# CHAPTER 5 Basic Procedures

Overview

Pupilx explore how procedures can be used to create fabulous drawing patterns, before creating their own.

### To do before the session

1. Look at the grid below and decide which optional and SEN activities you are going to include and exclude.
2. Print pupil worksheets for each activity chosen and staple into a booklet, one for each pupil.
3. Print marksheets for activities chosen to be placed where pupils can access them.
4. Download the code needed and place in a templates folder on your school network or add to a Scratch Studio or link on your learning platform.
5. Download the slides that go with the concept introduction.
6. Study the notes that go with the slides.
7. Examine the teacher help notes that are provided alongside every activity.

### To do at the start of the session

If you have not introduced **simple procedures** with this class before, do this first using the resources on page 15 as a whole class activity.

### To do after the concept has been introduced

Each activity has whole class notes to help you explain what is needed if it is the first time pupils have carried out this type of activity. There are also core instructions underneath in case you are sticking to the core activities only.

**How this module fits into a programming progression**

- Sequence and Input
- Count-controlled loops
- Indefinite loops
- Conditional selection
- Conditional selection in a loop
- **Procedures**
- Variables

**Vocabulary**

Procedure, task, name, call a procedure

| Resource Name | Core Optional SEN | Teacher | Pupil Grouping | How Assessed | SCRATCH ACCESS |
|---|---|---|---|---|---|
| **CONCEPT** Simple procedures | **CORE** | Leads Session | Solo whole class activity | Formative | NO |
| **PARSONS** | OPTIONAL SEN OPTIONAL ALL (predict or parsons not both) | Support Poor Readers | Solo or Paired (Teacher choice) | Pupil Marked Marksheet Provided | YES Exploring Basic Procedures Parsons |
| **FLOW** | OPTIONAL ALL If you do predict I recommend you do flow first | Can be done as a Whole Class or with a Large Group | Solo or Paired (Teacher Choice) | No Marked Outcome | NO |
| **PREDICT** | OPTIONAL ALL (predict or parsons not both) | Support Poor Readers | Paired | Pupil Marked Marksheet Provided | NO |
| **INVESTIGATE** | **CORE** | Support Poor Readers | Paired | Pupil Marked Marksheet Provided | YES Exploring Basic Procedures |
| **CHANGE** | **CORE** | Support Poor Readers | Paired | Pupil Marked Marksheet Provided | YES Exploring Basic Procedures |
| **CREATE** | **CORE** | Assesses Pupil Work and Checks Pupil Self-Assessment | Solo | Pupil Assessed & Teacher Assessed | YES Exploring Basic Procedures |

Core activities general instructions

1. Group pupils in roughly same ability pairs. For **investigate** and **change** worksheets, pupils will work in pairs, for **create** they will work separately.
2. Give out the pupil booklets and explain that pupils need to follow the instructions on the sheets to explore how **basic procedures** work.
3. Explain that each pupil will record separately while working alongside their partner and keeping to the same pace as their partner.
4. Demonstrate where they can find the template code and explain that pupils will share one device for investigate and change.
5. Explain that during each question only one person should touch the shared device and they should swap who that person is when there is a new questions.
6. Encourage them to discuss their answers with their partner. If they disagree with their partner, they can record a different answer in their own booklet.
7. Show pupils where it says they should mark their work on the sheet and where the answer sheets are in the classroom.
8. Remind pupils to return marksheets after marking, because there are not enough for every pair to have their own.

**Key Programming Knowledge**
A procedure is a small section of a program that performs a specific task.

**Simple Procedures**
Have a name
Are called or run by the name
Can be run many times in a programme
Found in My Blocks in Scratch
In Scratch has define first

**Naming**
Always name a procedure after the task that it does
Avoid naming procedures with spaces
Avoid using the same name as a variable

**Resources**
**Exploring basic procedures** https://scratch.mit.edu/projects/312212285/
**Parsons Exploring basic procedures** https://scratch.mit.edu/projects/623333620/

| | |
|---|---|
|  | On the sheet, if it says no Scratch, they must work only on the sheet. |
|  | If it says Scratch with a green tick, they can use one device between the pair. |
|  | If it says work with a partner, they must work at the same speed as their partner. |
|  | If it says work on their own, they must do this using a separate device each working alone. |

**Scottish Curriculum for Excellence Technologies**
I understand the instructions of a visual programming language and can predict the outcome of a program written using the language. TCH 1-14a

I can explain core programming language concepts in appropriate technical language TCH 2-14a

I can demonstrate a range of basic problem solving skills by building simple programs to carry out a given task, using an appropriate language. TCH 1-15a

I can create, develop and evaluate computing solutions in response to a design challenge. TCH 2-15a

English Computing National Curriculum Programs of Study

Pupils should be taught to:

- **design, write and debug programs that accomplish specific goals**, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.
- **use sequence,** selection and **repetition in programs;** work with variables **and various forms of input and output**
- **use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs**

Welsh National Curriculum Relevant Strands

Progression Step 3.

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

## BASIC PROCEDURES PARSONS

Start Scratch and load Parsons exploring basic procedures

Work with a partner


Use the algorithm below to help you connect the Scratch blocks in the correct places in the Parsons exploring basic procedures.

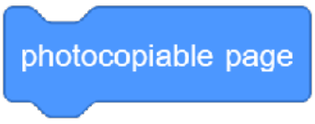
**Main Program**

Start
Point right
go to x –130 y 0
Clear all old lines
square procedure
Move forward 60
square procedure
Move forward 80
Do 5 times
    Right 72 degrees
    eqi_triangle procedure
    Pause 1/2 second

**Procedures**

| | |
|---|---|
| define eqi_triangle | define square |
| Start drawing pen down | Start drawing pen down |
| loop 3 | loop 4 |
| move 50 forward | move 30 forward |
| right 120 degrees | right 90 degrees |
| stop drawing pen up | stop drawing pen up |

Now mark your work using the Parsons marksheet



## SUPPORTING PARSONS

### Whole class advice

Load Parsons exploring basic procedures code and then use the algorithm on this page to build the code. When you have completed it, run the code and check your answer with the marking sheet.

### Send advice

Parsons problems can be made less complex by connecting more blocks in the example Scratch code and saving that version as a new template.

### Understanding programming

You can find out more about Parsons problems in the teacher book that accompanies this series.

### Individual advice

Point out that the code inside a loop is indented in the planning algorithm and in the code. This can help some pupils connect those aspects in both.

Use the algorithm below to help you connect the Scratch blocks in the correct places in the Parsons exploring basic procedures

### Notes on the activity

This allow pupils to build part of the code first before investigating, modifying and creating code of their own. The algorithm is written in language similar but also different to the code. This helps pupils by enabling them to see an example of planning which will help them when they come to plan their own project. On its own, it is not enough deep thinking about the code to enable agency, but as a starter or SEN activity it is useful to see how code can be built.

### Able advice

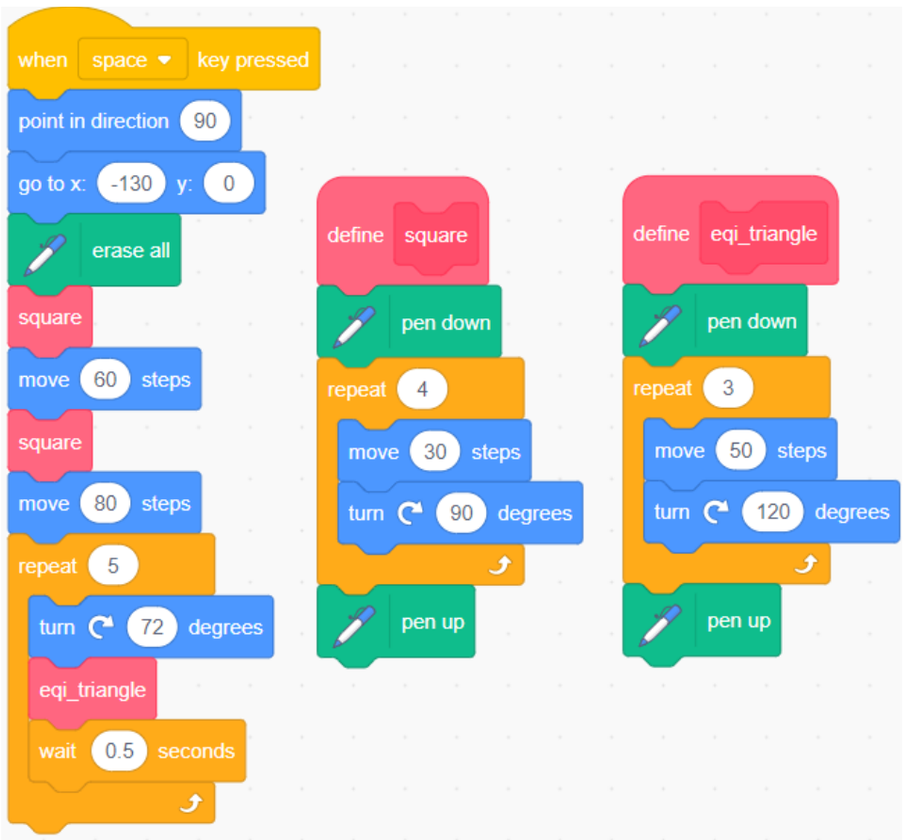
Parsons problems can be made more complex by separating more blocks in the example Scratch code and saving that version as a new template.

**Procedures**

| define eqi_triangle | define square |
|---|---|
| Start drawing pen down | Start drawing pen down |
| loop 3 | loop 4 |
| move 50 forward | move 30 forward |
| right 120 degrees | right 90 degrees |
| stop drawing pen up | stop drawing pen up |

**Main Program**

Start
Point right
go to x –130 y 0
Clear all old lines
square procedure
Move forward 60
square procedure
Move forward 80
Do 5 times
    Right 72 degrees
    eqi_triangle procedure
    Pause 1/2 second

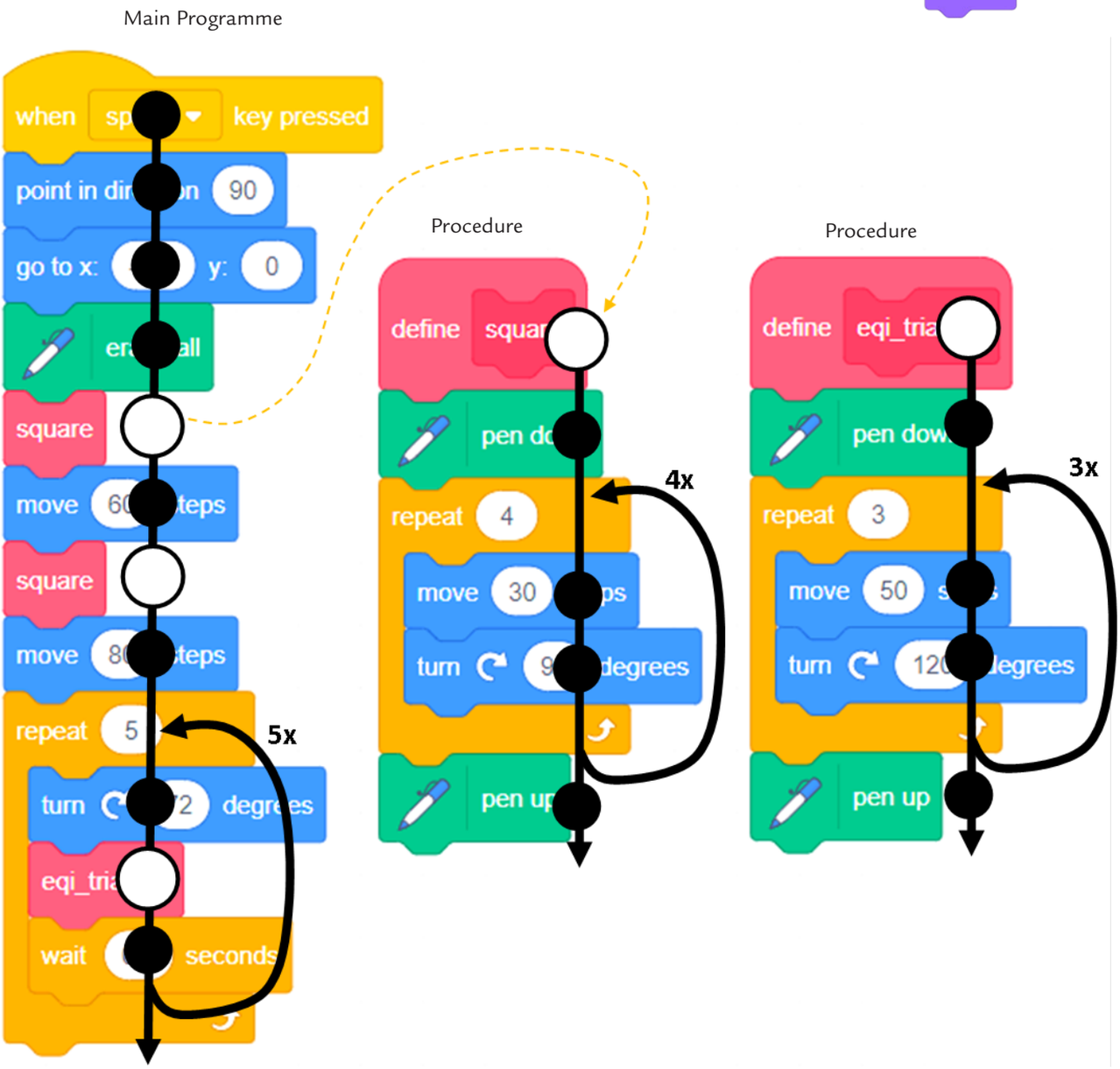


## BASIC PROCEDURES

Work with a partner


### FLOW

Read the code carefully with your partner and follow the order the code is run. Make sure you jump over to the procedure when it is started in the main programme.




photocopiable page

# Teaching Primary Programming with Scratch

Pupil Book – Year 6

**PHIL BAGGE**

A research informed scheme of work by Phil Bagge HIAS Computing Inspector/Advisor
Part of the HIAS Teaching Primary Programming from Scratch Series

Published in 2023 by University of Buckingham Press,
an imprint of Legend Times Group
51 Gower Street
London WC1E 6HJ
info@unibuckinghampress.com
www.unibuckinghampress.com



Published by arrangement with Hampshire Inspection and Advisory Service (part of Hampshire County Council)



ISBN 978-1-91505-4-289

# CONTENTS



Book resources can be downloaded from https://computing.hias.hants.gov.uk/course/view.php?id=5

# INTRODUCTION

## Scheme

This book is a complete scheme of work for teaching primary programming using Scratch in Year 6 for 10–11 year olds.

## Part of a Series

It is part of a five-book series. Three other books include projects for other year groups.

*Teaching Primary Programming with Scratch, Year 3*

*Teaching Primary Programming with Scratch, Year 4*

*Teaching Primary Programming with Scratch, Year 5*

If you are interested in the methodology and research-informed practice behind this series as well, as well as a wealth of other insights gained from teaching block-based programming for thousands of hours, then this will be an informative read:

*Teaching Primary Programming with Scratch – Research-Informed Approaches.*

## Permissions

It includes permission to photocopy the pupil worksheets and answer sheets for your class and school. These are clearly marked.

It includes links to example code, project templates and slides to introduce new programming concepts.

## Progression

There is a clear, research-informed progression through the series, and the graphic on the next page on a grey background shows which programming concepts are introduced in this book.

## Pedagogy in a Few Paragraphs

### Introduction to Programming Concepts Away From Code

Pupils are taught key programming concepts away from programming to lower cognitive load and make it easier to transfer these ideas from one programming language to another.

## Paired Programming

Pupils are encouraged to work in same ability pairs for some parts of the projects, because this has shown to be particularly helpful for pupils working within or below the expected outcomes.

## PRIMM

Pupils are encouraged to read and understand code before they create their own code. We use the PRIMM method in this book.

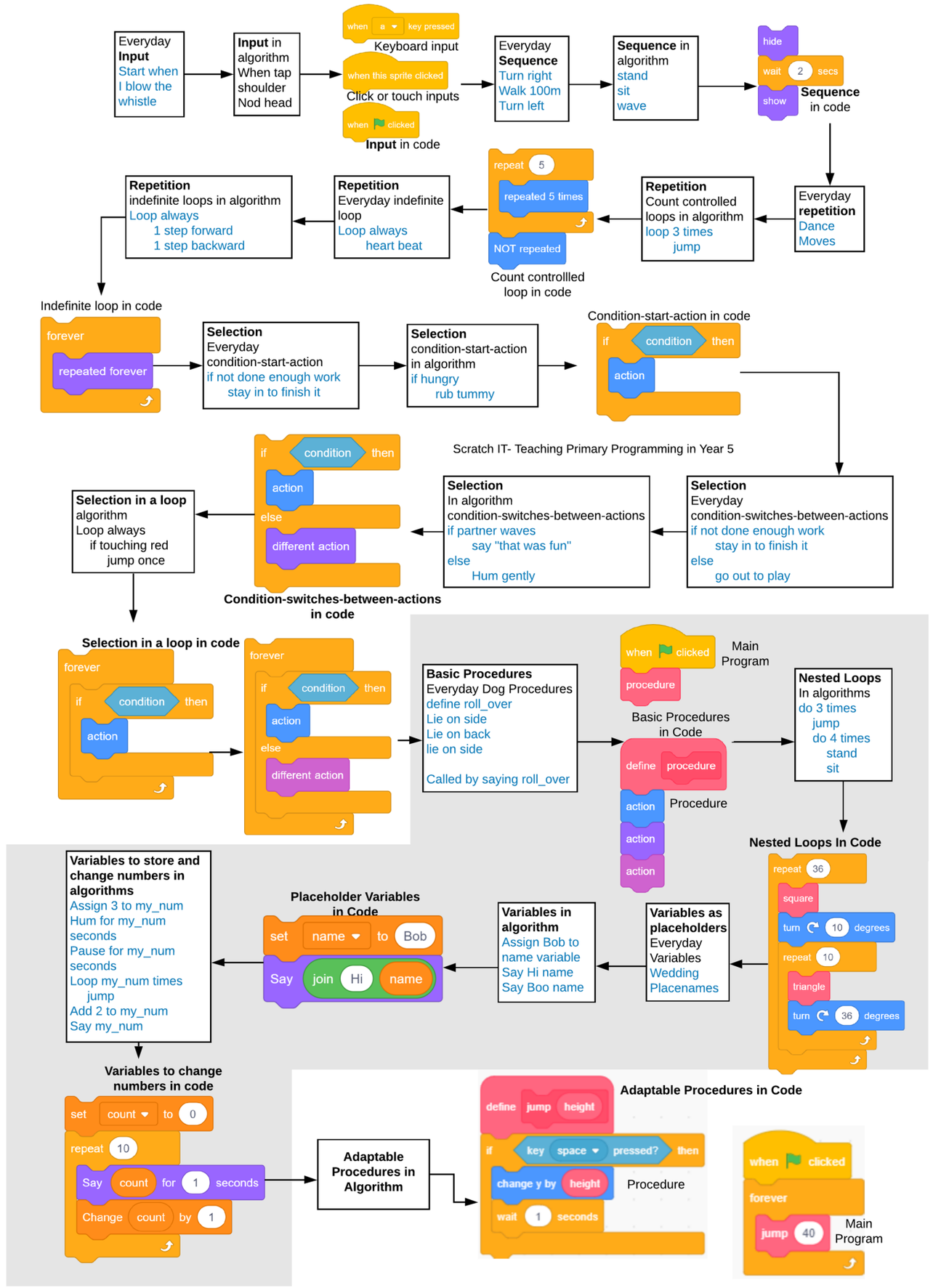
Predict

Run

Investigate

Modify (change)

Make


Everyday
Input
Start when
I blow the
whistle
Input in
algorithm
When tap
shoulder
Nod head
when a key pressed
Keyboard input
when this sprite clicked
Click or touch inputs
when clicked
Input in code
Everyday
Sequence
Turn right
Walk 100m
Turn left
Sequence in
algorithm
stand
sit
wave
hide
wait 2 secs
show
Sequence
in code
Everyday
repetition
Dance
Moves
Repetition
Count controlled
loops in algorithm
loop 3 times
jump
repeat 5
repeated 5 times
NOT repeated
Count controllled
loop in code
Repetition
Everyday indefinite
loop
Loop always
heart beat
Repetition
indefinite loops in algorithm
Loop always
1 step forward
1 step backward
Indefinite loop in code
forever
repeated forever
Selection
Everyday
condition-start-action
if not done enough work
stay in to finish it
Selection
condition-start-action
in algorithm
if hungry
rub tummy
Condition-start-action in code
if condition then
action
Scratch IT- Teaching Primary Programming in Year 5
Selection
Everyday
condition-switches-between-actions
if not done enough work
stay in to finish it
else
go out to play
Selection
In algorithm
condition-switches-between-actions
if partner waves
say "that was fun"
else
Hum gently
if condition then
action
else
different action
Condition-switches-between-actions
in code
Selection in a loop
algorithm
Loop always
if touching red
jump once
Selection in a loop in code
forever
if condition then
action
forever
if condition then
action
else
different action
Basic Procedures
Everyday Dog Procedures
define roll_over
Lie on side
Lie on back
lie on side
Called by saying roll_over
when clicked
procedure
Main
Program
Basic Procedures
in Code
define procedure
action
action
action
Procedure
Nested Loops
In algorithms
do 3 times
jump
do 4 times
stand
sit
Nested Loops In Code
repeat 36
square
turn 10 degrees
repeat 10
triangle
turn 36 degrees
Variables as
placeholders
Everyday
Variables
Wedding
Placenames
Variables in
algorithm
Assign Bob to
name variable
Say Hi name
Say Boo name
Placeholder Variables
in Code
set name to Bob
Say join Hi name
Variables to store and
change numbers in
algorithms
Assign 3 to my_num
Hum for my_num
seconds
Pause for my_num
seconds
Loop my_num times
jump
Add 2 to my_num
Say my_num
Variables to change
numbers in code
set count to 0
repeat 10
Say count for 1 seconds
Change count by 1
Adaptable
Procedures in
Algorithm
Adaptable Procedures in Code
define jump height
if key space pressed? then
change y by height
wait 1 seconds
Procedure
when clicked
forever
jump 40
Main
Program

## Creative

Each project provides time and stimulus to be creative in code within the zone of proximal development provided by the taught concepts and explored projects. In other words, it has reasonable projects that can be created independently or with minimum teacher support.

## Knowledge

Key knowledge is introduced in the concept introductions and reinforced in each of the activities.

## Revisiting Learning

It is important to revisit prior learning, so some modules have questions and activities which revise learning from Year 4 on loops and conditions in Year 5.

## Assessment

### Summative Assessment

Summative assessment is baked into every stage of the PRIMM process, providing a wealth of data to determine progress.

If you have used earlier versions of these resources on the code-it website, then you will enjoy the new project assessment grid that combines pupils self-assessment and quick teacher assessment, ideally within the lesson.

### Self-Assessment

Pupils self-mark to help them see how they have progressed, reducing teachers' workload and enabling teachers to concentrate on the pupils that might need more support.

## Hints & Tips

Every pupil's resource also includes a copy of the resource annotated with extra information to further teachers' programming knowledge, hints and formative assessment opportunities in case pupils are stuck, and tips to adapt or support whole class teaching.

Many of these extra hints and tips will not be needed, but the more informed the teacher is the better quality learning opportunity pupils will have.

Yellow highlighted hints and tips are whole class suggestions

Lilac highlighted hints and tips are information to help teachers extend their programming knowledge and sometimes explain why something has been included.

Green highlighted hints and tips are suggestions to help the teacher support individual pupils stuck on a specific question.

## Can We Start Here?

If pupils have never programmed with Scratch before a basic introduction project, *Teaching Primary Programming with Scratch, Year 3* is a must.

I would also recommend a single module of count-controlled loops and one on indefinite loops found in

*Teaching Primary Programming with Scratch, Year 4*

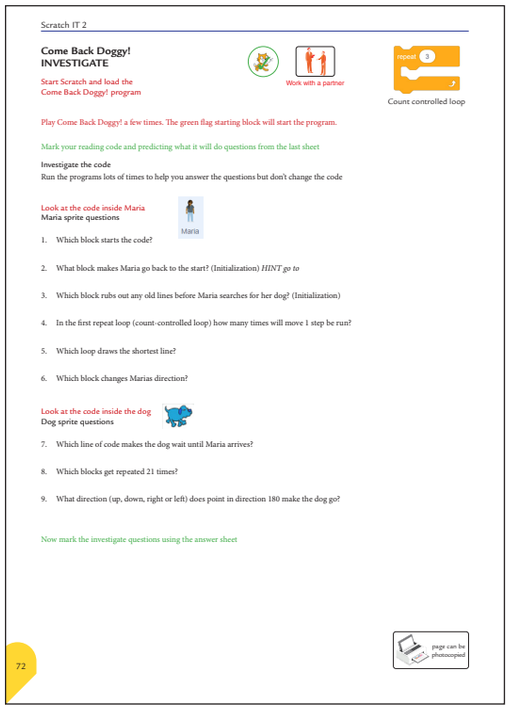
I would also recommend covering conditions using Making Choices and one of the gaming modules found in

*Teaching Primary Programming with Scratch, Year 5*

Many of the projects include revision questions to remind pupils about prior learning.

## Committed to Improvements

HIAS, Hampshire's Inspection & Advisory Service, is committed to developing and improving these resources. We recognize that primary programming is still its infancy in comparison with other subjects, and that new research and primary practice will refine and improve teaching and learning in this area. All royalties earned from this series will be used to write more computing books and revise these resources as needed.



Scratch IT 2

**Come Back Doggy!**
**INVESTIGATE**

Start Scratch and load the Come Back Doggy! program

Work with a partner

Count controlled loop

Play Come Back Doggy! a few times. The green flag starting block will start the program.

Mark your reading code and predicting what it will do questions from the last sheet

Investigate the code
Run the programs lots of times to help you answer the questions but don't change the code

Look at the code inside Maria
Maria sprite questions

Maria

1. Which block starts the code?
2. What block makes Maria go back to the start? (Initialization) *HINT go to*
3. Which block rubs out any old lines before Maria searches for her dog? (Initialization)
4. In the first repeat loop (count-controlled loop) how many times will move 1 step be run?
5. Which loop draws the shortest line?
6. Which block changes Marias direction?

Look at the code inside the dog
Dog sprite questions

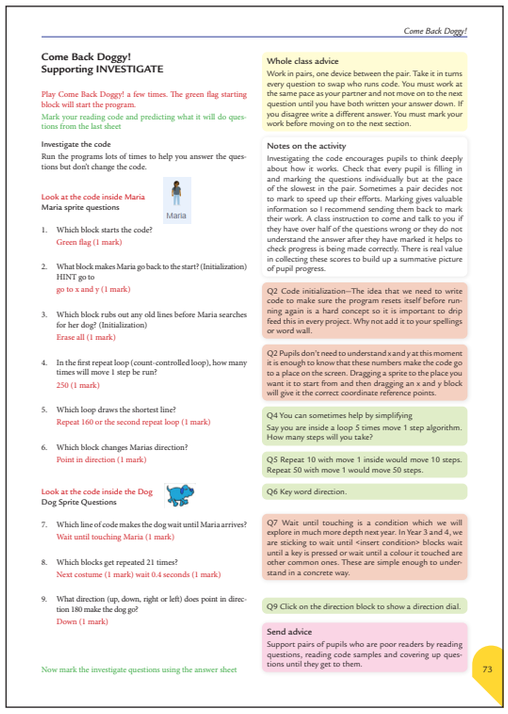
7. Which line of code makes the dog wait until Maria arrives?
8. Which blocks get repeated 21 times?
9. What direction (up, down, right or left) does point in direction 180 make the dog go?

Now mark the investigate questions using the answer sheet

page can be photocopied

72

**Photocopiable resource for pupils**



*Come Back Doggy!*

**Come Back Doggy!**
**Supporting INVESTIGATE**

Play Come Back Doggy! a few times. The green flag starting block will start the program.
Mark your reading code and predicting what it will do questions from the last sheet

Investigate the code
Run the programs lots of times to help you answer the questions but don't change the code.

Look at the code inside Maria
Maria sprite questions

Maria

1. Which block starts the code?
   Green flag (1 mark)
2. What block makes Maria go back to the start? (Initialization) HINT go to
   go to x and y (1 mark)
3. Which block rubs out any old lines before Maria searches for her dog? (Initialization)
   Erase all (1 mark)
4. In the first repeat loop (count-controlled loop), how many times will move 1 step be run?
   250 (1 mark)
5. Which loop draws the shortest line?
   Repeat 160 or the second repeat loop (1 mark)
6. Which block changes Marias direction?
   Point in direction (1 mark)

Look at the code inside the Dog
Dog Sprite Questions

7. Which line of code makes the dog wait until Maria arrives?
   Wait until touching Maria (1 mark)
8. Which blocks get repeated 21 times?
   Next costume (1 mark) wait 0.4 seconds (1 mark)
9. What direction (up, down, right or left) does point in direction 180 make the dog go?
   Down (1 mark)

Now mark the investigate questions using the answer sheet

Whole class advice
Work in pairs, one device between the pair. Take it in turns every question to swap who runs code. You must work at the same pace as your partner and not move on to the next question until you have both written your answer down. If you disagree write a different answer. You must mark your work before moving on to the next section.

Notes on the activity
Investigating the code encourages pupils to think deeply about how it works. Check that every pupil is filling in and marking the questions individually but at the pace of the slowest in the pair. Sometimes a pair decides not to mark to speed up their efforts. Marking gives valuable information so I recommend sending them back to mark their work. A class instruction to come and talk to you if they have over half of the questions wrong or they do not understand the answer after they have marked it helps to check progress is being made correctly. There is real value in collecting these scores to build up a summative picture of pupil progress.

Q2 Code initialization—The idea that we need to write code to make sure the program resets itself before running again is a hard concept so it is important to drip feed this in every project. Why not add it to your spellings or word wall.

Q2 Pupils don't need to understand x and y at this moment it is enough to know that these numbers make the code go to a place on the screen. Dragging a sprite to the place you want it to start from and then dragging an x and y block will give it the correct coordinate reference points.

Q4 You can sometimes help by simplifying
Say you are inside a loop 5 times move 1 step algorithm. How many steps will you take?

Q5 Repeat 10 with move 1 inside would move 10 steps. Repeat 50 with move 1 would move 50 steps.

Q6 Key word direction.

Q7 Wait until touching is a condition which we will explore in much more depth next year. In Year 3 and 4, we are sticking to wait until <insert condition> blocks wait until a key is pressed or wait until a colour it touched are other common ones. These are simple enough to understand in a concrete way.

Q9 Click on the direction block to show a direction dial.

Send advice
Support pairs of pupils who are poor readers by reading questions, reading code samples and covering up questions until they get to them.

73

**Teacher Hints & Tips on the same photocopiable resource**

# WE ARE LEARNING ABOUT PROCEDURES AND VARIABLES IN PROGRAMMING

Variables are used to store information to be referred to and changed in a computer programme or algorithm

## Year 6 Algorithm & Programming

Variables Algorithm

## Variables

Have a name and a value

Algorithms and programs read the name but act on the value

Values can be changed during the algorithm or programme

When writing the value of a variable, we call it assigning

Assign 30 to length variable
Pen down to start drawing
do 4 times
    Move length steps
    Turn right 90 degrees
Pen up to stop drawing

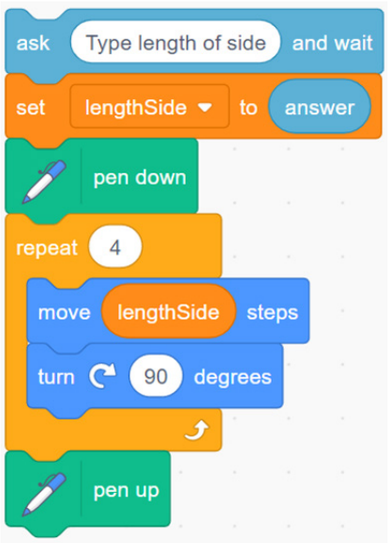
## Naming

Always name a variable after the data that it stores or the task that it does

Avoid naming variables with spaces; use teamScore (camelCase) or

user_name (underscore)

Avoid using the same name as a procedure




Define walk
Move right leg forward
Wait
Move left leg forward
wait

Everyday Procedures

Loop always
    breathe
    if need to go somewhere
        walk

Main Algorithm
Calls Procedures

Define breathe
Breathe in
Breathe out

Procedure Algorithm

**Variable Vocabulary** assign, value **Procedure Vocabulary** define, sub-routine, sub-programme

## Simple Procedures

Have a name

Are called or run by the name

Can be run many times in a programme

Found in My Blocks in Scratch

In Scratch has define first

## Naming

Always name a procedure after the task that it does

Avoid naming procedures with spaces

Avoid using the same name as a variable

Procedures are a set of instructions bundled together to complete a part of a program

photocopiable page