Computing – Programming: Computational Thinking

abstraction – identifying the important detail and ignoring irrelevant information

algorithm design – creating clear step-by-step instructions to make something work

computational thinking – using logic to solve problems step by step

decomposition – breaking a problem down into smaller, easier steps

logical – makes sense and follows a clear order or pattern

pattern recognition – finding similarities or repeated parts in a problem to help solve it more easily

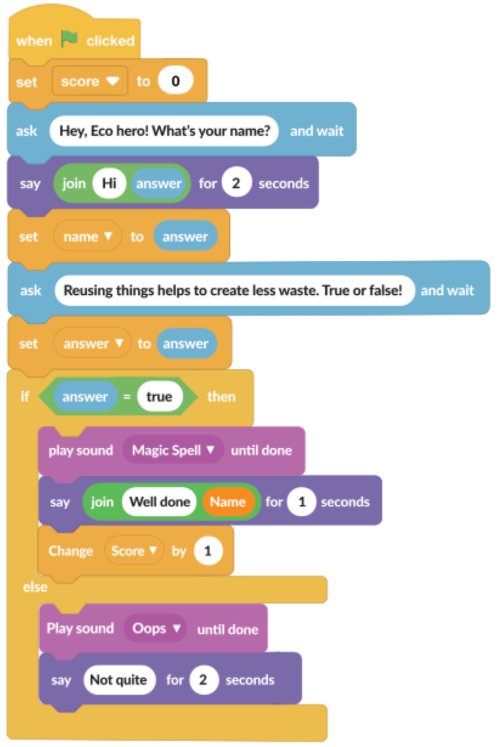
sequence – steps arranged in the correct order to make something work

Remixing code

Remixing code saves time by using ideas from existing projects.

Pattern recognition helps to understand how the code works and algorithm design helps to change it.

Programmers edit code to fix problems, add new features or make it work better.



```
when 🏴 clicked
set score ▾ to 0
ask Hey, Eco hero! What's your name? and wait
say join Hi answer for 2 seconds
set name ▾ to answer
ask Reusing things helps to create less waste. True or false! and wait
set answer ▾ to answer
if answer = true then
    play sound Magic Spell ▾ until done
    say join Well done Name for 1 seconds
    Change Score ▾ by 1
else
    Play sound Oops ▾ until done
    say Not quite for 2 seconds
```

Real-life examples of computational thinking

Planning a journey

When planning a journey, the route, stops and transport are decomposed into smaller steps.

Cooking a recipe

Cooking requires a sequence of steps to be followed.

Solving a jigsaw puzzle

Looking for patterns and grouping pieces together helps solve a jigsaw.