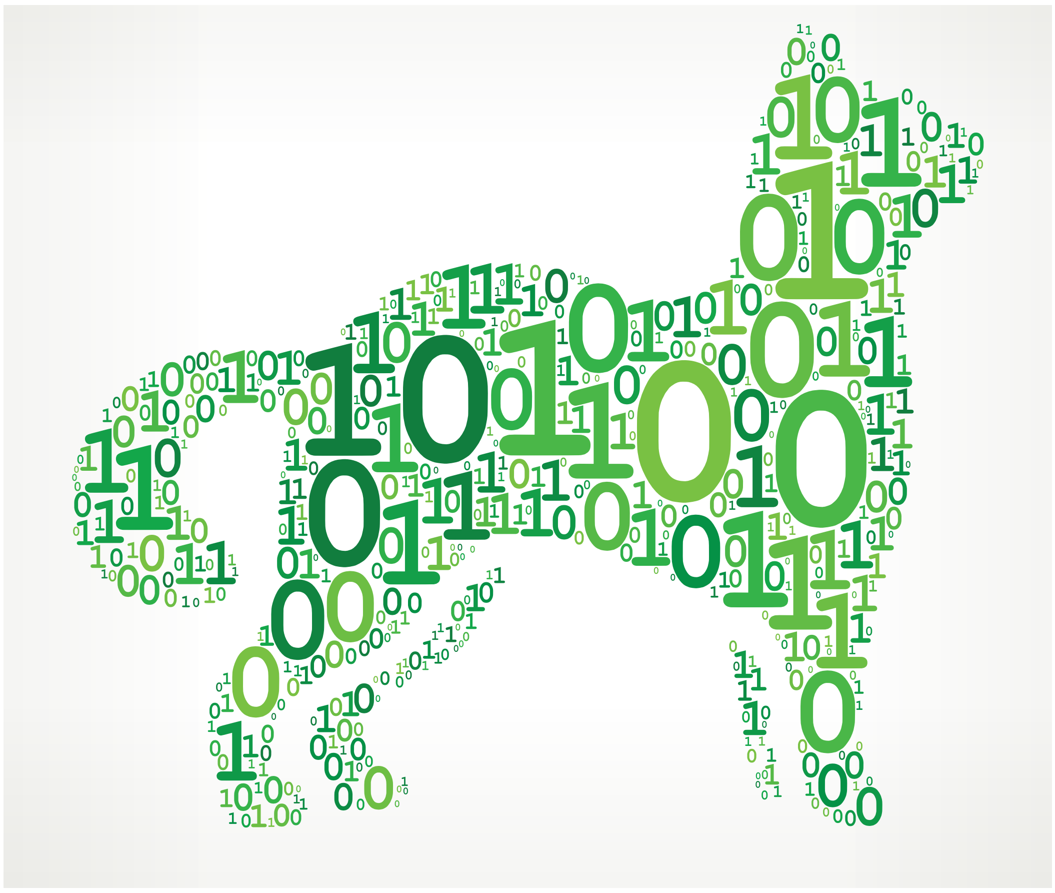
Programming for a purpose 1

	In this module you will:	Pass/Merit	Done?
1	Plan an interactive program using abstraction	P	
2	Create and test an interactive program using selection, input and output	P	
3	Predict the output of an interactive program that uses input and selection	P	
4	Create and formally test an interactive program using selection, input and output	M	
5	Correct (debug) a short interactive program containing more than one error.	M	

In this module you are going to develop skills to help you work towards your final project, which will be using Scratch to create a game where a player can look after a virtual pet. The player of your game will need to feed and play with the pet to make sure it doesn't get too hungry or bored!



You will learn how to plan, design, create and test a program before you start to make your game. Before you can start planning your game, you need to learn some new skills in Scratch. These skills will be used as part of the final game.

1 Programming for a purpose

You will learn how to:

- use variables
- broadcast
- change the costume of a Sprite
- change the background of a game
- detect collisions
- use timers and wait
- use random numbers.

Once you have learnt what these new skills are, and how they are used, you will be able to think about them when you design your game.

Before you start

You should:

- have used Scratch to create simple programs, including sequences, repetition and procedures
- have fixed problems in your own programs to make sure they work
- have had experience of using and creating flowcharts to plan a program.

Introduction

Computer programs need to be planned and designed before they are created. This is usually done following something called the **Software Development Cycle**. This is a structured sequence of actions that allow you to plan, design, create, **test** and then improve a computer program.

It is important that programs are planned first for many reasons, such as:

- to make it clear what the program has to do
- to make sure everyone on the team working on it fully understands the requirements
- to make sure the person (or group of people) who want the game are happy with what the game will do.

From this plan, you can design how the program will work, for example using a flowchart. This will let you find any problems before you start, and it means you're not making it up as you go along!

Once you have created your program you need to test it to make sure it works fully and that there aren't any problems. If you have been asked to make a particular program for a client, they will have a clear idea of what they want. If you cannot deliver the program they want, you may not get paid for the work you have done – if it doesn't work, for example.

Key terms

Software Development Cycle: a formal set of processes followed to plan, design, create and test a system.

Test: to make sure a section of code runs correctly.

Cambridge University Press
978-1-108-46355-3 — Cambridge ICT Starters On Track Stage 2
Victoria Ellis , Sarah Lawrey , Edited in consultation with Doug Dickinson
Excerpt
More Information

Key terms

Interactive: a program that has both input and output for the user.

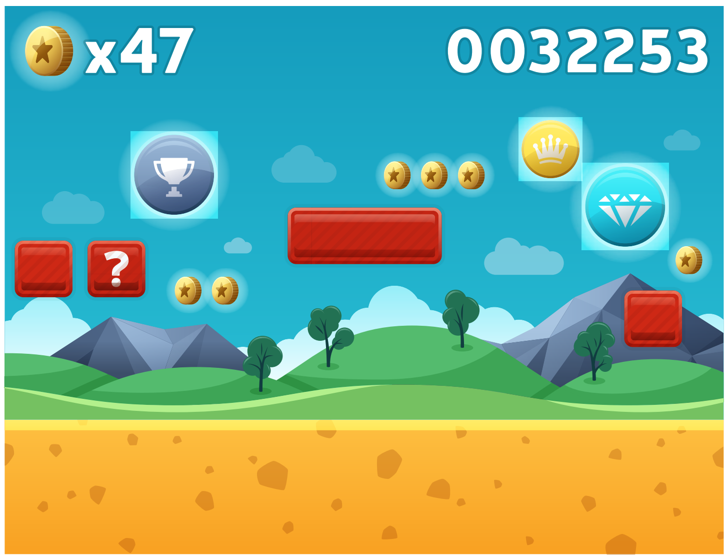
Interface: the part of a program that lets the user input data and that produces output for the user.

Input: putting data into a computer, for example typing, clicking buttons.

Output: data being given from the computer to the user, for example on screen, from speakers.

Analogue: data in the real world (that is not in a computer).

Programs usually require user **interaction** using an **interface**. The interface is the part of the program that allows the user to interact with it. It includes the on-screen buttons, text, images and so on, that the user can then click, type into and look at.



An interactive program means that the user is involved in the program and they are able to:

- put data into the computer. This is **input**, for example clicking buttons, typing text and numbers. In a computer game, this might be by using a hand-held controller.
- get data from the computer. This is **output**, for example you can see images and text, and hear sound.

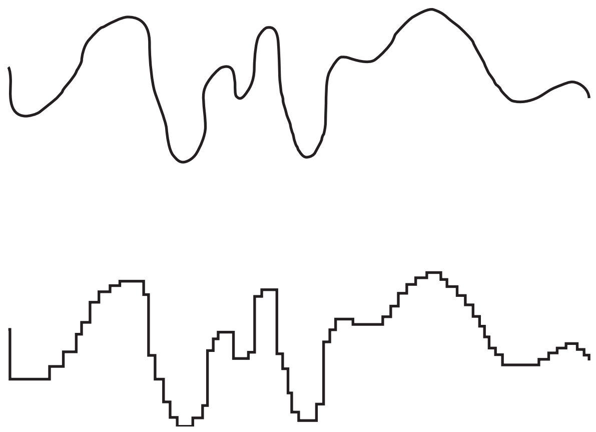
Stay safe!

You might be using the 'Scratch Internet' version. Make sure you stay safe while online, and only use the website you have been given. Do not talk to other people online, or visit other websites without the permission of your teacher.

Most data that is in the real world (that is not in a computer) is **analogue** data. This can be any value, represented in any form, for example sound or images. This data must be converted for a computer to understand it.

A computer only understands **digital** data. This is 1s and 0s. If you want to input data into a computer, it has to be turned into 1s and 0s. This can be done with any type of data, for example letters, numbers, sound and images. This is **data capture**.

This image shows an analogue sound wave at the top, that can have a large range of values, and a digital wave at the bottom where the data has been encoded as 1s and 0s. The digital sound wave is not identical to the analogue sound wave.



A special form of input and output is known as **feedback**. This is when the output from a process becomes the input into a process. So it is all run automatically, and the data that is created changes what happens next.

Key terms

Digital: data in a computer; it is stored in 1s and 0s.

Data capture: gathering data from the real world and turning it into a form the computer understands.

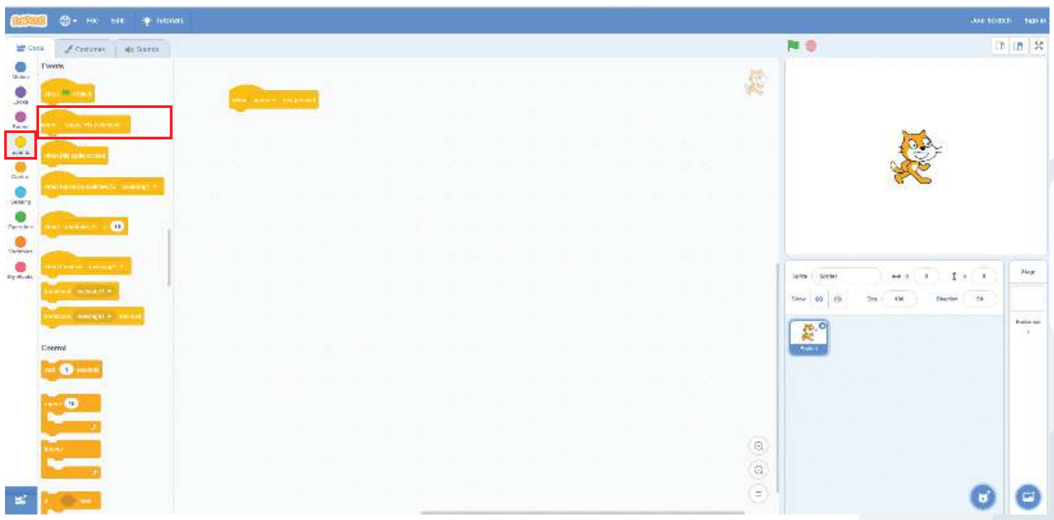
Feedback: the input changes the program, which then produces output. The output becomes the new input in the program.

Skill 1

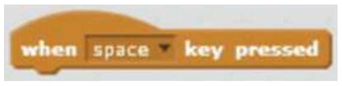
P

Detecting user interaction 1: detecting click buttons

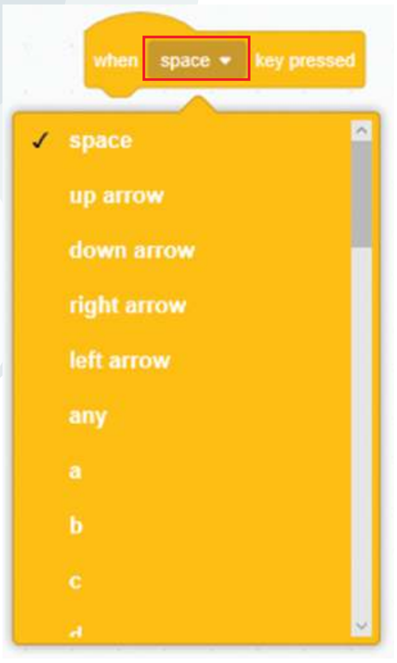
A computer game might need the user to press buttons, for example a keypress. In a car driving game, the car is going to move forward when the up arrow is clicked. The game needs to know when the up arrow has been clicked by the user so it knows when to move the car.



To do this in Scratch, you will need to click on **Events** and then drag



.



Next, you need to click on the down arrow next to the word ‘space’.

You would then choose the key from the drop-down box.

You would then add instructions for when this key is pressed below this block.

Activity 1.1

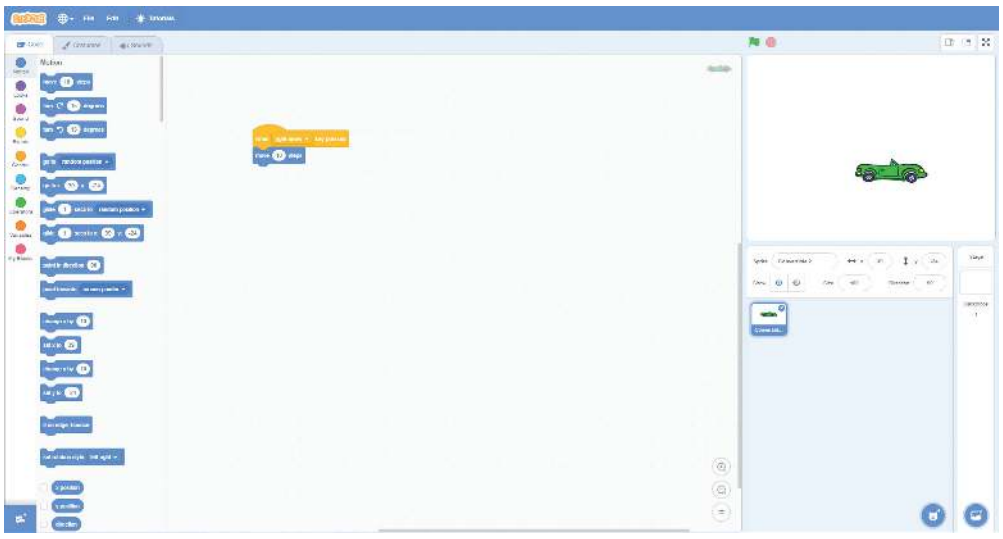
Open a new Scratch window.

Right-click on the cat Sprite.

Select ‘Delete’.

Click on the ‘New Sprite’ button.

Add a car Sprite to a new Scratch file.



Make sure it is facing to the right of the screen.

Add blocks so when the right arrow key is pressed, the car moves ten spaces.

Test your code works.

Activity 1.2

Add new blocks so when the left arrow key is pressed the car moves back ten spaces.

Test your code works.

Activity 1.3

Add new blocks so when the up arrow key is pressed the car rotates left 90 degrees.

Test your code works.

Activity 1.4

Add new blocks so when the down arrow key is pressed the car rotates right 90 degrees.

Test your code works.

Tip

Moving back will have a negative number, for example –10.

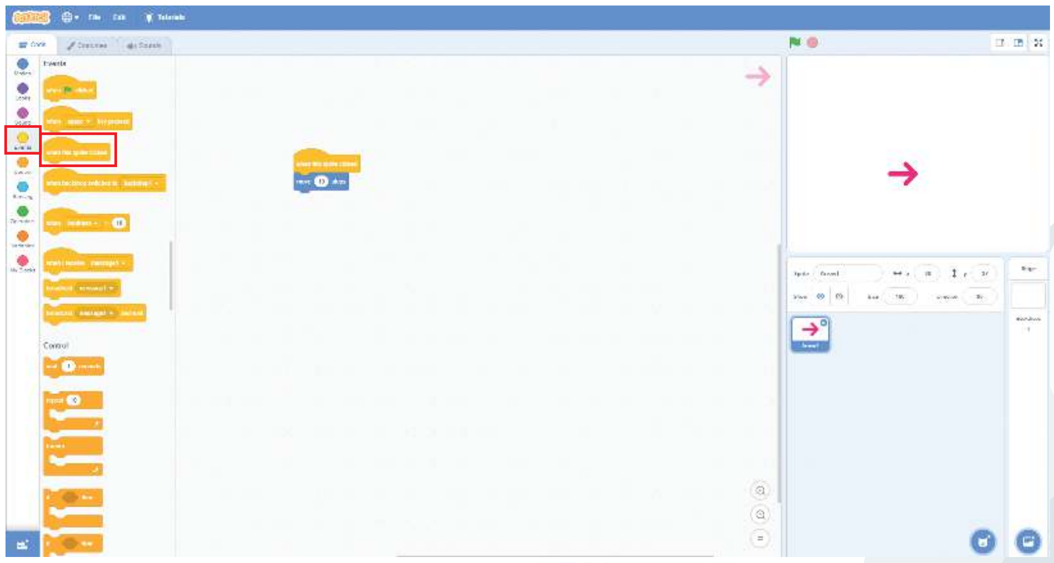
Skill 2

P

Detecting user interaction 2: user clicking objects

You can get your user to interact with your game by clicking on objects in the actual game, for example when the user clicks on the car, the car moves forward.

Start by clicking on the Sprite.



Then you can click on **Events** and then drag when this sprite clicked .

Finally, you can add your instructions for when this Sprite is clicked below this block.

Activity 2.1

Add a Sprite to a new Scratch program.

When the Sprite is clicked make the Sprite move forward.

Test your code works.

Activity 2.2

Change the program from **Activity 2.1**. When the Sprite is clicked, make the Sprite:

1 Move forward 20 steps
2 Turn left 90 degrees
3 Move forward 20 steps
4 Turn right 90 degrees
5 Move forward 20 steps

Test your code works.

Tip

To slow your Sprite down so you can see it move, add a 'Wait' block after each instruction.

Skill 3 P

Broadcasting: make a car move when you click on a Sprite

When you add code to a Sprite, you can affect only that Sprite.

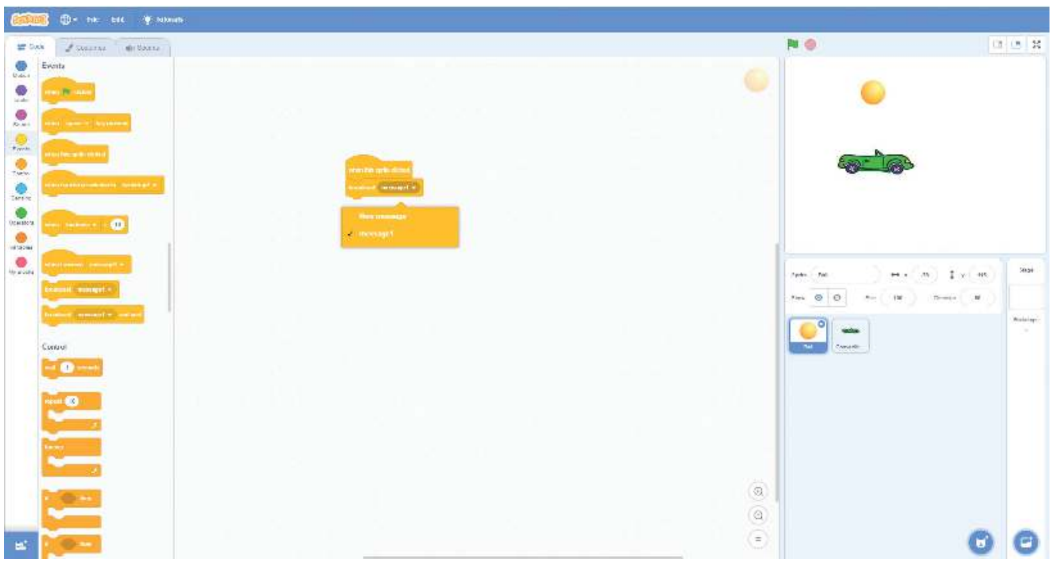
Broadcasting lets you send out a message to tell another Sprite to do something, for example move.

This is how to make a car move ten steps when the ball is clicked.

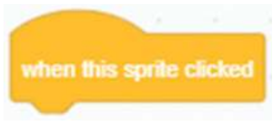
First, you need to add a car Sprite and a ball Sprite to your Scratch stage.

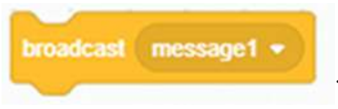
Key term

Broadcast: sending a message in Scratch that other Sprites can see, and then react to.



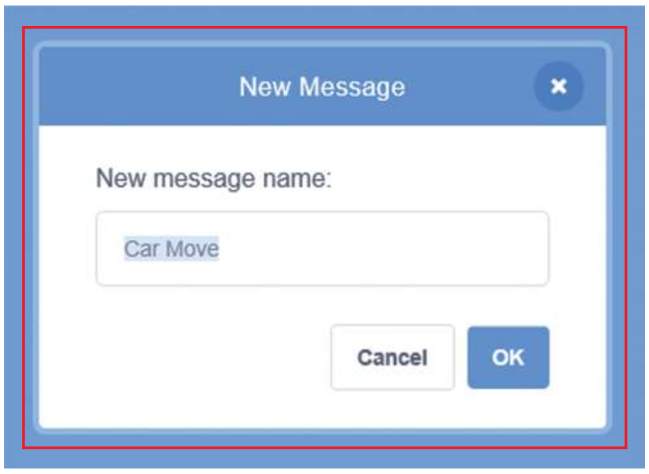
For the ball Sprite

Next, you need to click on **Events** and then drag 

and also drag 

.

You would then click on the drop-down menu and choose 'New Message'.



New Message

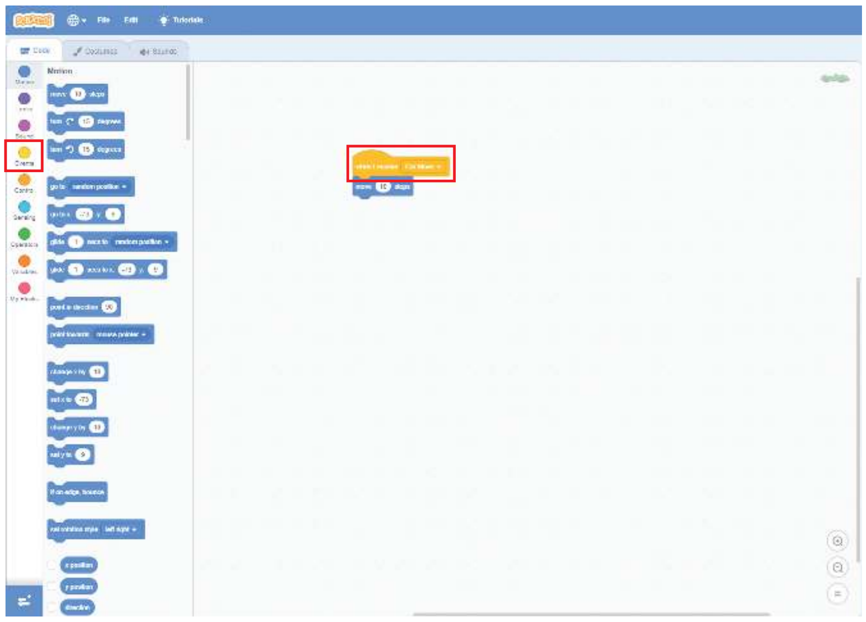
New message name:

Car Move

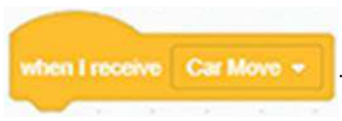
Cancel OK

You will need to change the message to make it meaningful. It should describe its purpose, for example Car Move.

For the car Sprite



You would start by clicking on **Events**.

You would then drag 

.

Next, you need to choose the correct broadcast message from the drop-down menu.

You can then add your action blocks below it.

Activity 3.1

Open a new Scratch window.

Add two car Sprites.

When the first car is clicked, make the second car move forward.

When the second car is clicked, make the first car move forward.

Test your code works.

Activity 3.2

Open a new Scratch window.

Add a dinosaur Sprite.

Add four other Sprites, one to move the dinosaur forward, one for moving back, one for turning left, one for turning right.

Add blocks so that when moving forward is clicked it broadcasts a message. When the dinosaur receives this message it moves forward. Repeat for moving back, and turning left and right.

Test your code works.

Key term

Variable: a space in memory where you can store data temporarily.

Did you know?

Variables are a fundamental part of programming, without them you can't store any data! They are only temporary though. To store data permanently, it needs saving to a file.

Skill 4 P

Variables

A **variable** allows you to store data in a program, for example a number. This data is stored in the computer's memory, a bit like putting something in a box. You give the variable a name so you can remember what it is called, and use it later. You might have lots of variables, so they all need to have names that describe what they are storing.

In a game where you are catching stars, you might want to count how many stars have been caught. This would be stored in a variable; it could be named Stars.

In a game where your spaceship is flying through space, you might want to store the number of planets it has visited. This would be stored in a variable; it could be named Planets.

This variable is called Points. At the moment there is the number 0 in the Points box.

Points
0

You can change the value in Points. For example, you could change Points to the number 2.

The variable now has 2 stored in it.

Points
2

You could ask the variable what is in it. If you asked Points what is in it, it would tell you ‘2’.

You can add to the value in Points. For example, you could add 1 to the current value.

Points currently has 2 in it, so 2 + 1 = 3.

The variable now has 3 stored in it.

Points
3

You could add 10 to it.

Points currently has 3 in it, so 3 + 10 = 13.

The variable now has 13 stored in it.

Points
13

Now you could subtract 3 from it.
Points currently has 13 in it, so 13 – 3 = 10.

The variable now has 10 stored in it.

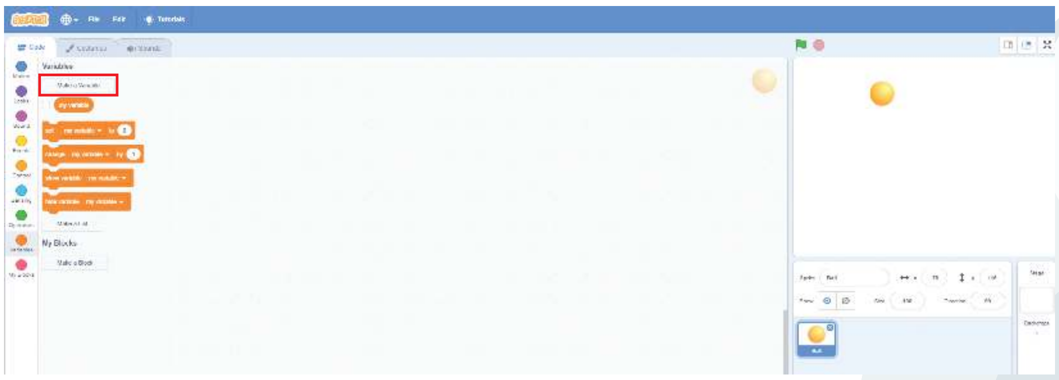
Points
10

If you asked Points what it has in it now, what would it say? 10.

Creating a variable

1 To create a variable, first you need to click on **Variables**.

2 Then you would click on ‘Make a Variable’.


Variables
Make a Variable
my variable
My Blocks
Make a Block