INTRODUCING

DIGITAL NUMERACY with SCRATCH

by

Seamus O'Neill

Introduction

Dear Teacher,

Thank you for your expression of interest in the topic of Numeracy from Scratch (or *Digital Numeracy with Scratch* as I prefer to call it) which I recently posted on the teacher's mailing lists, CESI and DICTAT. Perhaps I should start by briefly explaining why I began to develop *Digital Numeracy* strategies in Scratch and say a few words about my interests in Maths and computer programming.

I first tried Scratch programming in 2011 and I have been tutoring teachers since then in Scratch for Literacy and Numeracy in Navan Education Centre. Two years ago I set up our local CoderDojo where I teach Scratch to children. I have observed that there is quite a difference when it comes to the objective in teaching Scratch in the two situations. What do CoderDojo children and their mentors want? What do primary teachers need?

I began to develop two different approaches. The first, I call *Scratch From Scratch* and it's about teaching game programming to highly motivated Code Club kids. The approach that I take with teachers, *Digital Numeracy with Scratch,* has the purpose of improving children's Numeracy skills and encouraging their logical thinking skills using Scratch in a school situation. *Scratch From Scratch* would also be ideal for many children in senior primary classes at school, especially after a year or two of *Digital Numeracy with Scratch*.

Over the following pages I will gradually introduce some *Digital Numeracy with Scratch* strategies through easy-to-follow workshop activities. Please note, these are intended for teachers and other adults who work with children. These are the notes I use when I give staff training in schools and they are not meant to be used by the children. I have drafts of over 40 children's worksheets in *Digital Numeracy with Scratch* up through the primary levels (ages 7 to 12 years)and I share them as part of the workshop activities.

I rely on feedback from practicing teachers, tutors, mentors and other practitioners to assist me in developing the concept and the children's worksheets. With the benefit of practical classroom experience, teachers may develop other strategies, create new worksheets or have ideas in mind that they would like to see included. I welcome any suggestions teachers might have, should they want to become involved in creating further materials or resources.

Digital Numeracy with Scratch is a support methodology for teaching mathematics in the context of the current primary school curriculum. Yet, it should be evident from many of the projects, that you are not just supporting a Numeracy objective. You are also introducing the children to many basic concepts of computer programming (or coding). It's my belief that teachers who try some of these *Digital Numeracy* strategies for themselves will realise that they don't need to have a degree in computer science to be a better Maths teacher or to teach Maths today using Scratch code. SO'N

My name is Seamus O'Neill
seamusoneill@eircom.net
I am a former primary teacher, tutor of teachers in Navan Education Centre and co-author of *Mathemagic.* Since 2009, I programmed and developed almost 200 IWB primary school resources. All are free online. I am author of a new book *Scratch From Scratch* for Schools and Clubs. I shared authorship of a previous edition of *Scratch From Scratch* with Stephen Howell, former Computing Lecturer in Interactive Media and Software Development in the Institute of Technology, Tallaght. Stephen now works for Microsoft Corp. The earlier title was based on Scratch 1.4. The current version is Scratch 2.0 which is cloud based and most graphics in its sprite library are vectors – these are two significant developments from the Scratch MIT team.

Acknowledgement and Thanks:
Scratch Copyright (c) 2009 Massachusetts Institute of Technology
Scratch is developed by the Lifelong Kindergarten group at the MIT Media Lab.
See www.scratch.mit.edu

See also
www.scratchfromscratch.com
www.scratch.mit.edu/users/scratchfromscratch2
www.scratch.mit.edu/users/scratchfromscratch
www.weandus.ie/ maths.html

The Scratch Interface

To get started go to **http://scratch.mit.edu** and click **Create**. There's no need to download.

I am assuming that the reader of these materials has some familiarity with Scratch. If necessary, please visit **www.scratchfromscratch.com.** There you will find information about the various symbols and interactive features of the Scratch Project Editor.

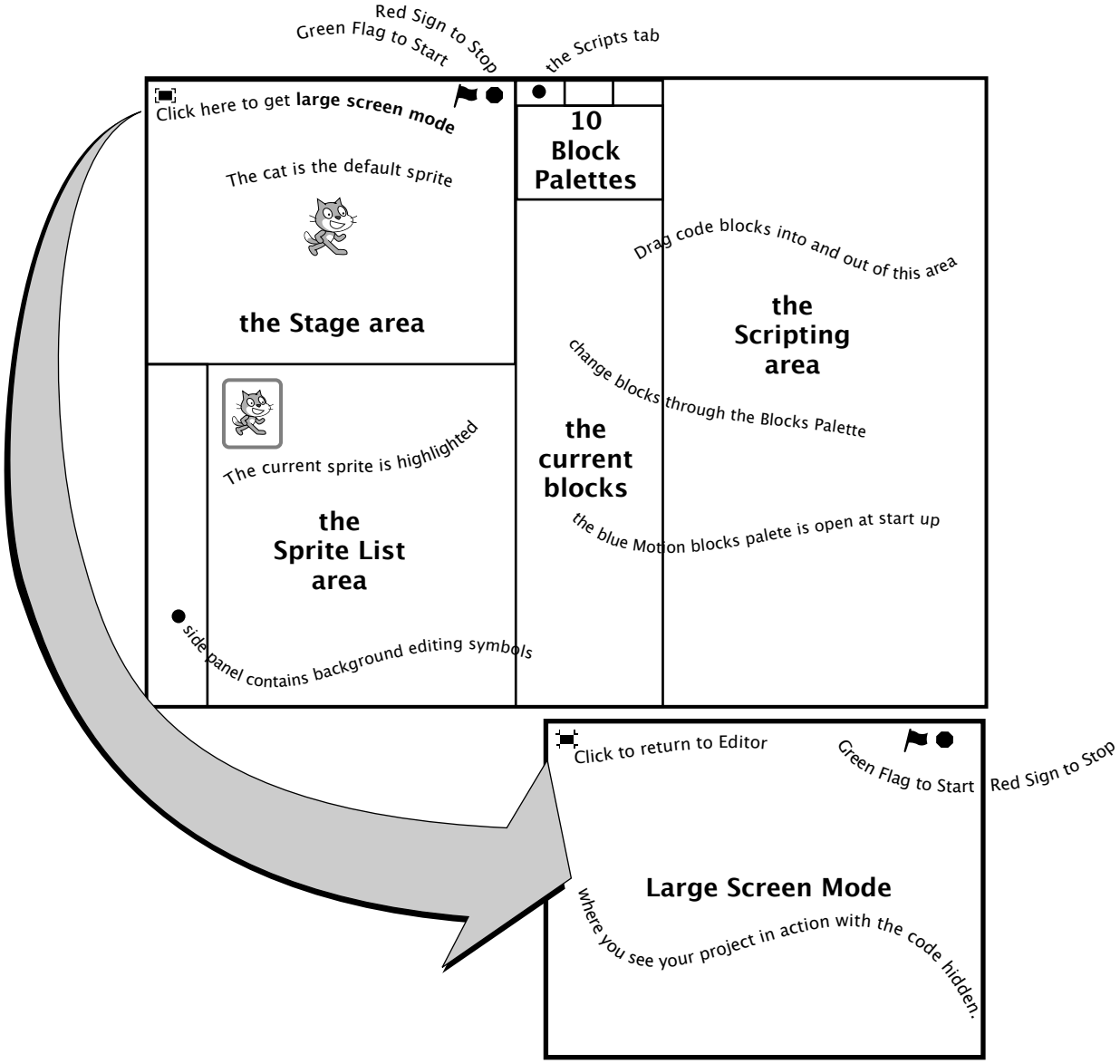
Take time to look around the PROJECT EDITOR. There are tips, videos and a 13-step tutorial, 'Get Started with Scratch' under the Help menu.

Scratch 2.0 is a totally free, web-based, easy-to-learn programming language for young and old. Scratch 2.0 works on laptops running Flash Player but it does not work on the iPad.

To use Scratch offline you need to download the offline version of the project editor at **http://scratch.mit.edu/scratch2download**.

You can see the main areas of the Scratch Project Editor below. Some main features are mentioned over the following pages.

The main areas of the Scratch Project Editor



There are online examples of *Digital Numeracy* projects at www.scratch.mit.edu/users/scratchfromscratch2 Some examples refer to children's worksheets which can be downloaded from www.scratchfromscratch.com/extra-resources

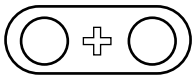
The Strategies

Strategy 1: Operators, Command blocks & Events

The following pages are intended as a simple introduction to Digital Numeracy with Scratch for teachers. They are not designed as worksheets for children. Children's worksheet can be found in a separate section.

Find each of the following three blocks in Scratch. Write underneath each which **palette** it's in and what colour it is.

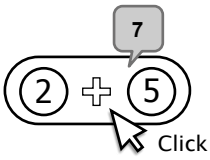
Experiment with plus operators first.

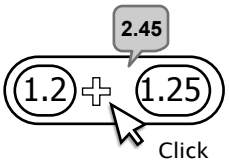
1 

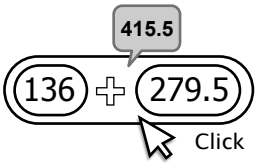
This is the **plus operator** block. It can be found in the **Operators** palette.
It has two **input windows** which only accept numeric input, not text.

A Drag a **plus operator** into the scripting area. Type numbers into the windows.

EXAMPLES:



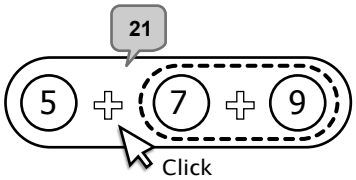




Click the block with the mouse-pointer and it returns a number just like a calculator.
Blocks with rounded ends are also called **reporters**.

B Combine two plus operators into one **combo** block.

EXAMPLE:



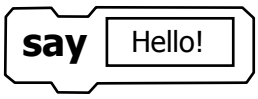
What if I want to add 4 numbers?
Try a combination of 3 plus operators.

The plus operator is one of four **arithmetic operators** in the **Operators** palette. Try some one-step calculations in subtraction, multiplication and division. This exercise shows how to make very basic use of an arithmetic operator. It shows how to combine several operator blocks and how to make the computer return a value by clicking the code block. Beyond that, it's not very exciting! It's not much different from a calculator.

Also, the cat sprite just sits idly in the centre of the **stage** with the cat thumbnail highlighted in the **sprite list** area. In Scratch, when you drag a code block into the scripting area, you are adding code to the current sprite. Scratch is all about coding sprites. Activities **1A** and **1B** are not making the cat do anything, even though you are programming it. If you go into large screen mode, all you see is the cat. You can neither see nor engage with the code.

Let's command the cat to* say *something. Next ...

The say block is a Command block in the Looks palette (purple)

2 

This is the **say** block and it is found in the **Looks** palette, coloured purple.
The word 'Hello!' is default text. Delete or change it to anything you like.

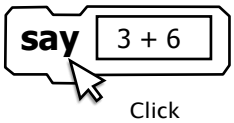
Most of the blocks in the Looks palette have a 'notch' at the top and a 'bump' at the bottom. Blocks of this shape are known as **stack** blocks. Stack blocks lock into each other and into other blocks with 'bumps' or 'notches'. You can snap these blocks together into stacks.

Notice that there are no stack blocks in the Operators palette. All operators have either rounded or pointed ends! They can't form stacks. Instead you drag operators into the input windows of other blocks. *You can put a round or hexagonal block into a rectangular window!* Operator (or reporter) blocks are used as inputs to other blocks.

The **say** block is also a **command** block. When clicked, or *triggered*, the computer makes the sprite *say* something (with a speech bubble). We will pick up from activity 1 by making the cat say something using expressions in maths.

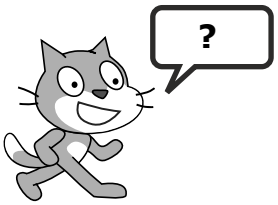
A Drag a **say** block into the scripting area.

Replace 'Hello!' by typing an arithmetic expression into the rectangular window.



Click

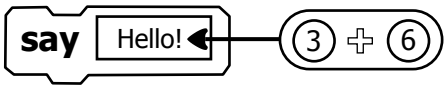
Does the sprite add the numbers?

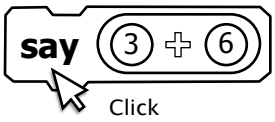


the STAGE

B Drag a **plus operator** into the **say** block.

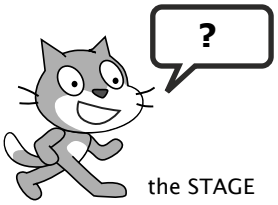
There's no need to replace or delete the text.





Click

Does the sprite add the numbers?



the STAGE

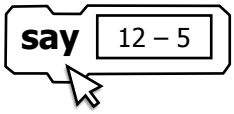
- Why do you think the cat responded in a different way in A and B?

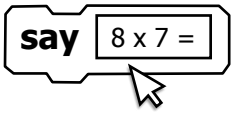
Look at the other arithmetic operators in the palette (bright green)

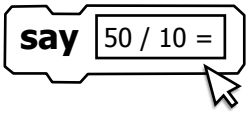
C Try some one-step operations in subtraction, multiplication and division.
(i) Make the cat say the expression. (ii) Make the cat report the answer.

EXAMPLES:

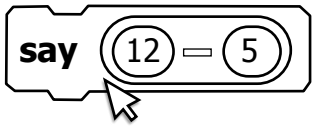
(i)

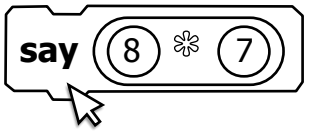


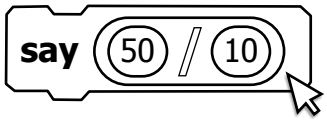




(ii)





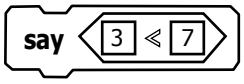


EXTRA:

(a) Use the **round operator** in division e.g.

(b) Find out what the sprite reports when you use a **comparison** operator. e.g.

Comparison operators have pointed ends. They report **Boolean** (true/false) values.



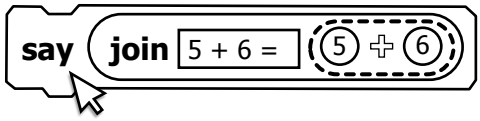
Use the join operator to make reading sense. Next ...

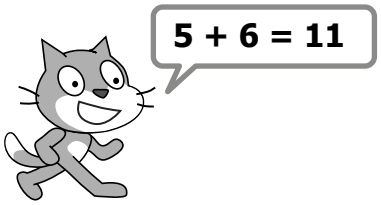
Joining text and code together is often called concatentating

3 

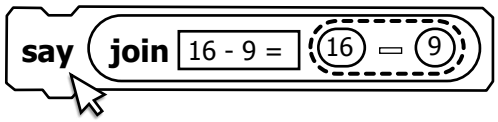
The **join** block is in the **Operators** palette. It has two input windows. Type text or drag reporter blocks into the input windows.

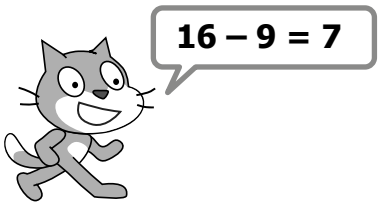
A Construct each combination and click to make the sprite say the full equation.

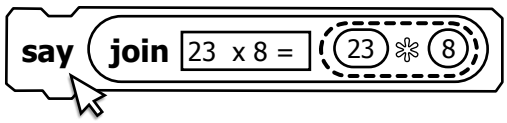
(a) 

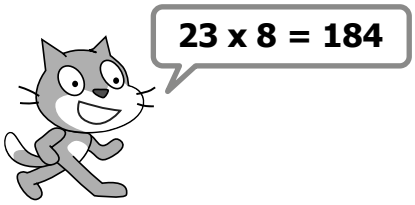


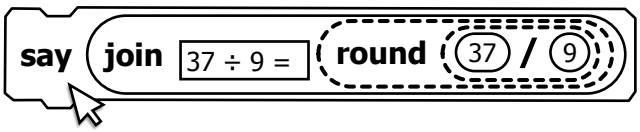
The first window in the **join** operator takes the text.
The second window receives the arithmetic operator.
Type a space after the equal sign.

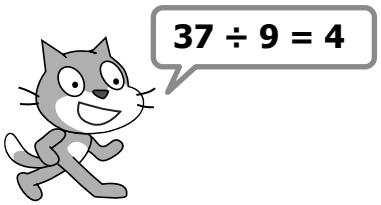
(b) 



(c) 



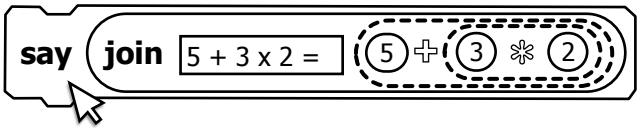
(d) 

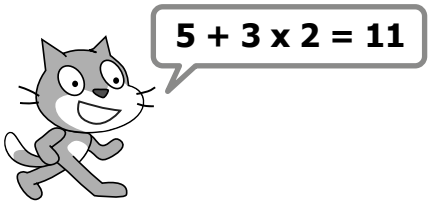


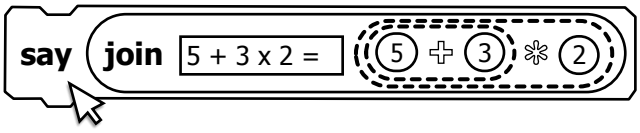
Use ÷ or / for division. (Tip: Hold down Alt key and type 246 to get the usual division sign ÷)

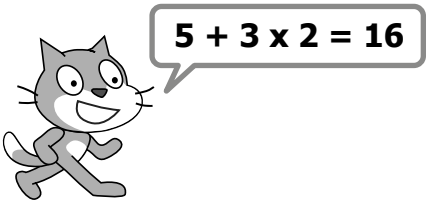
Two ways to combine operator blocks

B ● Only one of the following is correct. Which one (a) or (b)?
What is causing the error?

(a) 



(b) 



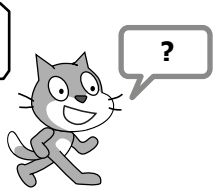
More blocks for Strategy 1: Next …

Strategy 1 contd: **Event Blocks**

Code the cat sprite with two scripts.

(i) when clicked
say 3 + 7 = ?

(ii) when this sprite clicked
say 3 + 7

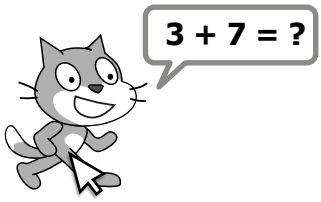


Find these two blocks with rounded tops.
In which palette are they?
What colour are they?

The green flag to click is above the stage.
It does not refer to the flag pictured on the block!

What do you think will happen
(i) when the green flag is clicked?
(ii) when the sprite is clicked?

4 The blocks with rounded tops look like hats and they are often referred to as *hat blocks*. Almost all the blocks in the **Events** palette are **hat blocks**. They are coloured brown. They go on top of a script to trigger the events below them. Scripts under the **sprite clicked** trigger block will run only when the sprite is clicked. All scripts that have the **green flag** trigger block will start simultaneously when the green flag above the stage is clicked. This is programming! You can run your programme from inside the editor or run it in large screen mode (the **Scratch Player**).

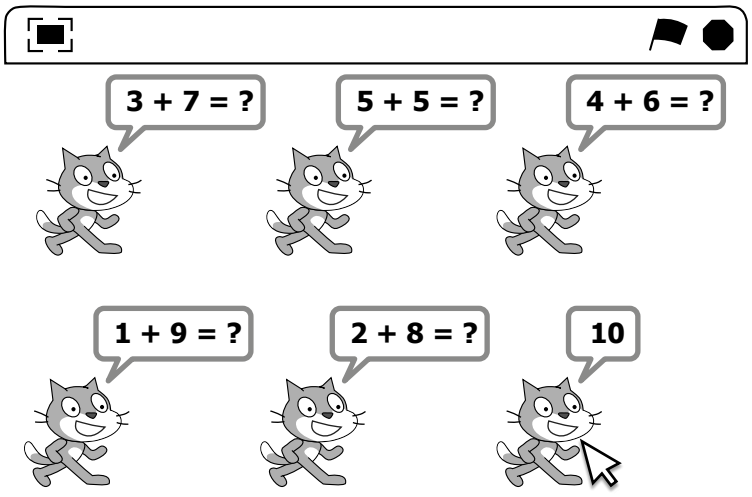


When the **Green flag** above the stage is clicked it triggers the event that makes the cat put the question.

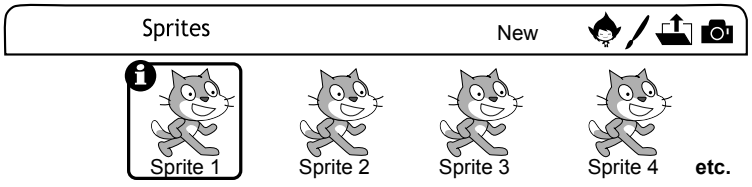
10

When you click the sprite it makes it say the answer.

5 **Duplicate the cat five times** and notice that the cat's two scripts are also replicated on each copy of the cat. Select each cat sprite in the **sprite list** one by one and edit the input values in the **say** blocks, so that they show the story of 10. Now you have several sprites on the stage together. When the green flag is clicked each will have a speech balloons and a different maths puzzle. Click the sprite itself to make it say the answer.

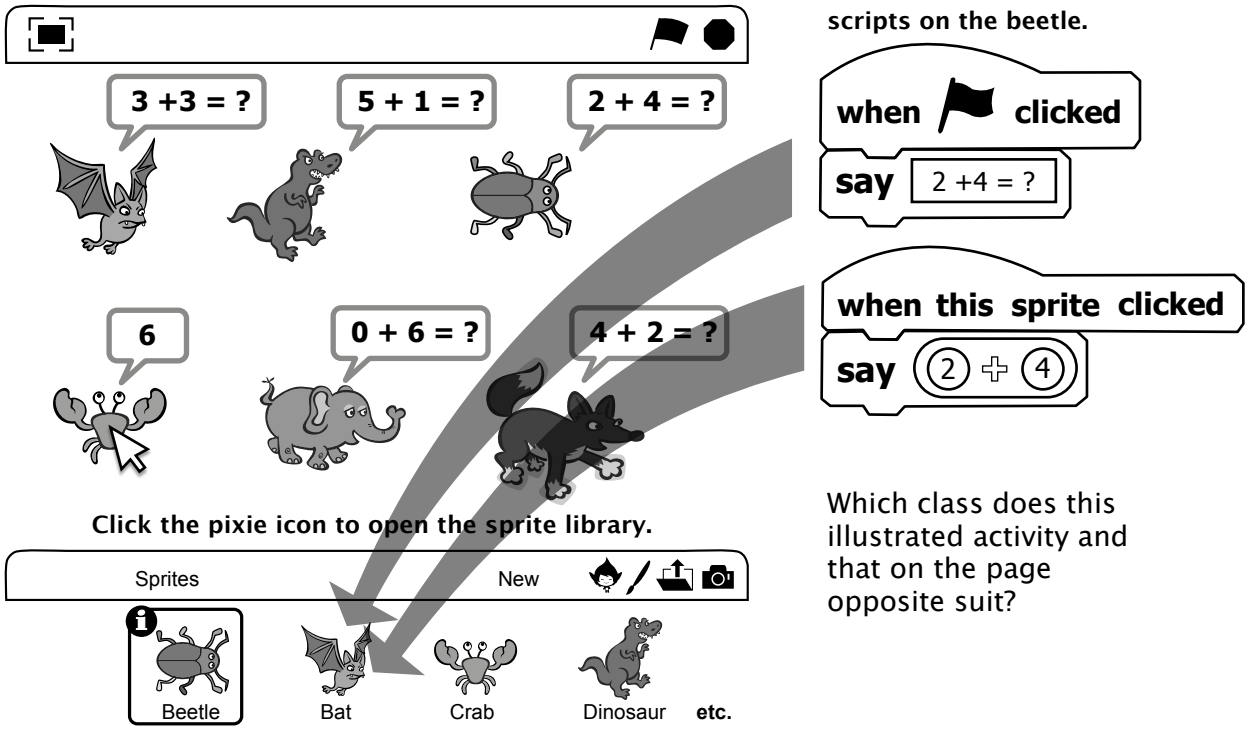


There are several ways to duplicate the cat!



Experiment and Find out more. Next ...

6 To add variety and to facilitate classroom discussion get different sprites from the **sprite library**. You need only script one sprite e.g. the beetle. Drag the two scripts from the scripting area on top of the thumbnail of each sprite in the **sprite list**. The script will copy and 'sling' back to the beetle when you release the pointer. Edit the values in each sprite's **say** and **operator** block.



scripts on the beetle.

Which class does this illustrated activity and that on the page opposite suit?

Strategy 1 (single and multi-step operations) at each class level

7 Now you're programming *Digital Numeracy with Scratch*. Try similar programming at higher level addition, subtraction, multiplication and division. Children can work out a set of maths problems as usual (with pencil and paper) then programme the set in Scratch (as in activites 5 and 6)

Examples for each class level:

(a) Add two 2-digit numbers

(b) Subtract a 2-digit numbers with regrouping

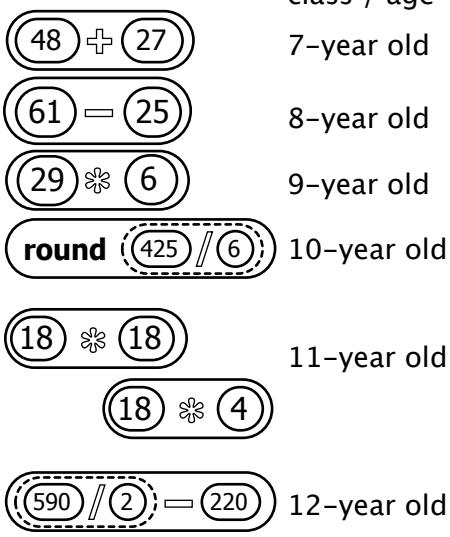
(c) Multiply a 2-digit number by a single digit number

(d) Divide a 3-digit number by a single digit and **round** the answer

(e) code the area of a square and perimeter of a square whose length is a 2-digit number of units.

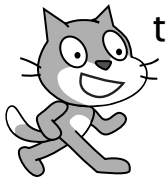
(f) programme the length of the shorter side of a rectangle given a 3-digit perimeter and the longer side.

Examples below in the operator. Create more similar problems then code them to RUN when the green flag is clicked and show correct answer when sprite is clicked.

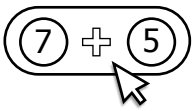


- **See online project examples at www.scratch.mit.edu/users/scratchfromscratch2**
  See children's worksheets at www.scratchfromscratch.com/extra-resources

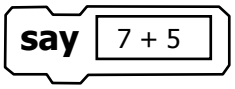
Worksheet based on Strategy 1

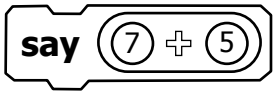


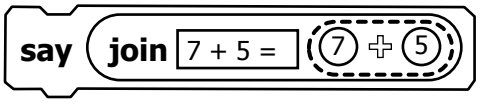
the sprite **code one script at a time and click**

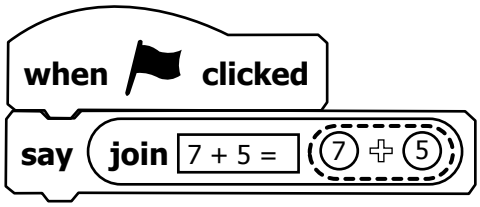
A 

Click the code

B 

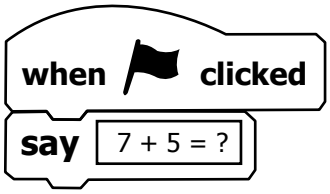
C 

D 

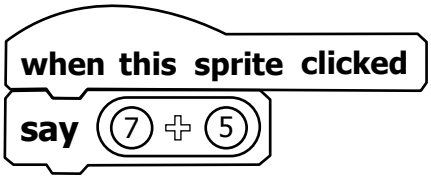
E 

Run in large screen mode
(also called Scratch Player)

F



2 scripts on the sprite



Run in large screen mode
(also called Scratch Player)

G Create several copies of the sprite and edit the values to

4 + 5 5 + 5 6 + 5 8 + 5 9 + 5

Look at some online examples at:

http://scratch.mit.edu/users/scratchfromscratch2

Extend the Activity Use as many as possible of the same combinations

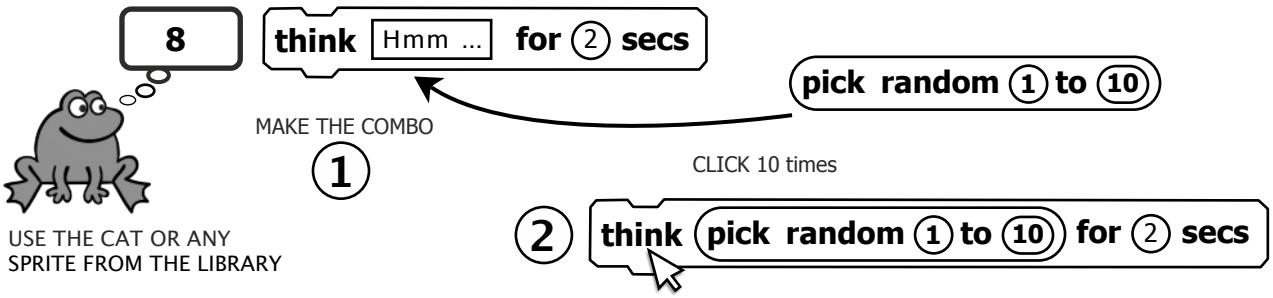
1. three number operations 

2. Explore subtraction / multiplication / division
3. Programme the average of three or four numbers
4. Programme Scratch to answer this: **5 + 4 x 3**

Strategy 2: The **Pick Random Operator**

Introduce Logical Thinking with the **pick random** block

Get random Numbers with the **pick random** operator in the think block



Write the random number after each click.
Answer the questions and discuss the outcome.

RANDOM NUMBERS (Round 1)

What's the highest number?

What's the lowest number?

What's the most common number?

Which number between 1 and 10 didn't appear (if any)?

Record another 10 clicks.

RANDOM NUMBERS (Round 2)

What's the highest number?

What's the lowest number?

What's the most common number?

Which number between 1 and 10 didn't appear (if any)?

Compare the two rounds. Discuss

Was there a different highest number?

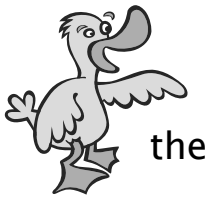
Was there a different lowest number?

Was there a different most common number?

Which number between 1 and 10 didn't appear in either round?

Worksheet based on Strategy 2

Pick random



the sprite

code one script at a time and click

i 

Experiment with 10 clicks

ii 

Change the default range

iii 

What's the outcome of a single click?

What's the highest possible result? lowest possible?

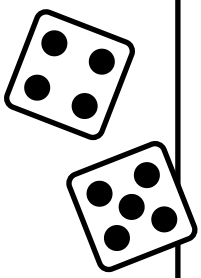
Experiment with subtraction/ multiplication / division. What difficulties can arise?

Other ideas

- Change the default values of 1 to 10, to **1 to 6**.
  What common object returns values of 1 to 6?
- Use pick random to select from a larger range of numbers: e.g. Randomly select a month/ select a random number from the total number of pupils in your class.
- etc

With more experience of Scratch you can develop a variety of uses for the pick random operator

(a) The **pick random** block simulates the effect of rolling a die.
What do you need to know to create your own virtual dice ising Scratch code – the graphics, conditional coding. i.e. if **pick random** returns a *four* your code must make the four-face of the die show itself, you might want to simulate the rolling / turning of the die etc. **E.g.** take a look at the **http://scratch.mit.edu/projects/23018053**



(b) **Explore data:** frequency, block graphs
What if you could programme two dice to roll simultaneously?
http://scratch.mit.edu/projects/23629779
What are the chances of each number from 2 to 12?

(c) Create a number spinner?

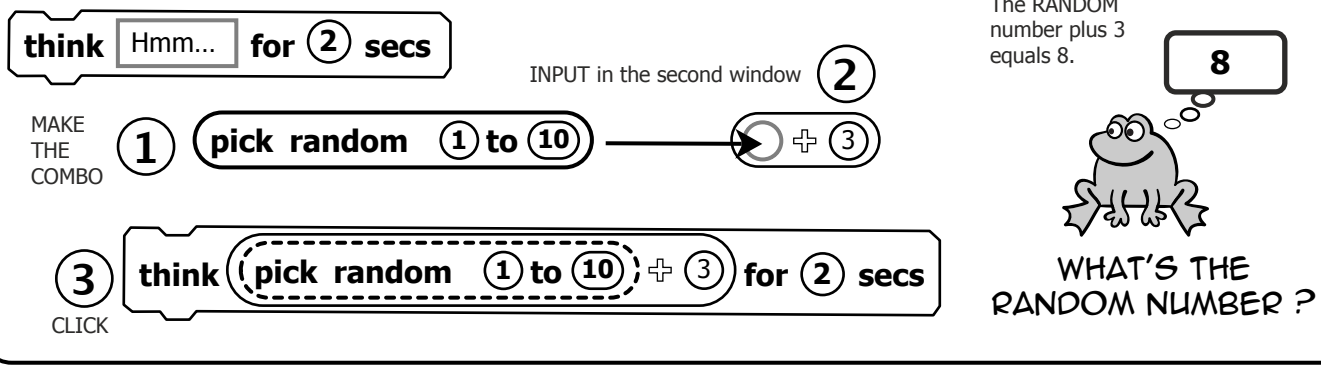
(d) **pick random** is used frequently to randomise movement, turning, positioning on the stage, timing, size, colour effects, sound, costume animation etc.

See the sample children's worksheet based on the pick random block (next page)

There are more project examples at
http://scratch.mit.edu/users/scratchfromscratch2
http://scratch.mit.edu/users/scratchfromscratch

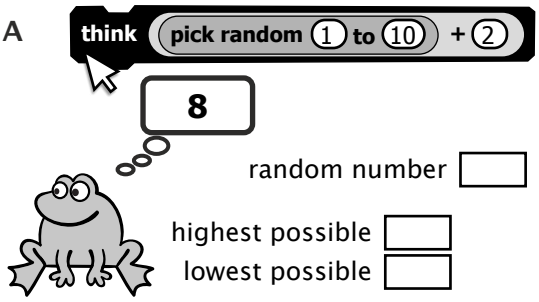
RANDOM NUMBER PLUS A NUMBER

Random Numbers and the *plus operator*

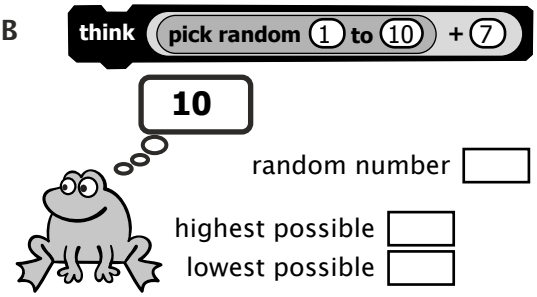


Write the random number and the highest and lowest possible answer the sprite could give.

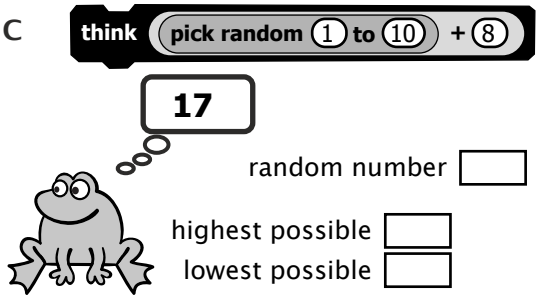
A



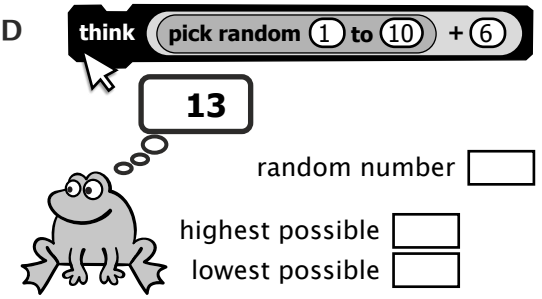
B



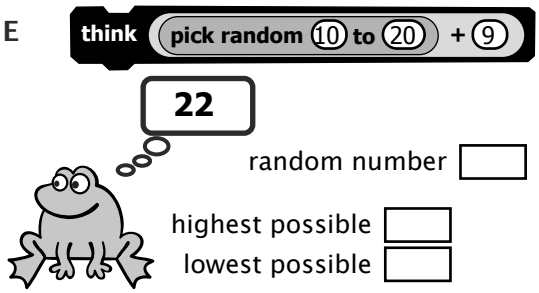
C



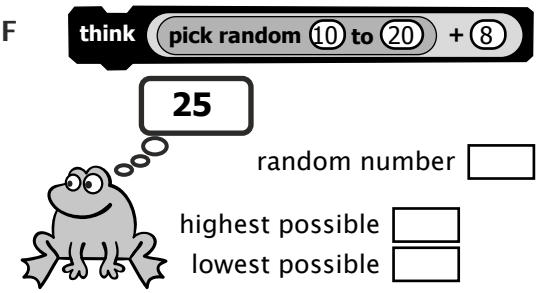
D



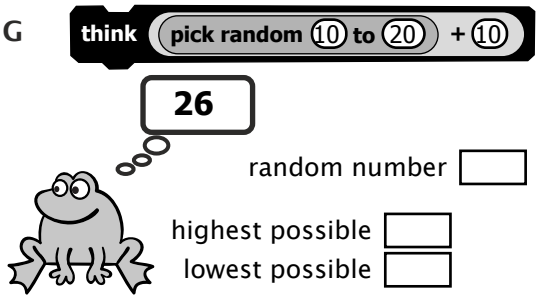
E



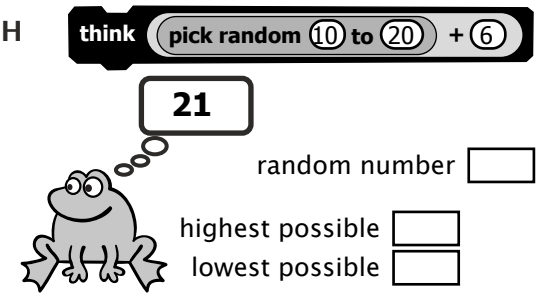
F



G



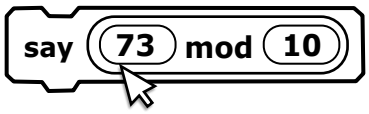
H



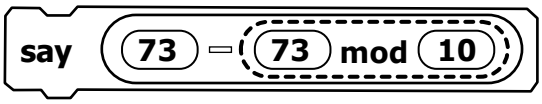
Strategy 3: The Modulo Operator

'mod' is short for **modulo** (or modulus). Find out what it means through experimentation. Explore its relevance in several areas of primary maths. e.g. Tens & Units

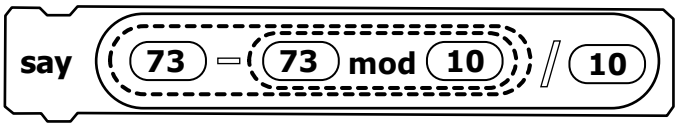
Tens & Units

i 

What number does this report?
What does it tell you about this 2-digit number?

ii 

What number does this report?

iii 

What number does this report?
What does it tell you about this 2-digit number?

Experiment further until you see the relationship between these three steps and how to programme Scratch to tell the **Tens** and **Units** of a 2-digit number.

Use **join blocks** and your own input text to make the sprite say of the number 28: "There are 2 tens". "There are 8 units".

As you gain experience of Scratch you can make greater use of the modulo operator in the area of Tens & Units (see opposite) and in other areas also.

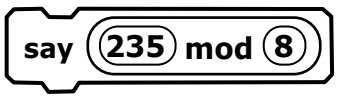
Odd and Even numbers

Use 'mod' to get Scratch to tell the divisibility of numbers.
e.g. a number with **mod 2 = 1** is an odd number; when **mod 2 = 0** it is even

Factors, Divisibility of numbers

e.g. A number with **mod 3 = 0,** is divisible by 3
A number with **mod 7 = 0,** is divisible by 7
A number with both **mod 3 = 0** and **mod 7 = 0,** is divisible by 21

If the number **235 mod 8 = 3** what is the nearest number to 235 that's divisible by 8?



Prime Numbers

It should be possible to use the **mod operator** to discover the prime numbers less than 100 in the Sieve of Erathosthenes

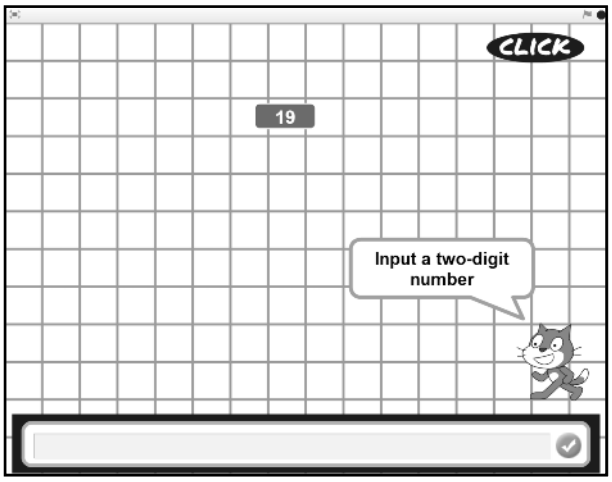
Go to http://scratch.mit.edu/users/scratchfromscratch2 to locate these projects.

Tens And Ones online

Click **See inside**. You can see in among the code the **mod operator** used with some familiar blocks and some not so familiar blocks.

The small rounded orage blocks are **variable** reporters and they are introduced in Strategy 5. Much of what you see in the code, is similar to what you have been doing with the **mod** on the opposite page.

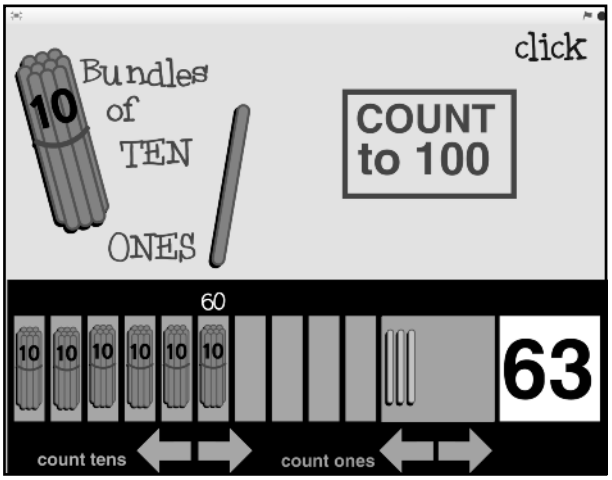
An important feature of the online project is the input bar, which programmes the sprite to tell the tens and ones of any two-digit number you input at the prompt.



Bundles of Tens

Click **See inside**. You can this has a lot more code than the Tens and Ones project. Fortunately it is shared online and it can be downloaded and used in class. With more experience in Scratch you might prefer to remix the code and personalise it to your own style.

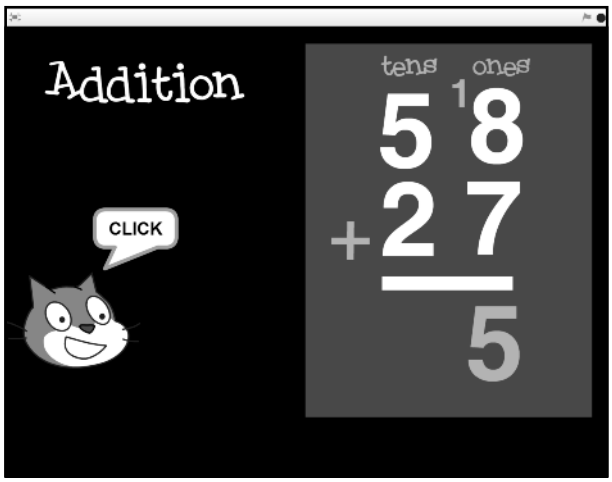
Note: The vast majority of teachers and children will enjoy and gain insight from the level of coding with the **mod** block as on the opposite page. There's no need to re-invent the wheel!



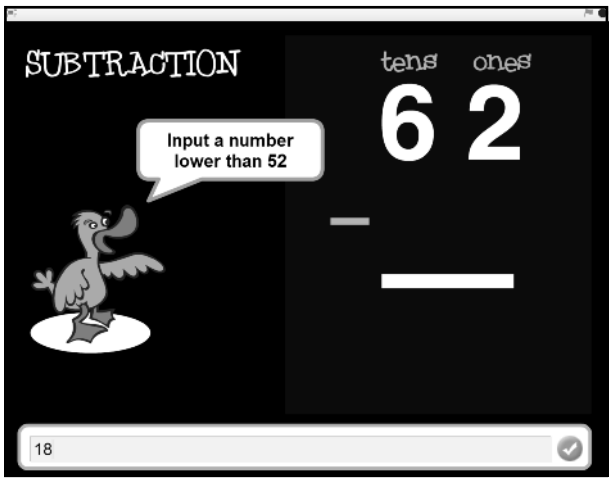
Add to 99 / Subtract within 99

Locate the projects and click **See inside**.

"Addition to 99" app. Child inputs any numbers within definite limits. This is a tutoring app, demonstrating addition skills with or without renaming with help from the Scratch cat. Child learns to differentiate when renaming is necessary.
Click **See inside** to study the code..



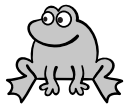
"Subtract within 99" app. Child inputs any numbers within definite limits. This is a tutoring app, demonstrating subtraction skills with or without regrouping with help from the Scratch duck. Child learns to differentiate when regrouping is necessary.
Click **See inside** to study the code.



Strategy 4: The **Repeat Loop** and **Create Clone** command

The report loop is a much used block in Scratch coding. The following code is as easy as coding gets. A creative teacher of six-year olds could get several fun-filled lessons on the IWB from these few blocks.

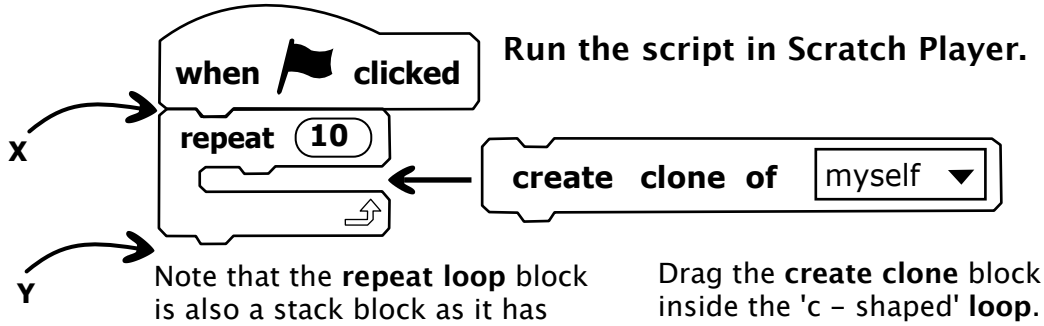
Worksheet to explore Strategy 4



the sprite

i Make the sprite draggable in Scratch Player.
(Hint: Click ⓘ on the sprite thumbnail and tick 'can drag in player' in the **sprite info box** that opens up)

ii Make 10 copies of the draggable frog

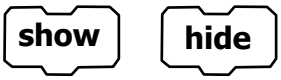


Run the script in Scratch Player.

Note that the **repeat loop** block is also a stack block as it has both a notch and a bump!

Drag the **create clone** block inside the 'c - shaped' **loop**.

How many frogs did you intend to have?
How many are there?
Drag and separate the sprites with the mouse pointer.



find in the **Looks** palette

When you create a clone you see the original sprite as well as the clone. Drag **show** and **hide** blocks to **X** and **Y respetively** to correct the problem of the original sprite.

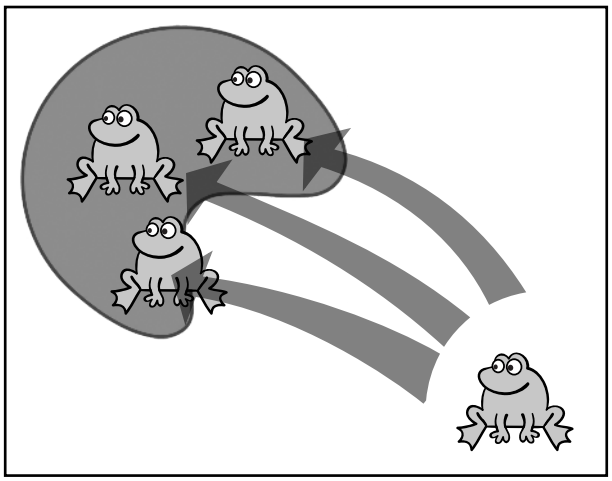
On and Off the Lily-pad

Drag-and-drop activities can be used across many class levels eg.

- for analysis of number at junior level
- for grouping and sharing when teaching multiplication, division
- to drag images and solution sprites to their correct target etc.

As you gain experience of Scratch you will use Sensing blocks from the light blue coloured **Sensing** palette. You will use variables to make numbers increment when one sprite collides with another e.g. to simulate the frog landing the lily-pad. **Variables** are introduced in Strategy 5.

You can have loops inside loops and use Move blocks to create shapes, rows & columns etc. Study the code in the projects on the opposite page.



You can draw the lily-pad sprite in the Paint Editor. It's very easy to do using the Ellipse and Reshape Tools in the Vector Editor. First you must click the 'Paint new sprite' icon under the right hand side of the stage to open the Paint Editor.

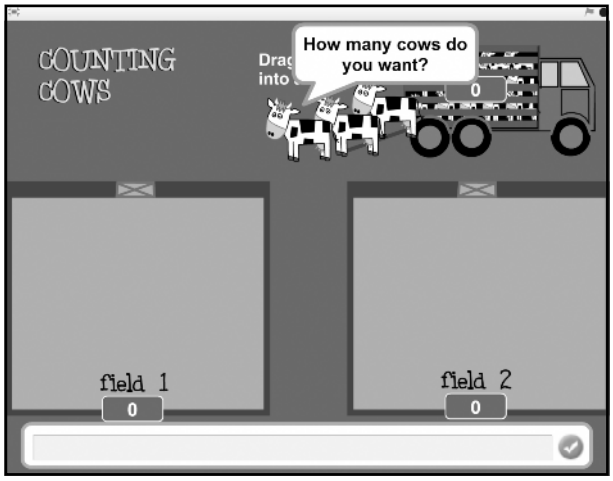
See how this basic script can be developed for sorting, grouping, shape-drawing activities etc.

Go to **http://scratch.mit.edu/scratchfromscratch2** to locate these projects.
or go straight to the project at **http://scratch.mit.edu**./projects/25386237

Counting Cows and Sheep

Click **See inside** and look at the sprites and their code. You can see several sprites in the sprite list area, under the stage. The fields are sprites, but they don't have much code attached to them. The *truck, Cow2* and *Cow3* have no code at all.
The main code is on the sprite named *cow1.* Click on the *cow1* thumbnail. It contains the variables *field1, field2* and *number*. These are made in the **Data** palette.
Apart from the variables and the **ask** for input code, the first script on *cow1* is quite similar to the basic script on the opposite page to this. Or look at the *sheep Counting* project, which does not include the **ask** block.

. . . /projects/25386237



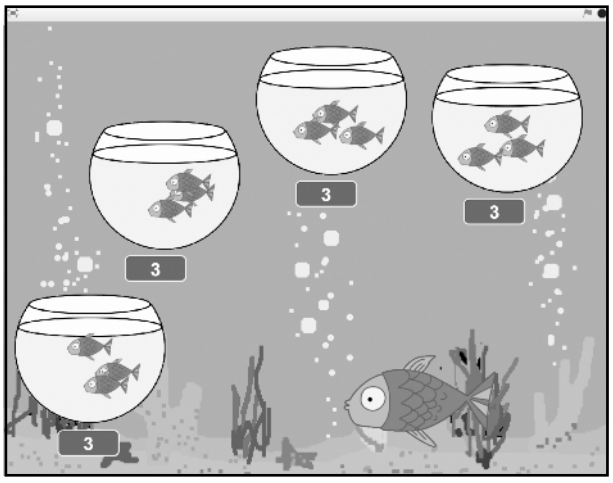
Goldfish App

After you become familiar with variables, through Strategy 5 you should begin to make great use of **loops** when using Digital Numeracy with Scratch to teach Maths.

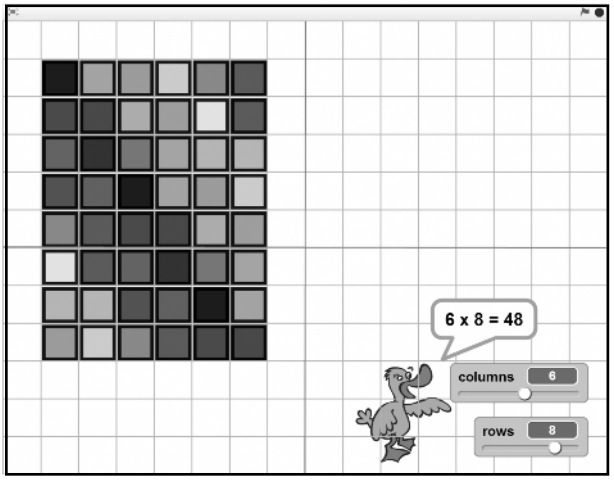
The goldfish in the fishbowls is pitched at 9 – 10 years olds, but the code is not much more difficult than on the opposite page.

After you become familiar with variables, through Strategy 5 you should begin to make great use of **loops** when using Digital Numeracy with Scratch to teach Maths.

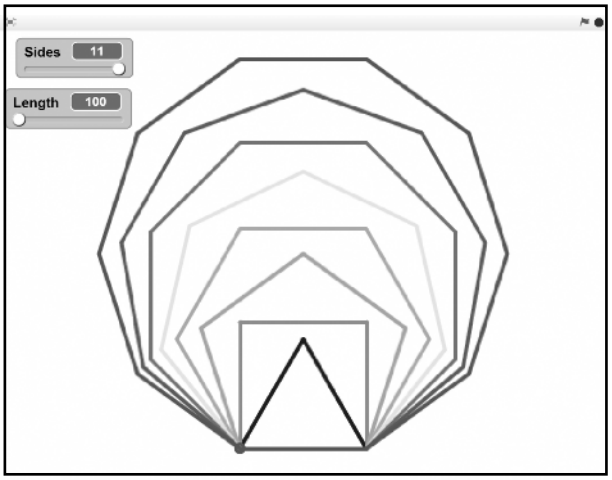
. . . /projects/25520526



Rows_by_Columns. . . /projects/25451945



variable_Polygons. . . /projects/23015562



Strategy 5: Make a **Variable**

A variable holds data that changes. Variables are one of the most important elements of any programming language. They can be made so easily in Scratch and used by children 7 years or younger! You make variables in the **Data** palette. You can give you variable any name you like – but it should make sense to you.

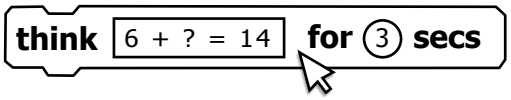
Complementary Addition

Here's a puzzle?

What number when added to 6 makes a total of 14?

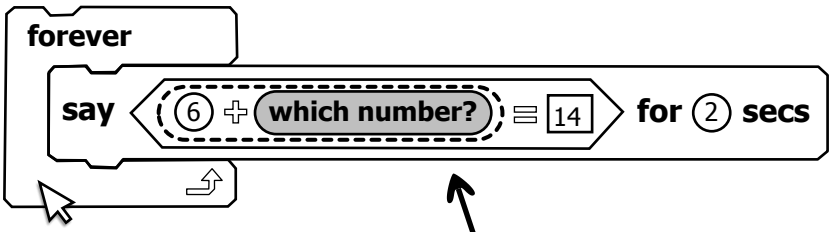
the sprite

6 + ? = 14



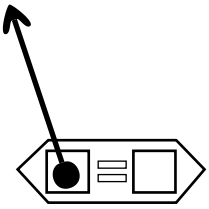
The code to solve the puzzle

Make a variable and give it the name *which number*. Tick the check box to make the **monitor** visible. Click the monitor window twice to get the slider version. Right-click the monitor window and set Min to 0 and Max to 10.

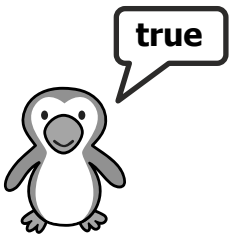


Note that the **forever loop** has no bump at the bottom!

What do you think is the significance of that?

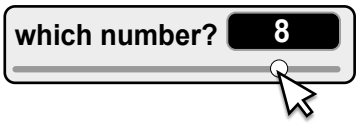


A **plus operator** with the **variable reporter** (orange) in its second window, are slotted into the first input window of an **equality reporter** block.



DRAG the SLIDER

until the sprite says 'true'



Dragging the slider is testing a value. When the **say** block is in a **forever loop** you don't test just once, you keep on testing. DRAG the slider. The penguin says ***true*** when number **8** shows. That's the number that complements 6 to make a total of 14.

Boolean (true / false) values

Equality and **inequality reporter** blocks are also known as comparison reporters, because they compare the value in the left window with the value in the right window and return a Boolean (true / false) value.

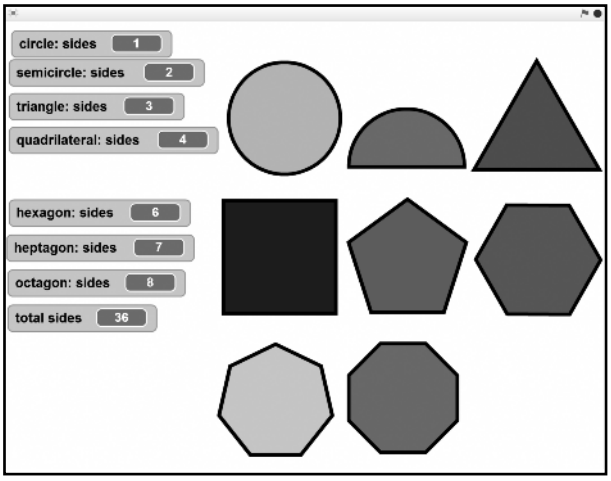
Learn more about making variables from the online projects already mentioned. See how a variable is used to store children's input numbers. See how other variables increment after drag-and-drop 'collisions' between sprites e.g. when a frog touches a lily-pad or a cow touches a field.

Go to **http://scratch.mit.edu/scratchfromscratch** to locate this polygon-shapes project.
or go straight to the project at **http://scratch.mit.edu**./projects/23015571

totalSides

Click **See inside** and look at the shape sprites. Unlike the **variable_Polygons** project (two pages back) this project just adds up the total sides of the shapes, which are pre-drawn.

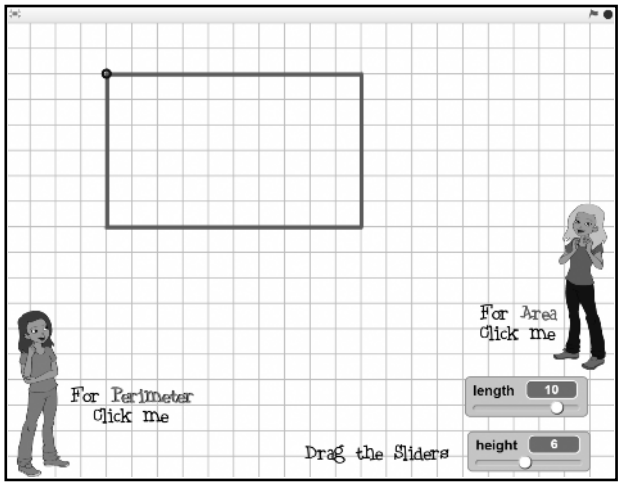
When making a variable you are asked if your variable should be 'For all sprites' or 'For this sprite only' – a private variable. Generally go with 'For all sprites'. In this project, a variable of the same name *total sides* will need to hold a different value for each shape. So *total sides* was made as a private variable on each sprite 'For this sprite only.' Click the **stage icon** to the left of the sprite list and you see that the variable is set to zero (for all sprites) when the green flag is clicked.



The next page ends this introduction to Digital Numeracy with Scratch with a sample worksheet based on Strategy 6:

Now that you have been introduced to *Digital Numeracy with Scratch* and have become familiar with Strategies 1 to 5, you should be able to evaluate its benefits as a support methodology to teach Maths against a background of calls to introduce coding to children at school. You may now feel better informed to agree with my belief that teachers who try some of these *Digital Numeracy* strategies for themselves will realise that they don't need to have a degree in computer science to be a better Maths teacher or to teach Maths today using Scratch code.

The final strategies are still at 'alpha' stage but there are many examples of Strategies 6 (the Pen palette: and Drawing Shapes), 7 (the Grid and the Move and Turn commands) and 8 (Code for Puzzles and Problems) in the online examples.



a. Programme the computer to draw the **six** rectangles given below.

1. 6 squares by 4 squares
2. 5 squares by 6 quares
3. 8 squares by 3 squares
4. 9 squares by 4 squares
5. 7 squares by 5 squares
6. 6 squares by 7 squares

b. Make the computer draw each rectangle when the **space bar** is pressed

c. Programme a sprite to **say** the **area** of the shape when the ***a** key* is pressed and to **say** the **perimeter** when the ***p** key* is pressed.

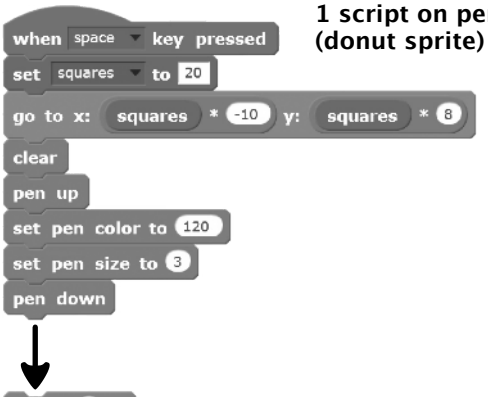
Create a 20-steps background grid and centre it. (Teacher will help.)

Create a small donut sprite to replace the cat. (Teacher will help.)

Make a variable and give it the name **squares**. Set squares to **20** in the initial script.

Initialise the pen and set in **down** state to draw the following rectangles.

1 script on pen (donut sprite)



This script will draw the first rectangle.

2 scripts on girl sprite

This script will make the sprite tell the area.

Script 1

This script, on the same sprite, will make it tell the perimeter.

Script 2

Chicken Run Puzzle

You can have several shapes of rectangles all with the same perimeter but with different areas.

I keep chickens and I have 26 metres of chicken-wire. I want to make a rectangular area in which to keep the chickens. What is the biggest area I can enclose in the chicken run.

How many whole-metre options do I have?

Create a Scratch programme to illustrate the story.