**Year 5 Computing Curriculum – Spring Term 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Theme: Selection in Quizzes | | | | | | | | |
| **Curriculum objectives** | | | **Vocabulary** | | | | | **Links across the curriculum** |
| - 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 | | | **Keyword** | Definition | sequences | a pattern or process in which one thing follows another. | | [**Computing**](https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study)   * 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 | |
| design | to think up and plan out in the mind | decompose | Break down into smaller and manageable ‘chunks’ | |
| Count-controlled loop | A count-controlled loop is a form of repetition in which a set of commands are carried out a specific number of times. | Condition-controlled loop | A condition-controlled loop is a form of repetition in which a set of commands stop being carried out when a condition is met. | |
| debug | to fix | conditions | Conditions are statements that need to be met for a set of actions to be carried out | |
| commands | to order or instruct | program | a plan of what will be done | |
| Infinite loop | An infinite loop is a loop that commands the instruction/set of instructions to repeat forever. | algorithms | a determined and finite procedure for solving a problem | |
| **Prior Knowledge:**  EYFS – To follow two step instructions. Year 1 – Commands for a robot. Year 2 – plan and debug algorithm  Year 3 - Sequencing Sounds Year 4 – repetition in Shapes; Year 5 – Selection in Physical Computing | | | | | **Future Knowledge:**  Year 6 - To choose how to improve a game by using variables | | | |
| **Lesson Sequence** | | **Key Knowledge** | | | | | **Key Skills** | |
| Exploring conditions | | In this lesson, learners revisit previous learning on ‘selection’ and identify how ‘conditions’ are used to control the flow of actions in a program. They are introduced to the blocks for using conditions in programs using the Scratch programming environment. They modify the conditions in an existing program and identify the impact this has. | | | | | To explain how selection is used in computer programs   * I can recall how conditions are used in selection * I can identify conditions in a program * I can modify a condition in a program | |
| Selecting outcomes | | In this lesson, learners will develop their understanding of selection by using the ‘if… then… else...’ structure in algorithms and programs. They will revisit the need to use repetition in selection to ensure that conditions are repeatedly checked. They identify the two outcomes in given programs and how the condition informs which outcome will be selected. Learners use this knowledge to write their own programs that use selection with two outcomes. | | | | | To relate that a conditional statement connects a condition to an outcome   * I can use selection in an infinite loop to check a condition * I can identify the condition and outcomes in an ‘if… then… else…’ statement * I can create a program that uses selection to produce different outcomes | |
| Asking questions | | In this lesson, learners consider how the ‘if… then... else…’ structure can be used to identify two responses to a binary question (one with a ‘yes or no’ answer). They identify that the answer to the question is the ‘condition’, and use algorithms with a branching structure to represent the actions that will be carried out if the condition is true or false. They learn how questions can be asked in Scratch, and how the answer, supplied by the user, is used in the condition to control the outcomes. They use an algorithm to design a program that uses selection to direct the flow of the program based on the answer provided. They implement their algorithm as a program and test whether both outcomes can be achieved. | | | | | To explain how selection directs the flow of a program   * I can explain that program flow can branch according to a condition * I can design the flow of a program that contains ‘if… then… else…’ * I can show that a condition can direct program flow in one of two ways | |
| Designing a quiz | | In this lesson, learners will be provided with a task: to use selection to control the outcomes in an interactive quiz. They will outline the requirements of the task and use an algorithm to show how they will use selection in the quiz to control the outcomes based on the answer given. Learners will complete their designs by using design templates to identify the questions that will be asked, and the outcomes for both correct and incorrect answers. To demonstrate their understanding of how they are using selection to control the flow of the program, learners will identify which outcomes will be selected based on given responses. | | | | | To design a program that uses selection   * I can outline a given task * I can use a design format to outline my project * I can identify the outcome of user input in an algorithm | |
| Testing a quiz | | In this lesson, learners will use the Scratch programming environment to implement the first section of their algorithm as a program. They will run the first section of their program to test whether they have correctly used selection to control the outcomes, and debug their program if required. They will then continue implementing their algorithm as a program. Once completed, they will consider the value of sharing their program with others so that they can receive feedback. Learners conclude the lesson by using another learner’s quiz and providing feedback on it. | | | | | To create a program that uses selection   * I can implement my algorithm to create the first section of my program * I can test my program * I can share my program with others | |
| Evaluating a quiz | | In this lesson, learners will return to their completed programs and identify ways in which the program can be improved. They will focus on issues where answers similar to those in the condition are given as inputs, and identify ways to avoid such problems. Learners will also consider how the outcomes may change the program for subsequent users, and identify how they can make use of ‘setup’ to provide all users with the same experience. They will implement their identified improvements by returning to the Scratch programming environment and adding to their programs. They conclude the unit by identifying how they met the requirements of the given task, and identifying the aspects of the program that worked well, those they improved, and areas that could improve further. | | | | | To evaluate my program   * I can identify ways the program could be improved * I can identify the setup code I need in my program * I can extend my program further | |
|  | | | | | | | | |
| **Themes and links** | | | | | | | | |
| **Computing themes** | **Where these are covered:** | | | | | | | |
| **Technology around us**  Autumn 1 | * Crumble links to the real world and computer games the children know. | | | | | | | |
| **Digital painting**  Autumn 2 | * Design, make and evaluate process | | | | | | | |
| **Programming A**  Spring 1 | * the concept of selection in programming using the Crumble programme | | | | | | | |
| **Data /information**  Spring 2 | * Storing the commands and the effect on language on the outcome of your commands. | | | | | | | |
| **Creating media**  Summer 1 | * Your own designs of Crumble | | | | | | | |
| **Programming B**  Summer 2 | * Using crumble to implement an algorithm as a code | | | | | | | |