Year 5 – Programming B - Selection in quizzes

Unit introduction

In this unit, pupils develop their knowledge of selection by revisiting how conditions can be used in programs 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 answer 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; further ways it could be improved.

There are two year 5 programming units:

- Programming A - Selection in physical computing
- Programming B - Selection in quizzes

This is unit B which should be delivered after unit A.

Overview of lessons

Lesson	Brief overview	Learning objectives
1. 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 and identify the conditions needed to control gadgets in the house of the future. They are introduced to the command 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
2. 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 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 with different outcomes using selection
3. Asking questions	In this lesson, learners consider how the If… Then… Else… structure can be used to identify two responses to a binary question (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 design, using an algorithm, 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 if 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 which contains if… then… else… • I can show that a condition can direct program flow in one of two ways

4. Planning 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 design by using a storyboard 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 which 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
5. Testing a quiz	In this lesson, learners will use the Scratch programming environment to implement the first section of the algorithm as a program. They will run the first section of their program to test if 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 get feedback. Learners conclude the lesson by using another's quiz and providing feedback on it.	To create a program which 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
6. Evaluating a quiz	In this lesson, learners will return to their completed programs and identify ways that 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	To evaluate my program • I can identify ways the program could be improved • I can identify what setup code my project needs • I can extend my program further

	improvements by returning to the Scratch programming environment and adding to their program. 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.	

Progression

This unit assumes that learners will have prior experience of programming using block-based construction (e.g. Scratch) and understand the concepts of sequence, repetition and have some experience of using selection. Ideally, learners will have completed programming unit A (selection in physical computing) before undertaking this unit as this will provide them with the required knowledge of selection.

Please see the learning graph for this unit for more information about progression.

Curriculum links

Computing

- 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

Assessment

Formative assessment

Assessment opportunities are detailed in each lesson plan. The learning objectives and success criteria are introduced in the slide deck at the beginning of each lesson, and then reviewed at the end. Pupils are invited to assess how well they feel they have met the learning objective using thumbs up, thumbs sideways, or thumbs down.

We recommend the use of teacher accounts in Scratch to help with assessment throughout this unit. For guidance on setting up teacher accounts, please visit the Scratch website (scratch.mit.edu/educators/faq).

Summative assessment

- Please see the assessment question and answer documents for this unit.

Subject knowledge

This unit focuses on developing learner's 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 pupils' 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, 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 if 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.

Enhance your subject knowledge to teach this unit through the following training opportunities:

Online training courses

- Raspberry Pi Foundation online training courses

Face-to-face courses

- National Centre for Computing Education face-to-face training courses