RISING STARS

Learn to Code

Practice Book 3

TEACHER GUIDANCE

Help your pupils to become super-coders!

Introduction

This booklet supports Learn to Code Practice Book 3.

The practice book is broken into three sections, with one coding tool per section. In most cases, the activities within a section are sequential and should therefore be followed in order. Ensure children know how and where to save their work, where relevant, so they can quickly pick up from where they left off.

Try out the activities in the practice book before you give them to children. This is a great way to develop your own confidence in coding.

Contents

Published in association with:

HES
Dedicated to education

Tool: Kodu

www.kodugamelab.com

Watch: Queen of Kodu video guides: bit.ly/1AlMakK
Before you start: Install Kodu Game Lab on all machines. Familiarise yourself with the software and have a go at the activities.

KS2 Computing objectives covered:

- Design, write and debug programs that accomplish specific goals.
- Use sequence and selection 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.
- Use software to design and create programs.

Activity 1: Programming how characters move (page 8)

How to use this activity

This activity introduces children to the Kodu software and guides them through the process of programming the character, Kodu, to move in response to arrow keys being pressed.

Use these activities to build on the idea of selection in programming. Programming in Kodu is based on 'when' and 'do' instructions. For example, you tell Kodu: 'When this happens, I want you to do this'. When there is choice in an algorithm and the possibility of more than one outcome, this is called selection.

Tips

- Use this activity to introduce the idea of different forms of input to children. Explain that Kodu can be played on an Xbox and controlled using a gamepad as well as a PC keyboard. Children may wish to explore this further at home if they have this device. The keyboard will be the input device in these four activities.
- Be aware that if using a command tile like *slowly*, using more than one tile will make Kodu move even more slowly.
- Children will need to enter play mode to test the game. They can do this at any point.
- If children add the wrong tile, they can right-click on the tile and click on *Cut Tile* to remove it.

Assessment

All children should be able to:

- create a game that allows a user to control Kodu by using the arrow keys on their keyboard.

Most children should be able to:

- change the keys with which Kodu is controlled.

Some children should be able to:

- adjust the speed at which Kodu moves.

Activity 2: Programming characters to collect objects (page 12)

How to use this activity

Now that Kodu has been programmed to move using the keyboard, children will add a variety of objects to their world for Kodu to collect. Kodu can be programmed to behave differently when it encounters different objects.

Tips

- In step 2, children are shown how to add more terrain so their game can be played across a larger area. You could take this further and allow them time to explore the different design tools and build a landscape with different features if they haven't already done this.
- There are plenty of different objects that can be added to a Kodu world. Children can choose different objects from the examples used in the activity and program them in the same way.

Assessment

All children should be able to:

- program Kodu to eat the stars.

Most children should be able to:

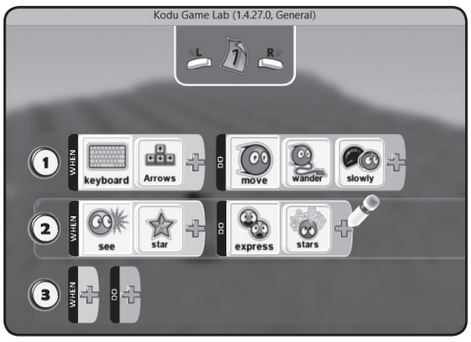
- program Kodu to react to the stars when he sees them.

Some children should be able to:

- change the objects Kodu eats and program him to react in different ways.

About Kodu

Kodu is a visual programming language that enables children to create their own video games on a PC and on an Xbox. Creating a game in Kodu requires no previous programming knowledge. Programming in Kodu is based on dragging, dropping and sequencing different command tiles to produce different outcomes. Kodu also has a comprehensive set of design tools, which enable programmers to design and build the world in which their game will take place.



Activity 3: Programming objects to move (page 16)

How to use this activity

This activity can be used to introduce or consolidate the computing concepts of repetition and infinite loops (repetition that never ends). Using the *always* tile when programming the random movement of the balloon object means it will continue to move repeatedly. This activity will explore two ways to make objects move: random movement and following a preset path, called a motion path.

Tips

- Children can change the size of objects by right-clicking on the object and selecting *change size* in the menu. They can then use the slider to adjust the object's size.
- Children are shown how to copy and paste objects in this activity. This is useful because when an object is copied and pasted, its code is also copied and pasted so the duplicated object will behave in the same way as the original.
- To create a motion path, children will need a bird's eye view of the terrain in their world and will need to use the *Move Camera* tool, which looks like a hand, to change their perspective. They could plan their motion paths on paper by drawing a map of their world before implementing them in Kodu. They will program a jet to follow their motion path.

Assessment

All children should be able to:

- program an object to move randomly and continuously.

Most children should be able to:

- program an object to follow a motion path.

Some children should be able to:

- alter their script so that objects, such as the balloons, only start moving if and when Kodu bumps into them.

Activity 4: Programming a points system (page 20)

How to use this activity

This activity provides good links to the computing concept of variables (things that can change) using the score in the game. A points system motivates the player so it is a vital element in a computer game. In this activity, children will be shown how to allow the user to score points in two ways: by firing at objects and by eating objects.

Tips

- The first thing the children will be guided to do is program Kodu to fire a blip upwards when the *M* key is pressed on the keyboard. They can explore using other keys for this too. This will enable Kodu to shoot an object above it – in this case the balloons and jets they set up in activity 3.
- They will also set up a scoring system so the user scores five points for hitting a balloon and 10 points for hitting a jet. Again, they can change these values, but it's worth encouraging them to think of the user and how having different scores for different objects makes the game more interesting.
- Points will also be scored if Kodu bumps into a star. Children should think about awarding points based on how difficult the task is. The more difficult the task, the more points should be awarded if it is achieved. For example, it is easier to bump into a star than fire at the balloons and jets, so this only scores one point.

Assessment

All children should be able to:

- program the *M* key to enable Kodu to fire upwards.

Most children should be able to:

- award different scores for different tasks that are achieved.

Some children should be able to:

- add another Kodu to make their game a two-player game.

Tool: Logo

mswlogo.en.softonic.com/

Before you start: Install MSW Logo on all machines. Familiarise yourself with the program and activities. Mathematical knowledge of 2D shapes, especially calculating external angles, is vital to these activities.

KS2 Computing objectives covered:

- Design, write and debug programs that accomplish specific goals.
- Use repetition in programs.
- Work with variables.
- Use logical reasoning to explain how some simple algorithms work, and to detect and correct errors in algorithms and programs.

Activity 5: Writing repetition procedures (page 24)

How to use this activity

This activity is a great introduction to text-based programming languages and for programming on-screen turtles in general. There are strong links to maths throughout, in particular, properties of shapes and angles. This activity shows children how to use repetition to program the turtle to draw regular shapes and quadrilaterals. Repetition works here because regular shapes have sides with the same length and angles of the same size.

Tips

- To start again at any point, children can type in the command *HOME CS* to reset the turtle and erase the current drawing.
- In step 3, it's a good idea to make links with what children know about the properties of squares, e.g. 'Why is the code in the brackets repeated four times?'
- In step 4, children program the turtle to draw an equilateral triangle. Take care with the angles here. To program in Logo using the *RT* and *LT* commands, the numerical value needs to be the *external* angle in the shape: 360 degrees divided by the number of angles in the shape. For example, 360 ÷ 3 in this case, so the code will read *RT 120* (not *RT 60*). Ideally, children will have some knowledge and understanding of external angles before trying this activity.
- In step 8, children start to program the turtle to draw quadrilaterals, such as a rectangle. As this doesn't have four equal sides and angles, two sets of *FD* and *RT* commands must be used within the brackets.

Assessment

All children should be able to:

- use repetition to program the turtle to draw regular shapes.

Most children should be able to:

- use repetition to draw a range of quadrilaterals.

Some children should be able to:

- alter the code to create shapes of different sizes.

Activity 6: Writing nested programs (page 28)

How to use this activity

This activity can be used to introduce the concept of nesting, where programs have a repetition command within another repetition command. This allows children to create patterns based on repeating and rotating shapes. There are strong links with rotating shapes and angles.

Tips

- Remind children that the commands within the brackets are repeated, which is why there are two sets of brackets in these nested programs.
- Encourage children to spot the repetition commands for the shapes they drew in the previous activity, within the nested program code.
- In step 6, the *FD* command value is decreased so the whole pattern will fit on the screen.
- In step 8, a new command, *BK*, is introduced, which programs the turtle to move backwards. As with the *FD* command, it needs to be accompanied by a numerical value.
- At any point, allow children to explore the number of repetitions and see if they can predict what the pattern will look like before they run the program.

Assessment

All children should be able to:

- use nested programs to create rotating patterns.

Most children should be able to:

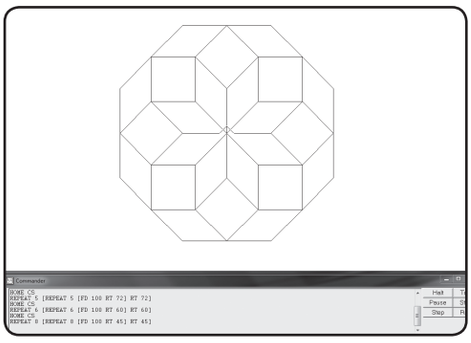
- use the *BK* command in a nested program to create patterns that repeat horizontally or vertically.

Some children should be able to:

- create their own repeating patterns using a nested program.

About Logo

Logo is a text-based programming language, so the commands have to be typed in. Logo allows you to program a 'turtle' – an on-screen object that draws a pen line behind it. Several Logo interpreters are available. MSW Logo is used in these activities: its 'turtle' is a triangle. Basic commands used in these activities are *PD* (pen down), *PU* (pen up) *FD* (forward), *RT* (right turn) and *LT* (left turn). Some commands, such as *FD*, must be followed with a numerical value so the turtle knows how much to move, e.g. *FD 100* means 'move forward 100 steps'.



Activity 7: Creating and naming procedures (page 32)

How to use this activity

This activity builds on children's understanding of nested programs. They create a repeated or nested program and name it so they can use it again easily in their code, without having to type out all the code again.

Tips

- To define a procedure you always start by typing in *TO*, followed by the name of the procedure. In step 2, children type in *TO SQUARE*, which allows them to then enter the commands that instruct the turtle to draw a square. Whatever word follows the *TO* will be the name of the procedure.
- Children then go on to create a pattern using their *SQUARE* procedure and they also name the procedure for their pattern. Children should understand that they can define any procedure they create and give it a name. Once this is done it can be used within a Logo program, just like any command that is already preset.

Assessment

All children should be able to:

- define the procedure for drawing a square.

Most children should be able to:

- use their own procedure for drawing a square to create a pattern.

Some children should be able to:

- define procedures for drawing other shapes and patterns.

Activity 8: Creating procedures using variables (page 36)

How to use this activity

This activity builds on children's understanding of defining their own procedures and makes strong links with the computing concept of variables, which are items within a program that can change. They have already created a procedure for drawing a square and another pattern; now they will combine this with variables so they can alter the size of the shape their program produces each time.

Tips

- In step 2, children are introduced to the command for a variable *:N* which is going to stand for the length of the sides. They are using *N* in this instance but it could be any letter. What makes *N* a variable is the colon in front of it. This is how Logo recognises variables.
- Putting the variable *:N* after the *FD* command will allow the user to choose the value of the *FD* command each time rather than it being programmed in, as it has been before.
- In step 6, two variables are used: one for the *FD* command and one for the *RT* command. This allows the user to choose not only the size of each side but also the number of sides before running the program. There are clear links here again to maths and external angles of shape, which children may need to recap.

Assessment

All children should be able to:

- use variables to create programs that draw shapes of different sizes.

Most children should be able to:

- use two variables to draw shapes of different sizes and with a varying number of sides each time.

Some children should be able to:

- use variables to create programs that draw repeating patterns of different sizes and with different shapes within it.

Tool: Snap!

snap.berkeley.edu/

Before you start: Ensure you have internet connectivity on all machines. Familiarise yourself with Snap! and try the four activities to see how they build on one another to eventually create a text-to-Morse-code converter.

KS2 Computing objectives covered:

- Design, write and debug programs that accomplish specific goals.
- Solve problems by decomposing into smaller parts.
- Use sequence and selection 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.

Activity 9: Converting text into sound (page 40)

How to use this activity

Use this activity to demonstrate how coding can be used to achieve more complex programming. In the first activity, children begin to build their text-to-Morse-code converter by creating a procedure that converts Morse code text into electronic sound.

Tips

- Before beginning these activities, children should be familiar with Morse code and the fact that it is a method of sending messages as electronic pulses. There are short pulses (written as dots) and long pulses (written as dashes), which are combined in different ways to represent each letter of the alphabet.
- In step 2, children are introduced to the term 'tempo' as the speed of a piece of music. The tempo will be altered to create the short and long pulses.
- In step 3, children begin to build their own block. Emphasise that this is useful because they can use the commands in the block later without having to repeat them again. Breaking code into smaller parts like this is called decomposition. It makes big problems much easier to tackle. This is an important computing concept.
- In step 8, parameters are mentioned. This is an advanced computing concept, which children do not need to know at this stage, but can be explained as a type of variable. Parameters can be placed into custom functions so they can behave differently each time they are called upon.

Assessment

All children should be able to:

- create a procedure that converts text Morse code into sound.

Most children should be able to:

- change the tempo of the notes played.

Some children should be able to:

- alter their code so a bulb sprite also lights up when the Morse code is played.

Activity 10: Converting text into Unicode (page 44)

How to use this activity

Use this activity to build on the concept of functions (programs that can be used within a main program and can also feed information back into it) and to introduce the idea of Unicode (computers store text as Unicode numbers, which is a numerical representation) and lists (these allow us to store information for later use in a program). This activity guides children through the process of creating a function that converts a string of letters into a list of Unicode numbers.

Tips

- In step 2, children are introduced to a type of block called a reporter block: it returns a value (it reports back). Here it will return the text as a list of Unicode numbers.
- In step 6, an iteration block is mentioned. Iteration is when a set of instructions is repeated a certain number of times, or until a specific condition is met. In this activity, it will keep working through the text that has been inputted into the function and will stop when this has been done.
- The report block mentioned in step 9 is needed because the user will type in text and the report block will return this as a series of Unicode numbers. The list of Unicode numbers it returns will be used in the main program. Remember, this activity builds a function that will be used within the main program that will, after all four activities have been completed, convert text into Morse code and play it.

Assessment

All children should be able to:

- build a function that converts text into Unicode numbers.

Most children should be able to:

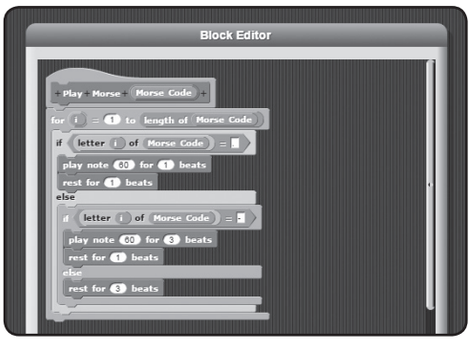
- test lower and upper case letters in their function.

Some children should be able to:

- find out how numbers and other characters are represented in Unicode.

About Snap!

Snap! is a visual programming language based on the idea of dragging and dropping command blocks to create programs. Its interface looks very similar to Scratch. It allows you to build your own blocks as well as using inbuilt ones. These four activities involve considerable levels of challenge. Depending on the experience of your class, you may wish to consider additional support for these activities or using this with Year 6 pupils. You could also swap these activities with some from Book 4.



Activity 11: Using the list lookup function (page 48)

How to use this activity

In the previous activity, children built a function that returned text as Unicode. Now they need to convert the Unicode numbers into the Morse code equivalents as dots and dashes, character by character. They will build another function to do this. This function will then be combined with the other functions in the final activity to build the text-to-Morse-code converter.

Tips

- The block created in step 2 is a reporter block because it will return a list of Morse code to the main program.
- The lookup table mentioned in step 5 is the long list of what each letter looks like in Morse code. In step 6, the children have to enter the Morse code for the whole alphabet. It is a really good idea to get children to work in pairs and check this for each other as mistakes could easily be made when doing this.
- In step 10, the *join* block is used to ensure there is a space between each Morse code character that is returned.

Assessment

All children should be able to:

- build a function that converts Unicode into its Morse code equivalents.

Most children should be able to:

- test upper and lower case letters in their functions to see if they produce the same Morse code.

Some children should be able to:

- work out the Unicode of a word and then use their function to turn it into Morse code.

Activity 12: Combining functions and procedures (page 52)

How to use this activity

This final activity shows children how to combine the functions and procedures they made in the previous three activities to create a program that converts text into Morse code and plays it as a series of electronic pulses.

Tips

- The string variable created in step 2 is where the user will input the text string they want converted into Morse code. It is a variable because it will change depending on what text the user types in. The word ‘hello’ is used as the first test here.
- The iteration block in step 6 means it will work its way through the Unicode list one number at a time and turn each one into its Morse equivalent. The length of this list will change depending on the length of the text string the user enters initially.

Assessment

All children should be able to:

- combine procedures and functions to create a program that converts text into Morse code and plays it back.

Most children should be able to:

- test out their converter using single words and phrases that need spaces in between.

Some children should be able to:

- extend the program so it can deal with Morse for digits and punctuation too.

Glossary

- **Command:** an instruction given to an object or character to make something happen.
- **Debug:** to find and get rid of errors within code.
- **Do** [in Kodu]**:** what your object must 'perform' or do as the result of a command (see 'When [in Kodu]').
- **FD** [in Logo]**:** the command used to move Logo forward, followed by a numerical value to set the distance moved, e.g. FD 40.
- **Function:** a section of code that makes a specific process happen, and importantly returns a result (or answer).
- **Input:** information given to a computer to make something happen, e.g. a number, a mouse click or button press.
- **Iteration:** the repetition of a command based on the result from the last time the command was used.
- **Iterator:** an object that allows the user to work through a list.
- **List:** a series of connected items.
- **Nesting:** to embed (put) an object inside another, e.g. nesting functions within a spreadsheet.
- **Output:** something that a computer produces when given an instruction, e.g. a number, an on-screen image, a sound or vibration.
- **Parameter:** an amount that can be set by a user.
- **Path:** this allows you to create a 'journey' for your graphic within a piece of software.
- **PD** [in Logo]**:** pen down; the command used to instruct Logo to 'draw' on the screen. What is drawn is set by the other commands input by the user.
- **Procedure:** a small section of code that makes a specific process happen.
- **Program:**
  1. a sequence of instructions written to perform a task or solve a problem, using a programming language (noun)
  2. to create or change a program (verb).
- **REPEAT** [in Logo]**:** the command used to instruct Logo to carry out a command again, or for a certain number of times, e.g. REPEAT 25.
- **RT** [in Logo]**:** the command used to turn Logo right, followed by the angle to turn, e.g. RT 45 means 'turn to the right 45 degrees'.
- **Test:** to run the program or set of commands to check how well (if at all) the code works.
- **Unicode:** a computing language that is used to represent text characters as numbers.
- **User:** the person who uses a program or application.
- **Variable:** a piece of data that changes or can be changed.
- **When** [in Kodu]**:** the command your object must follow for an action to be performed (see 'Do [in Kodu]').

Rising Stars, part of Hodder & Stoughton Ltd, Carmelite House, 50 Victoria Embankment, London, EC4Y 0DZ

www.risingstars-uk.com

Author: Claire Lotriet

Published 2015, reprinted 2019