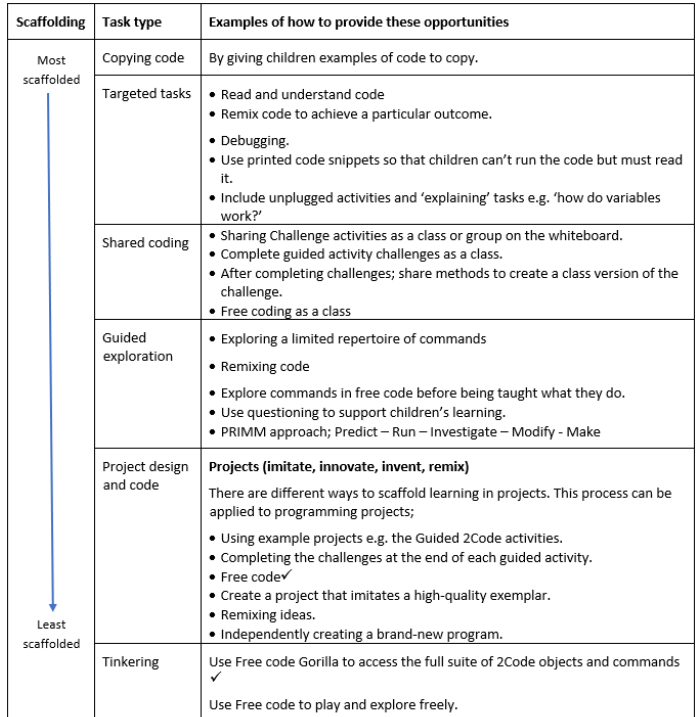
|  |  |  |
| --- | --- | --- |
| **Term:** Autumn 1 | **Year:** 4/5 | **Topic/Unit: Year 4 coding crash course** |
| **Key Vocabulary**  **Lesson 1- coding, event, program, algorithm, object, action, command**  **Lesson 2**- **instructions, algorithm, scene, predict, execute, debug, collision detection**  **Lesson 3- button, object, action, key event**  **Lesson 4- timer, interval, sequence, flowchart**  **Lesson 5-selection, IF statement, IF/ELSE Statement, command, predict**  **Lesson 6-** **variable, number variable**  **Lesson 7- interactive, trigger, show/hide, nesting, alert** | | |

**Evidence: Please can all class teachers generate an example of work that is Working towards, Working at and Working to GD for each unit and upload them to the computing curriculum folder (or email to Louise) for monitoring. Evidence is collected for Computing Folder in SLT room.**

**Important:**

Follow PRIMM method: Predict… what this code will do, Run… the code to check your prediction, Investigate… trace thought 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.

Adaptions: If this is one of your first coding lessons with the micro:bit, it may be hard to know which children will need more support. Each lesson contains tasks that all children will have a go at completing. Some children will need more peer and/or teacher support to complete these successfully. Working in mixed ability pair groupings may help to ensure all pupils are able to participate in each lesson’s main activities. Each lesson also contains extension ideas to challenge those children who complete the task more quickly than others. The extension activities are optional and may not be completed by all pupils. Please see the Assessment Guidance section at the end of this document to support your teacher judgements on pupil outcomes throughout the lessons.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **National Curriculum** | **Week** | **NC Coverage** | **Skills taught** | **Knowledge** | **Activity Outline** |
| design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various  forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs  select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information | 1 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various  forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs | To understand what coding means in computing.  • To use code to make a computer program.  • To understand what objects, actions and events are.  • To use an event to control an object | • Children can explain what coding is.  • Children know that for the computer to make something happen, it needs to follow clear  instructions.  • Children can create a program using event, object and action code blocks.  • Children can explain what events, objects and actions do in a program. | *Preparation: This is a longer session- ensure you have enough time.*  *•Set Fun with Fish as a 2Do.*  *• Set Bubble Coding as a 2Do.*  *• NB: This lesson introduces quite a few fundamental coding concepts in one session.*  *Depending on the previous coding experience and ability levels of your children, this lesson may need to be split over two sessions. Set using the following objectives:*    Discuss new vocabulary with the children on vocab slide.  Slide 5:  Slide 6: introduce children to coding/computer programming.  Activity 1: slide 7 to explain to the children that you are now going to be the programmer and they are all the robots.  Reveal the instructions on the board as symbols. Get children to ‘act’ out / follow the instructions you have displayed as symbols – a twirl, a hand next to a toe and a hand next to an ear - the children should twirl, touch their toes then touch their ears.  Activity 2  Use slide 8 to display a hand next to an up arrow and see if the children can see that this would be ‘hand up’! Ask children to use small whiteboards draw symbols for ‘hand down’, what  about touch nose.  Slide 9: Now that children have practised receiving instructions in code represented as symbols, reiterate the introduction using this slide. Explain that a coder writes instructions in code for the computer to follow, this is called the input. These instructions make our programs work, our programs are the output.  Slide 11: Explain to the children that the light blue code blocks  represent objects, and the dark blue code blocks represent actions. This set of instructions can be called an algorithm.  Slide 12: Explain that a single instruction is called a command.  Slide 13: demonstrate Fun with Fish: . Open Purple Mash and go to 2Dos, click on Preview within the Fun with Fish 2Do to show children the Fun with Fish lesson. Open stage 1 and click on OK to close the instruction screen. Click on ‘Design’ (in the top right-hand corner) and discuss what can be seen – a fish in the sea. Explain to children that they will use 2Code to program the object (fish) to do an action (move right).  Slide 14: Click  to go back to the code view.  Open the instruction screen by clicking on the  . Watch the video for stage 1.  Slide 15: Complete stage 1 as a class; emphasise the need to give the computer clear instructions for moving the fish.  The available actions for the fish object pop-up as soon as the fish is dragged into the code window.  Show the children what to do if they click on the wrong direction - click on the direction again and select the correct one.  Show the children where the Run button is to run the code and emphasise that the code has programmed the object to do an action.  Show them how to move to the next stage of the activity or stop the code running to make changes.  Complete stage 2 together as a class.  Slide 16: Ask children to log in to Purple Mash, go to their 2Dos and click on ‘Start’ on the Fun with Fish 2Do. Challenge them to complete stages 1 and 2. Ask them to use the code blocks to make their Tuna move, and then move onto the next challenge to make the Crab move.  Slide 17: Load stage 3 and explain that this is a stage where you must fix the code that the monkey has got wrong. We call this debugging. Complete this stage as a class (show children that if you want to change an action you can click on it).  Children to then complete stage 3 in Purple Mash independently.  8. Review progress together - did they get lots of code monkey stars? The maximum is 5; they lose stars for using hints.  Slide 18: Look at stage 4 together – this is the challenge stage. All the guided activities have this challenge stage, and this is where children deepen their understanding of the code that they have been working on. Take a few suggestions from the class about how to improve the fish tank by adding new objects – fish / crabs – add one new object then switch to the code screen to notice it then appears as a blue object code piece, show how to program it to move and test it out using the Run button. Add one or two more objects and show children how to use the event. Ask children to complete the challenge stage and then save their work before they exit.  Review children work together against the lesson aims – this could be done by sharing some good examples from the 2Dos folder. Did any children try using the ‘when clicked event’? What did that do?  Slide 19: Start by telling children that the ‘when clicked’ code block is an example of what is called an event in coding. Ask the children to describe how the ‘when clicked’ event worked in the last stage of Fun with Fish.  Slide 20: This physical activity will help children understand what events are and how they make things happen:  Get the bubbles out! Blow bubbles. Ask children:  What is the event? (What do you do to make something happen?) What are the objects? (bubble wand, bubble) What is the action? (float).  Show the children what the code might look like for blowing bubbles.  This command block has three parts: the event (when blow); the objects – the bubble wand and the bubble and the action - float.  Show the children what the code could look like for blowing bubbles. (It should look like this:)    Slide 21: discuss other event – object – action examples children might be familiar with (e.g. push – swing – swing - forward, kick – football - football –roll). Which part of the code is the event? Which is the object? What action does the object do?  Rearrange the code for the football example (this could be done physically using Example Code printed on paper or as a drag into place activity on the board using slide 19).    Plenary:  Review meaning of vocabulary (click on words to reveal the definitions). Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hand |
| 2 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various  forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs | • To understand what an algorithm is.  • To create a computer program using an algorithm | •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. | *Preparation:*  *Set Air Traffic Control as a 2Do. Use these objectives for assessment:*    Discuss new vocabulary with the children on vocab slide. (use the coding vocab quiz and the vocabulary cards to find the answers and display in the classroom)  Go through PM slides.  Activity 1: . 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.**  Slide 9. 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.  Slide 10: discuss the process of making a computer program. Look at the plan for the airport program. This time you want children to concentrate on implementing this algorithm. Discuss what the objects in this program are (the planes and the helicopter); what are the actions? (the planes go up and right); what events are used to make these actions happen? (a click event is needed to make the objects move)  Activity 2: Use it to open Air Traffic Control.  Watch the video for stage 1 together as a class. Remind children that there are hints if they need them, and that once they’ve clicked on OK they can get back to the hint by clicking on the instruction at the top.  Ask children to come up to the front of the class and use the code blocks to make the plane take off when it is clicked on:  Click on  to test the code and see if it works as they were expecting. When you do this, notice with them that the code highlights orange when it executes (you may need to click on the stop button and re-run the program to point it out). Explain that if you click on the plane before the code executes it won’t take off, they need to make sure the code executes first.  Get children to watch the code and see what happens when you click on the planes, which bit of code executes when?  **Children then have a go themselves for stage 1 and 2.**  Use slide 13 to support looking at stage 3 together as a class. Discuss what debugging is – detecting and fixing any errors in the code. Demonstrate how you can drag code around to move it into a different place and click on actions to change them. Fix the code together and click on Next Challenge. **Set children off to complete the debugging stage.**  Activity 2: Air Traffic control. Ask children to return to 2Code and continue working through Air Traffic Control until they reach the final stage, then challenge them to start adding the code to make the first 3 steps of the algorithm work. Use slide 18 to 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.  When the majority of the class have programmed the click events, draw their attention back to slide 19 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)  Ask children to hand work in- Ask children to work with a partner to read through each other’s code and predict what will happen when they run the program.  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  **Good questioning for evaluations for chn to discuss to aid your marking:**  • Does it work as you expect?  • If not, do you need to debug the code and download it again?  • How good is the project?  • Would you recommend it to a friend? • How could you improve it?  • Could it have other uses?  • How does it work?  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions). Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hands. |
| 3 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.  use sequence, selection, and repetition in programs; work with variables and various forms of input and output.  use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs. | • To understand that different objects have different attributes.  • To create a program using a given design.  • To understand the function of buttons in a program | • Children can create a computer program that includes different object types.  • Children can create a computer program that includes a button object.  • Children can modify the attributes of an object and a button to fit their program design.  • Children can explain what a button does in their program. | *Preparation:*  • set Free Code Chimp as 2Do. (Set Road scene as 2Do for less confident children).    Recap vocabulary from previous work, introduce new vocab for lesson. Go through purple Mash slides. (lesson follows same process as previous one; create, test, debug, run and evaluate.  Objects and Actions: Display slide 5 and introduce today’s topic of objects and actions.  Snail Race: Display slide 6. 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 7. Ask children to predict what will happen when the program is run. Run the program Turtle and Character, interact with it and see what happens.  Designing a Scene: Talk through slide 8 and then open Free Code Chimp in front of the class. Follow the instructions on the slide to set the scene.  Choosing Objects: Return to your design in Free Code Chimp and look at the object types to choose from on the left. Add a turtle and 3 other objects that would move (tell children that in this part of the lesson we are using any object type apart from the button – we are going to look at the button object type later).  Changing Objects: Display slide 10. Talk through how to change the objects and the size of the objects. Recap how to move the objects around. This is the first-time children have used Free Code Chimp in coding lessons so spend a bit of time in 2Code browsing the clipart galleries – pointing out the categories and search option.  Activity 1: Use slide 11 to 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 ‘See Code’ to start adding some code. Talk through slide 12, thinking about the events and the ‘when’ key.  The ‘when’ key is an event command. It makes code run when you press the specified key on the keyboard. In the example on this slide, the when key event will run when you press the up arrow on the keyboard.  Actions for Objects: Use slide 13 to re-cap how to program objects to do actions. Point out that the actions for a turtle are different to the other ones. When adding code for the turtle, in this lesson we are going to program it just 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). In 2Code open the design you created earlier in the lesson and, with the children, add code to program some of the objects to move. Ask them to predict what will happen, then run the code.  Adding to Your Code: Display slide 14. Challenge the children to add their own code to their programs. To give them more support you could return to your program in 2Code, delete the code you had in there and ask children to help you use timers to make the objects start at different times. (They will learn more about timers in the next lesson, so you may wish to leave it out for now.) 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.  Adding Buttons to Your Code: Display slide 15 to introduce what a button is and what it does.  Activity 2: Add a Button- Display slide 16. In 2Code, add a button to your scene.  Button Attributes: Display slide 17 and look at the button attributes. Name the button and set the text for the button. You can also change the text size, text colour and background colour of the button.  Coding Your Button: Display slide 18. Look at the code blocks and ask children to help you add code to make one of your objects move when the button is clicked on.  Activity 3: Program Your Scene: Display slide 19. 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. They could also add in more buttons to make objects move.  NB: If pupils are coding on tablets, the when key event is not available. Instead, they could try using when clicked or when swiped events.  Display slide 20 and ask children to save their designs. Share great examples with the class, discussing the code that has been used to make them work  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  **Good questioning for evaluations for chn to discuss to aid your marking:**  • Does it work as you expect?  • If not, do you need to debug the code and download it again?  • How good is the project?  • Would you recommend it to a friend? • How could you improve it?  • Could it have other uses?  • How does it work?  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |
| 4 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various. forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs. | • To understand that there are different types of timers.  • To be able to select the right type of timer for a purpose. | • Children can create a program that uses a timer-after command.  • Children can create a program that uses a timer-every command.  • Children understand there can be different ways to solve a problem | *Preparation:*  *• Set Magician as a 2Do.*  *• Set Night and Day as a 2Do.*  *• Set Tick Tock Challenge as a 2Do.*  *You can select the following objectives for assessment:*    Recap vocabulary from previous work, introduce new vocab for lesson. Go through purple Mash slides. (lesson follows same process as previous one; create, test, debug, run and evaluate.  Instructions With Delays: Display slide 8. Look at the flowchart together and then ask the children to follow the instructions it gives. Ask children to draw their own version on an individual whiteboard or piece of paper. Ask children to swap their flowchart with a partner and have a go at following each other’s instructions.  Magician: Work through slide 9 and watch the video for stage 1 of the Magician guided lesson.  Magician - Stage 1: Display slide 10. Use the slide to talk through stage 1. This is a bit like using an event code block – it sets a timer and after the specified time the object (rabbit) will hide.  Open Magician and work through the first stage as a whole class. Watch the videos and remind them that they can unlock a hint if they get stuck.  Make mistakes as you add the code and get the children to help you debug and fix the problems.  Magician - Stage 2: Display slide 11. Work through stage 2 together. Add code incorrectly first (see below), then test it and ask the children to help you debug:    This code doesn’t work because when you run the program both timers start at the same time (if you click stop and Run 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. 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 the solutions shown on the slide would work.  Activity 1: Magician Display slide 12. Ask children to start Magician from their 2Dos and work through stages 1-4 independently.  Activity 2: Night and Day: Display slide 13. Ask children to start the Night and Day activity from their 2Dos and see if they can complete it. This works in a very similar way to Magician.  Activity 3: Tick Tock Challenge: Display slide 14. Once they have completed Night and Day and they have recapped using timer-after, tell children that there is another kind of timer, and they are going to learn about it by working through the Tick Tock Challenge. Set them to start and complete this challenge from their 2Dos.  Review how they have got on – what have they learnt? Ask children: What is the difference between timer-after and a timer-every?  Activity 4: Extension Display slide 15. Ask children to look at the scene and read the code, then predict what would happen when the code is run. Discuss with children how they could use a timer-every command to develop this program.  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  **Good questioning for evaluations for chn to discuss to aid your marking:**  • Does it work as you expect?  • If not, do you need to debug the code and download it again?  • How good is the project?  • Would you recommend it to a friend? • How could you improve it?  • Could it have other uses?  • How does it work?  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions). Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hands. |
| 5 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various. forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs.  . | To begin to understand selection in computer programming.  To understand how IF and IF/ELSE statements works. | • Children can create a program that includes an IF and IF/ ELSE statement.  • Children can interpret a flowchart that depicts an IF and an IF/ ELSE statement.  • Children can read code that includes an IF and an IF/ ELSE and explain how they work. | *Preparation:*  *Set ‘Lost’ 2Code Example as a 2Do.*  *• Set Free Code Gibbon as a 2Do.*  *You can select the following objectives when setting the 2Dos to make future assessment easier:*  *• Print copies of the* [*IF/ELSE Flowchart Template*](https://static.purplemash.com/mashcontent/applications/flashcards/y4_4_Coding_Crash_Course_Template/IF%20ELSE%20Flowchart%20Template.pdf)  *Teacher video to watch that aids understanding*  Lesson outline:  Go through PM slides, introducing vocabulary.  IF Statement: Say to the children ‘IF my class is quiet for 30 seconds, then I will [insert action/ activity here!!]. Start a timer and then check IF statement is true. If it is, carry out stated action/ activity.  In pairs, ask one child to write an IF statement on their small whiteboard, then the other to check if it’s true and run the action if it is, or not if it isn’t.  Discuss as a class: When tested, were any not true?  Explain that in code we can use IF statements to help our programs work – for example, IF the countdown has reached 0 the game is over, or IF the score equals 10 the ‘amazing’ sound plays.  Selection Video: Display slide 7. Play Selection video to children (Video should play from slide).  Is It Raining?: IF Statement: Use slide 8 to display Is It Raining? 2Code activity – show how the chart in the video looks in a program – look at the design together; two people under some rain clouds and a hidden umbrella (you can hide objects at the start using the attributes table). Talk through the code –it starts with a prompt for input. If the user notices the rain clouds and puts ‘yes’ into the input, the IF statement runs and shows the umbrella.  Comparing IF and IF/ELSE statements: Use slide 9 to introduce an IF/ ELSE statement using the ‘Is it Raining?’ example:  In this flowchart, if the answer to ‘Is it raining?’ is yes, then the  umbrella shows. If the answer was no, then nothing changes. However, we now want to program something to happen if the  condition is not met e.g. program something to happen in our scene if it is not raining. There is a suggestion on the slide - if it is not raining, the clouds will disappear and everyone will be happy!  Ask the children to discuss how the flowchart could be adapted for this example.  Use slide 10 to show children what the [Is it Raining IF ELSE Flowchart](https://www.purplemash.com/app/games/2diy/example_flowchart_rainIFELSE_2021) could look like:      Talk it through with the children, then go back to the Is It Raining? 2Code activity and ask them to help you develop the program to reflect the changes to the flowchart (ask them to look at the code blocks and see if they can pick out IF/ ELSE as a sensible one to use, if not direct them to it). The code could look like this:  Run the program a couple of times, once typing ‘yes’ in the prompt for input box, once typing ‘no’. Notice how the code executes and what happens each time.  Lost: Follow slide 11: Look at the design for the ‘Lost’ program together and notice that there is a background and 2 objects.  Look at the code and see if children can ‘read’ the code and predict what will happen when the program is run. Use the link on the slide to open’ Lost’ in 2Code.  Run the program twice, putting in different inputs to see what  happens. Delete the code and see if the children can help you put it back in again – you may need to emphasise the difference between alert and prompt for input. Click on ‘Design’ and remind children how to change the backgrounds and objects – remind them to change the name of an object in the attributes table if they change it so the name matches what it is.  Activity 1: Lost :IF/ELSE Use slide 12 to ask children what could happen if the fish says no, they don’t want to go to the sea. What could happen instead? Tell children that in this lesson they will be making their own ‘Lost’ program in 2Code. Ask them to think about how this example could be developed to include an IF/ ELSE statement, then challenge them to draw the flowchart for their program on the IF/ ELSE flowchart template – either using the template on the front or by drawing their own on the back.  Ask children to open the ‘Lost’ code activity from their 2Do area and work independently or with a partner to modify the ‘Lost’ code –changing the IF statement into an IF/ELSE statement that matches their planned ideas  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |
| 6 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various. forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs. | To understand what a variable is in programming.  To use a number variable. | • Children can explain what a variable is in programming.  • Children can create and use variables when programming.. | *Preparation:*  *Set Genie as a 2Do.*  *• Set Night and Day as a 2Do (if using extension)*  *• You can select the following objectives when setting the 2Dos to make future*  *assessment easier.*  *• Print and cut up 2 copies of number cards 0-23.*  *• Print and stick 4 Variable Game Cards under 4 children’s chair*  Go through PM slides, introducing vocabulary.  Slide 6. Use this slide to help you explain what a variable is. On this slide each box is a variable. They both have names - the first variable is called ‘team1score’ and the second variable is called ‘team2score’.  The variable values are determined by how well (or not) each team does in a quiz.  Slide 7. Put two actual boxes on a table so that all the  children can see them, and label them ‘team1score’ and  ‘team2score’ as in the slide. On the board, write 'team1score=' and team2score='.Read the IF/ ELSE statements that will have an impact on these variables –‘If a Team 1 answer is correct, the value of team1score will increase by 1, else the value of team1score will stay the same’.  ‘If a Team 2 answer is correct, the value of team2score will increase by 1, else the value of team2score will stay the same’.  Tell children that they will be taking part in a class quiz that will have an impact on these variables. Split the class into Team 1 and Team 2.  Display the quiz on slide 8 and play it: Team 1 play the first question. When they answer, check if it is correct and then react in the way the IF/ ELSE statement directs –use a number card to show the variable value in the team1score variable box (put the number card in the box) and add a value to ‘team1score = ‘on the board.  Ask team 2 the next question – repeat as above until all the  questions have been answered – changing the number cards in the boxes, and the values of the relevant variables each time an answer is given.  Emphasise that the variable value is replaced with the new value  each time – a variable holds only 1 value.  Look under your chair: Display slide 9. Ask children to look under their chairs – four should find the Variable Game Cards you stuck underneath them. The cards say something like the following:  Bonus points: add 4 to your team’s score.  Bonus points: double your team’s score.  Disaster card: subtract 2 from your team’s score.  Disaster card: halve your team’s score.  NB. At this point your variable value might increase to a number  higher than you have number cards for, if this happens create the relevant extra values out of post-it notes or scrap paper!  Who won? Discuss how the answers impacted the value of the  variables and emphasise the importance of naming variables  sensibly.  2Code Genie: Use slide 10 to open the Genie activity.  Stage 1: Complete together – when you create the variable point  out that there are different types of variables and in this lesson, we are choosing ‘number’. Creating the variable is a bit like making the box, our box in the classroom was named score, this one will be named ‘count’ as it keeps a count.    When you click on Run to run the program point out the variable  watch box:    Explain that you can’t see this variable in the scene as it’s part of the code.  Display slide 11.  Ask children to open Genie from their 2Dos and try and complete it independently. Remind them that they can click on the instruction to return to the video or unlock hints if they get stuck.  Notes to support children with task:  Stage 2: When children have added in the click event, they will need to add the code block and then select count because they need to change the count variable when the lamp is clicked on. When they click on Run to run the code encourage them to look at the variable watch box and notice how the variable changes each time they click on the lamp.  Stage 3:  At stage 3 the children may make the following error -  Incorrect Code - the IF statement runs and checks to see if the count=3 as soon as you press Run (so only once), and it needs to be triggered to check if the count=3 every time the variable changes value.  Correct Code - the IF statement is  checked every time the lamp is  clicked on (the click event triggers the IF statement to run), so if/ when it’s true, the lamp can turn into a genie!  Extension: Night and Day (Gibbon)  Display slide 12. Children have a go at working through Night and Day Gibbon that you have set as a 2Do  **Ask children to ‘hand in’ tasks with an honest review of how**  **they got on.**  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |
|  | 7 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts   use sequence, selection, and repetition in programs; work with variables and various. forms of input and output   use logical reasoning to explain how some simple algorithms work and to detect and  correct errors in algorithms and programs. | To design and create an interactive scene | • Children can use the attributes table to set the attributes of objects.  • Children can plan their scene and algorithms before they create their program.  • Children can confidently make several different things happen in a program. | Preparation  • Print and copy a range of planning documents for children to use in Lesson 7.  • Open the example programs: Lightning Scene, Moon Phases, Solar System and Viking Discovery in 4 browser tab for easy access.  • Set Free Code Chimp for children to refer to in Lesson 7  Children might want to be able to use images that are photographs or not part of Purple Mash when creating their program. You could create a folder of topic-related images in the class folder for them to choose from when they are coding. You can upload these in the work area and then children can select ‘choose file’ from the galleries in 2Code to access them:    Children could alternatively source them, save them and upload them to their own folder or import them directly from their device **(be more aware with online safety and safeguarding if you chose this option)**  Design and Make an Interactive Scene: Display slide 5 and introduce the main activity which will be spread over two lessons. Explain that the main aim of this lesson is to create a plan that would be good enough for someone else to follow to make your program if you didn’t.  Lightning Scene: Display slide 6. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don’t like about each program, and how they might develop it – if they think it needs developing. **Leave the program open in a tab.**  Moon Phases: Display slide 7. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don’t like about each program, and how they might develop it – if they think it needs developing. **Leave the program open in a tab.**  Solar System Display: slide 8. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don’t like about each program, and how they might develop it – if they think it needs developing. **Leave the program open in a tab.**  Viking Discovery: Display slide 9. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don’t like about each program, and how they might develop it – if they think it needs developing. **Leave the program open in a tab.**  Object Attributes: Display slide 10 and talk through the object attributes for the animal object. Open up the program Unicorn Dog Seagull. Click on an object and look at the options in the attributes table with the children. How does changing each attribute affect the object? In the ‘Lightning Scene’ most of the objects in the design were hidden, and they were programmed to show when they were needed the sequence.  Set a sensible name for the objects in this scene. If ‘allow off screen’ is set to ‘yes’, the object will be allowed to move off the screen in the direction it’s moving and it will disappear. If it is set to ‘no’ it will move off the screen in the direction it’s moving, but come back on the other side, moving in the same direction. You could add a simple ‘animal right’ command to demonstrate this.  Different Objects: Display slide 11. Remind children that if there isn’t an object type that matches what they want to add (e.g. they want to add a car) they can choose one that is there (e.g. animal) and then change the image and the name. If you return to ‘Lightning Scene’ and click on the fire engine you’ll see in the attributes table that the fire engine is actually an animal object with the image and name changing it to a fire engine. Show/Hide – Don’t Forget Nesting!: Display slide 12 and ask children to read each example of code and predict what would happen when the program is run. Go back to Unicorn Dog Seagull and program the second example: set the show/ hide attribute for the seagull to hide and then add code to  make it show after 2 seconds and say ‘Hi!’ when it shows.  Emphasise the importance of nesting the speak command into the after timer or the seagull will speak but you won’t be able to see it!  Button Attributes: Display slide 13 and remind children of the button attributes. In Unicorn Dog Seagull, drag a button onto the scene and remind children how the attributes options are different.  Alert!: Display slide 14 and use it to remind children of the ‘Moon Phases’ program - notice how there are instructions at the start – explain that these were programmed using an alert.  Planning: Using slide 15, show the children the planning frameworks you’ve printed and copied and encourage them to choose a method of planning that suits them or that they think suits their program. They might favour a scene sketch, the Storyboard Planner or the Scene and Code Planner.  Display slide 16. Give children time to make their plan, recommend that they have Free Code Chimp open in front of them so they can explore backgrounds and clipart/ images available to them as they plan. If they finish they could start making their programs in Free Code Chimp.  **Ask children to ‘hand in’ tasks with an honest review of how**  **they got on.**  **Share positive examples on the board of childrens work. (Good time to add comment to children’s work for marking- expectation of everyone’s work commented on at least once per half term)**  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |