

# technoteach

## technocamps



Llywodraeth Cymru  
Welsh Government

Cyngor Cylido Addysg  
Uwch Cymru  
Higher Education Funding  
Council for Wales

**hefcw**

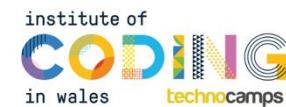


Prifysgol  
Abertawe  
Swansea  
University



Cardiff  
Metropolitan  
University

Prifysgol  
Metropolitan  
Caerdydd



# Pre - Scratch



# Scratch

- Scratch is a tool that is commonly used to teach programming/coding in schools (and universities!).
- It is block-based, so does not require lines and lines of instructions in the form of text.
- Scratch is excellent for teaching lots of important programming and for helping pupils to understand the logic of how to program and how to give instructions.
- The pupils can use the skills and logic that they learn in Scratch, and apply them to other programming languages in the future.

# Why use Scratch?

- Scratch is designed for school pupils to use, to learn about programming.
- Perfect way of incorporating Computer Science into the curriculum.
- When you program, you write algorithms, so you use important D.C.F. (4.1) skills.
- Scratch actually hits MANY strands on the D.C.F.

# Scratch in the Classroom

- Many pupils say that they can use Scratch if they have used it a couple of times, but they probably cannot even use 10% of its capabilities.
- Do not be fooled into thinking that if the pupils have made a couple of games, that they have mastered Scratch; it's not true.

# Scratch in the Classroom

- Scratch can be used to create:
  - Games;
  - Quizzes;
  - Animations;
  - Stories;
  - Interactive presentations;
  - Timelines;
  - Fact files;
  - Educational tools (e.g. dice, random number/name generators, story writing prompt generators, etc...).
- A Scratch project can include collaboration, problem solving, computational thinking, creative thinking, self/peer assessment, etc...

# Useful Scratch Sections

- Scratch For Educators page, including information and resources for educators - <https://scratch.mit.edu/educators>
- Scratch Ideas page, including tutorials to get you started, and idea for projects - <https://scratch.mit.edu/ideas>
- Information about Scratch Teacher Accounts - <https://scratch.mit.edu/educators/faq>

# Programming – More Than Scratch!

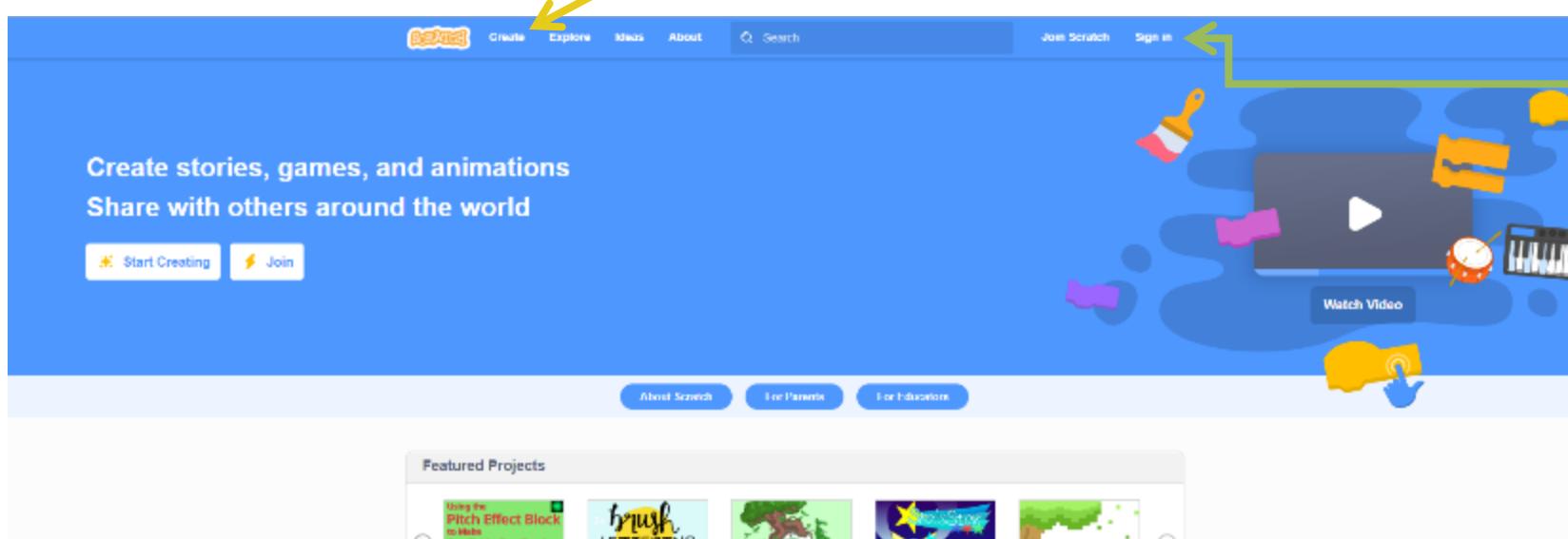
- Programming is more than just Scratch, Python, HTML etc.
- The skills and concepts exist outside of the programming language
- Digital industry requires people able to develop generic solutions to problems and then apply the solution in a programming language
- **Focus on developing the concepts and skills**

# Programming Concepts

Concept	Description
<b>Sequence</b>	Performing multiple instructions one after the other
<b>Input/Output</b>	Allows a user to interact with a program
<b>Selection</b>	Allowing conditions in a program to determine whether to execute instructions or not
<b>Iteration</b>	Repeating instructions while a condition is true
<b>Variables</b>	A data item that may take on more than one value during the runtime of a program
<b>Functions</b>	A defined set of instructions with a specific purpose
<b>Data Structures</b>	Methods of structuring data in a program
<b>Algorithm</b>	A sequence of instructions executed in a specified order to solve a problem

# Scratch

- Go to the Scratch website: <https://scratch.mit.edu/>
- You can 'Join Scratch' or 'Sign In', to be able to save your program to your Scratch account.
- Or you can just click 'Create' and use Scratch without an account.

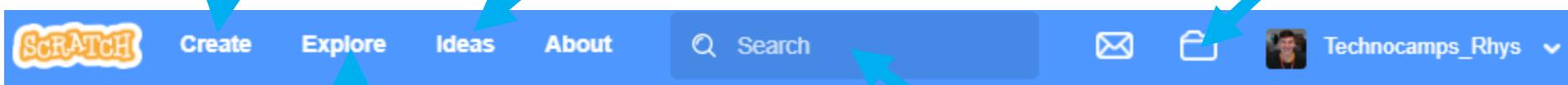


# Scratch Website

Create new projects of your own.

Gives you ideas for projects, as well as tutorials to follow.

View your projects.



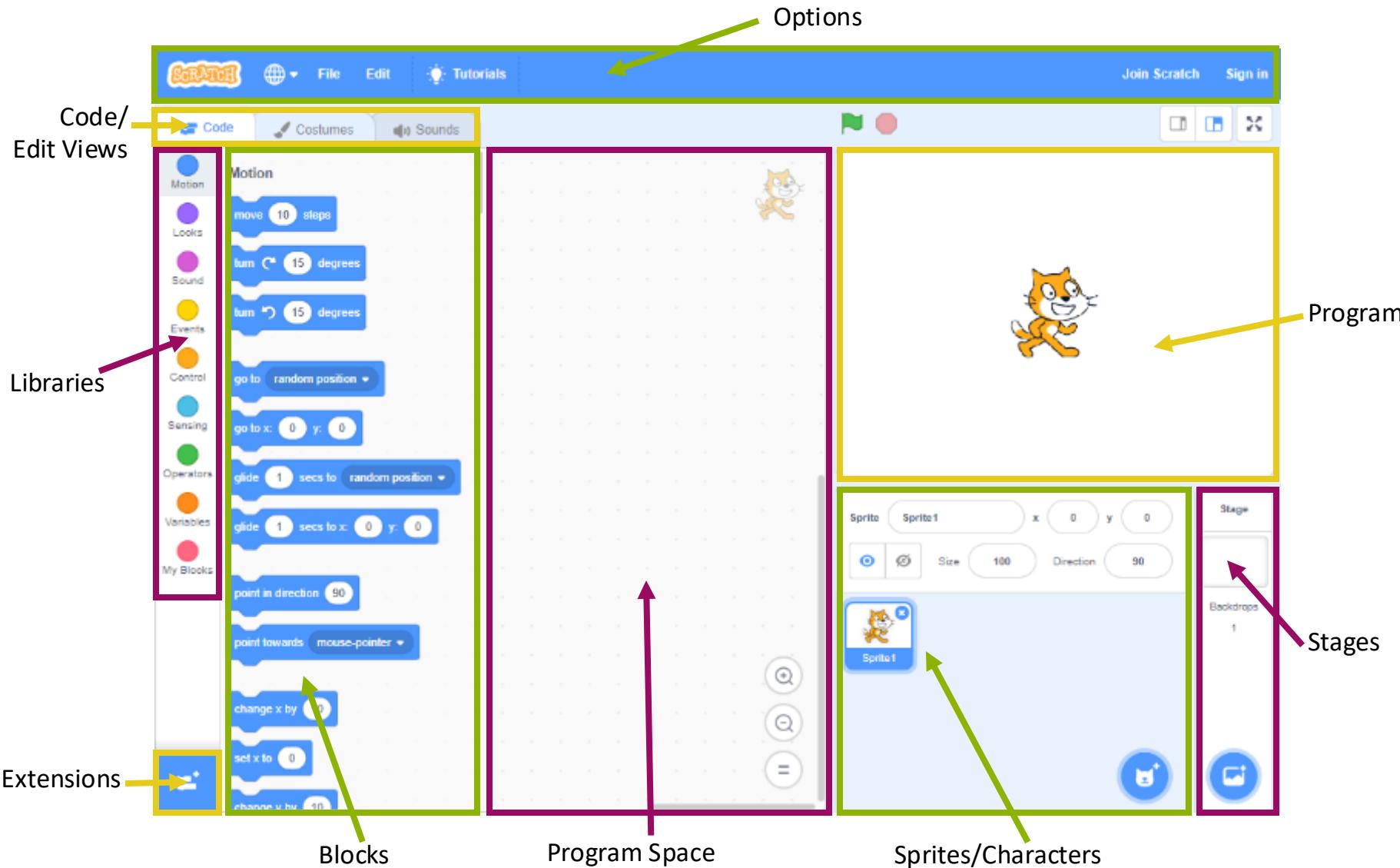
Find other people's projects, that you might be interested in.

Search for other people or projects.

# Scratch Website

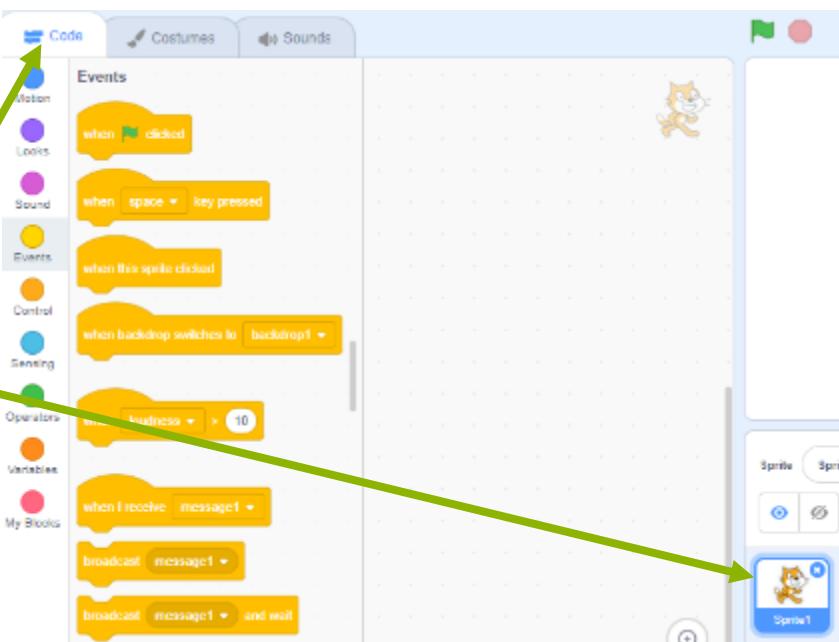
- On the Scratch website, you can:
  - View other people's projects;
  - Use other people's projects;
  - Look at the code other people used to create projects ('**See inside**');
  - Use the code that other people used to create a project ('**Remix**');
  - Follow other people's accounts, to see their creations;
  - Comment on and 'like' other people's projects.

# Scratch



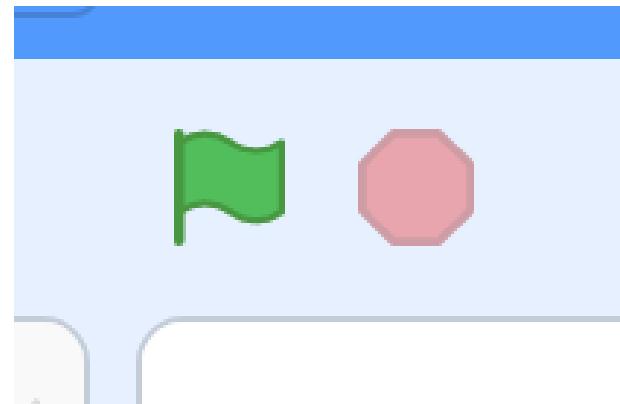
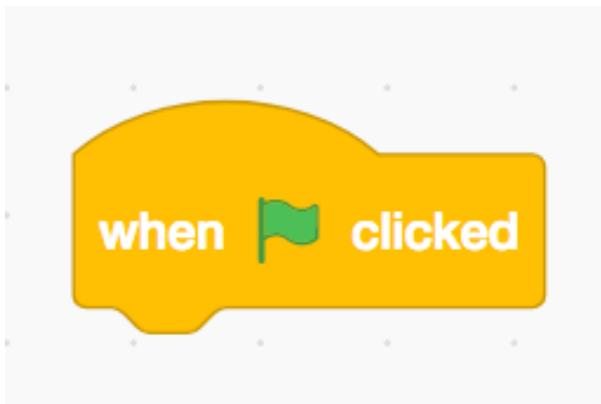
# Scratch Areas

- In Scratch, it is really important that you make sure that you are working in the correct area.
- If you want to give instructions to a specific sprite, then you must select that sprite and edit the code for that sprite.



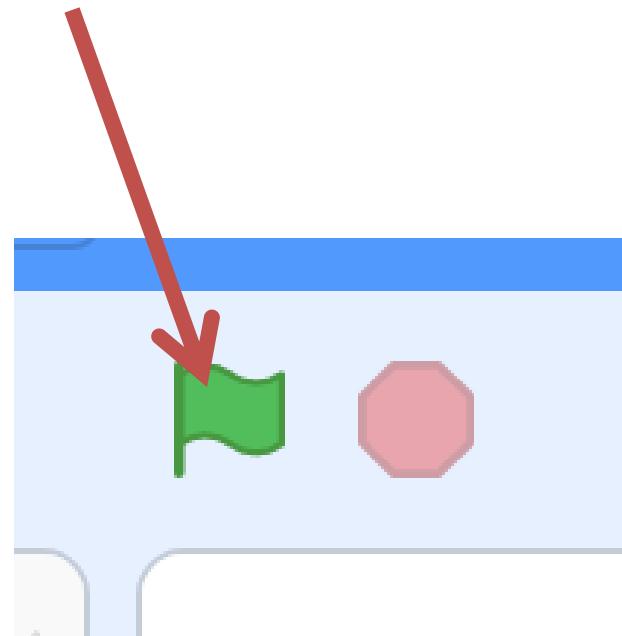
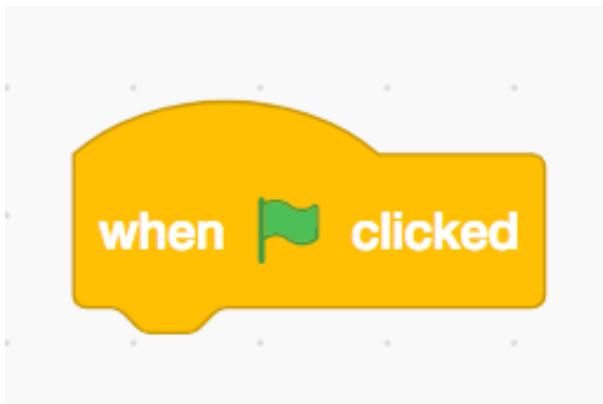
# Green Flag

- In Scratch, clicking the **Green Flag** is a good way to start your game/program.
- It is just like pressing a start button.

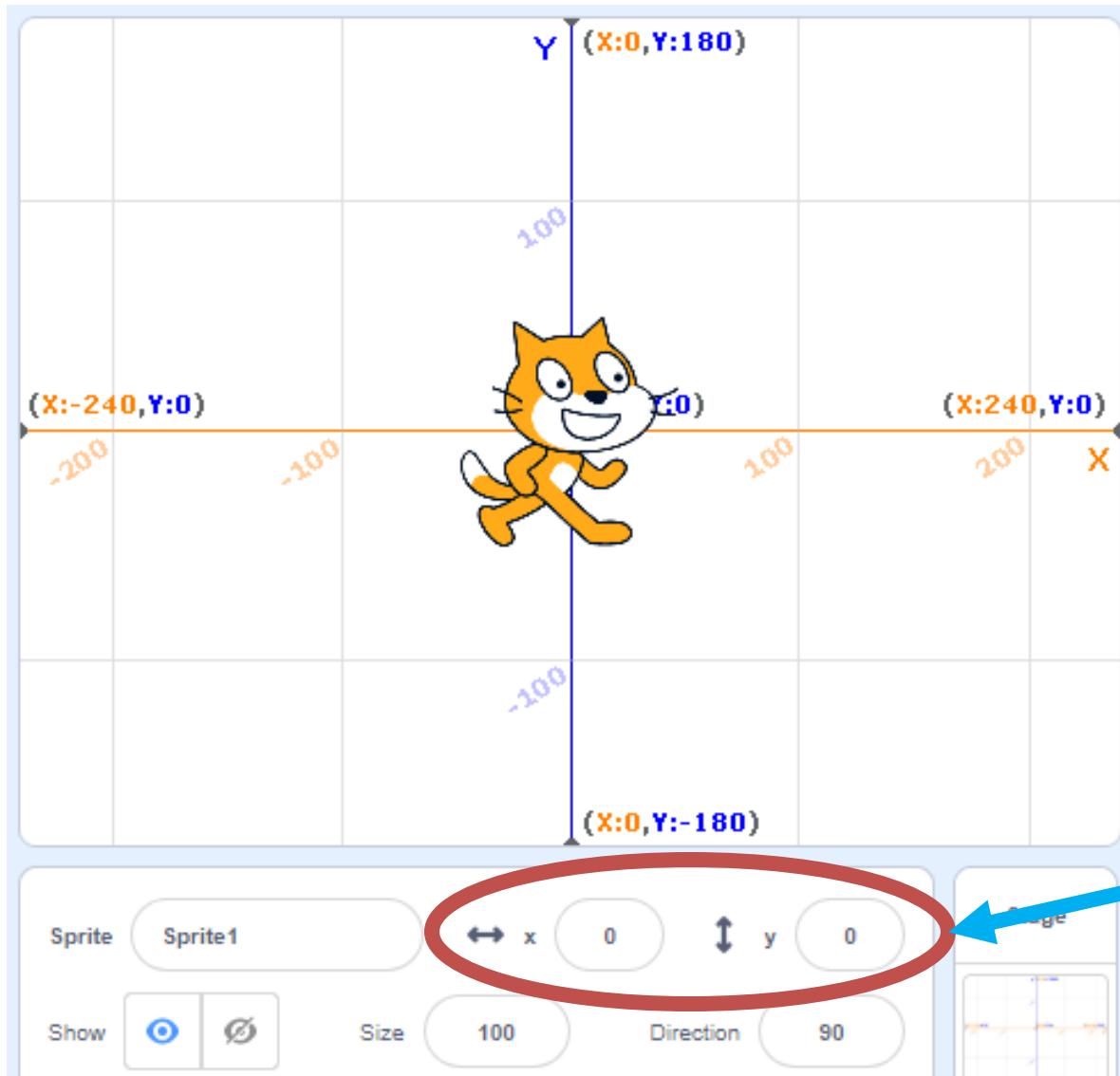


# Program Testing

- Also, whenever you want to test your program, to see if it is working as you expect it to, click the **Green Flag**, to run the program.



# Scratch Co-ordinates



# Scratch Co-ordinates



Co-ordinate  
location  
(200,-100).

# Scratch Co-ordinates





# Scratch Game (Variables)

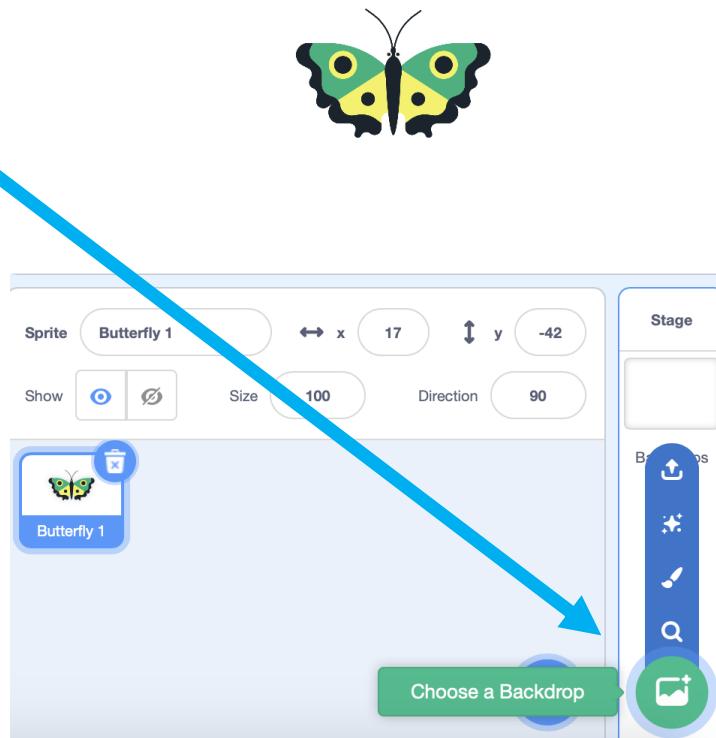
# Adding a Background

Money £ 0

In order to add a background:

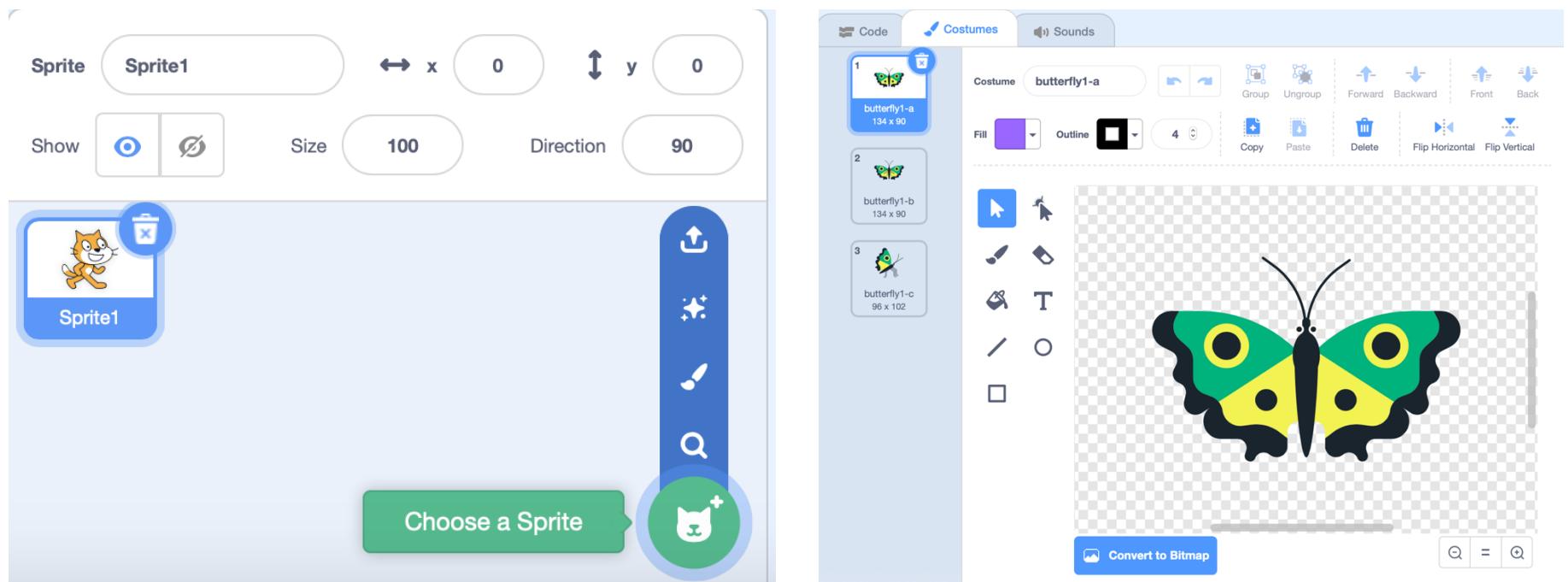
Click on the button in the bottom right corner and choose a background for your game.

Your butterfly will need to hide in this background with camouflage.



# Designing A Butterfly

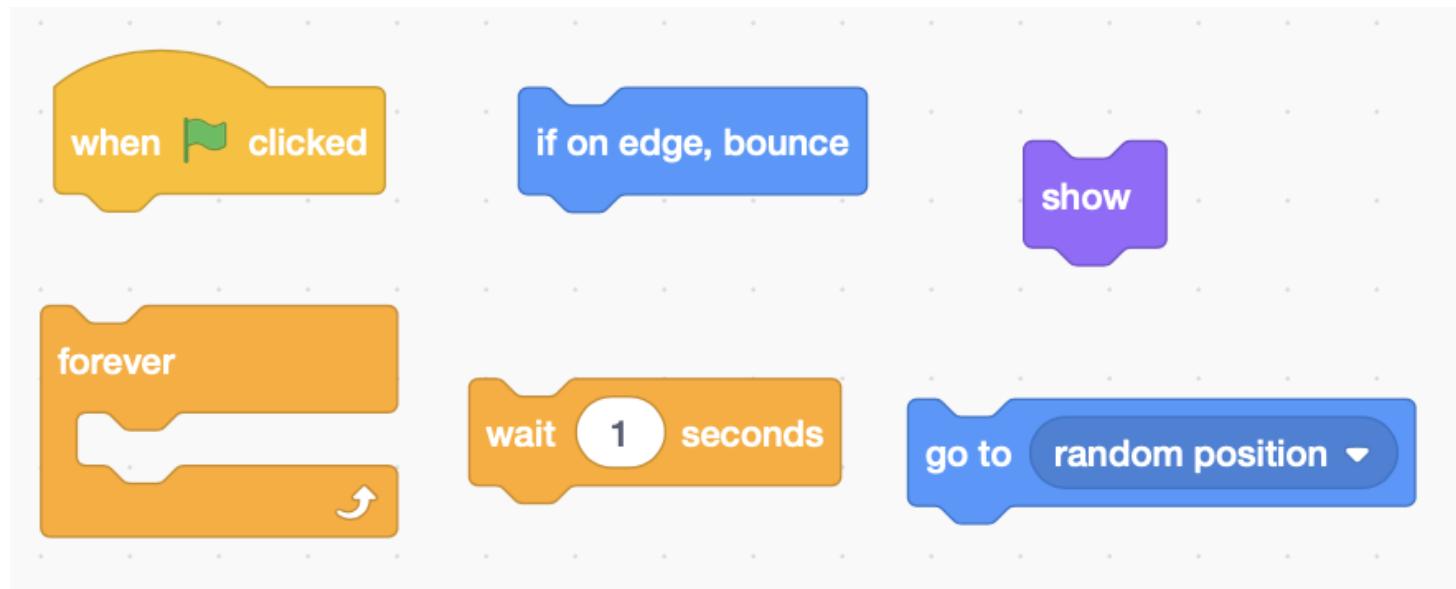
Select a new sprite, find the butterfly and edit the colours by clicking on Costumes in the top left corner.



# Butterfly Movement

When we click the Green Flag, we want the butterfly to:

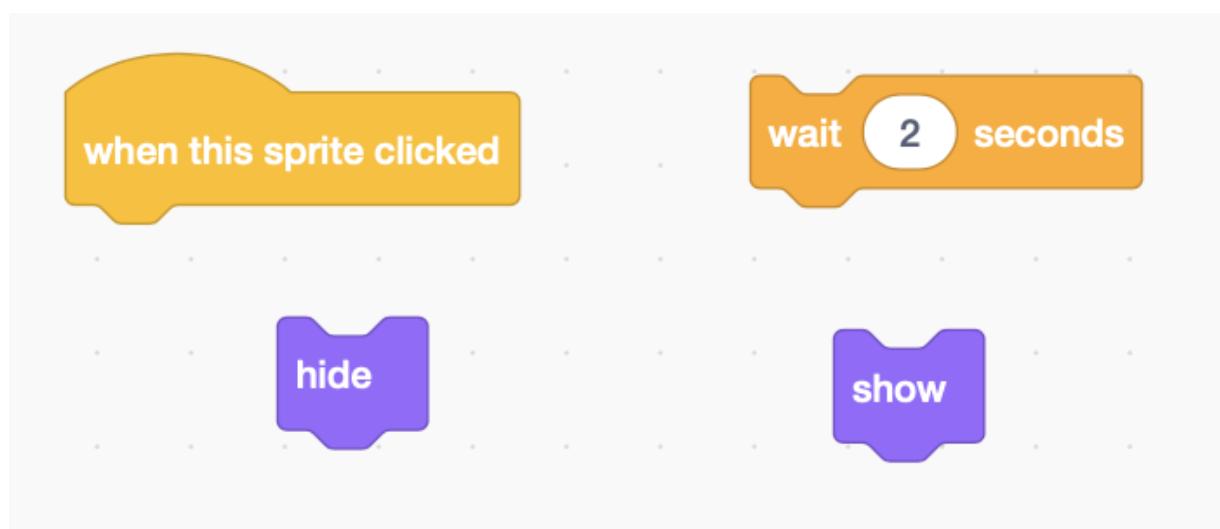
- show in a random place (and if touching the edge it bounces)
- wait some time before moving to a new position.
- repeat this forever



# Butterfly Clicking

We want the butterfly to react when it is clicked by:

- hiding
- waiting for some time
- showing again

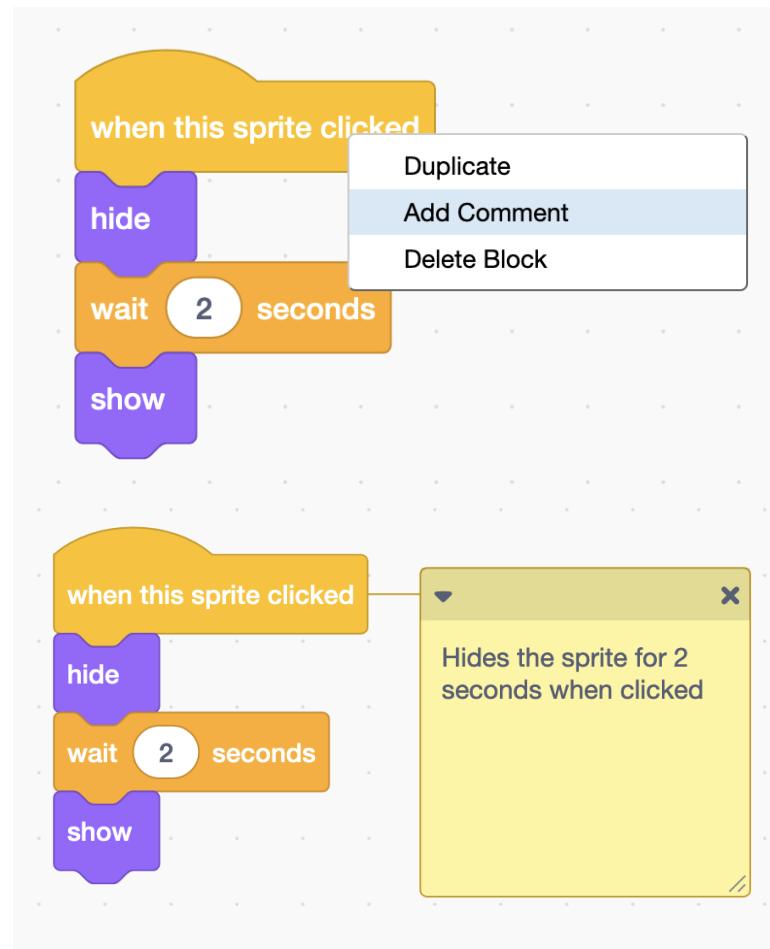


# Comments

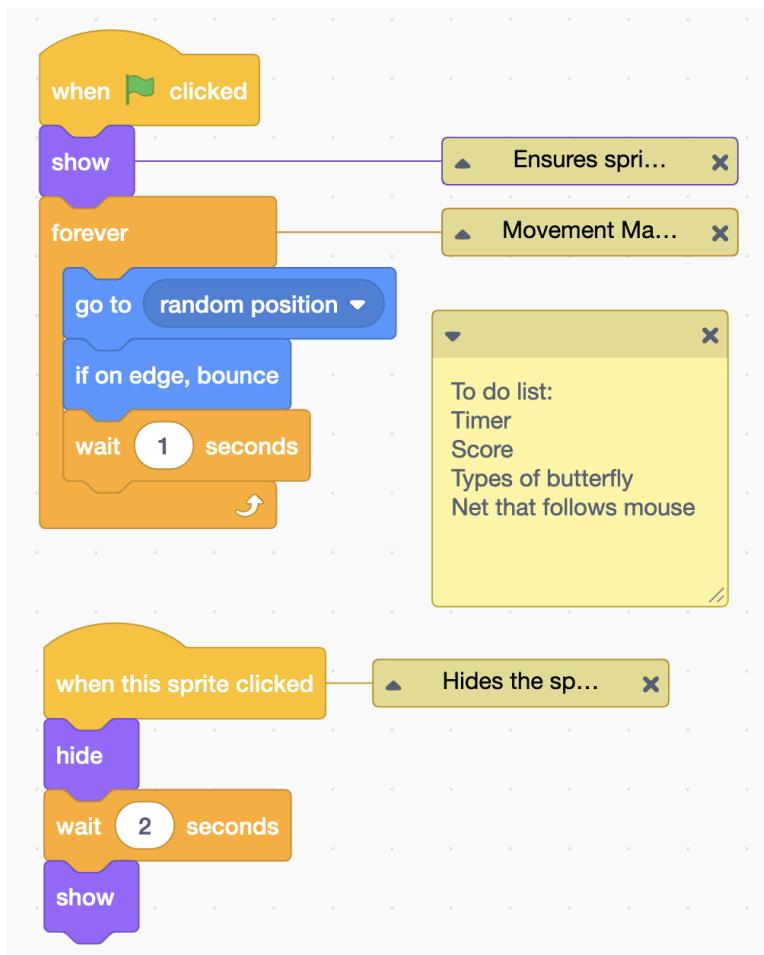
We have a lot of blocks on the screen and more than one stack – if we keep adding things could get messy!

Comments can improve the readability of our code and are good practice in any programming language

To add comments in Scratch, Right-click and select 'add comment'. If you right-click on a block the comment will be linked to it.



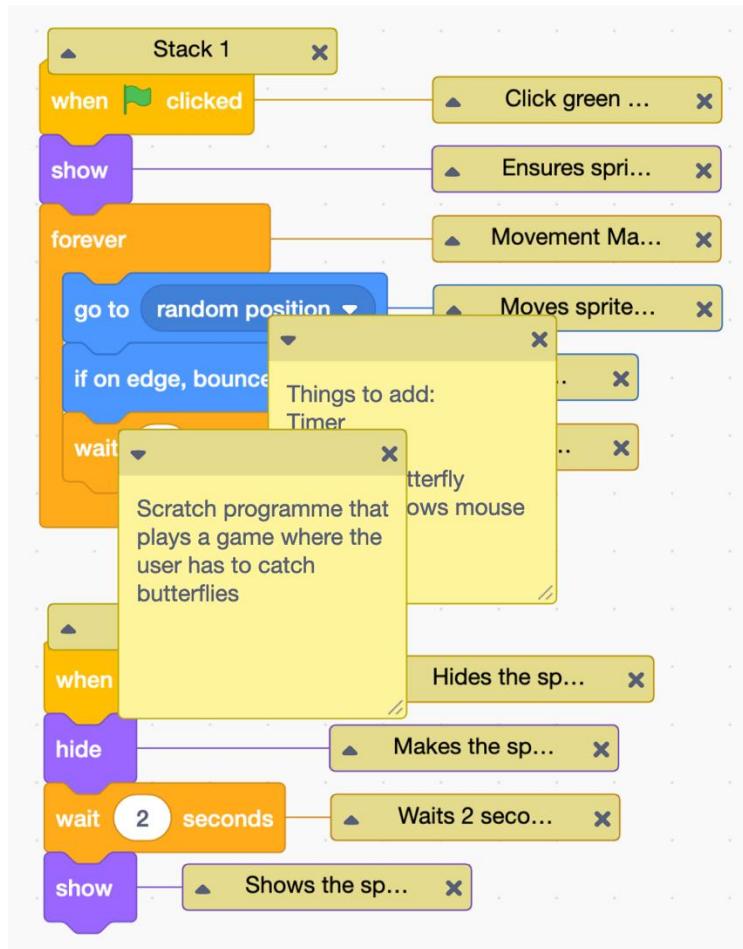
# Comments



Effective use of comments can help:

- You remember what each section does / how it works
- You pick up where you left off
- Someone else understand your code

# Comments



Effective use of comments can help:

- You remember what each section does / how it works
- You pick up where you left off
- Someone else understand your code

Just be careful not use so many that it clogs up your code and makes it harder to read!

# How Can We Improve the Game?

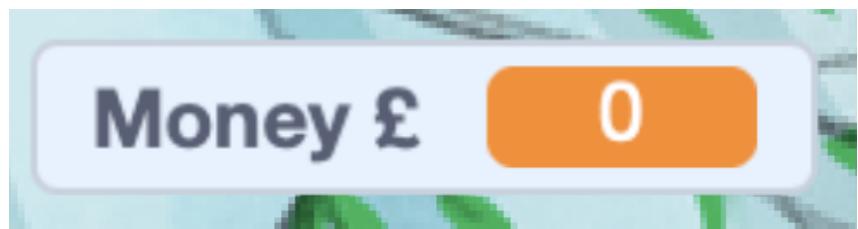
Do you have any suggestions about how we can make the game more fun or interesting?



# Adding a Score System

Wallace used to sell the butterflies he caught for money in order to pay for his trips.

So, we can add a **variable** to our game to make it add money each time we catch a butterfly.

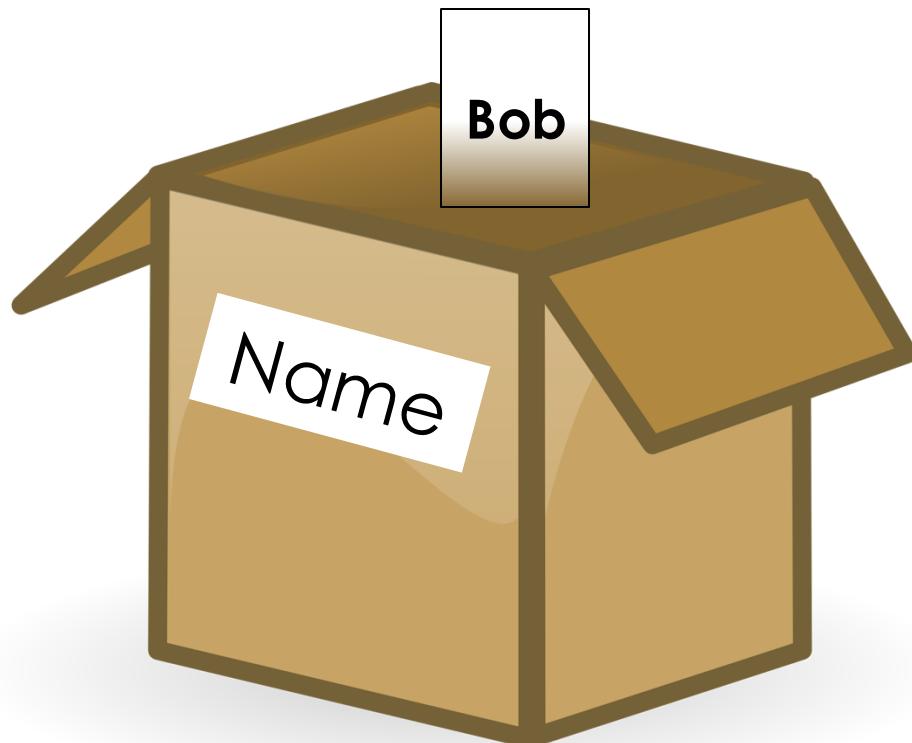


# What Is a Variable?

A **variable** is something that stores data in our program. It is like a box with a label on it.

I can store different things in the box, but the label stays the same.

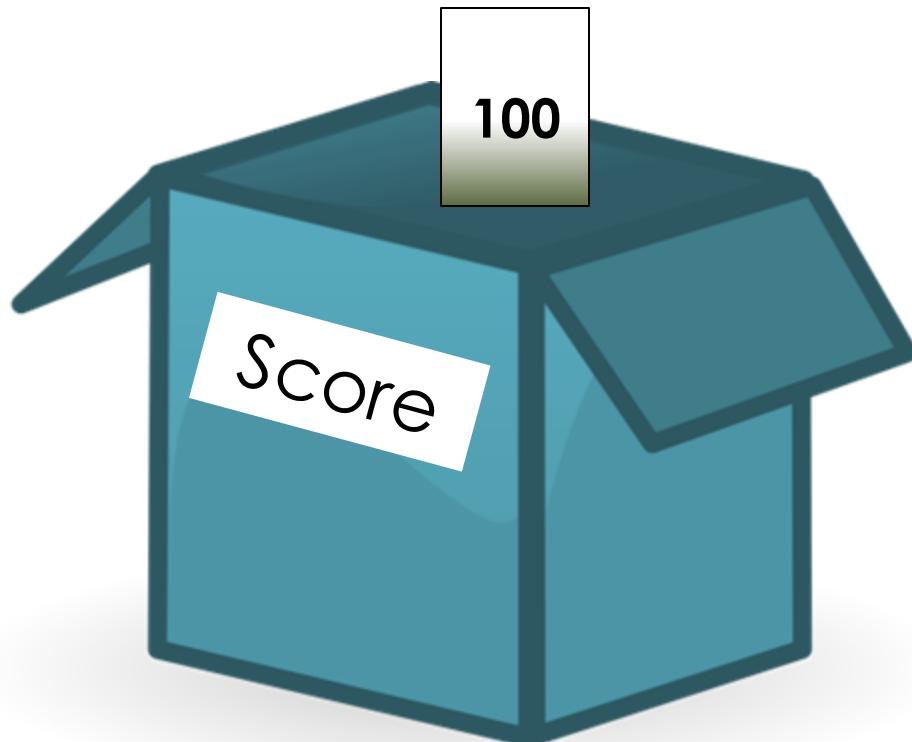
For example, I have stored the word “Bob” in my **variable** which is labelled “Name”.



# Score Variable

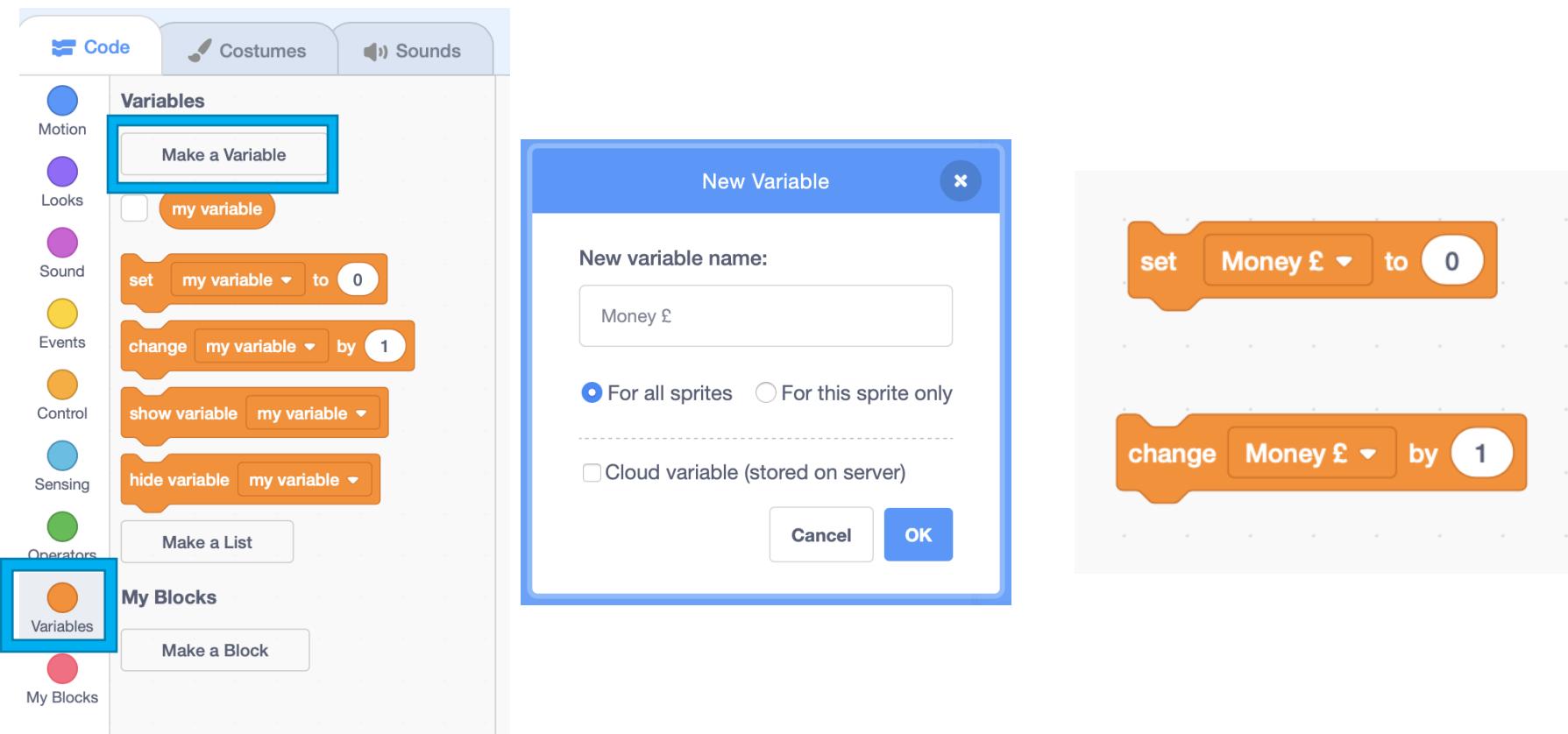
I could have another **variable** named Score which keeps track of our score in a game.

If we're good at the game, our score will go up and will change as we play.



# Adding a Variable in Scratch

To add a **Variable**:



The image shows the Scratch interface with the 'Code' tab selected. On the left, the 'Variables' palette is open, showing a 'my variable' slot and several blocks related to variables. A 'New Variable' dialog box is centered over the workspace, prompting for a new variable name ('Money £') and scope ('For all sprites'). To the right, a script is visible on the stage.

**Scratch Interface Elements:**

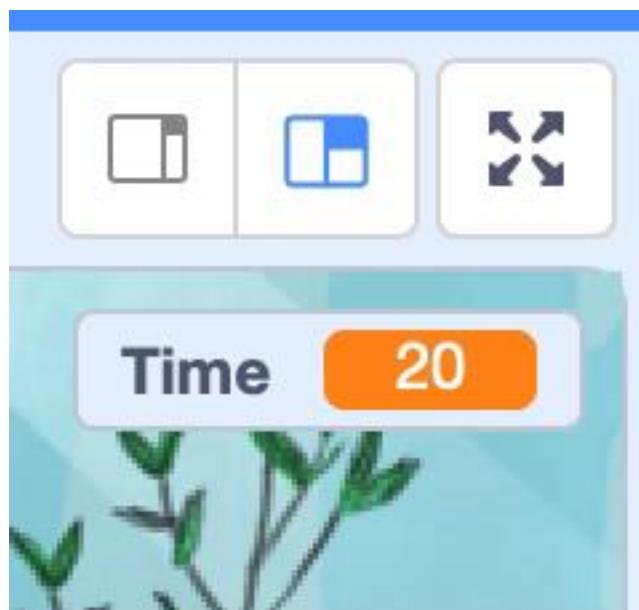
- Code Tab:** Selected tab.
- Costumes Tab:** Unselected tab.
- Sounds Tab:** Unselected tab.
- Variables Palette:** Shows a 'my variable' slot and various variable-related blocks (set, change, show/hide).
- New Variable Dialog:** Title: 'New Variable'. Fields: 'New variable name:' (Money £), 'Scope': 'For all sprites' (selected). Buttons: 'Cancel' and 'OK'.
- Script on Stage:** A script consisting of two blocks:
  - 'set Money £ to 0'
  - 'change Money £ by 1'

# Adding a Time Limit

To make our game more challenging we can add a time limit.

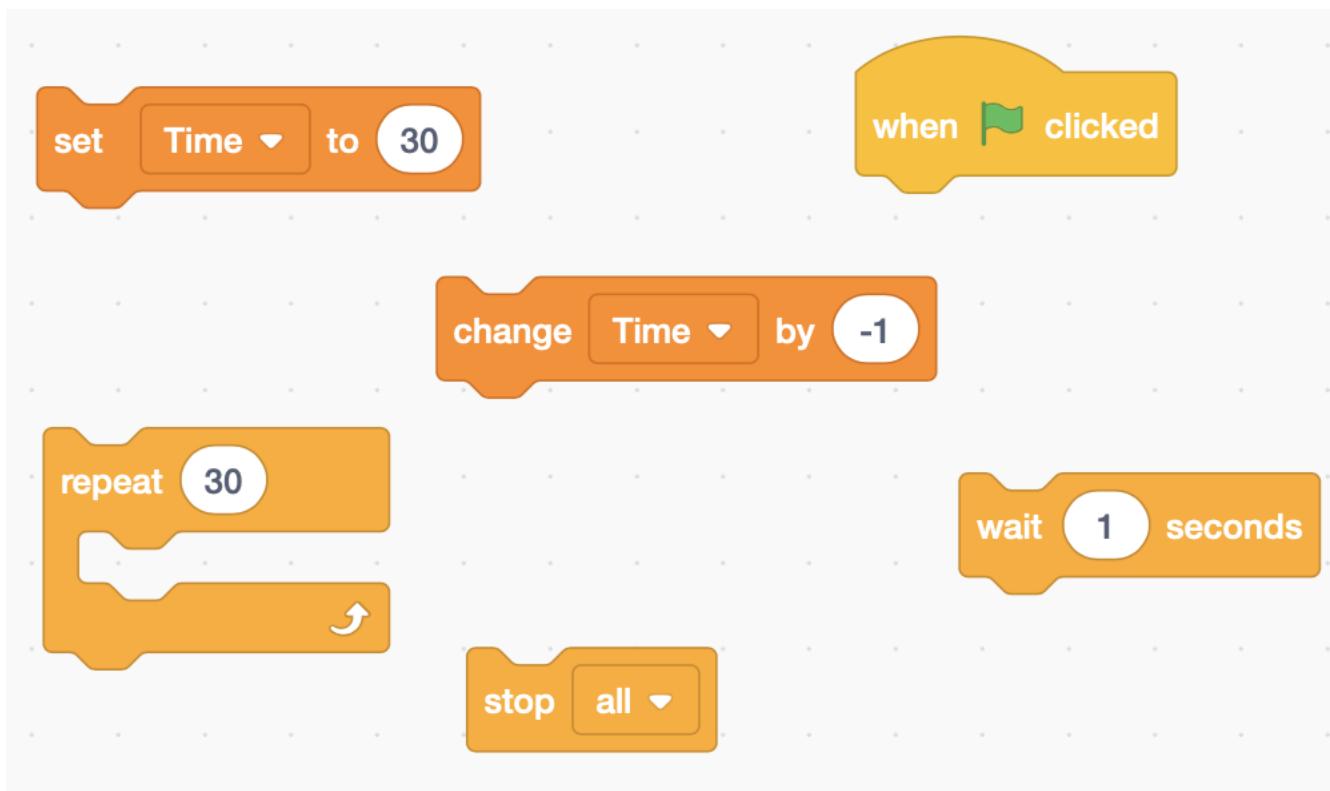
Just like adding our Money **variable**, we need to add a Time **Variable** to our game.

Can you remember how to add a **variable**?



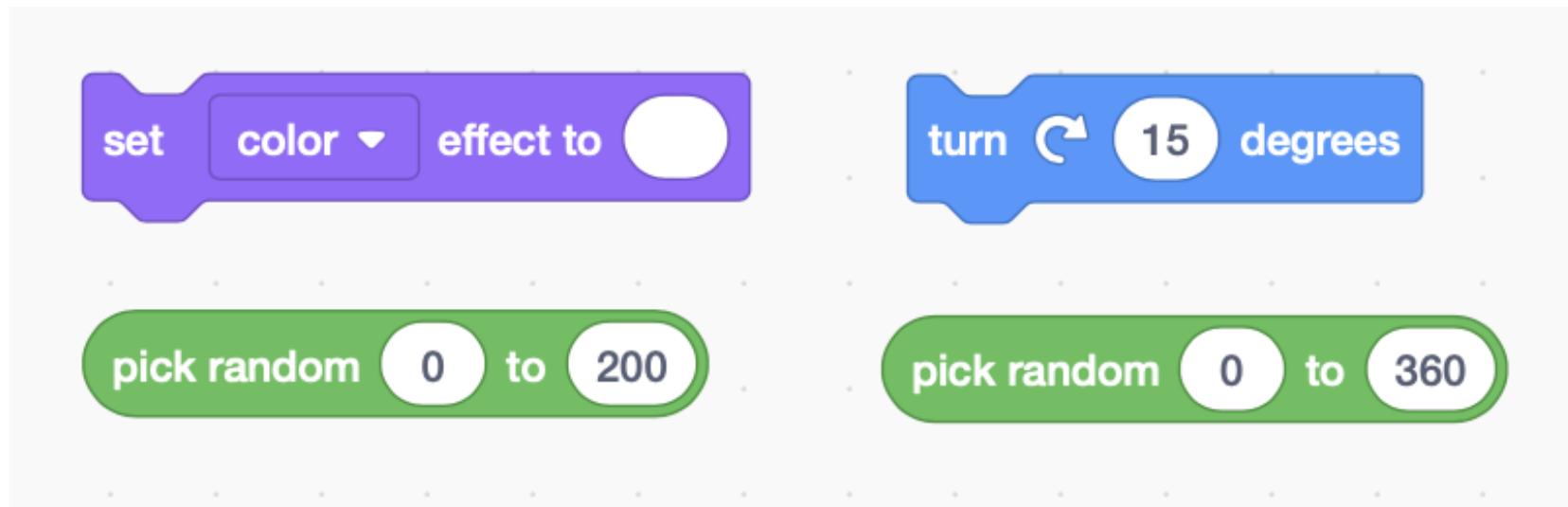
# Creating a Timer

Use the following blocks to create a countdown timer:



# Changing Butterfly Colours and Direction

We can change the Butterfly's colours everytime we've caught one using the following blocks:



Can you see how they fit together? Where would they go in the code?

# Ideas for Extensions

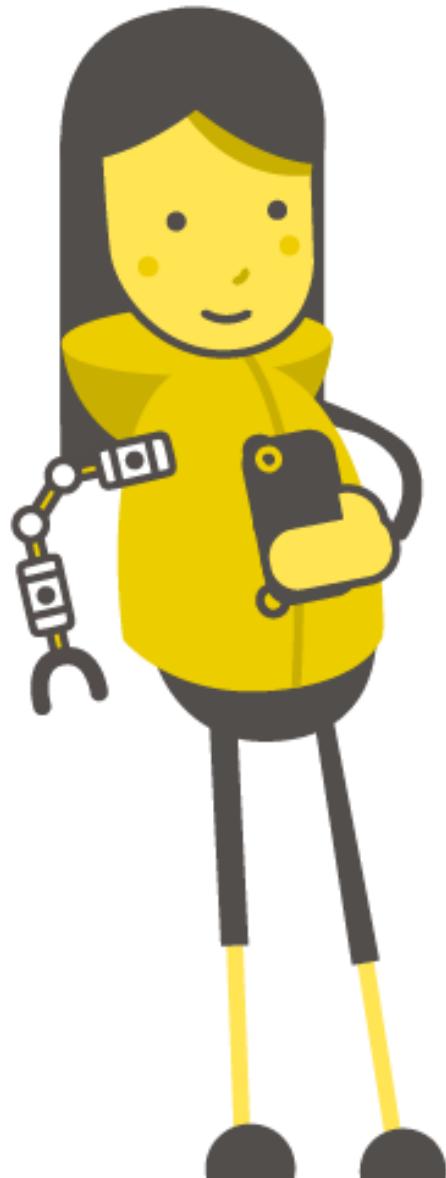
Add a second Butterfly or insect which is harder to catch but gets you more money if you do

Make a swiping noise when you catch a Butterfly

Adding a High Score **variable** to keep track of your best score.

Having a different background for when the game is finished.

Adding a message that says “Time’s Up” at the end.

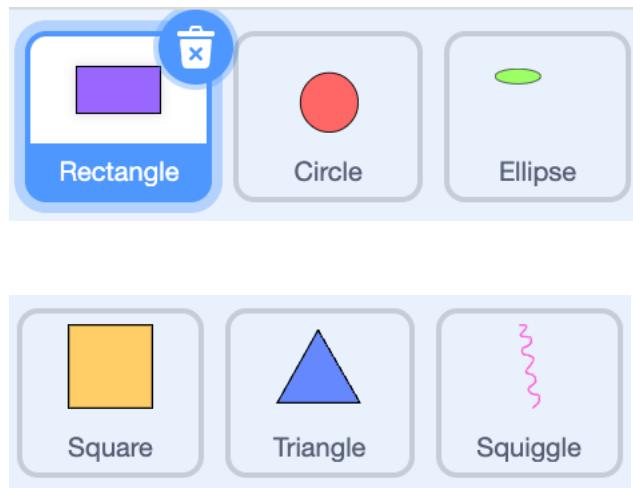


# Scratch Art (Algorithms)

# Algorithm Art



# Algorithm Art



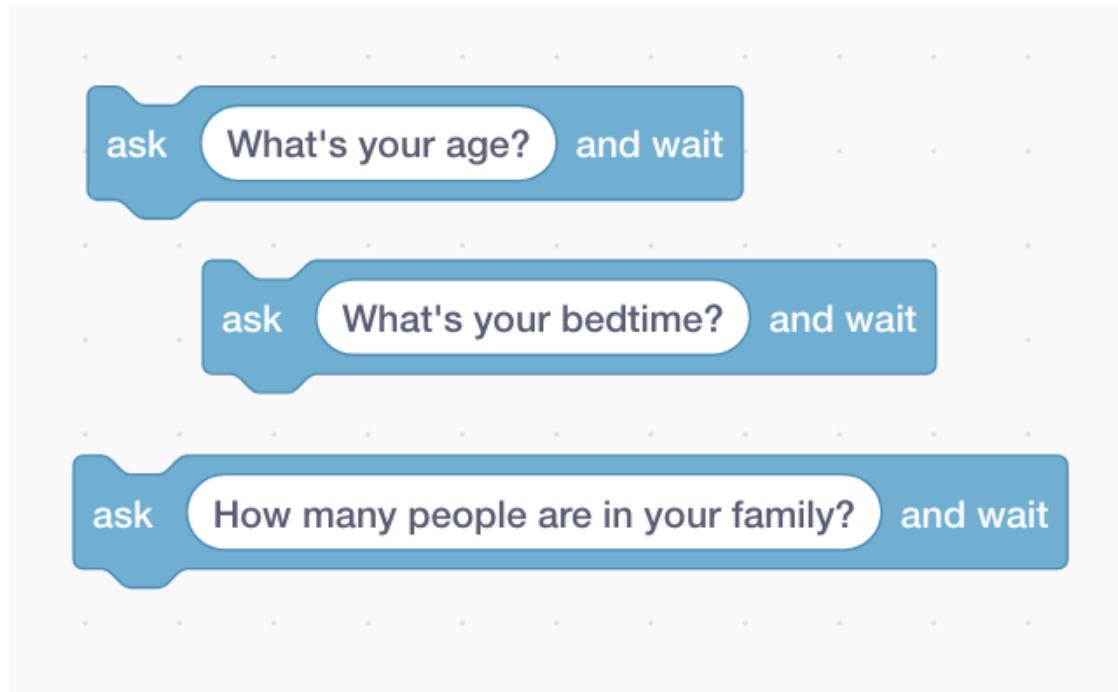
Create sprites of various shapes.

The size and colour of these will be altered by the algorithm to add diversity, so only distinctly different shapes are needed.

The same code will be copied into each shape.

# Algorithm Art - Questions

Produce 3 questions to give number values, these will be used to create our art:



# Algorithm Art- Variables

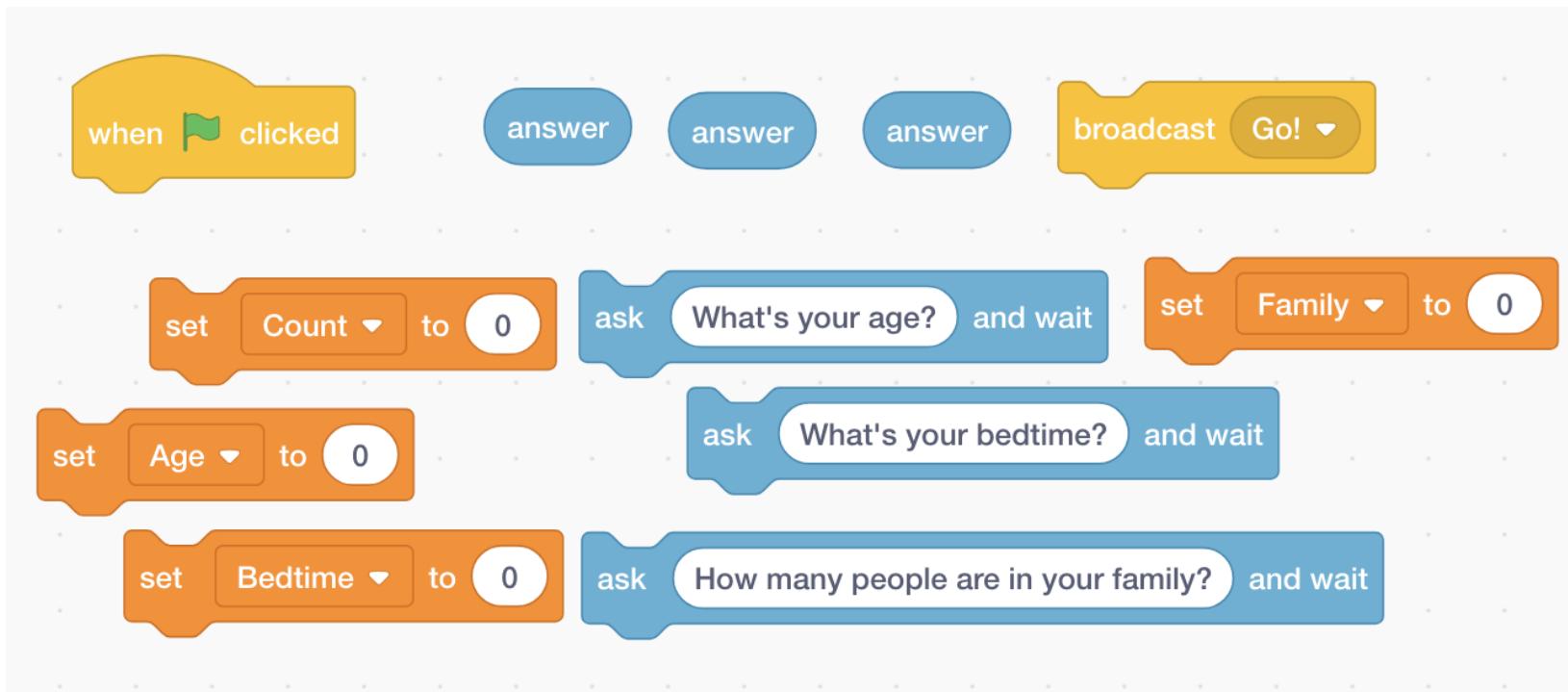


Make four new variables to begin. These will be user inputs to create the art, name them appropriately:

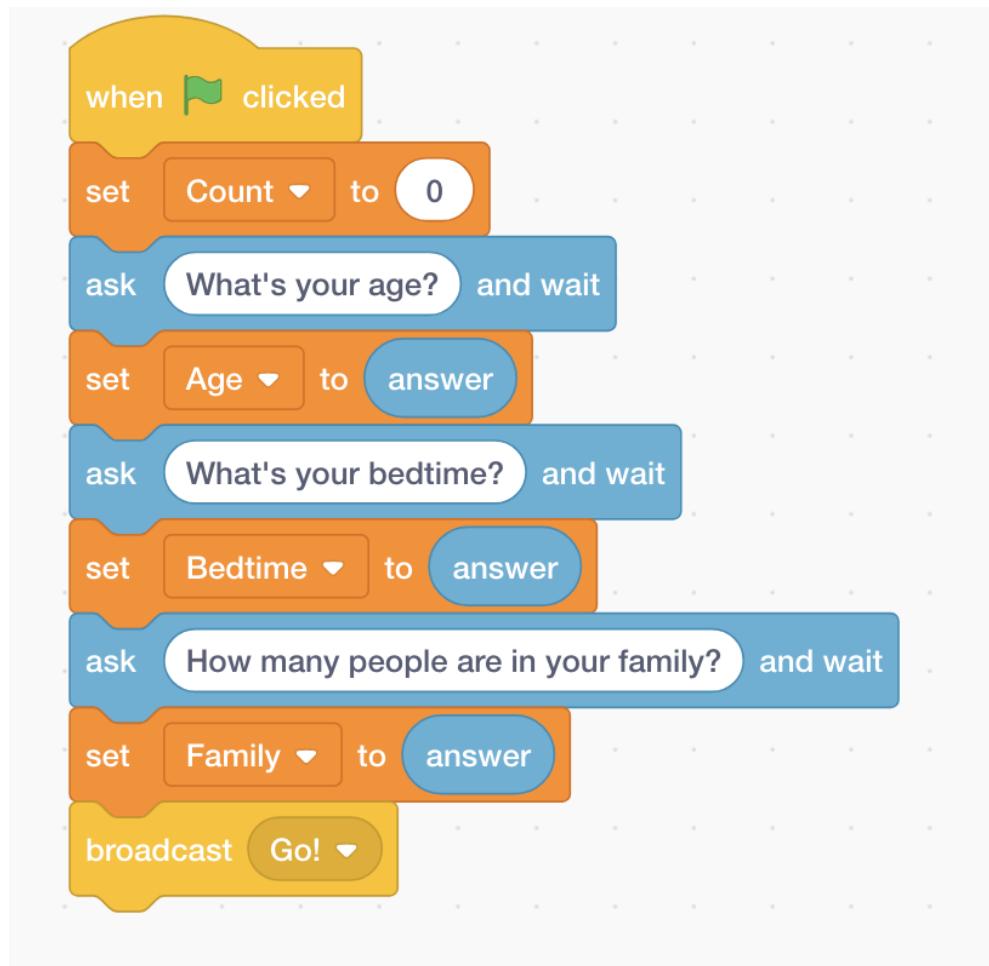
**count, answer 1, answer 2 and answer 3**

**These could be Age, Bedtime and Family**

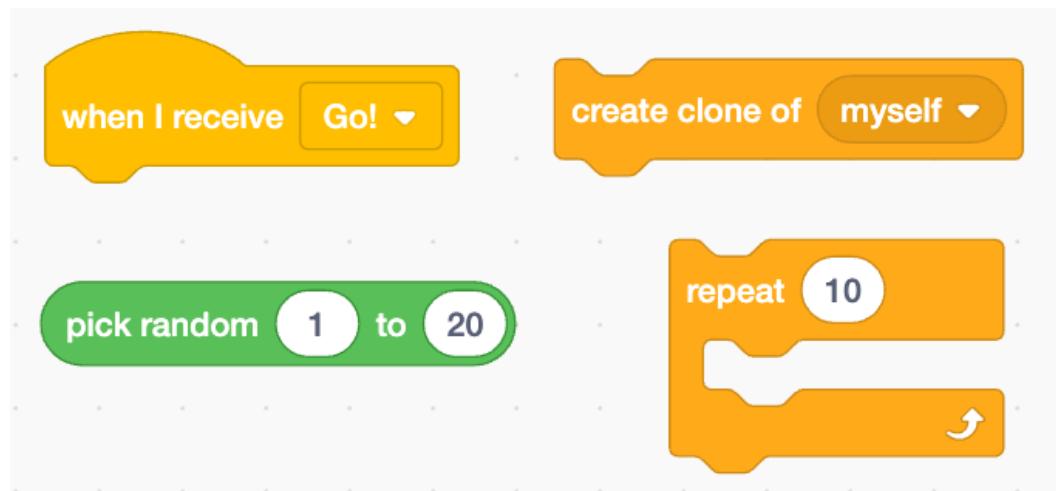
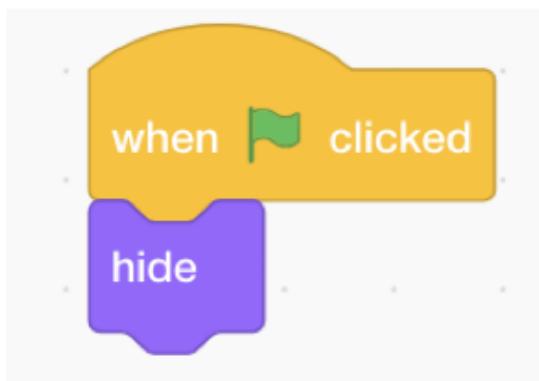
# Algorithm Art – Answers



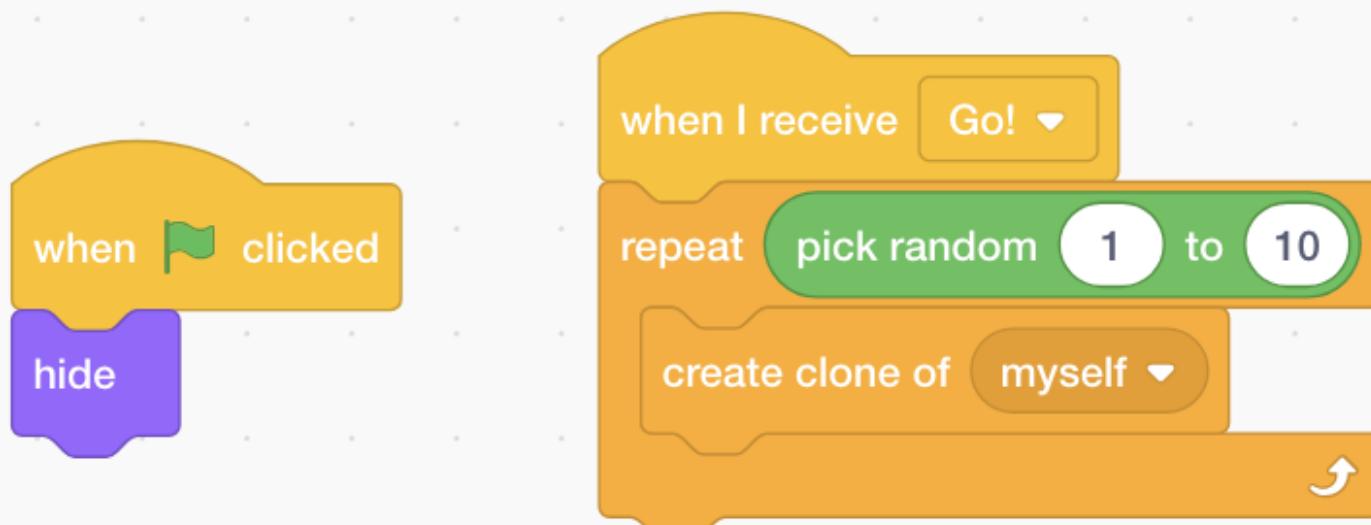
# Algorithm Art – Question/Answer



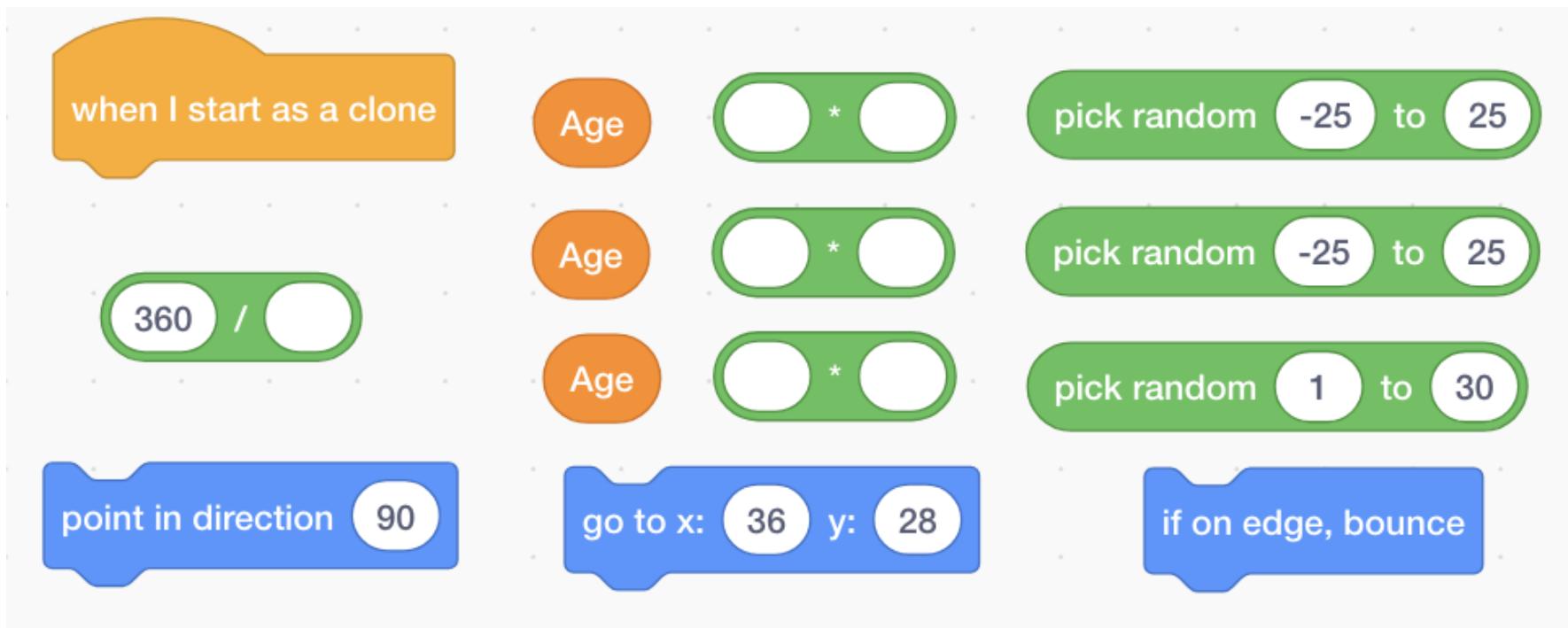
# Algorithm Art – Start Sprite



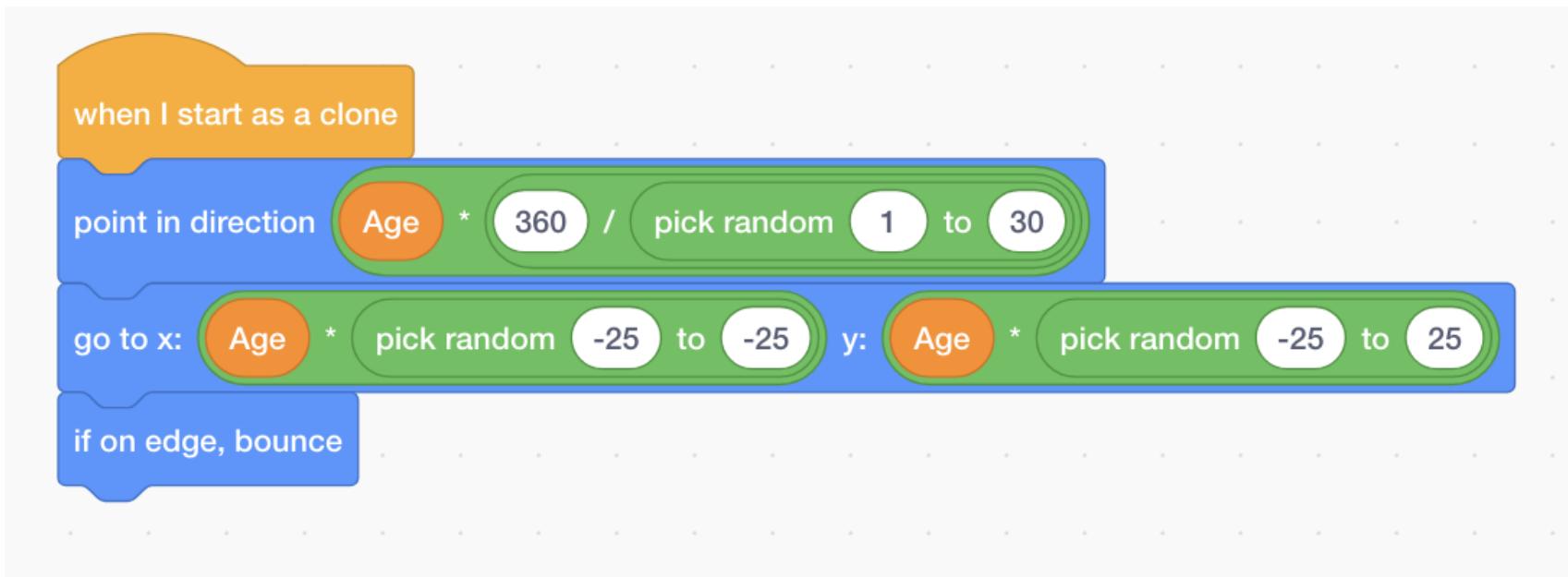
# Algorithm Art – Clones



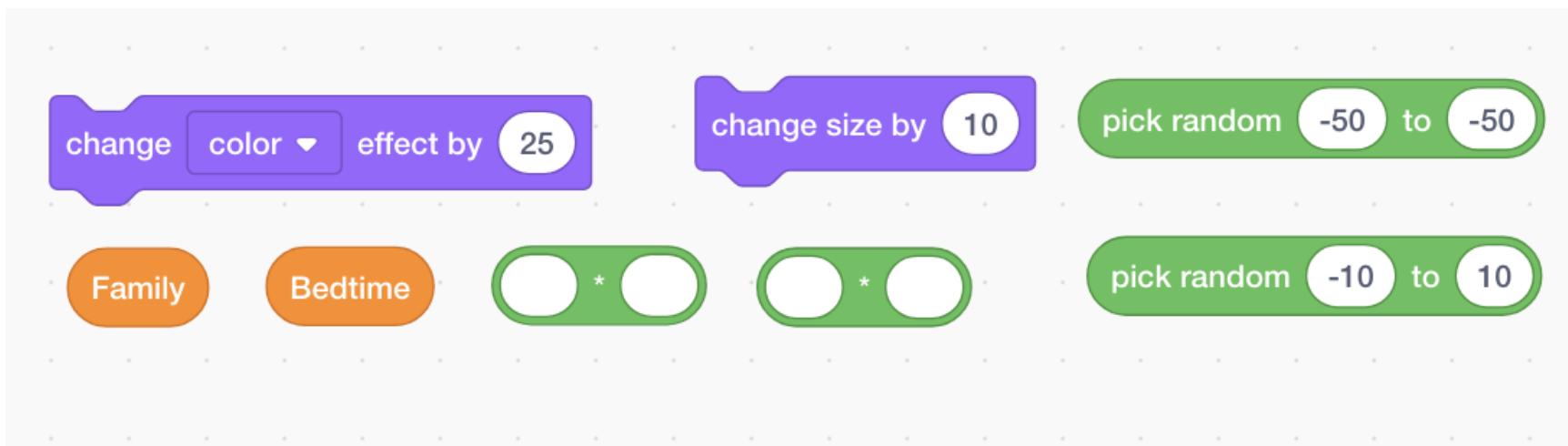
# Algorithm Art – Position



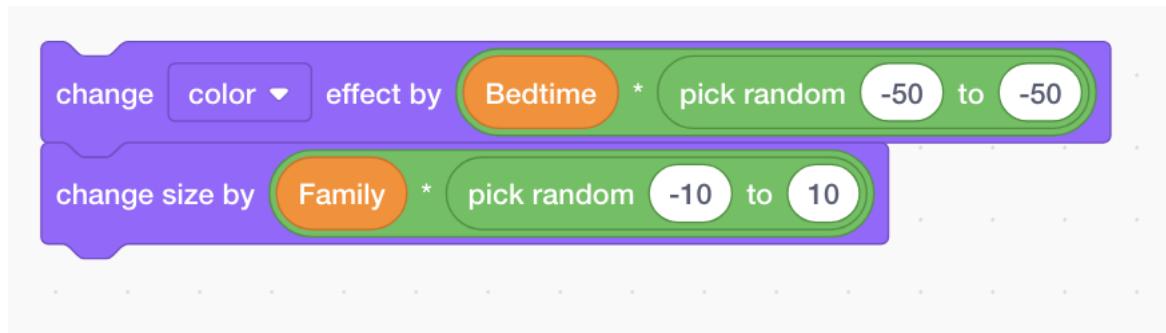
# Algorithm Art – Position



# Algorithm Art – Colour and Size



# Algorithm Art – Colour and Size

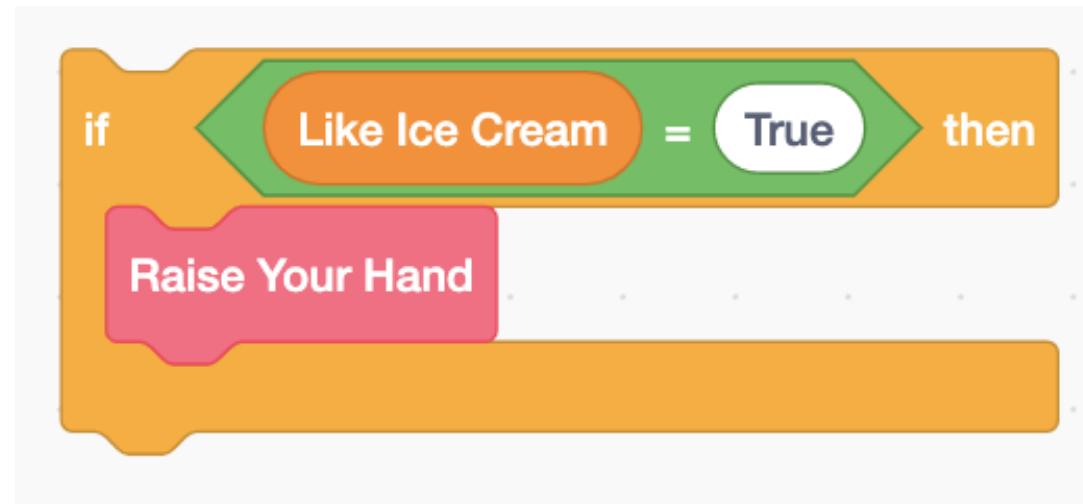


# Scratch Selection



# Ice Cream

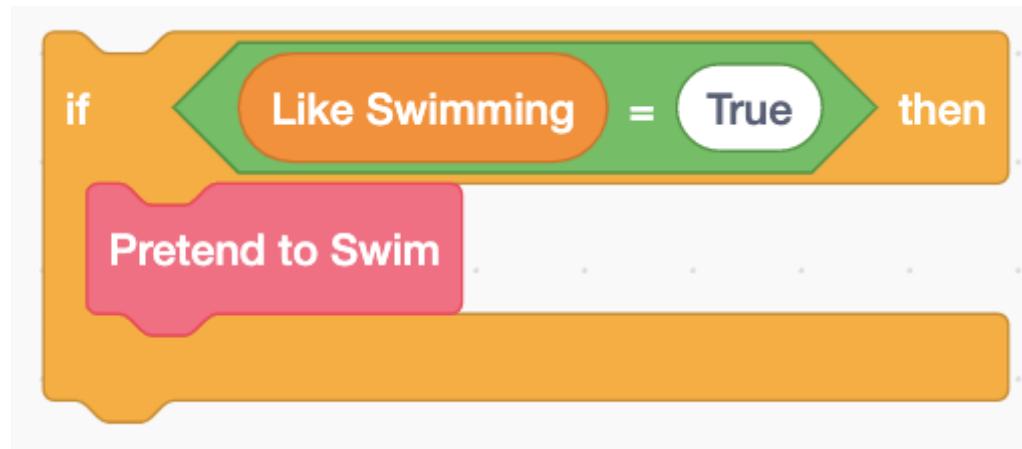
If you like ice cream, then raise your hand.

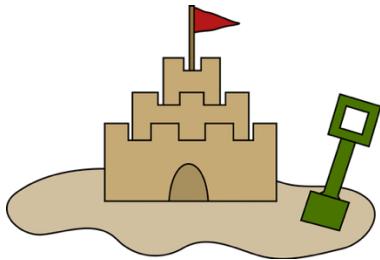




# Swimming

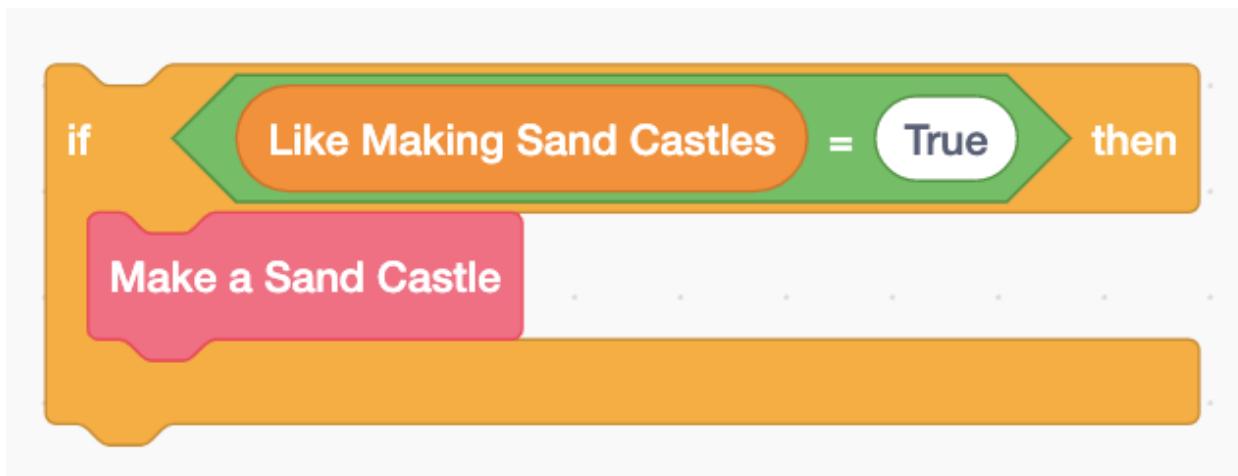
If you like swimming, then pretend you are swimming.





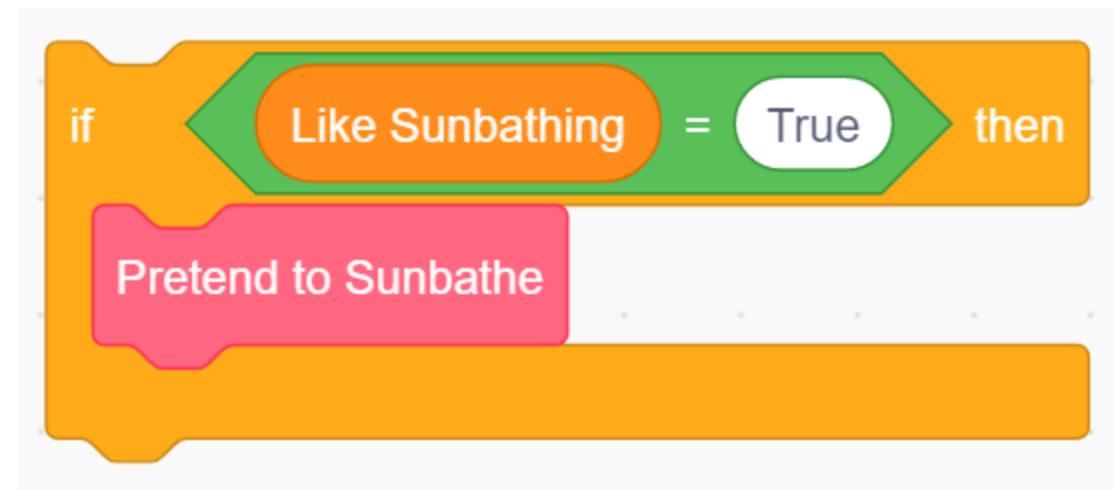
# Sand Castle

If you like to make sand castles, then pretend to make a sand castle.



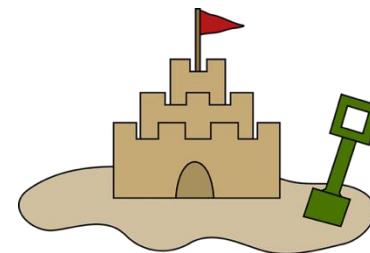
# Sunbathing

If you like sunbathing, then pretend you are sunbathing.



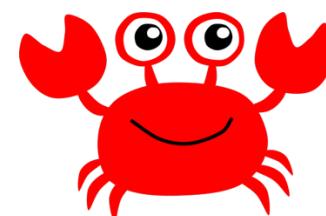
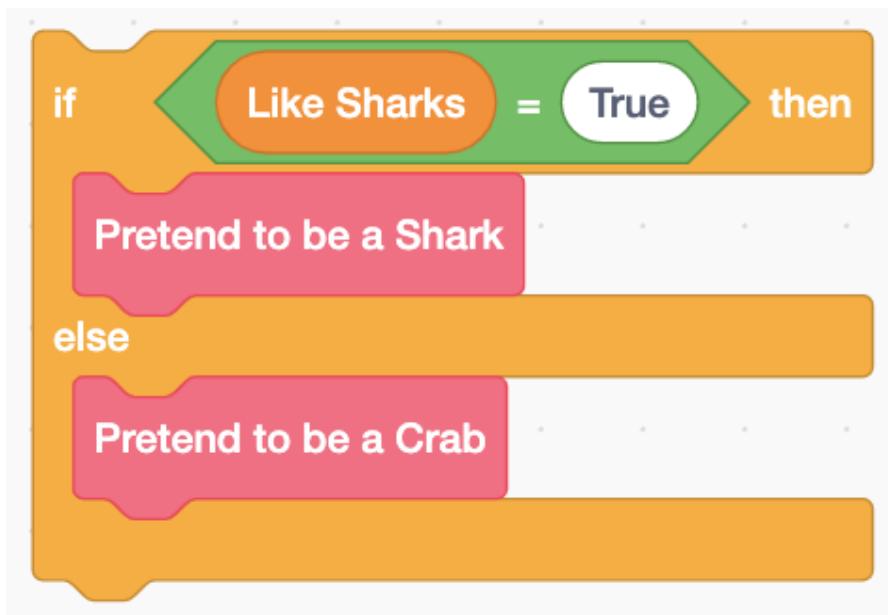
# Decision Making

If you like sand, then pretend to dig sand, **ELSE** pretend to swim.



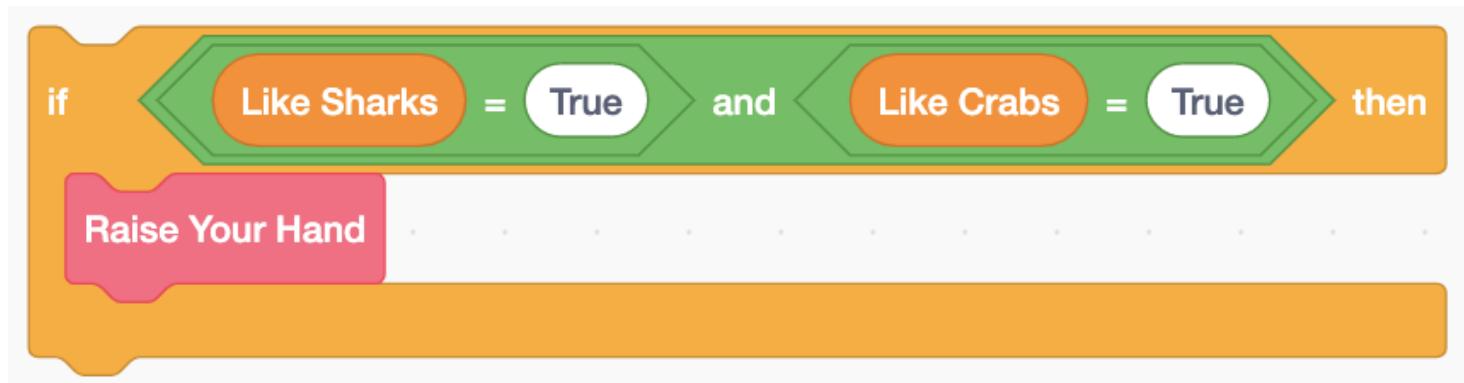
# Decision Making

If you like sharks, then pretend to be a shark,  
**ELSE** pretend to be a crab.



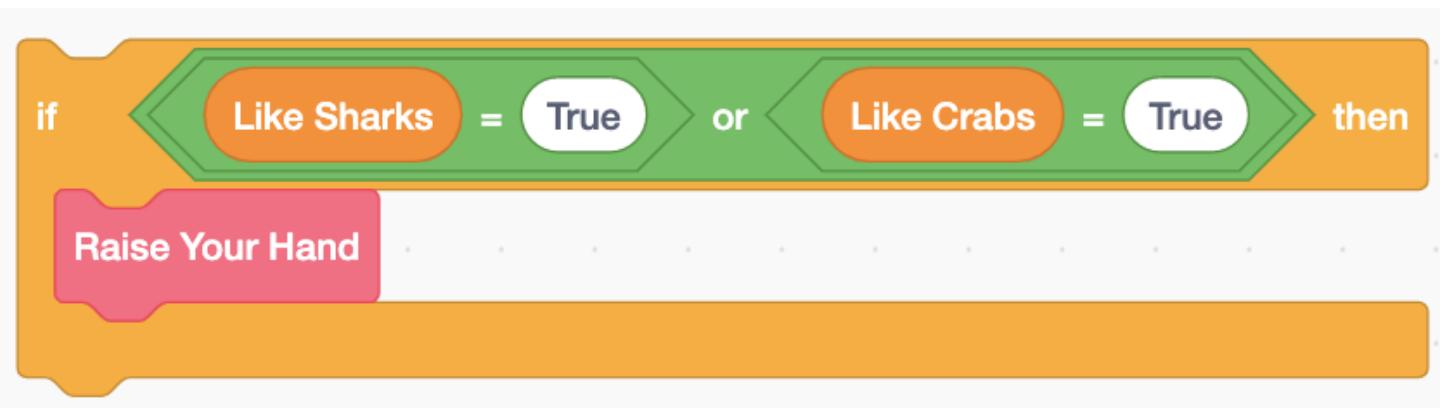
# Combining Conditions

If you like sharks **AND** you like crabs, then raise your hand.



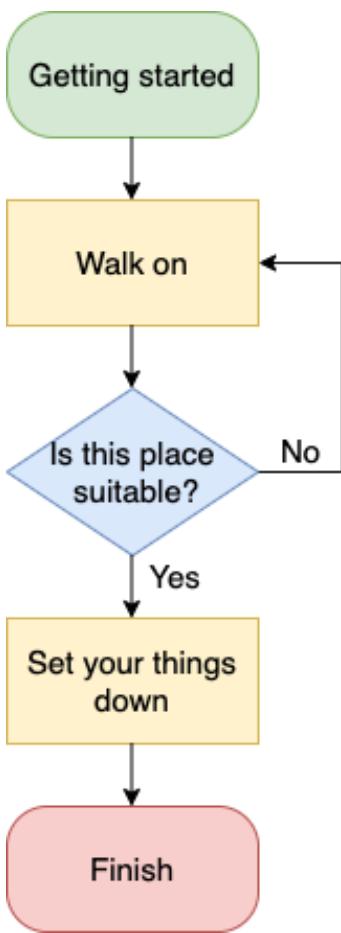
# Combining Conditions

If you like sharks **OR** you like crabs, then raise your hand.



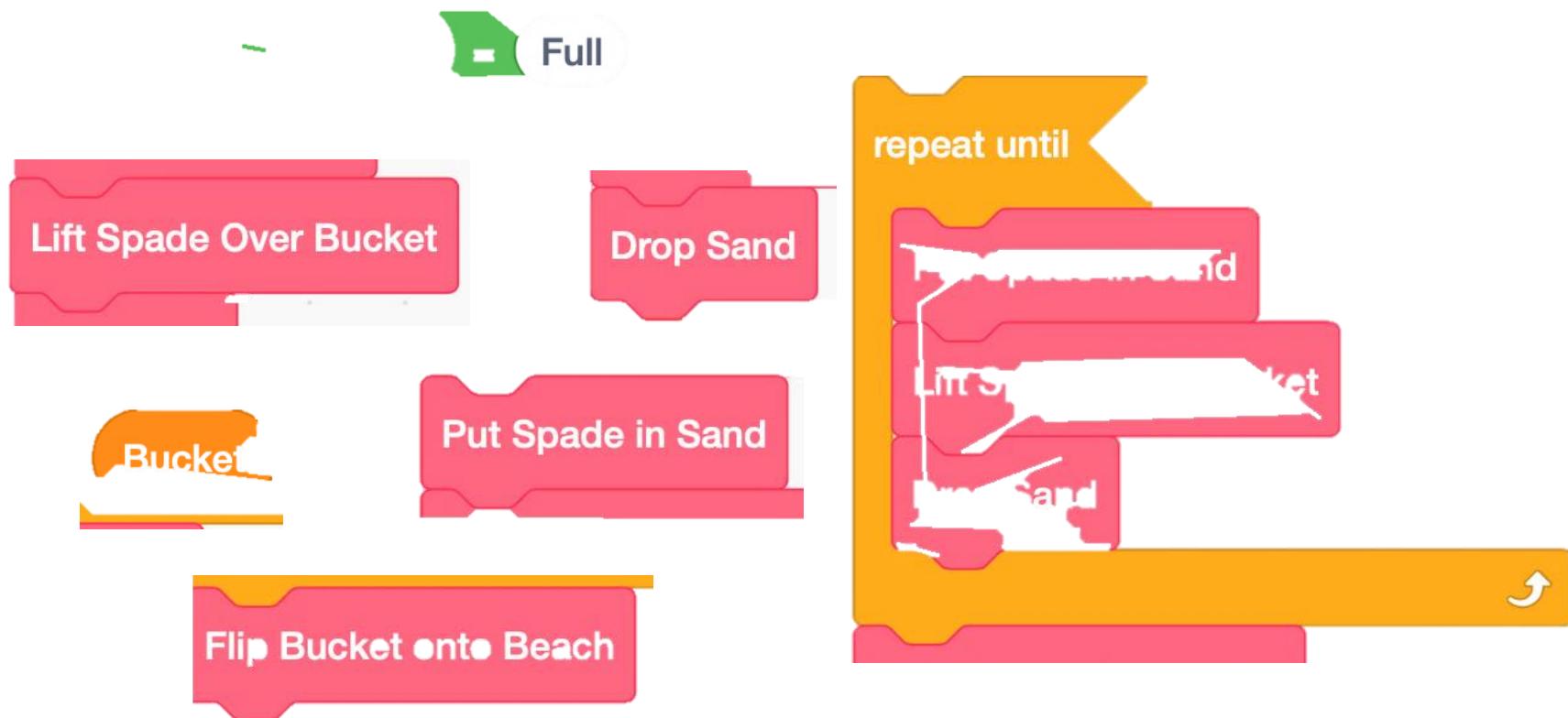
# Find Somewhere on the Beach

We'll have to find somewhere suitable on the beach to do everything we like!



# Activity: Create a Sand Castle

Put the cards in the correct order to create a Sand Castle.

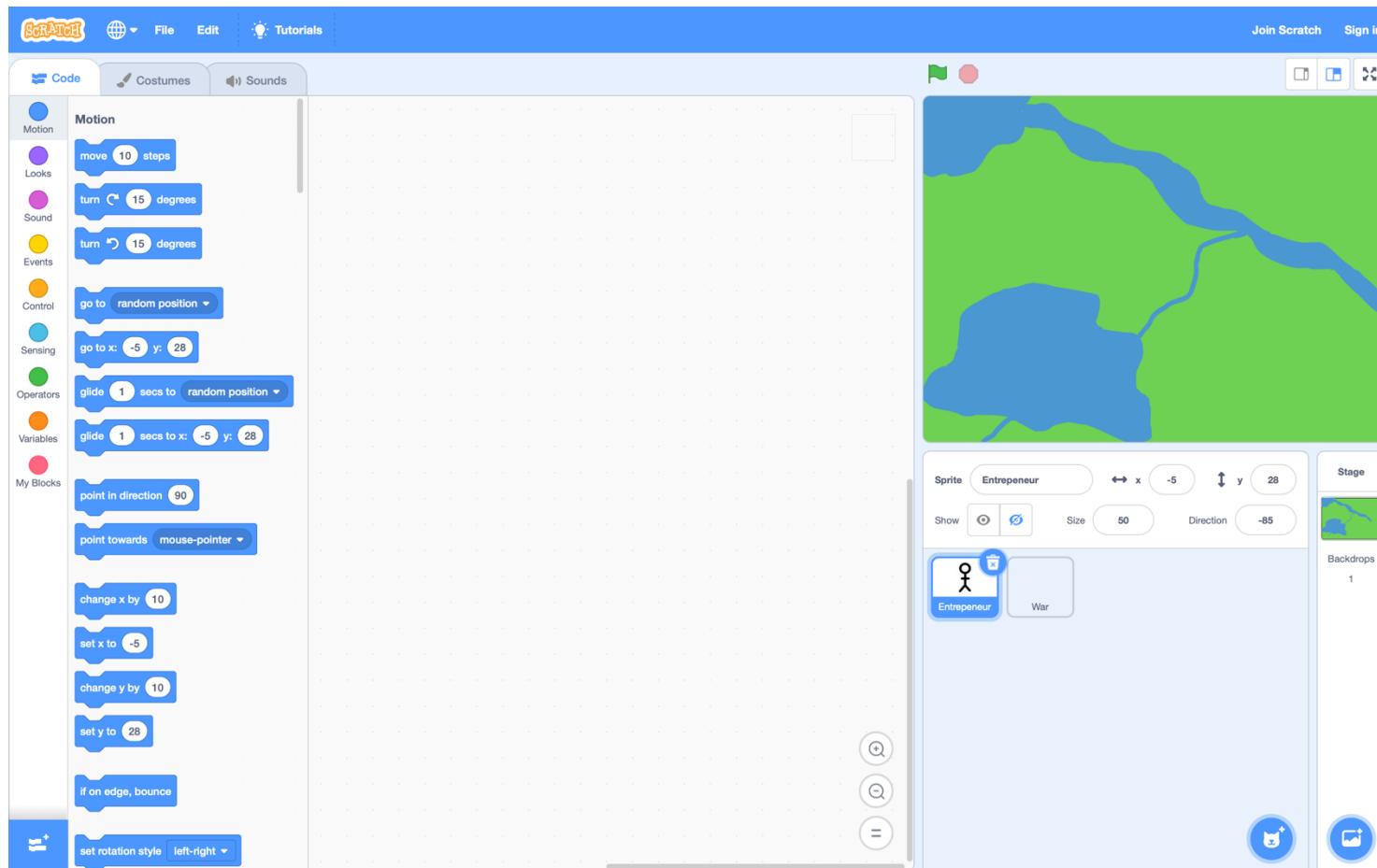




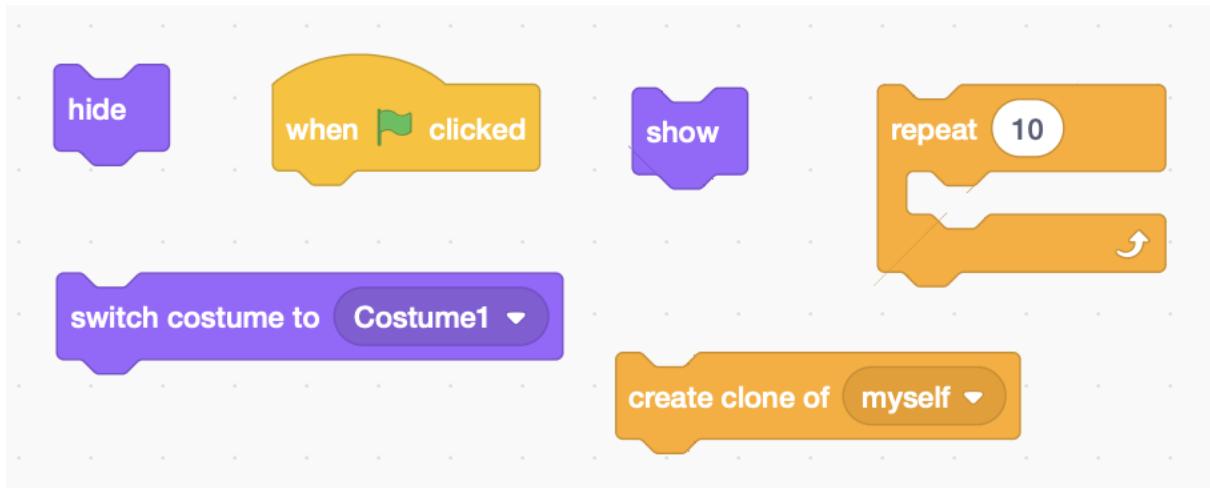
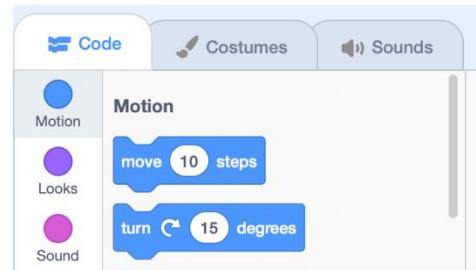
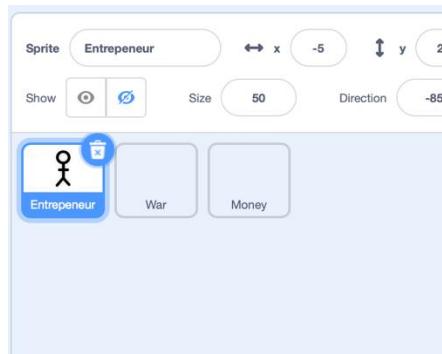
# Modelling In Scratch

# Open the project

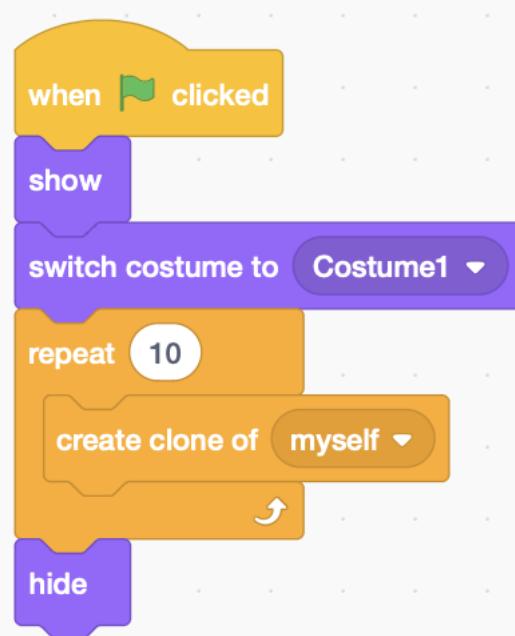
[tc1.me/whypeoplemove](http://tc1.me/whypeoplemove)



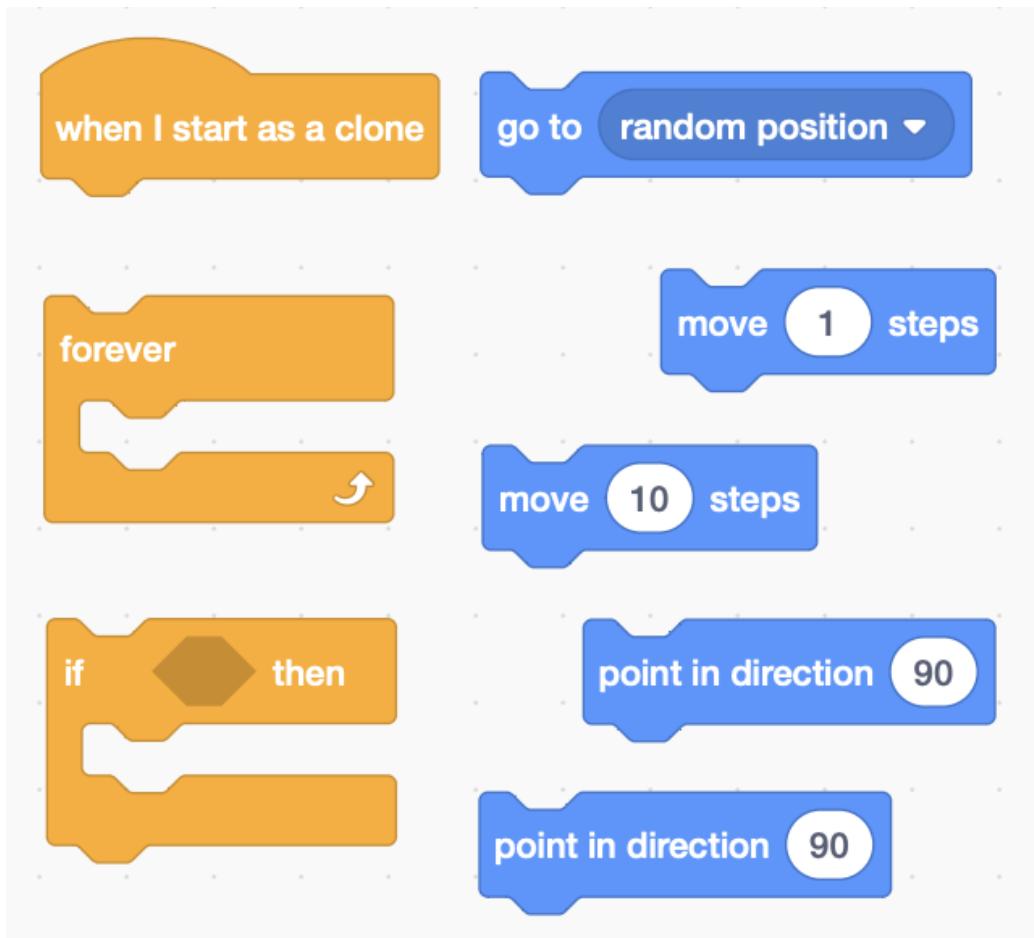
# Code for Entrepreneur Cloning



# Code for Entrepreneur Cloning



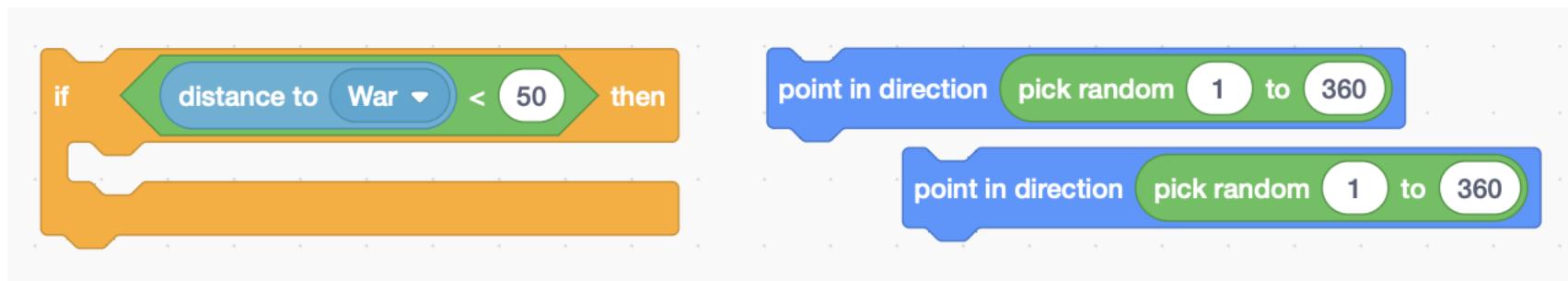
# Code for Entrepreneur Movement



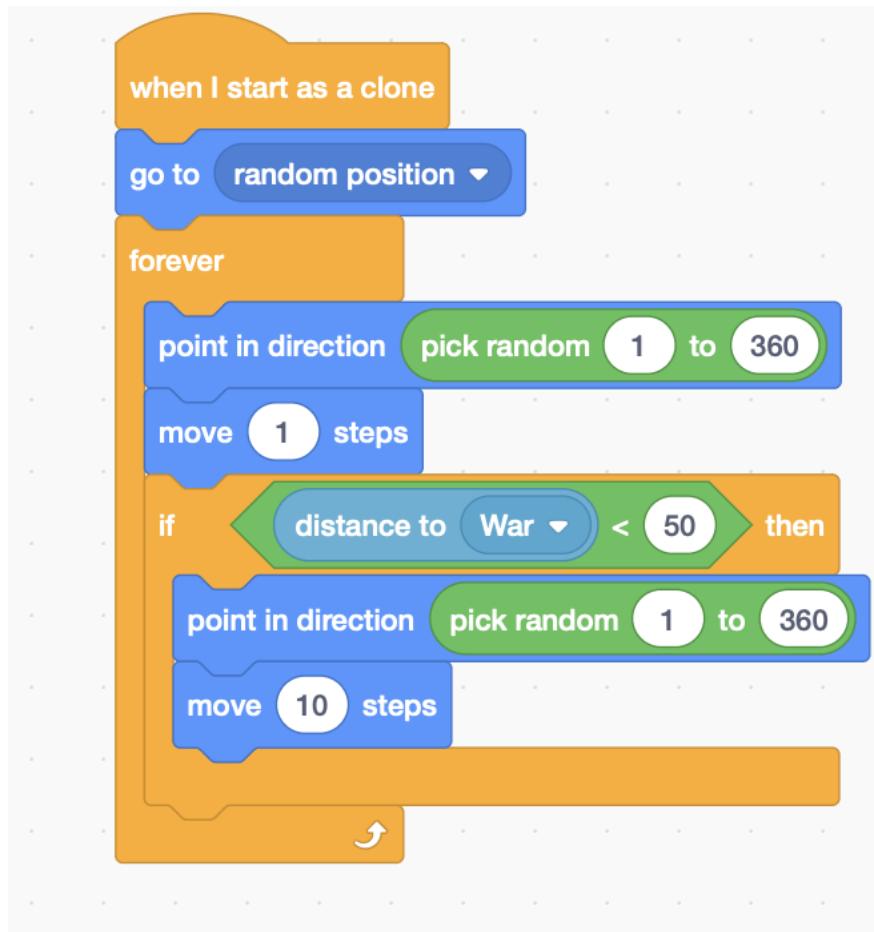
# Code for Entrepreneur Movement



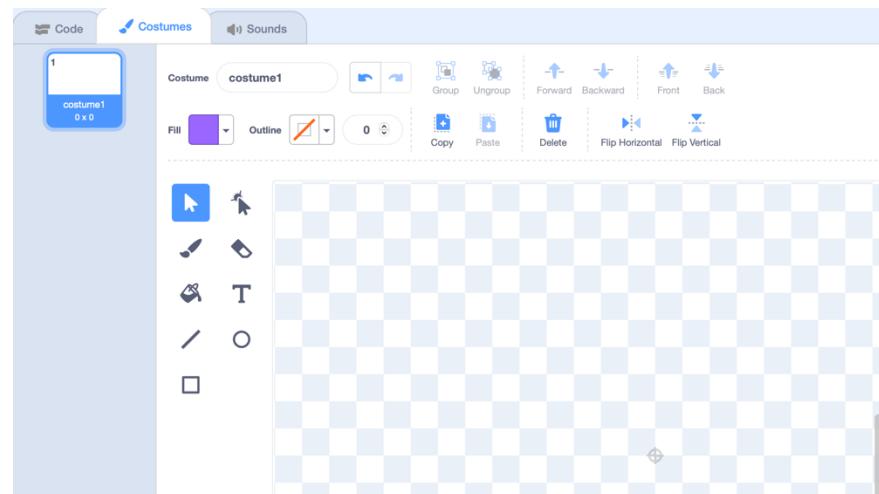
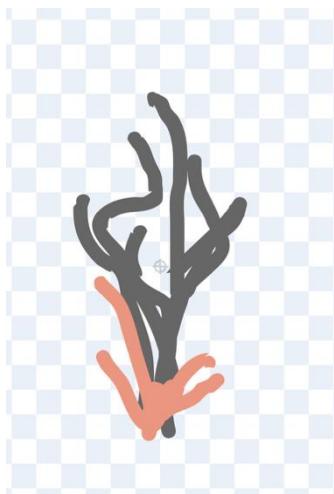
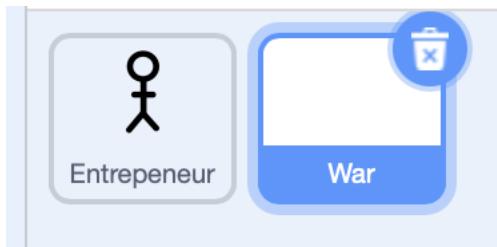
# Code for Entrepreneur Movement



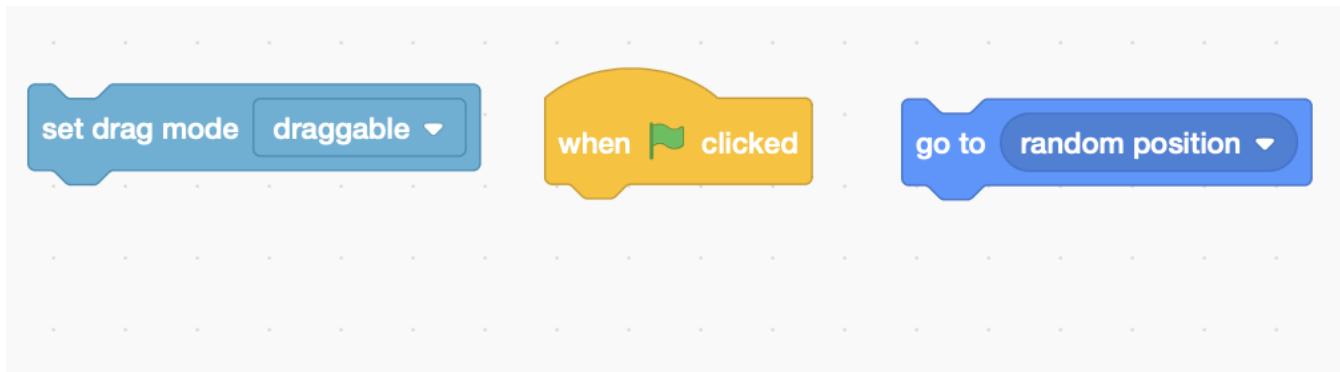
# Code for Entrepreneur Movement



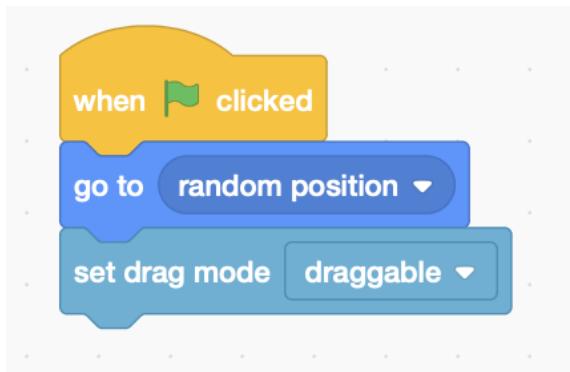
# Design War Sprite



# Code The War Sprite



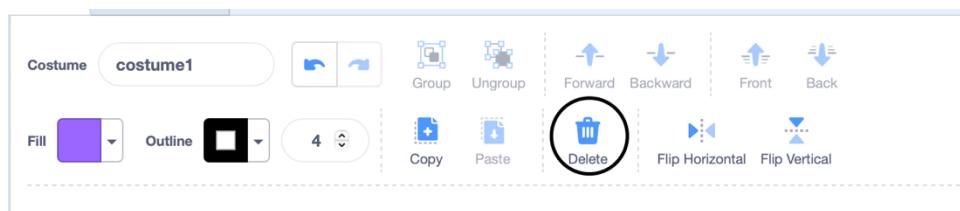
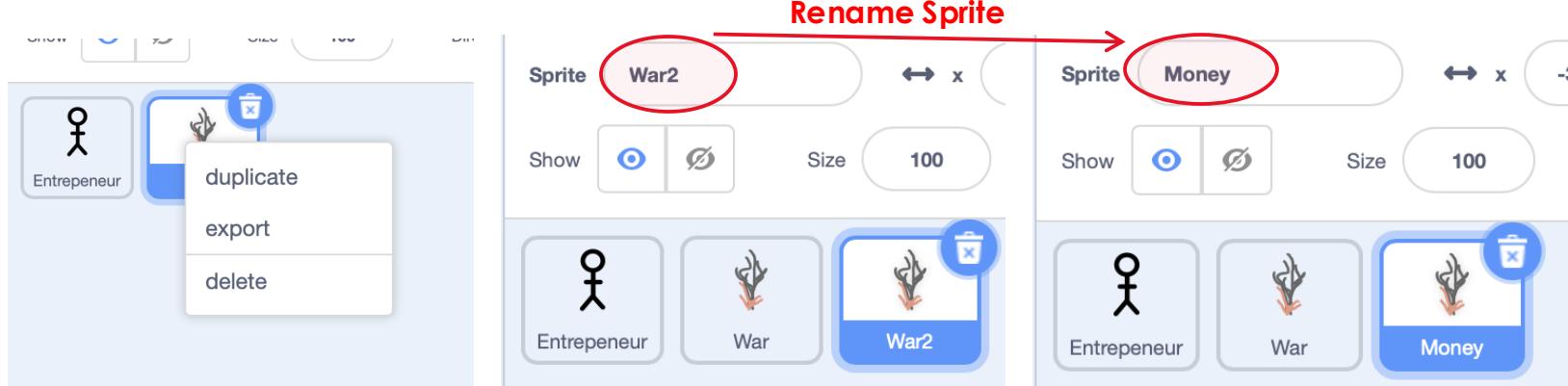
# Code The War Sprite





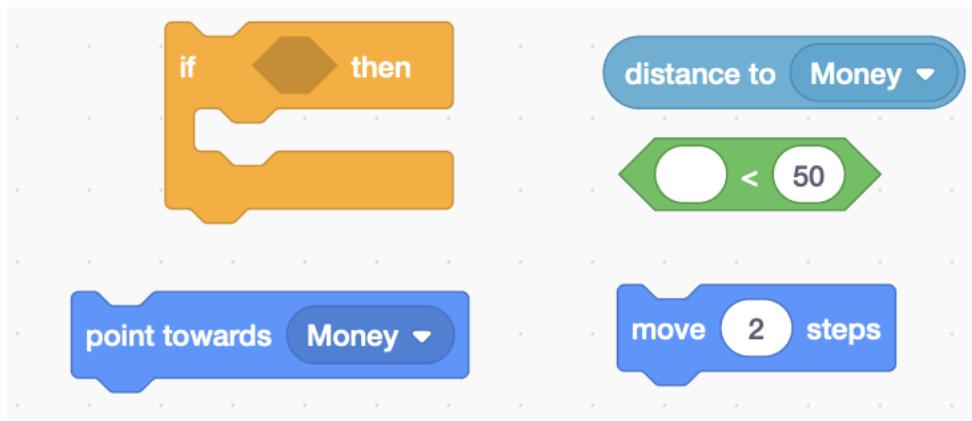
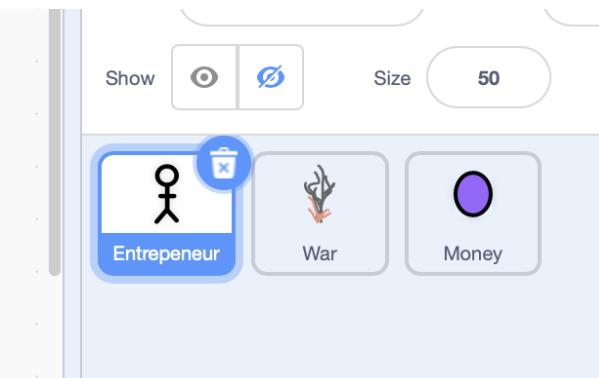
# Extensions!

# Money sprite

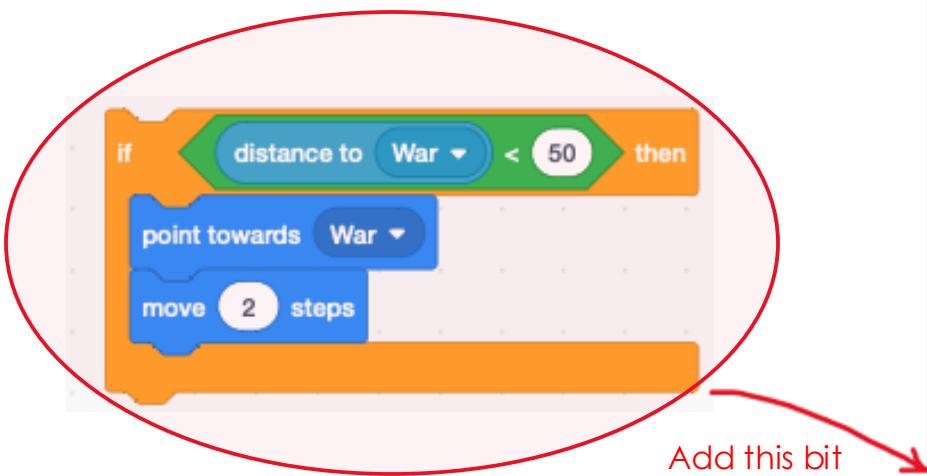


Design your own!

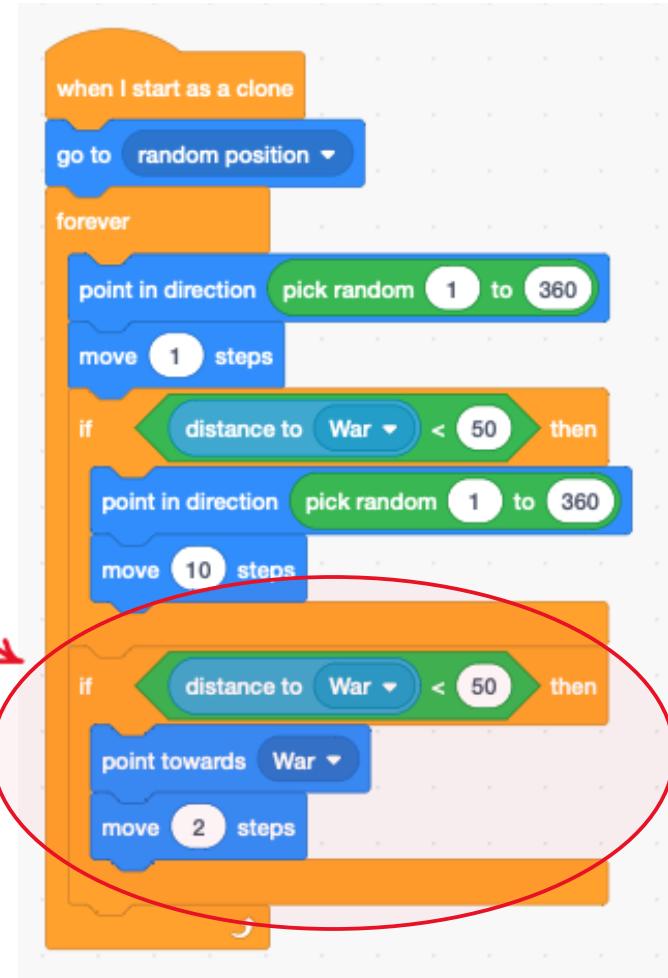
# Move To Money



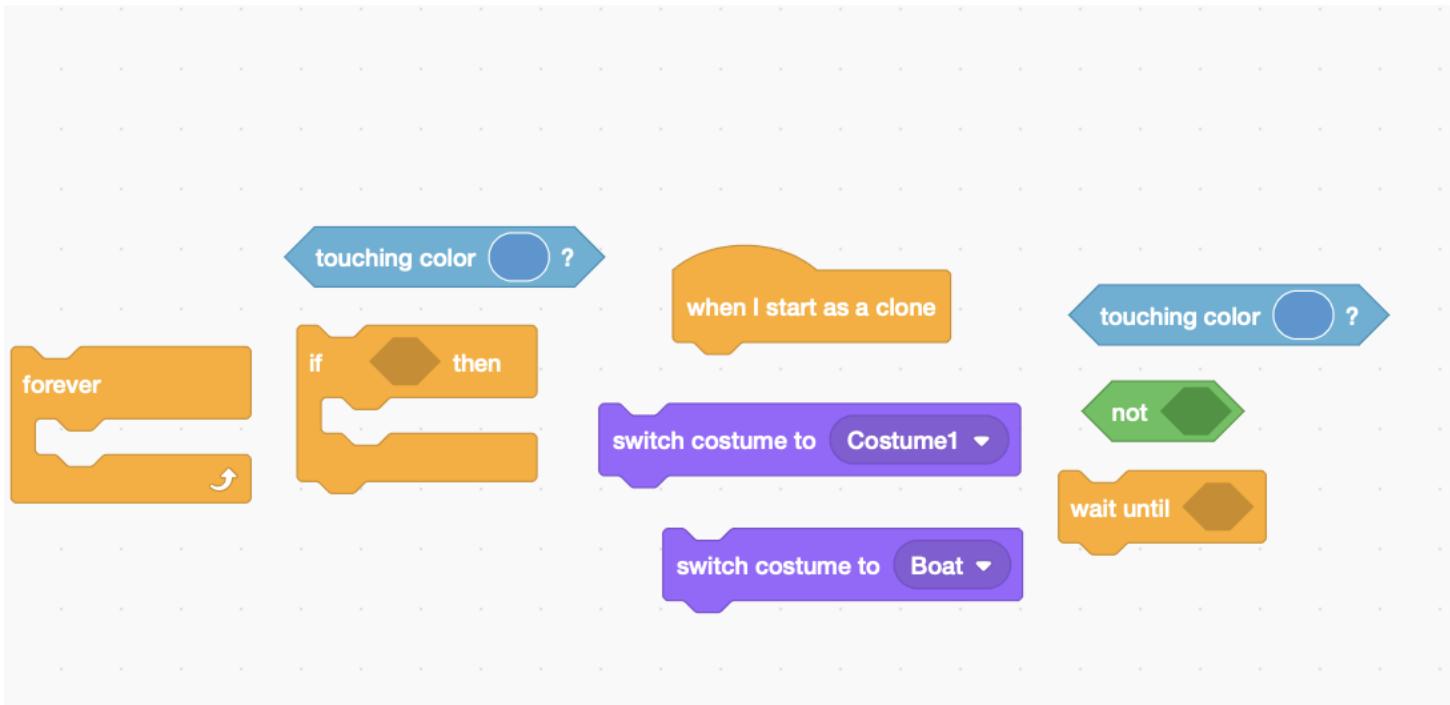
# Move To Money



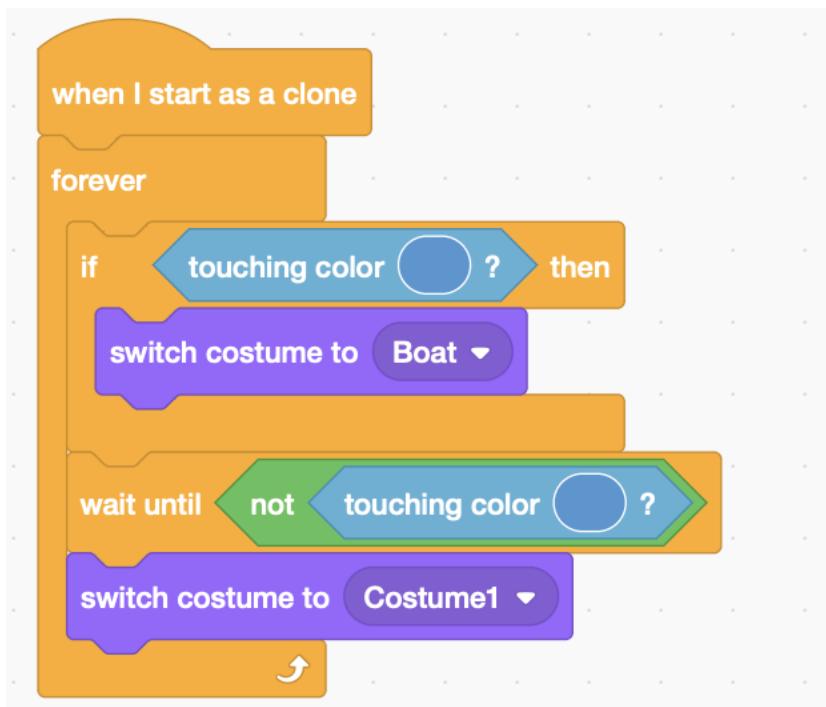
Add this bit  
of code

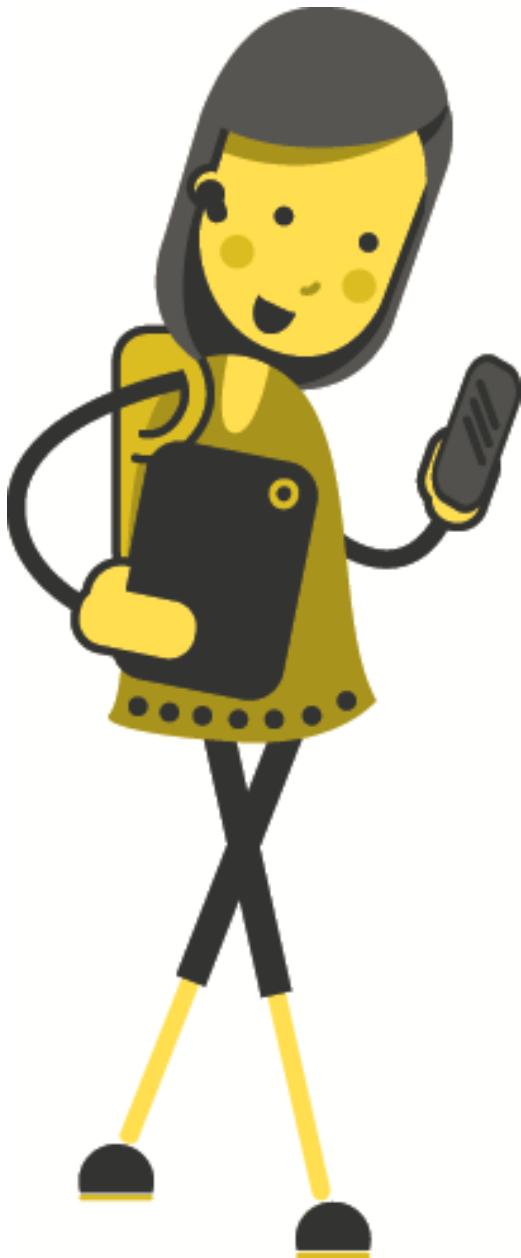


# Boat Travel



# Boat Travel





# Problem Solving using Scratch

# Problem Solving using Scratch

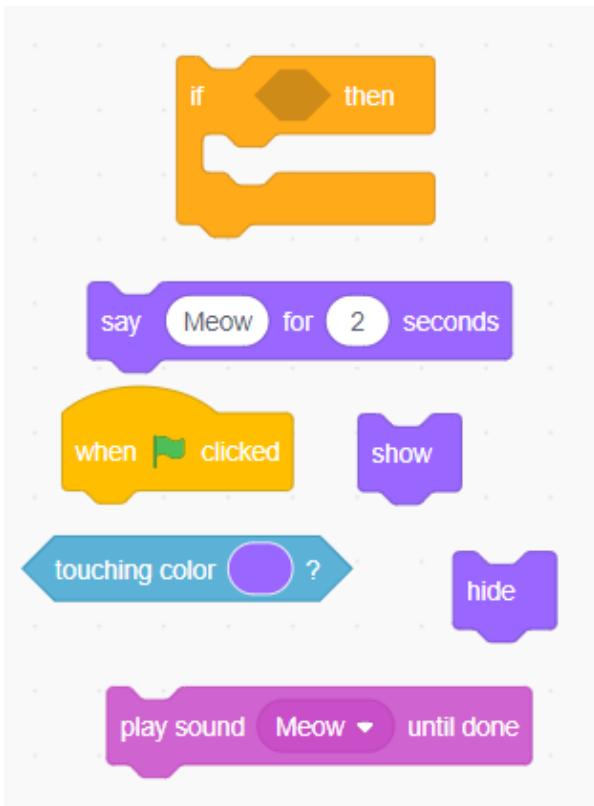
- When starting to set tasks for pupils, you can ask them to put together pre-selected blocks to solve simple problems.
- These problems can be differentiated by the problem itself, or the way the blocks are presented (some may already be placed together for less-able pupils).

## **Task:**

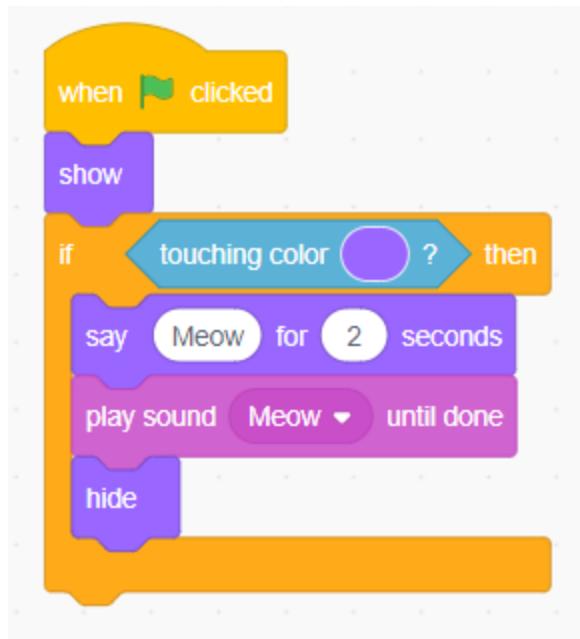
- Can you predict what these blocks will do when placed together in an algorithm?
- Open the program and test your predictions, if it helps.

# Problem Solving Task 1

Blocks:

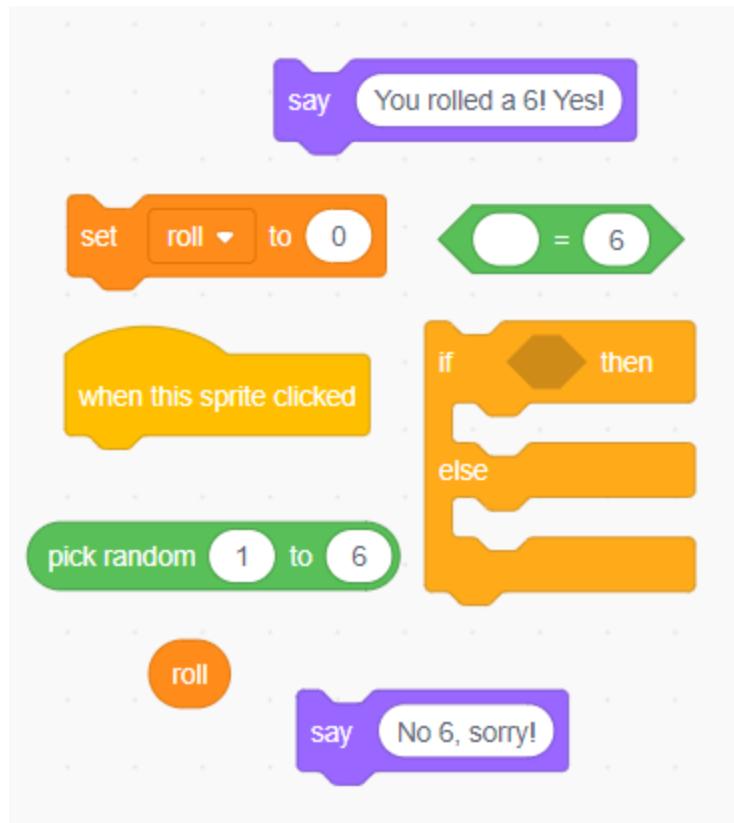


Solution:

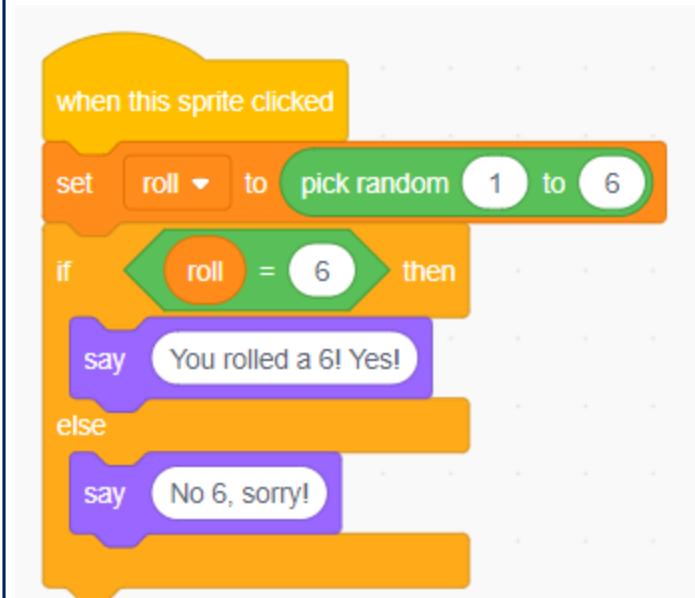


# Problem Solving Task 2

Blocks:



Solution:





# Testing and Debugging

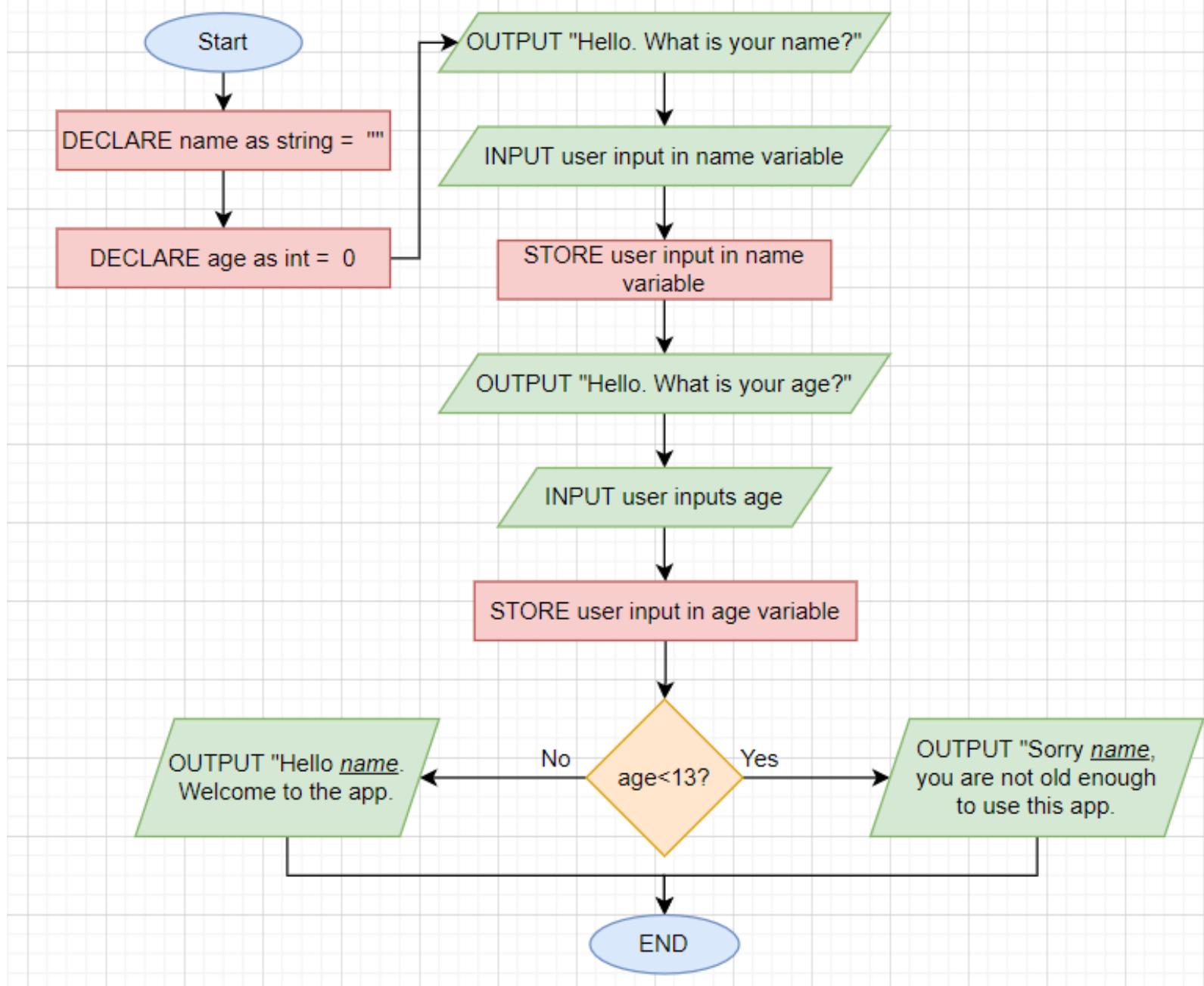
# Debugging using Scratch

To help pupils develop their debugging skills, we can use Scratch to set tasks and for peer-to-peer work/assessment:

1. Write and plan an algorithm (flowchart/pseudocode);
2. Write the program in Scratch (with ‘bugs’ in);
3. Get pupils to test the program, note what is not working and correct the mistake(s).

# Debugging using Scratch

- Look at the algorithm on the following slide.
- Read it and understand its purpose.



# Debugging using Scratch

## **Task:**

1. Follow this link to a Scratch program:  
[tc1.me/DebugScratch](http://tc1.me/DebugScratch)
  
2. Use the flowchart on the previous page, to test the program.
  
3. Note mistakes by right clicking on blocks, to add comments.
  
4. Correct the mistakes.

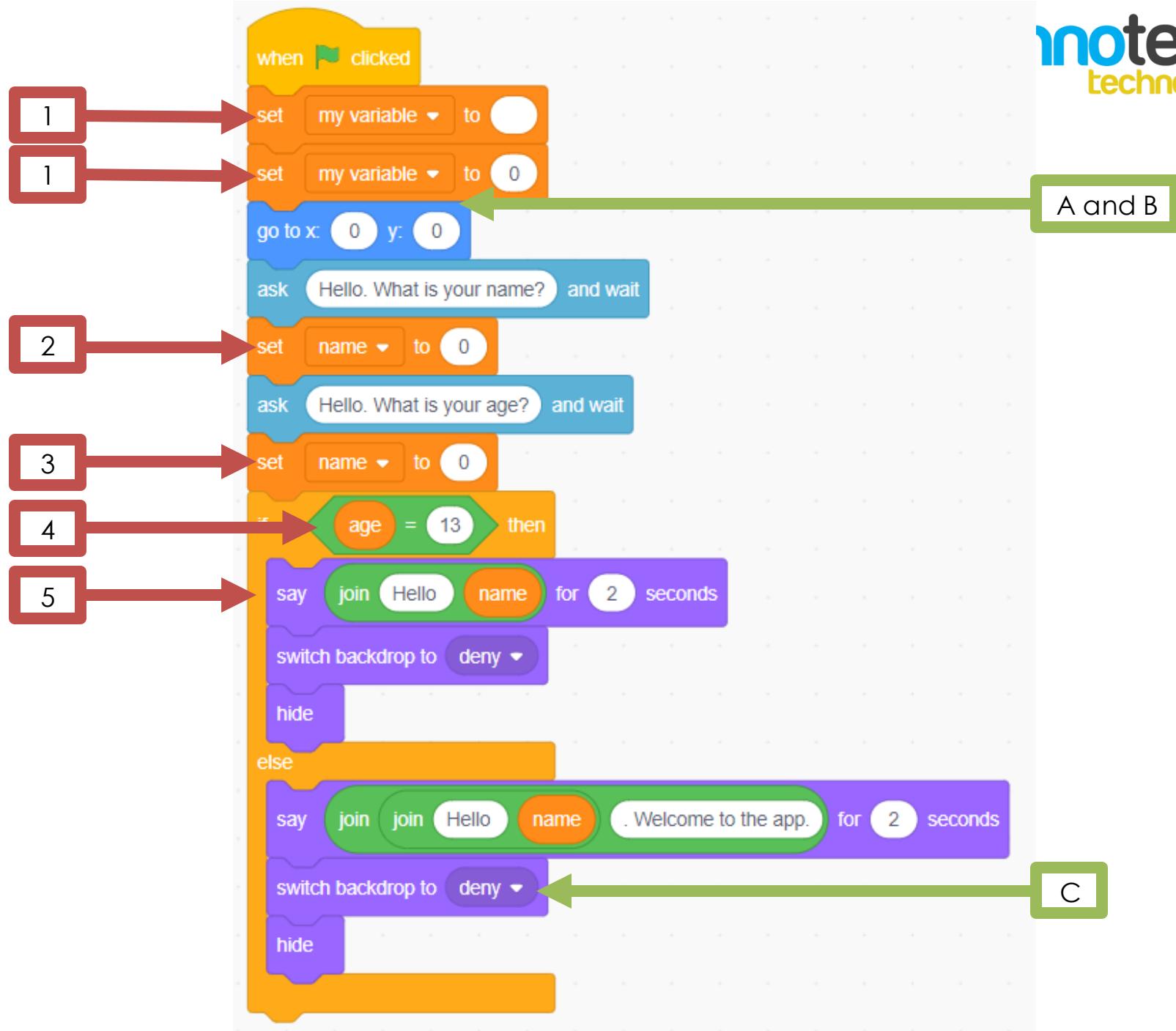
# Debugging using Scratch

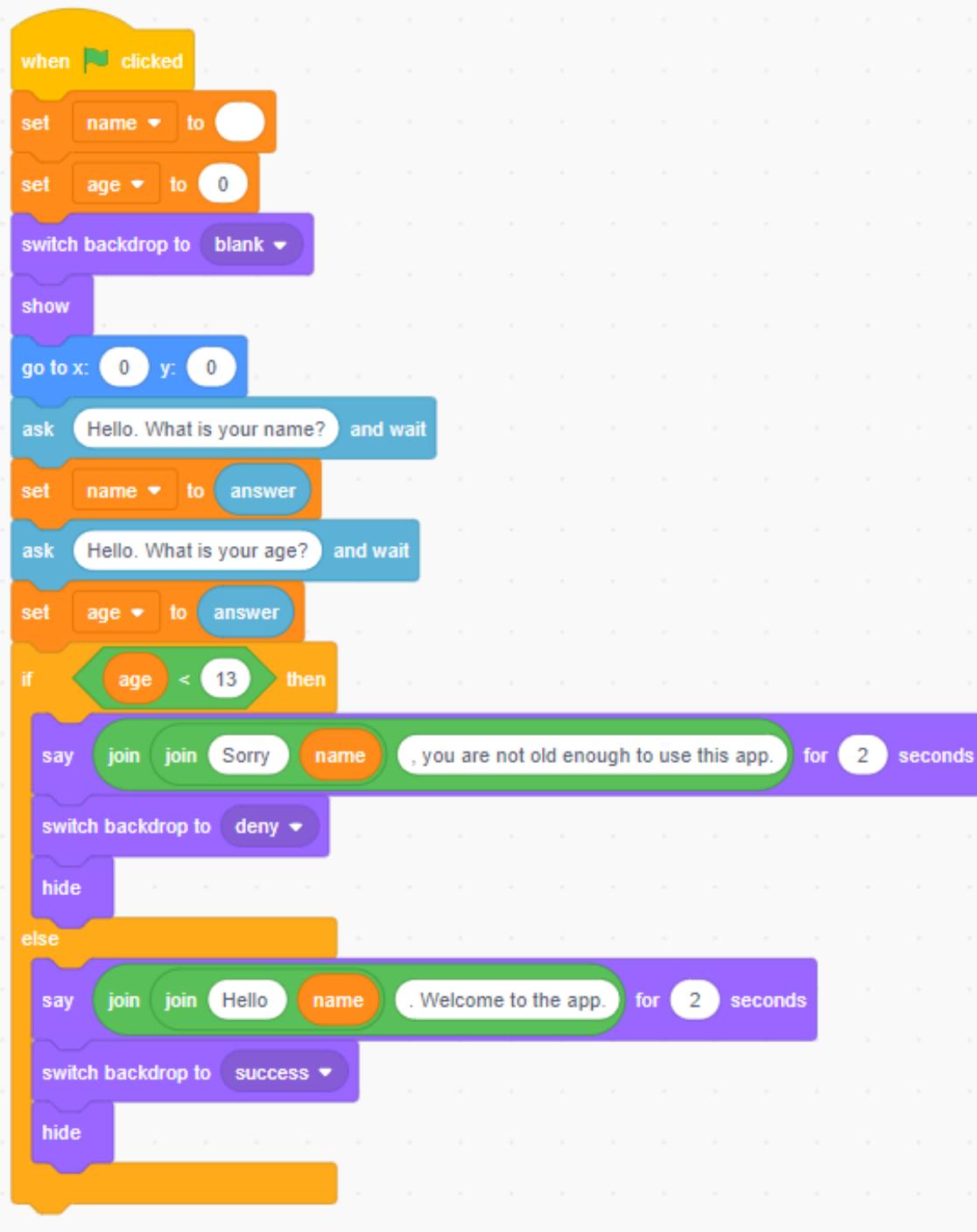
## Mistakes to debug:

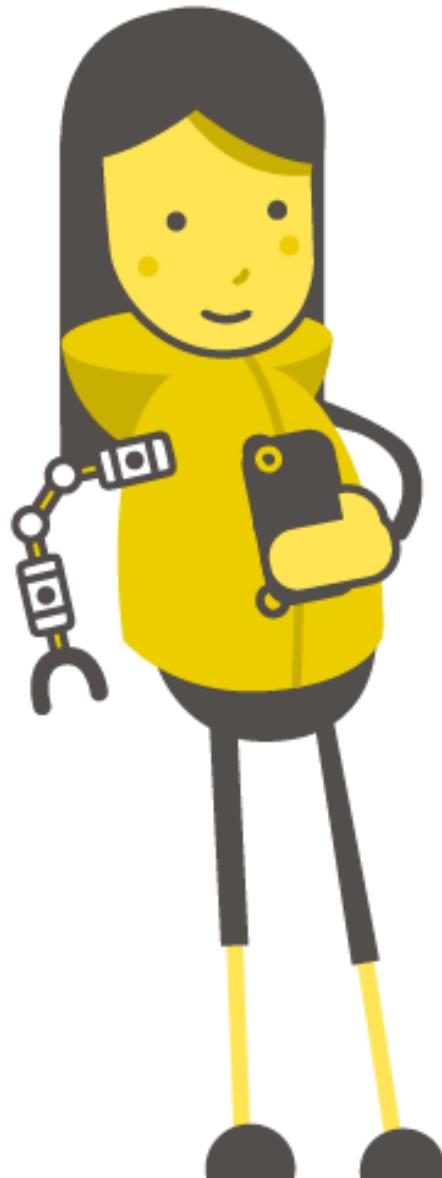
1. The **name** and **age** variables are not declared properly at the start of the program.
2. When you ask the user their name, save **their response** into the **name** variable.
3. When you ask the user their age, save **their response** into the **age** variable.
4. The condition to check should be '**age<13**'.
5. If the condition is TRUE, then a **specific** message should be shown.

## Extra mistakes:

- A. The cat should **show** at the start.
- B. The backdrop should be **blank** at the start.
- C. If the user is old enough, the backdrop should be '**success**'.







# Planning using Computational Thinking

# Using Computational Thinking to plan a Scratch program

Using your Computational Skills, and knowledge of Scratch, plan and make the following program:

- A simple 60 second timer.

# Planning: A 60 Second Timer

## **Decomposition:**

What do we need?

What does it need to do?

## **Pattern Recognition:**

Have we done something similar before?

Are there blocks in Scratch we can use?

## **Abstraction:**

Keep it simple.

What are the key things we need?

## **Algorithm:**

Write the algorithm.

# Planning: A 60 Second Timer

## **Decomposition:**

A counter to count from 60 to 0 in seconds; a number/counter with 1 second intervals.

## **Pattern Recognition:**

Counting – the angle in our spirals – variables?

Repeat blocks – timer repeats the same things 60 times.

When we count down, we have a number and then subtract 1.

## **Abstraction:**

Display number on screen; take away 1; repeat to 0.

## **Algorithm:**

Write the algorithm...

# Planning: A 60 Second Timer

## **Algorithm:**

Write the algorithm...

## **Example 1:**

- Start;
- When green flag clicked, set timer to 60 seconds;
- Start timer;
- Each second, reduce timer by 1;
- When timer = 0;
- Stop;

- To simplify this, we can use a loop. What kind of loop?

# Planning: A 60 Second Timer

## Algorithm:

Write the algorithm...

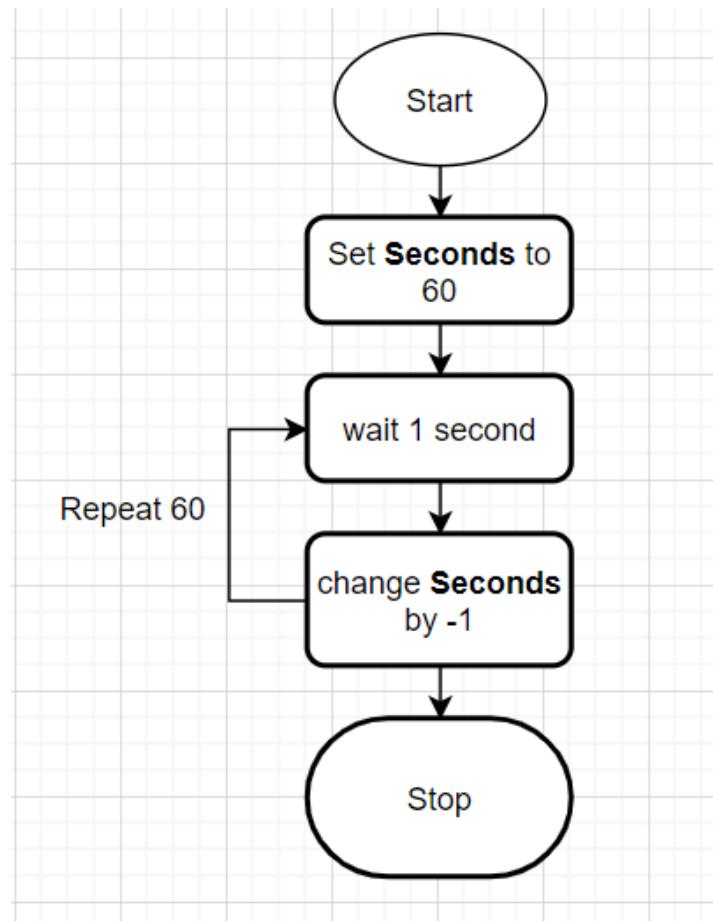
## Example 2:

- Start;
- When Green Flag clicked;
- Set timer to 60 seconds;
- **Repeat 60:**
  - wait 1 second;
  - change timer by -1;
- Stop.

# Planning: A 60 Second Timer

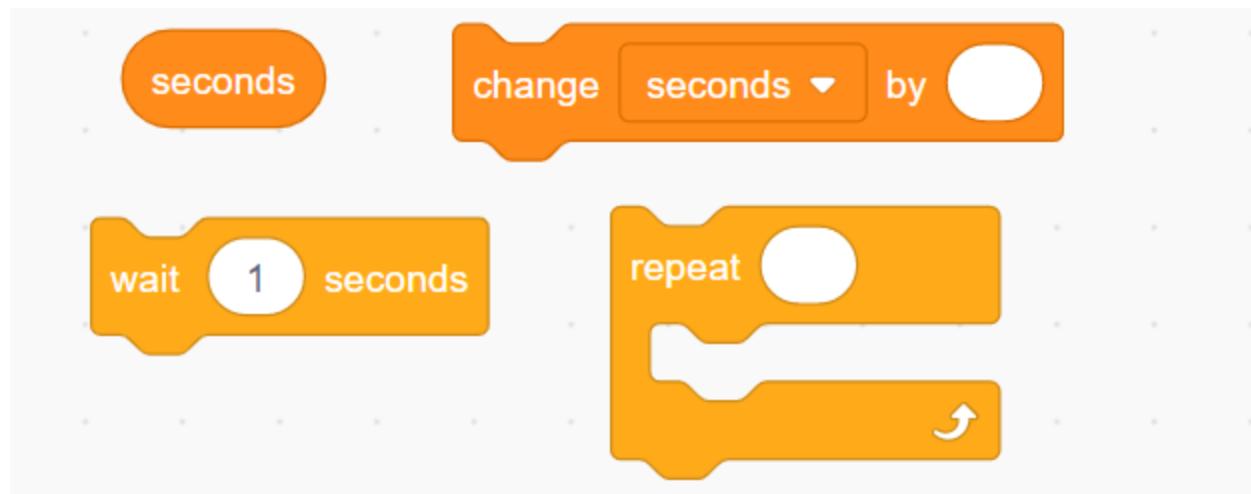
**Algorithm** – can you write a **Flow Chart** for this before programming in Scratch?

Now make your timer in Scratch.



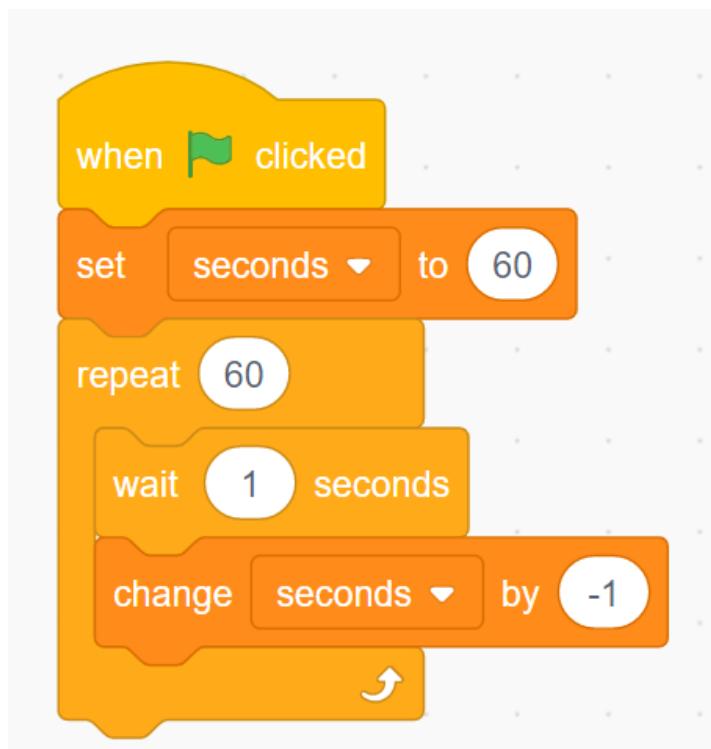
# Programming your 60 second timer in Scratch

## Hints:



# Programming your 60 second timer in Scratch

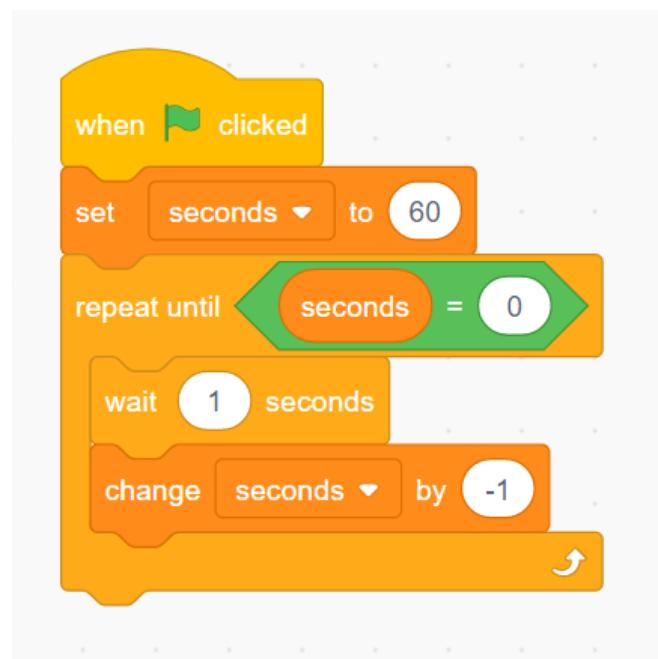
**Solution (not the only one!):**



# Alternative Timer?

Is there another way to make the timer?

Where we had used a count-controlled loop - here we have used a conditional loop instead.



# Alternative Timer?

What other improvements could be made?

- Ticking
- Alarm/bell
- Larger numbers

Could you set this as a task in class?

How many solutions can pupils program to the same problem?

# Timer Activity – in class

Could you set this as a task in class?

What do you think you would need to think about?

Extension:

Can they design more than one solution?

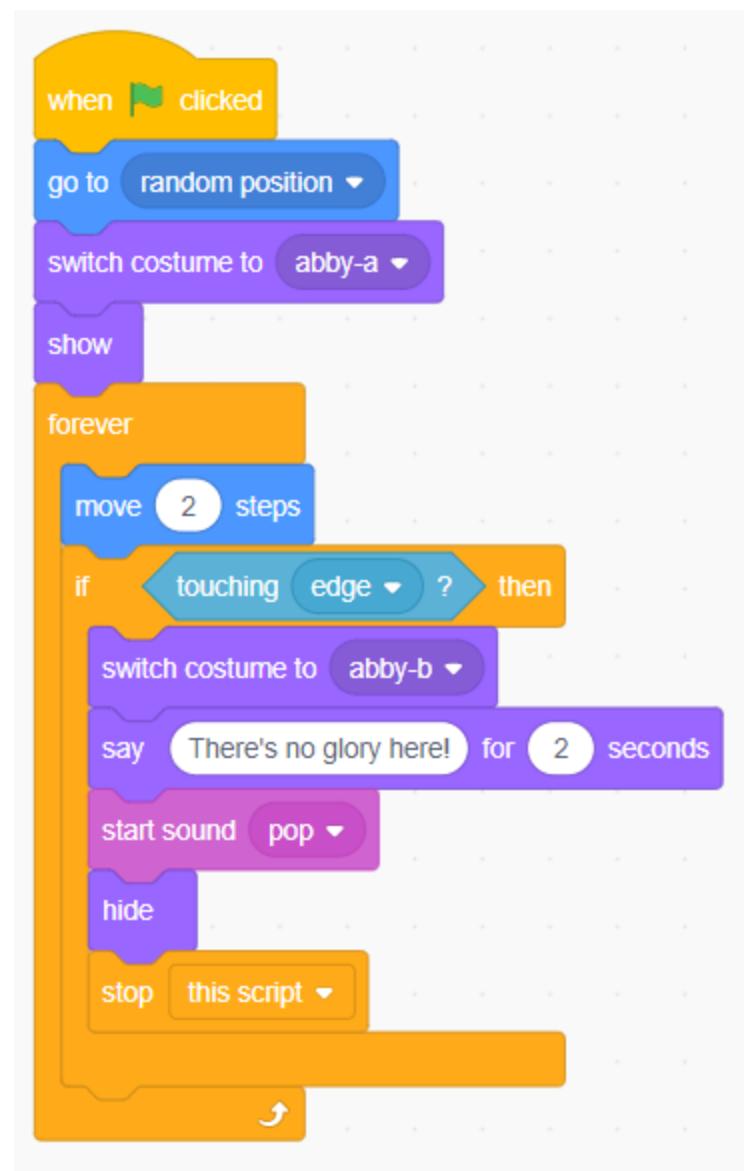
How many solutions can pupils program to the same problem?

# Programming from a Description



# Programming from a Description

- I need a program that contains the following. Please can you create it?
- **A sprite that starts each program in a different position, and then moves forward continuously.**
- **If the sprite hits an edge, then...**  
*make it look different,  
show a message,  
play a sound,  
and then disappear.*
- If you find it helpful, you could write an algorithm first.



# Machine Learning





# What is Machine Learning?

# Machine Learning

Machine Learning is a system with the ability to automatically learn and improve from experience without being explicitly programmed.

Machine Learning focuses on the development of computer programs that are provided with data and use it to learn by themselves.

# Why teach Machine Learning?

Element	Progression step 1	Progression step 2	Progression step 3
<b>Problem-solving and modelling</b>	I can identify, create and follow sequences and patterns in everyday activities.	I can break down a problem to predict its outcome.	I can create and refine algorithms and flowcharts to solve problems, making use of features such as loops, Boolean values and formulae.
	I can recognise and follow instructions in the appropriate order to perform a task.	I can detect and correct mistakes which cause instructions (a solution) to fail (debug).	I can understand the importance of the order of statements within algorithms.
	I can organise, select and use simple language to give instructions to others.	I can create and record verbal, written and symbolic instructions to test ideas, e.g. <i>the order of waking up through a diagram or flowchart</i> .	
	I can control devices giving instructions.	I can change instructions to achieve a different outcome.	
	I can identify errors in simple sets of instructions (algorithm).	I can identify repetitions or loops in a sequence, e.g. <i>identify where to shorten a set of instructions by repeating steps, for instance when learning a new song</i> .	

# Why teach Machine Learning?

<b>Data and information literacy</b>	I can collect data found in my environment.	I can collect, enter, organise and analyse data into different groups or formats, e.g. <i>tables, charts, databases and spreadsheets</i> .	I can construct, refine and interrogate data sets within tables, charts, spreadsheets and databases to test or support an investigation.
	I can sort and classify objects using one criterion.	I can extract and evaluate information from tables and graphs to answer questions.	I can use a range of spreadsheet formulae, e.g. $+ - / \times$ , <i>sum, average, max, min</i> .
	I can present and evaluate my data by creating simple charts, e.g. <i>pictogram</i> .		

# Machine Learning & the Science and Technology AoLE

## ❖ Computation is the foundation for our digital world.

### Progression step 1

I can identify, follow and begin to create sequences and patterns in everyday activities.

I am beginning to follow a sequence of instructions.

I can experiment with and identify uses of a range of computing technology in the world around me.

### Progression step 2

I can safely use a range of tools, materials and equipment to construct for a variety of reasons.

I can use computational thinking techniques, through unplugged or offline activities.

I can create simple algorithms and am beginning to explain errors.

I can follow algorithms to determine their purpose and predict outcomes.

I am beginning to explain the importance of accurate and reliable data to ensure a desired outcome.

I can follow instructions to build and control a physical device.

### Progression step 3

I can use conditional statements to add control and decision-making to algorithms.

I can identify repeating patterns and use loops to make my algorithms more concise.

I can explain and debug algorithms.

I can use sensors and actuators in systems that gather and process data about the systems' environment.

I can identify positive and negative design elements that affect user interactions.

# Machine Learning & the Science and Technology AoLE

▼ Computation is the foundation for our digital world.

## Progression step 1

## Progression step 2

## Progression step 3

I can explain how digital devices can be interconnected locally and globally.

I can explain the importance of securing the technology I use and protecting the integrity of my data.

I can explain how my data is used by services, which can help me make more informed decisions when using technology.

I can explain how data is stored and processed.

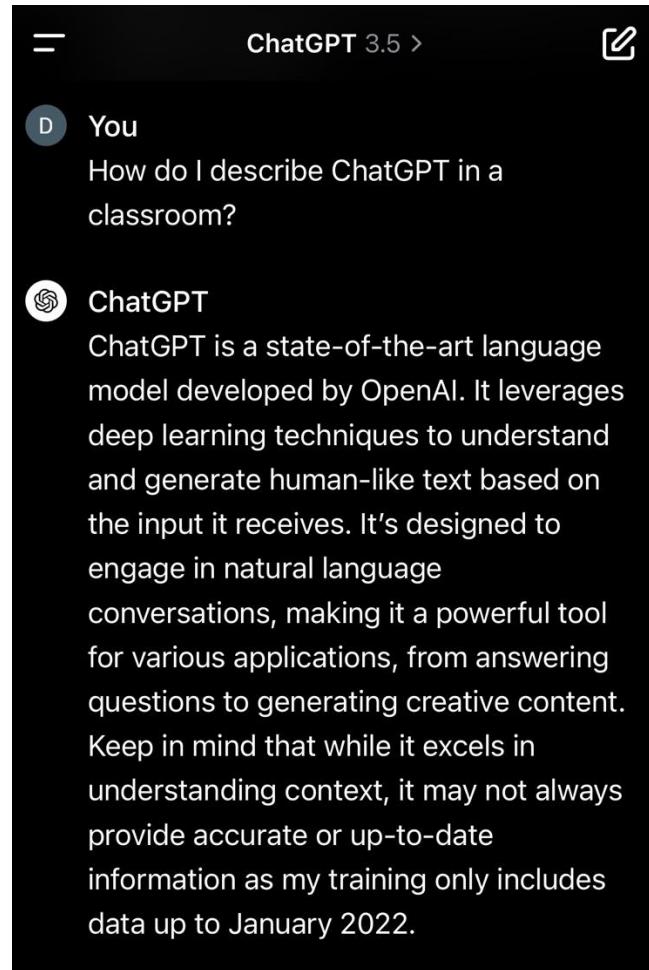
I can effectively store and manipulate data to produce and give a visual form to useful information.



**Do You Know  
Any Machine  
Learning  
Systems?**

# Generative AI

- ChatGPT is a tool that can generate convincing text responses, that can both appear accurate and human.
- ChatGPT is not always correct, it does not know anything, but rather uses Pattern Recognition to give you the answer it thinks you want!



ChatGPT 3.5 >

You  
How do I describe ChatGPT in a classroom?

ChatGPT  
ChatGPT is a state-of-the-art language model developed by OpenAI. It leverages deep learning techniques to understand and generate human-like text based on the input it receives. It's designed to engage in natural language conversations, making it a powerful tool for various applications, from answering questions to generating creative content. Keep in mind that while it excels in understanding context, it may not always provide accurate or up-to-date information as my training only includes data up to January 2022.

# Generative AI

A Van Gogh styled picture of a girl in traditional Welsh Costume with daffodils fighting a red dragon on a hill

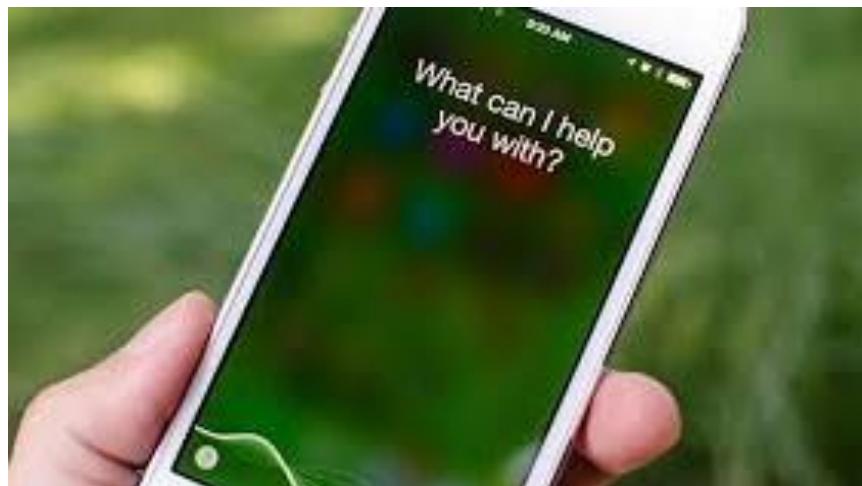
- DALL-E is a tool that can generate virtually any image you ask for – in any style you'd like.
- DALL-E is a very controversial tool, as who owns the pictures it creates?
- As it is also using Pattern Recognition has it stolen the original artists rights?



# Current Machine Learning Systems

## Virtual Personal Assistants

- Siri is a voice-activated assistant.
- Siri listens to your instructions, and, when possible, carries them out.
- Siri can open apps, play your favourite music and can even tell you a joke.



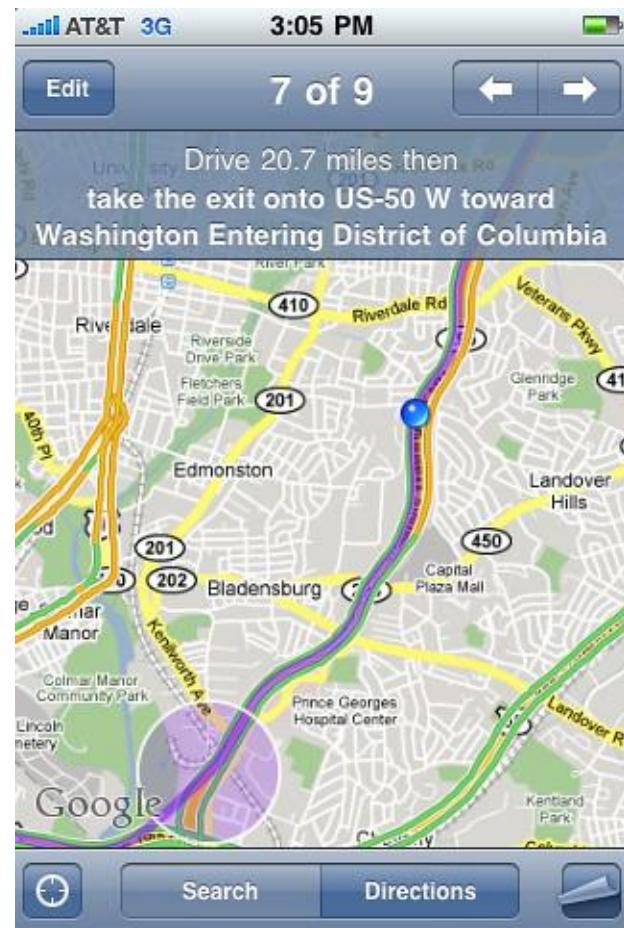
# Current Machine Learning Systems

## Google Maps



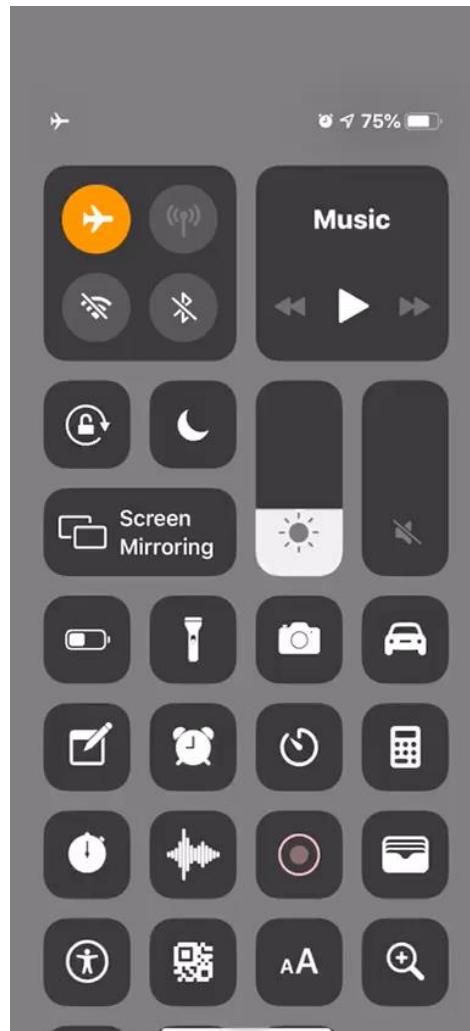
Google can use user's location data for things like:

- real-time traffic updates,
- estimating current traffic speeds, and
- adjust directions accordingly.



# Machine Learning Games

- Draw It - mobile app.
- Players compete with one another in an online real time drawing game.
- The goal is to draw one of the two categories given, so that the computer can make an accurate guess.
- The computer is taught through ML to recognise drawings.





# Turtle or Tortoise

# Turtle or Tortoise

We are going to do an experiment, there are two parts to this experiment. For the first part: I will be showing you a series of images one at a time. The images will either be a picture of a turtle or a tortoise.

I want you to tell me if you think it is a turtle or a tortoise.

Now you may already know the major differences between them but please do not share those details for now, just respond simply with either 'turtle' or 'tortoise'.

# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise



# Turtle or Tortoise





**How many did  
you get right?**

# Turtle or Tortoise

Turtle



Tortoise





# Turtle or Tortoise (Part 2)

# Turtle or Tortoise (Part 2)

Part 2: We are going to do the same experiment but this time I am going to provide a set of images that are 100% turtles, and another set that are 100% tortoises. The aim is to provide data that will help you make an informed guess.

I will again be showing you a series of images one at a time. The images will either be a picture of a turtle or a tortoise. I want you to tell me if you think it is a turtle or a tortoise.

Now you may already know the major differences between the two but please do not share those details for now, just respond simply with either 'turtle' or 'tortoise'.

# Turtle or Tortoise

Turtles



Tortoises



?

# Turtle or Tortoise (1)

Turtles



Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



?

# Turtle or Tortoise (2)

## Turtles



## Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



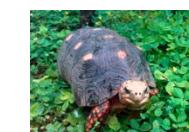
?

# Turtle or Tortoise (3)

## Turtles



## Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



?

# Turtle or Tortoise (4)

## Turtles



## Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



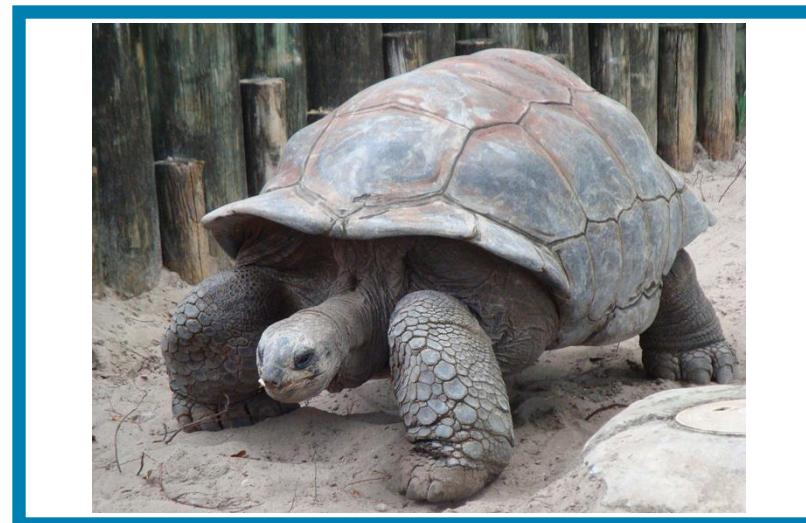
?

# Turtle or Tortoise (5)

## Turtles



## Tortoises

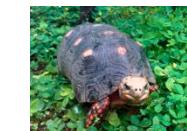


# Turtle or Tortoise

## Turtles



## Tortoises



?

# Turtle or Tortoise (6)

## Turtles



## Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



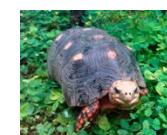
?

# Turtle or Tortoise (7)

## Turtles



## Tortoises



# Turtle or Tortoise

## Turtles



## Tortoises



?

## Turtles



## Tortoises





# Machine Learning in Scratch

# Make Me Happy

The image shows a Scratch project titled "make me happy". The stage features a yellow smiley face sprite with a neutral expression. A speech bubble above it says "type me a message!". The script on the stage is as follows:

```
when green flag clicked
  switch costume to [not sure v]
  ask [type me a message!] and wait
  if [recognise text answer (label) = kind_things v]
    switch costume to [happy v]
  else
    switch costume to [sad v]
```

The script uses the "when green flag clicked" event to start. It first switches the costume to "not sure". Then, it asks the user for a message. If the message contains the word "kind\_things", the costume changes to "happy". Otherwise, it changes to "sad".

The script palette on the left shows the following blocks:

- Motion: "make me happy"
- Looks: "recognise text text (label)", "recognise text text (confidence)"
- Events: "kind\_things", "mean\_things"
- Control: "add training data text kind\_things", "train new machine learning model", "Is the machine learning model ready?"
- Sensing: None
- Operators: None
- Variables: None
- My Blocks: "make me happy"
- Images: None

The stage properties on the right show the sprite is named "Sprite1" with a size of 100, direction of 90, and a backdrop number of 1.

# Make Me Happy

Go to [machinelearningforkids.co.uk](https://machinelearningforkids.co.uk)

Click on “Get started”.

Click on “Try it now”.

Click on “Projects” on the top menu bar.

Click the “+Add a new project” button.

Name your project “make me happy” and set it to learn how to recognise “text” and click the “Create” button.

# Make Me Happy

Training the machine

The screenshot shows a web-based machine learning interface. At the top, there is a navigation bar with links for 'About', 'Projects', 'Worksheets', 'News', 'Help', 'Log Out', and 'Language'. Below the navigation bar, the main title reads 'Recognising **text** as **kind\_things or mean\_things**'. There is a link '[< Back to project](#)' on the left. On the right, there is a button '+ Add new label'. Below the title, there are two large rectangular input fields. The left field is labeled 'kind\_things' and the right field is labeled 'mean\_things'. Each field has a small 'Add example' button at the bottom left corner.

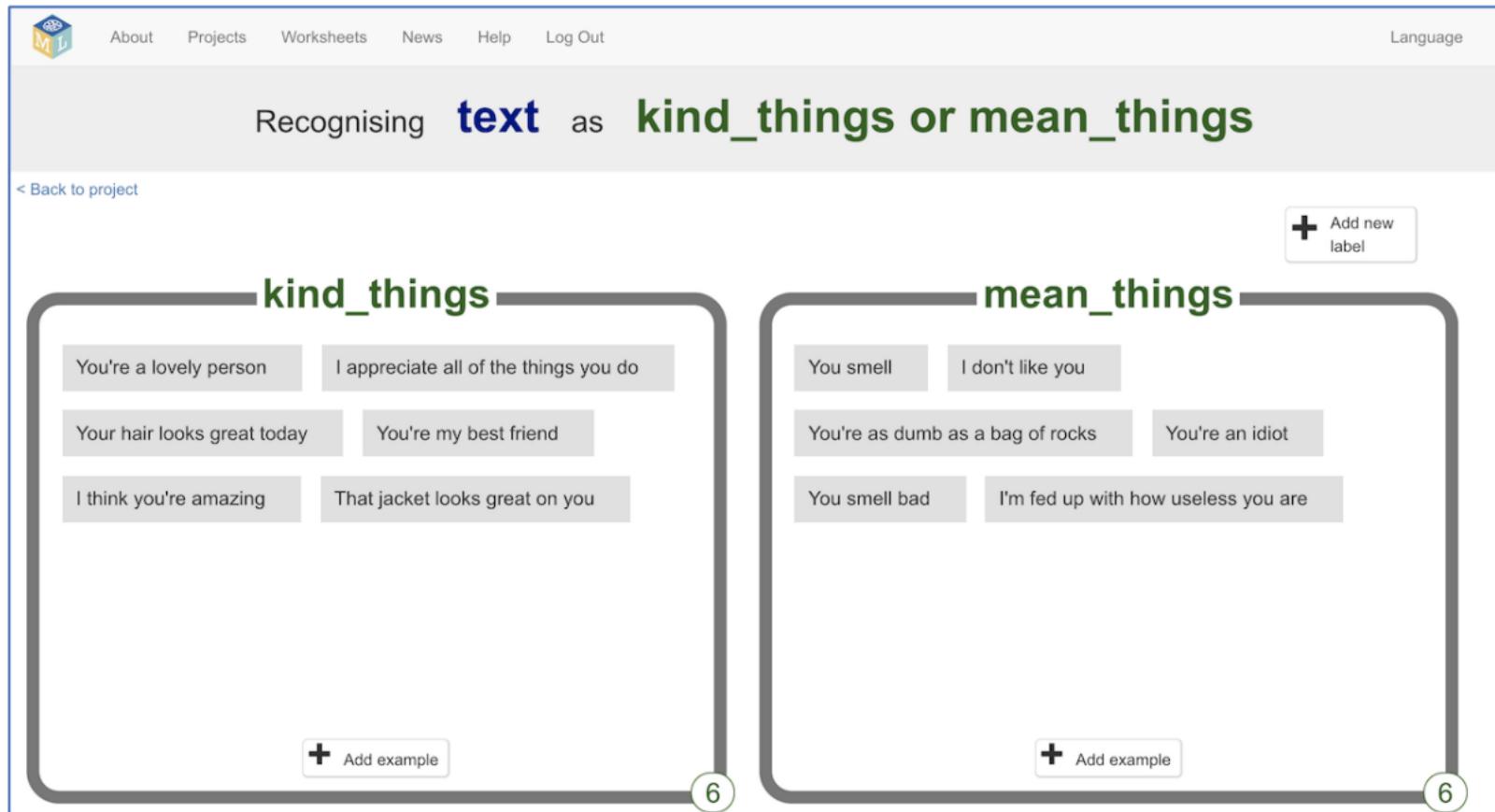
# Make Me Happy

Click the “Add example” button in the “kind things” label, and type in the nicest, kindest compliment you can think of.

Click the “Add example” button in the “mean things” label, and type in the meanest, cruelest insult you can think of (**be reasonable and appropriate**).

Write at least **six** compliments and at least **six** insults.

# Make Me Happy



About Projects Worksheets News Help Log Out Language

Recognising **text** as **kind\_things or mean\_things**

< Back to project

**kind\_things**

You're a lovely person I appreciate all of the things you do

Your hair looks great today You're my best friend

I think you're amazing That jacket looks great on you

**mean\_things**

You smell I don't like you

You're as dumb as a bag of rocks You're an idiot

You smell bad I'm fed up with how useless you are

**Add new label**

**Add example** 6

**Add example** 6

# Make Me Happy

Click on the “< Back to project” link.

Then click on the “Learn & Test” button.

Click on the “Train new machine learning model” button. As long as you have entered an **even** amount of examples, the computer should start to learn how to recognise messages from the examples you have given it.

# What have you done so far?

You have started to train a computer to recognise text as being kind or mean.

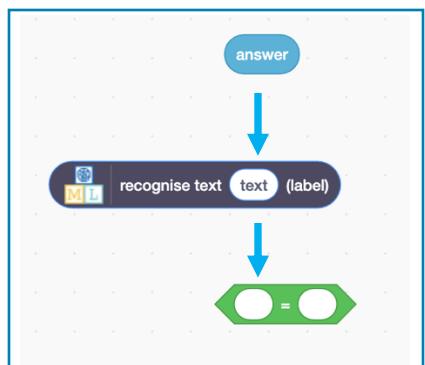
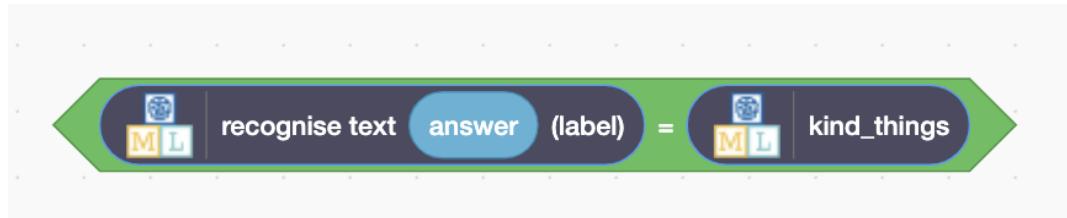
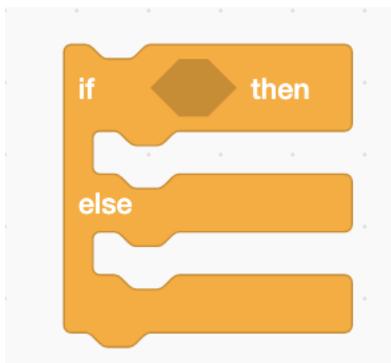
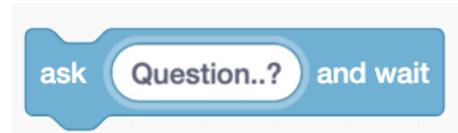
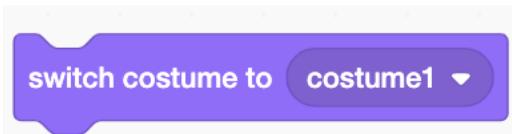
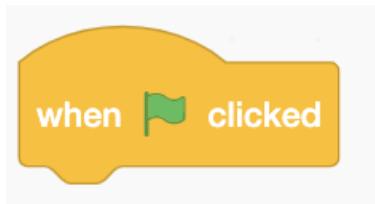
This is called “supervised learning” because of the way you are supervising the computer’s training. The computer will learn from patterns in the examples you have given it.

# Make Me Happy

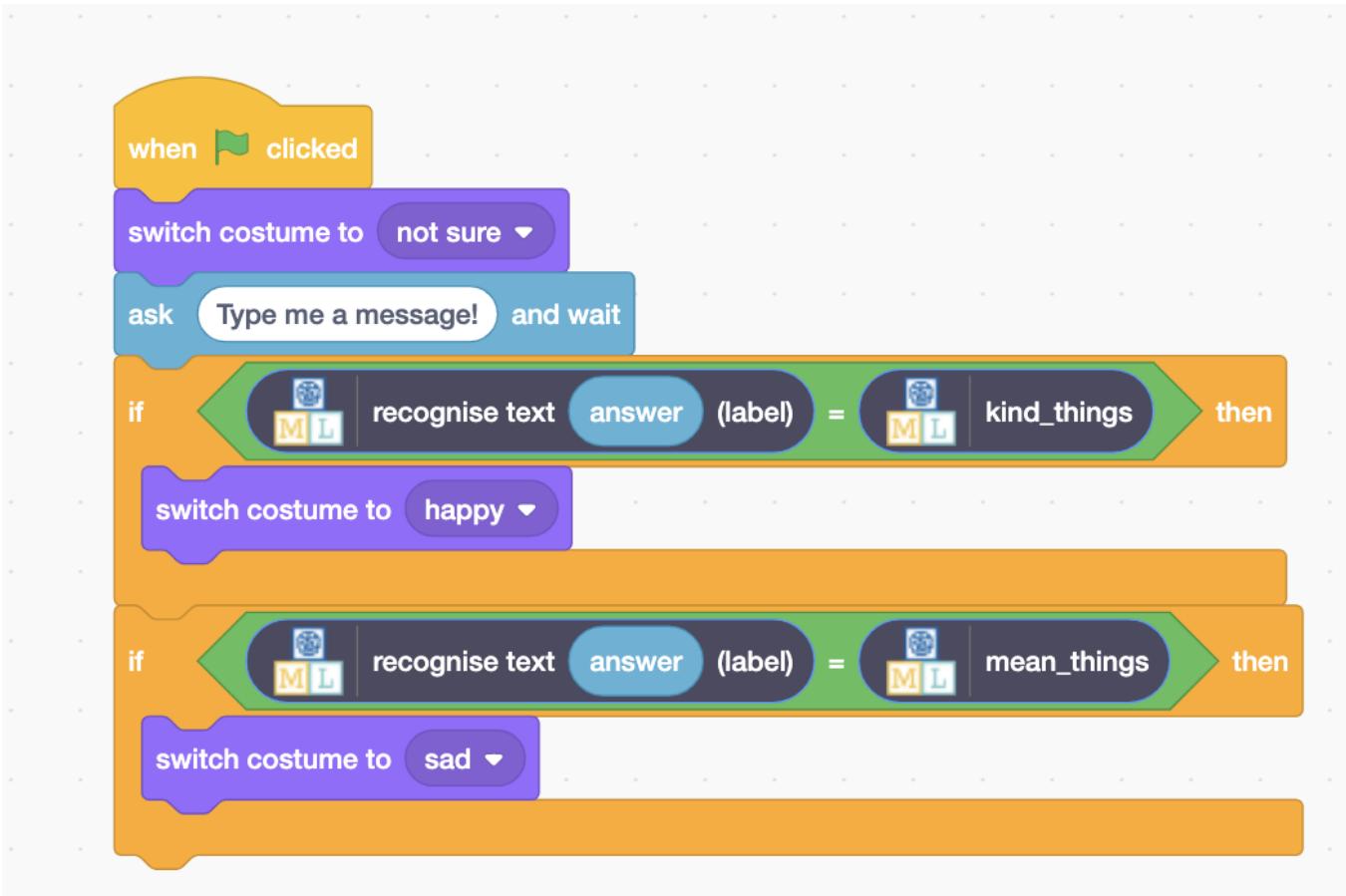
Program your sprite so that:

- 1) When we start the game, display not sure face.
- 2) Ask the question “Type a message: ”
- 3) If the answer is recognised as “kind” then, display happy face.
- 4) If the answer is recognised as “mean” then, display sad face.

# Make Me Happy



# Make Me Happy



# Tips

## **More examples!**

The more examples you give it, the better the computer should get at recognising whether a message is kind or mean.

## **Try and be even**

Try and come up with roughly the same number of examples for kind and mean.

If you have a lot of examples for one type, and not the other, the computer might learn that type is more likely, so you'll affect the way that it learns to recognise messages.

## **Mix things up with your examples**

Try to come up with lots of different types of examples.

For example, make sure that you include some long examples and some very short ones.

# What have you done?

You have modified your Scratch character to use machine learning instead of the earlier approach.

The more examples you give it, the better it should get at recognising messages correctly.

# Activity: Improvement

Now that you've finished, keep testing your character to see if it has the wrong reaction to anything you say!

If it has the wrong reaction – why?

Let's look back over our examples to see; are there any patterns we didn't intend on creating? Have we included similar words in another category?

Retrain your model with the new words and see if it has improved!

# Activity: Ideas and Extensions

Now that you've finished, why not give one of these ideas a try? Or come up with one of your own?

## **Write a reply (easy)**

Instead of just changing the way they look, make your character reply, based on what it recognises in the message!

## **Try a different character (medium)**

Instead of a person's face, why not try something different, like an animal?

It could react in different ways, instead of smiling.

For example, you could make a dog that wags their tail if you say something kind to it!

## **Different emotions (hard)**

Instead of kind and mean, could you train the character to recognise other types of message?

# Summary

We created our first machine learning scratch game by:

- Creating textual data to use to train the machine: kind things and mean things.
- We trained the machine with this data.
- And lastly using the trained data, we program a scratch game that recognises whether we have entered a kind or mean message.