

## 7.3 Programming Essentials in Scratch

### Revision Guide - 7.3 Programming Essentials in Scratch

This unit is the first programming unit of KS3. The aim of this unit is to build learners' confidence and knowledge of the key programming constructs. Importantly, this unit does not assume any previous programming experience, but it does offer learners the opportunity to expand on their knowledge throughout the unit.

The main programming concepts covered in this unit are sequencing, variables, selection, and count-controlled iteration. All of the examples and activities for this unit use Scratch 3.

#### Lesson 1: Introduction to programming and sequencing

##### Objectives:

- Compare how humans and computers understand instructions
- Define a sequence
- Predict the outcome of a simple sequence
- Modify a sequence

##### Key vocabulary

Sequencing, subroutines, instructions, execute

##### What is programming?

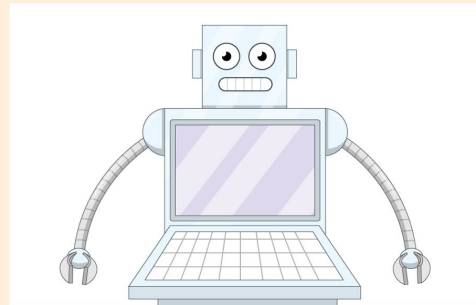
"Programming is how you get computers to solve problems."

"Programming is how *you* get computers to solve problems"

There are two key phases that are important here:

**You:** Without the programmer (you), the computer is useless. It does what you tell it to do.

**Solve problems:** Computers are tools. They are complex tools, admittedly, but they are not mysterious or magical: they exist to automate tasks.



Unlike builders who are constrained by physical limitations, such as the number of bricks they have, or the maximum height of a building, as a programmer, there are no such limits. The only thing that limits a programmer is the expanse of their imagination and their ability to use logic and code to solve those problems. In this unit, learners will start to become equipped to solve problems using logic and code.

## 7.3 Programming Essentials in Scratch

### What is a Sequence?

“The order in which instructions are executed.”

**Sequencing:** Instructions performed in order, with each executed in turn

As you have seen, computers will follow your instructions precisely and in the order in which you tell it.

Can you think of any non-computing related examples of where instructions need to be carried out in the correct sequence?



## 7.3 Programming Essentials in Scratch

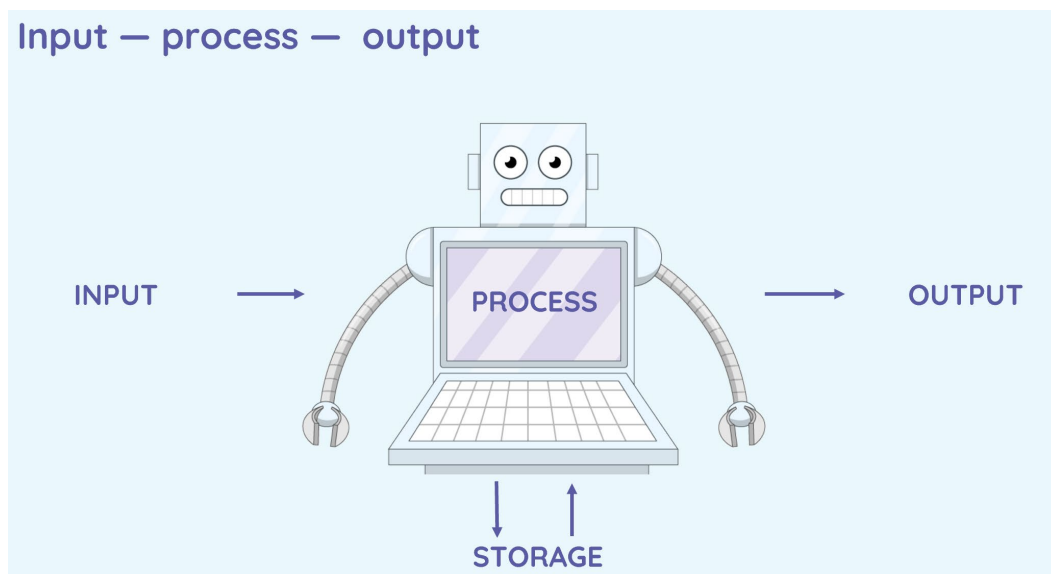
### Lesson 2: Sequence and variables

#### Learning objectives

- Define a variable as a name that refers to data being stored by the computer
- Recognise that computers follow the control flow of input/process/output
- Predict the outcome of a simple sequence that includes variables
- Trace the values of variables within a sequence
- Make a sequence that includes a variable

#### Key vocabulary

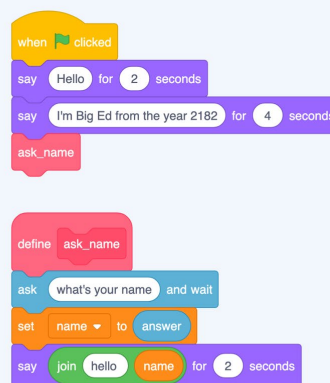
Variables, sequencing, subroutines, commands, execute, input, process, output, storage, tracing



Last lesson, the learners were introduced to the terms: sequence, selection, and iteration. These are all processes, but for computers to perform tasks there is more that is needed. Often a computer will take inputs (this might be automatic or via human input) as well as producing an output. A good example of this is when you use a keyboard and mouse, it inputs data into the computer to be processed and the output is visible on the computer monitor. Storage is also important. The computer stores data and needs to retrieve it to be processed at the appropriate times.

### PREDICT

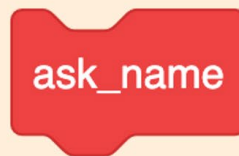
With a partner, spend time reading the code on the right. Predict what you think will happen.



## 7.3 Programming Essentials in Scratch

You might have accurately said that the ask\_name block runs the “define ask\_name” block. Although this is correct, you should use the correct terminology. Instead, say that the ask\_name block **calls** the **“subroutine”**.

How do the following two blocks relate to each other?



When your program reaches the ask\_name block, it **calls** the **subroutine** ‘define ask\_name’.

‘define ask\_name’ is a **subroutine**.



### What is a variable?

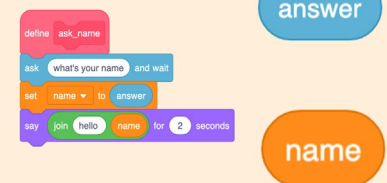
- A variable is a **location in memory** in which you can **temporarily store text or numbers**
- It is used like an empty box or the Memory function on a calculator
- You can choose a name for the box (the “variable name”) and change its contents in your program



Below ‘define ask\_name’, there are two variables being used.

What are their names?

1. Answer
2. Name



### INVESTIGATE: Answers

Why do you think it only says “Hello” and not “Hello” and the name you entered?

What can you learn from this?



It is because ‘name’ is being linked to ‘answer’ before the question is asked.

You must always set the value of a **variable** before using it.

### Tracing the value of the variable (What's the temperature?)

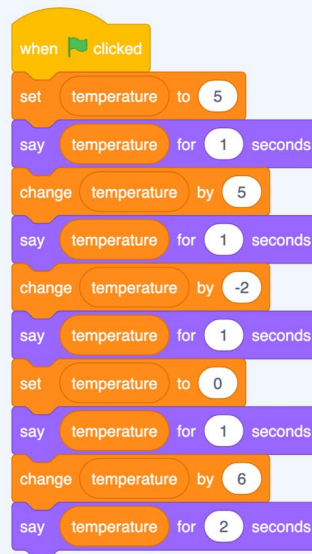
As described earlier, a variable is a location in memory that holds a value temporarily. This value may change frequently as in the code snippet below. When debugging code it is often a good idea to 'trace' the values stored by the variable. To trace the variable, write down the current value of the variable each time there is an output i.e. every time they see the following block:

### Trace the temperature variable: What will Big Ed say?

Big Ed has just arrived on a new planet and he's measuring the temperature of his new environment.

Use the activity sheet to trace (keep track of) the value of the temperature variable on each line that it is referenced.

Fill in your activity sheet and write down what Ed will say on each line.



## 7.3 Programming Essentials in Scratch

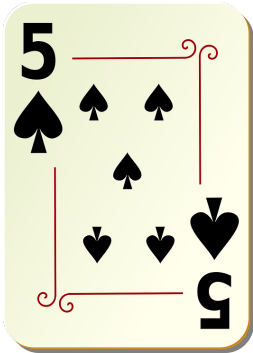
### Lesson 3: Selection

#### Learning objectives

- Define a condition as an expression that will be evaluated as either 'true' or 'false'
- Identify that selection uses conditions to control the flow of a sequence
- Identify where selection statements can be used in a program
- Modify a program to include selection

#### Key vocabulary

*Expressions, evaluate, conditions, selection, If statements, variables, sequencing, subroutines*



#### Conditions and expressions

You evaluated an **expression** to 'true' or 'false' and then performed an action depending on the outcome.

If "*the card is a heart*" is true:

Stand next to true

Else:

Stand next to false

A **selection** statement in programming allows a computer to **evaluate** an **expression** to 'true' or 'false' and then perform an action depending on the outcome.

The expression is "if the card is a heart". If it evaluates as 'true', then it will perform the action beneath it, else (i.e. otherwise) it will perform the action under the **Else** heading. Both **If** and **Else** would be written, because you don't know which statement will be executed.

## Conditions and expressions: guess who?

If "*the character has a hat*" is true:

Remove from game

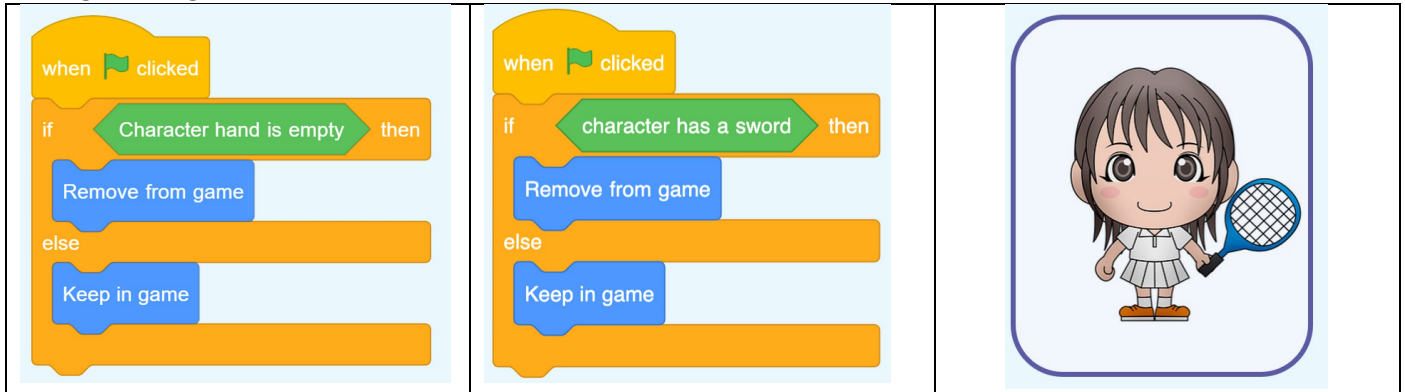
Else:

Keep in game

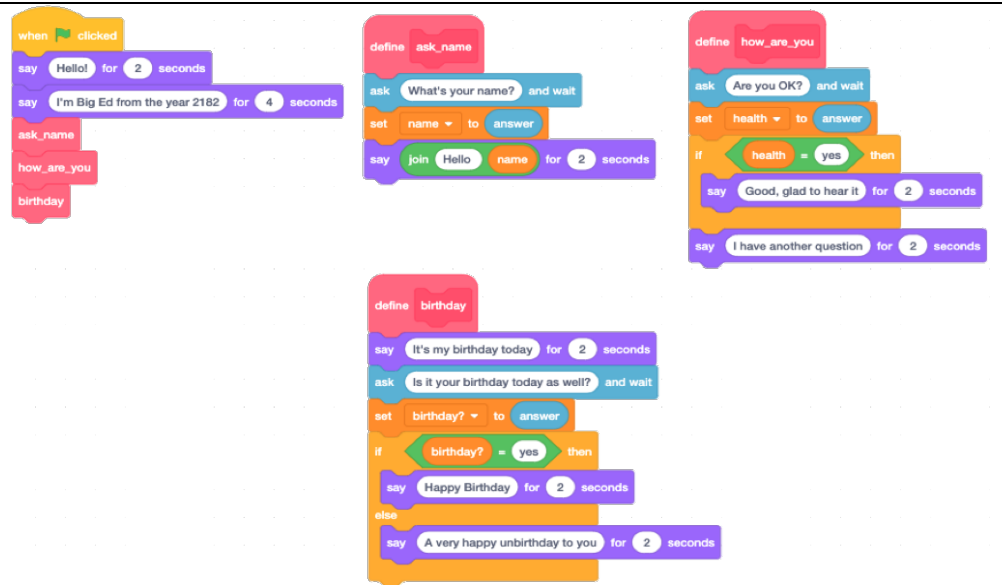




### 7.3 Programming Essentials in Scratch

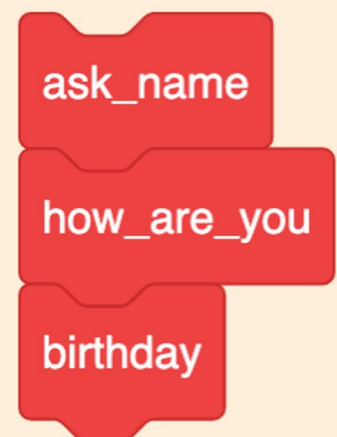


Read through the code snippets on the right which you will recall are from the 'Big Ed' Chatbot. Note how the 'birthday' subroutine has an 'else' statement which will run if the condition is not met. Compare this to the 'how\_are\_you' subroutine. There is no 'else' statement and the final block ('say I have another question') will always run after the if statement has completed.


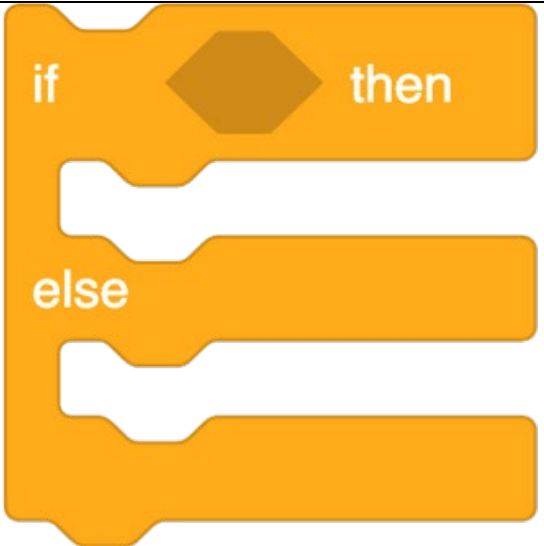


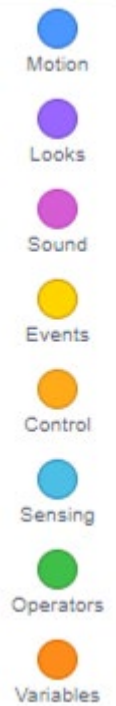
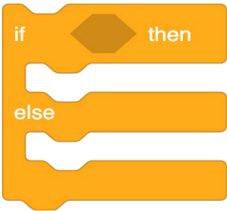
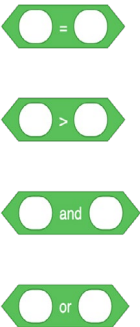
Look again at the code snippets above...

- What is the purpose of the following three blocks of code?
- They **call** the three subroutines in the specified sequence



### 7.3 Programming Essentials in Scratch

 An orange Scratch 'if then' block. It features a tab labeled 'if' on the left, a diamond-shaped condition slot in the middle, and a tab labeled 'then' on the right. Below the condition slot is a single slot for an operation.	<p>As we have discovered, an <b>If</b> block allows us to check a <b>condition</b> and perform an operation if the condition <b>evaluates</b> to 'true'.</p> <p>If the condition evaluates to 'false', the operation will not be carried out.</p>
 An orange Scratch 'if then else' block. It features a tab labeled 'if' on the left, a diamond-shaped condition slot in the middle, and a tab labeled 'then' on the right. Below the condition slot are two slots for operations, one for the 'then' branch and one for the 'else' branch.	<p>An <b>If</b> block with an <b>Else</b> allows us to perform a different operation should the <b>condition evaluate</b> to 'false', before the program continues.</p>

 A vertical list of Scratch menu categories, each with a colored circle icon and a label: Motion (blue), Looks (purple), Sound (pink), Events (yellow), Control (orange), Sensing (light blue), Operators (green), and Variables (dark orange).	<p>The selection statements (if statements) above can be found in the control section of the menu.</p>  An orange Scratch 'if then else' block, identical to the one in the previous table. <p>The comparison operators below (=, &gt;, &lt;, and, or) can be found under the operators menu.</p>  Four green Scratch comparison operator blocks: an equals sign (=), a greater than sign (>), the word 'and', and the word 'or'.
--	---

#### In this lesson, you...

Learned that a **condition** is an **expression** that will be evaluated as either '**true**' or '**false**'

Used **selection** in a program to control the flow of the **sequence**



## 7.3 Programming Essentials in Scratch

### Lesson 4: Operators

#### Learning objectives

- Create conditions that use comparison operators (>,<=)
- Create conditions that use logic operators (and/or/not)
- Identify where selection statements — that include comparison and logical operators — can be used in a program
- 

#### Key vocabulary



*Operators, logic, comparison, expressions, evaluate, conditions, selection, If statements, variables, sequencing, subroutines*

## Operators

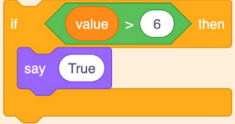
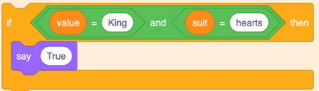
Last lesson you used a comparison operator in your program.

Can you spot where in this code?

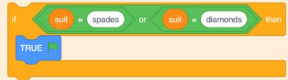



This condition compares the value of 'health' to 'yes' and checks to see if they match.

Comparison operators	Logic operators
<p>Comparison operators compare the values of <b>expressions</b>.</p> <p>What are each of these conditions checking?</p> <div> <div>variable1 = variable2</div> <div>That variable1 is <b>equal</b> to variable2</div> </div> <div> <div>variable1 &gt; 10</div> <div>That variable1 is <b>greater than</b> 10</div> </div> <div> <div>200 &lt; variable1</div> <div>That 200 is <b>less than</b> variable1</div> </div>	<p>Logic operators perform 'logical operations'.</p> <p>What are each of these conditions checking?</p> <div> <div>number &gt; 30 and city = Athens</div> <div>That 'number' is greater than 30 <b>and</b> 'city' is equal to Athens (when both are true)</div> </div> <div> <div>number &gt; 30 or city = Athens</div> <div>That 'number' is greater than 30 <b>or</b> 'city' is equal to Athens (when either are true)</div> </div> <div> <div>not age &gt; 18</div> <div>That 'age is greater than 18' is <b>not</b> true (i.e. it is false)</div> </div>

Expression 1	Expression 2
<p>value &gt; 6</p> 	<p>value = king and suit = hearts</p> 

7.3 Programming Essentials in Scratch

<p>Expression 3</p> <p>suit = spades</p> <p><b>or</b></p> <p>suit = diamond</p> 	<p>Expression 4</p> <p><b>not</b> colour = red</p> 
<p>Expression 5</p> <p>value = queen <b>or</b> value = 9</p> <p><b>and</b></p> <p>colour = black</p> 	<p>Expression 6</p> <p>value &lt; 5 <b>or</b> value &gt; 10</p> <p><b>and</b></p> <p>suit = clubs <b>or</b> suit = hearts</p> 

Does each statement evaluate to ‘true’ or ‘false’? Complete the table below:

Statement	Evaluates <u>to</u> ‘true’ or ‘false’?
7 > 6	True
9 = 9	True
10 < 9	False
(30 < 50) or (30 > 50)	True
(20 = 20) and (15 < 15)	False
not (20 = 20)	False

## 7.3 Programming Essentials in Scratch

### Lesson 5: Count-controlled iteration

#### Learning objectives

- Define iteration as the process of repeatedly executing instructions
- Describe the need for iteration
- Identify where count-controlled iteration can be used in a program
- Implement count-controlled iteration in a program
- Detect and correct errors in a program (debugging)

#### Key vocabulary

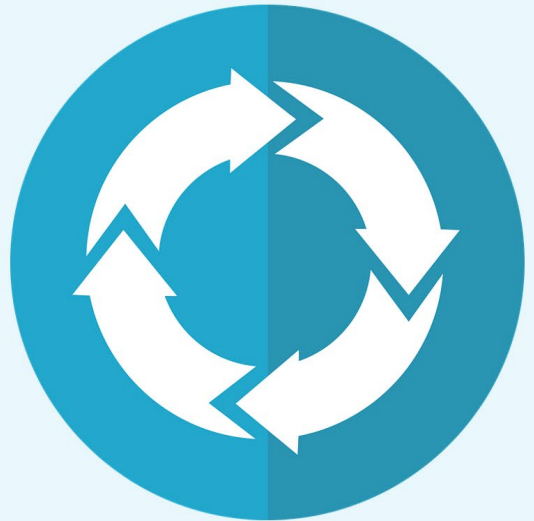
*Iteration, count-controlled, condition-controlled, debugging, variables, sequencing, subroutines*

## Iteration

**Iteration in computing is the process of repeatedly executing instructions**

Being able to repeatedly execute instructions is commonly referred to in computing as **iteration**.

Can you think of any repetitive tasks that computers or humans might be able to perform?



## Count-controlled or condition-controlled

### Count-controlled

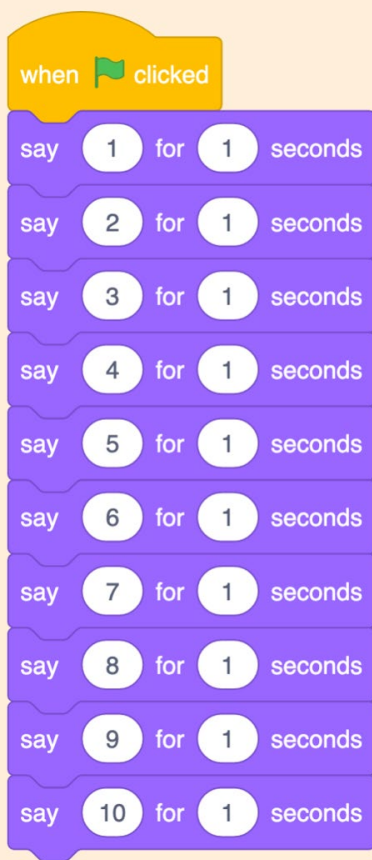
Count-controlled iteration will execute the commands a set number of times

**Example:** “Write out lines 100 times”

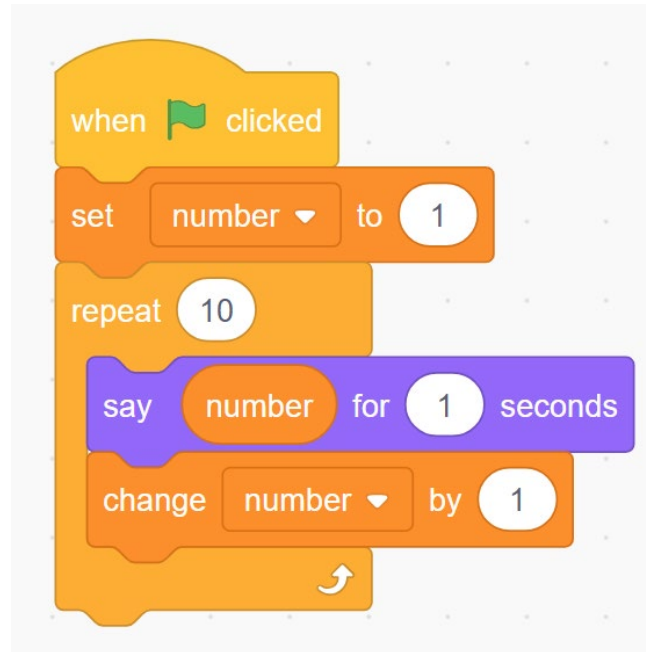
### Condition-controlled

Condition-controlled will execute the commands until the condition you set is no longer being met

**Example:** “Write out lines until 4pm”

**'Counting Cat' Program**

The 'Counting Cat' program to the left was inelegant. By adding iteration, the number of lines of code needed to solve a problem is reduced. The common programming term for this is that it makes the code more elegant. There is a misconception that this will make the code more efficient. It will be more efficient for the programmer to write, but not in terms of processing by the computer.



Note: In the count-controlled loop above, marks are awarded for:

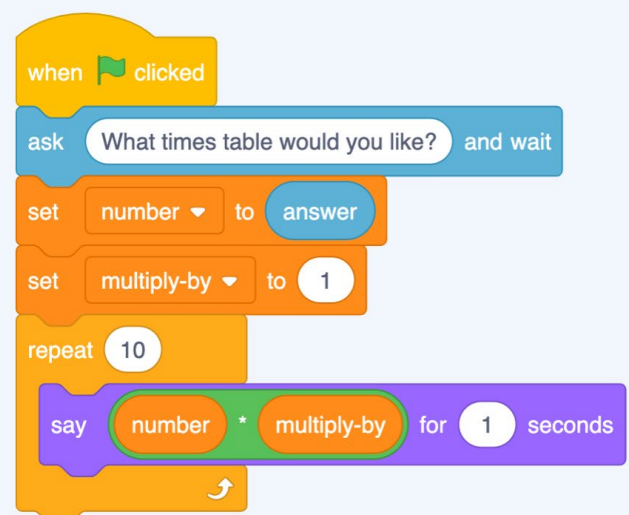
1. Reset count to 1 before loop commences
2. Use of count controlled loop
3. Storing 'number' in a variable
4. Incrementing number variable by +1 through each iteration.

**Debugging**

Debugging is the process of finding an error in your code and taking steps to fix the problem.

The code to the right has a problem that needs debugging.

Use your worksheet to help you find the bug and suggest a solution.



### 7.3 Programming Essentials in Scratch

When executed, the code above will see the cat on screen seemingly 'stuck' saying the number that was inputted. See below for the correct code:

