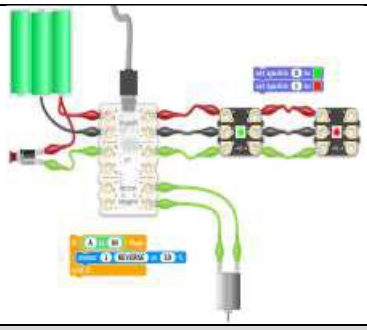
Jeavons Wood
Primary School

Knowledge Organiser for Year 5

Big question: How can using conditions and choices in a program change what happens?

KS2 National curriculum specification
• 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



In this unit, the children will:
Explain how selection is used in computer programs
Relate that a conditional statement connects a condition to an outcome
Explain how selection directs the flow of a program
Design a program which uses selection
Create a program which uses selection
Evaluate my program

Key vocabulary:	
Programming	Writing instructions that tell a computer or device what to do.
Circuit	A path that electricity flows through, usually made of wires and components.
Electricity	Energy that powers devices and flows through circuits.
Microcontroller	A small computer on a chip that controls electronic devices and runs programs.
Code	The instructions written in a programming language that a computer can follow.
LED	A Light Emitting Diode—a small light that turns on or off in electronic projects.
Algorithm	A step-by-step set of instructions for solving a problem or completing a task.
Motor	A device that uses electricity to create movement, often used in robotics.
Modify	To change or improve something, like adjusting your program or design.
Debugging	Finding and fixing mistakes (bugs) in your code so the program works correctly.

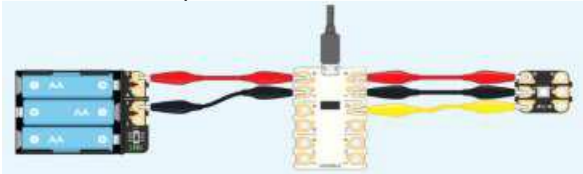
Microcontrollers, LEDs and Motors	Programming Commands
-**Microcontrollers:** A microcontroller is a small device that can be programmed to control devices that are connected to it. -One brand of widely used microcontroller is called a Crumble controller, which can be used to	-For programming, we should use the microcontroller software.

control many things, e.g. LEDs and motors.

LEDs:
-LEDs are output devices that are emit light. When electricity is passed through an LED it produces light. One type of LED light, controlled by a Crumble controller, is called a Sparkle.

Motors:
-Motors are another output device. A motor can start, stop, spin forwards, spin backwards, and go at different speeds.

Creating Circuits:
-The USB port connects the microcontroller to a computer. Crocodile clips pass electricity and data through to the LED/motor.
-The + and - power pads on the Crumble should be connected with the + and - power pads on the Sparkle and battery box. The D pads on the Crumble and Sparkle should also be connected.



-Crumble uses command blocks (like Scratch).
-**Adding/Removing Commands**: To add a command block, drag it from the menu towards the program. When the grey arrow appears, the command will snap into the program. To remove a command block, drag it away from the program and back to the menu.
-Modifying Commands: Clicking on the colour square in the command block allows us to change the Sparkle's colour. To change the time of commands, click on the value. Delete the current value and type in the new value. Press enter after completed.
-Count Controlled Loops: These allow us to put programs on a loop. Count Controlled Loops are found in the 'Control' options. Drag the desired program into the Count Controlled Loop command block. 'Do until' loops allow commands to happen until a condition is met.

Sequencing and Algorithms	Trialling and Debugging
-A **sequence** is a pattern or process in which one thing follows another. -We design **algorithms** (sets of instructions for performing a task) to help us program sequences involving multiple output devices (e.g. LEDs and motors). -**Programming** is the process of keying in the code recognized by the computer into the software (using your algorithm). 	-Programmers do not put their computer programs straight to work. They **trial** them first to find any errors: -Sequence errors: An instruction in the sequence is wrong or in the wrong place. -Keying errors: Typing in the wrong code. -Logical errors: Mistakes in plan/thinking. -If your algorithm does not work correctly the first time, remember to **debug** it. 

Teacher Information:	
Subject Knowledge	This unit focuses on developing learners' understanding of selection in an on-screen context. It highlights what 'conditions' are and how they are used as part of 'selection'. This unit also develops learners' understanding of design in programming, using the approach outlined below. **Levels of abstraction** When programming, there are four levels which can help describe a project (known as

	Levels of abstraction). Research suggests that this structure can support learners in understanding how to create a program and how it works: • Task - this is what is needed • Design - this is what it should do • Code - this is how it is done • Running the code - this is what it does Spending time at the 'Task' and 'Design' levels before engaging in code-writing aids learners in assessing the 'do-ability' of their programs and reduces a learner's cognitive load during programming. Learners will move between the different levels throughout the unit and this is highlighted within each lesson plan. **Conditions** 'Conditions' are statements that need to be met for a set of actions to be carried out. They can be used in algorithms and programs to control the flow of actions. When a condition is met it is referred to as 'true' and when it is not met it is referred to as 'false'. You need to be able to identify and use conditions in algorithms in the form of statements to both start and stop sets of action. Additionally, you need to understand that conditions can be used in loops, and when they are, that the set of actions in the loop will be carried out repeatedly until the condition is true. For example, 'until button 'A' is pressed'. **Selection** When designing programs, there are often points where a decision must be made. These decisions are known as 'selection', and are commonly implemented in programming using 'if' statements. Selection is used to control the flow of actions in algorithms and programs by checking whether a condition (see above) has been met. If it has been met, the identified actions will be carried out. When selection is used in programs, infinite loops (see above) are often used to instruct the device to check the condition repeatedly. Without using loops, the condition would only be checked once following the sequence of the code
Progression	This unit assumes that learners will have prior experience of programming using block-based construction (eg Scratch), understand the concepts of 'sequence' and 'repetition', and have some experience of using 'selection'. Ideally, learners will have completed 'Programming A – Selection in physical computing' before undertaking this unit, as this will provide them with the required knowledge of 'selection'.
Unit overview	In this unit, pupils develop their knowledge of 'selection' by revisiting how 'conditions' can be used in programming, and then learning how the 'if… then… else…' structure can be used to select different outcomes depending on whether a condition is 'true' or 'false'. They represent this understanding in algorithms, and then by constructing programs using the Scratch programming environment. They learn how to write programs that ask questions and use selection to control the outcomes based on the answers given. They use this knowledge to design a quiz in response to a given task and implement it as a program. To conclude the unit, learners evaluate their program by identifying how it meets the requirements of the task, the ways they have improved it, and further ways it could be improved.