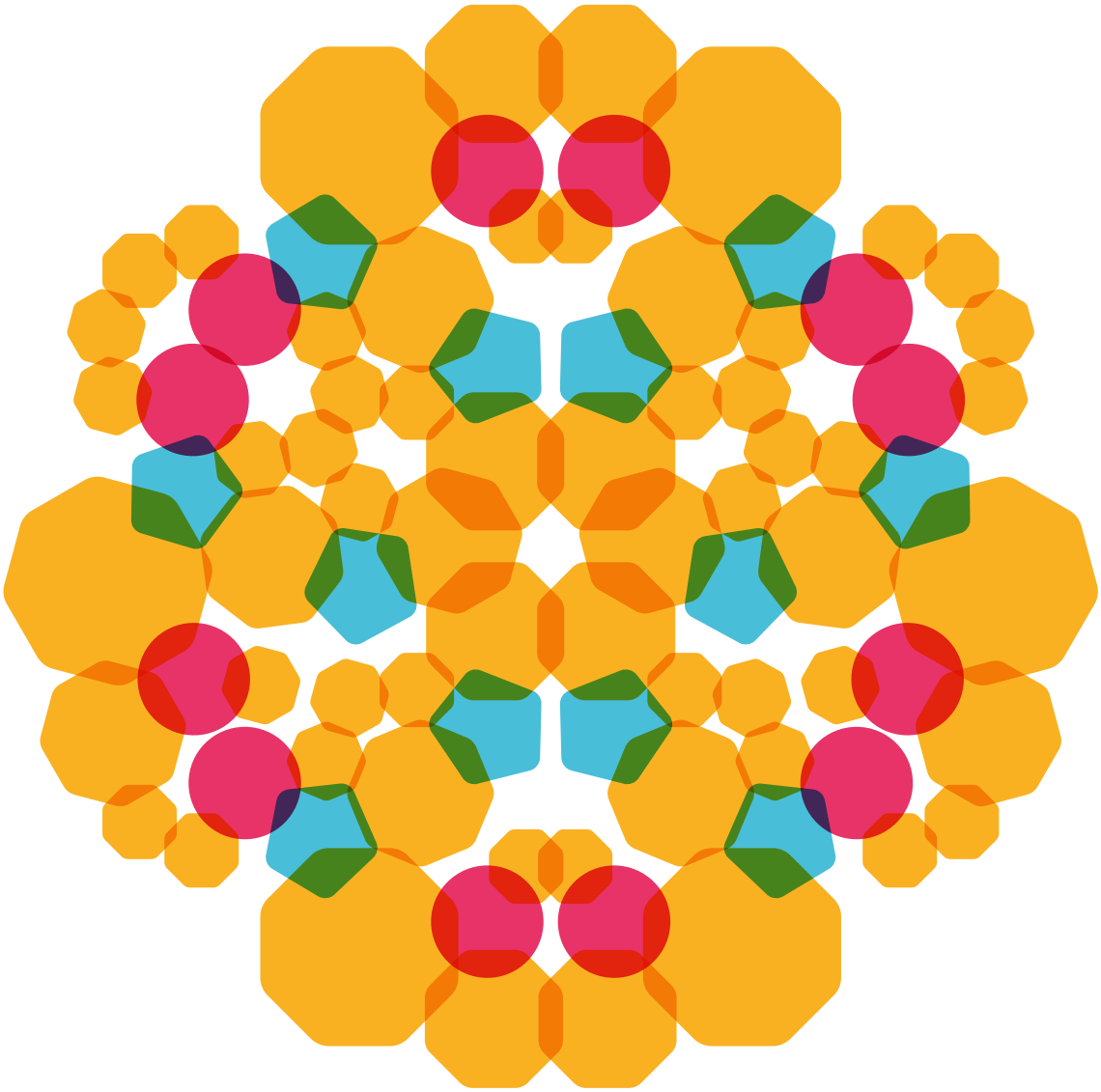
Oxford
*International*
*Curriculum*

# Computing

Subject Guide



OXFORD

## Contents



# I see learners equipped with the skills they need in a rapidly evolving digital world

The Oxford International Curriculum is a new approach to teaching and learning focused on wellbeing, which places joy at the heart of the curriculum and develops the skills your learners need for their future academic, personal and career success.

Computing is one of six subjects that make up the curriculum, part of a coherent and holistic approach that ensures continuity and progression across every student's educational journey.

Four strands encompass the full spectrum of skills and understanding to prepare students for future employment and participation in the digital world, whether or not they are destined to become digital specialists:

- Programming and computational thinking
- Using software for creativity and productivity
- The nature of technology
- Digital literacy

**What does the Oxford International Computing Curriculum offer you?**

- Integrated curriculum materials, continuous professional development, assessment and world-class resources
- A trustworthy and flexible route to equip all learners with the lifelong skills they need to fully engage with the digital world
- A practical framework, designed to be adaptable as technology changes, and to be flexible between communities where different types of technology are in everyday use.



# Curriculum at a glance

The Oxford International Curriculum for Computing offers end-to-end teaching and learning support with year-on-year progression of learning outcomes for every year group.

**Measurable and unambiguous assessment criteria are linked to every learning outcome in the curriculum.**

**The spiral development model means that learning themes are revisited each year, building on previous achievement, and giving coherence and structure to the learning journey.**

| Strand | Year 1 | Year 7 |
|---|---|---|
| | Students can: | Students can: |
| **1 Programming and computational thinking** | **1.1a:** Run and use a simple program made by somebody else<br>**1.1b:** Describe a program by saying what its inputs and outputs are<br>**1.1c:** Edit a program and say how that will change what it does | **7.1a:** Describe how program commands are stored and executed<br>**7.1b:** Use more than one programming language<br>**7.1c:** Write programs in a text-based language<br>**7.1d:** Remove a range of errors to improve a program |
| **2 Productivity and creativity** | **1.2a:** Make simple images using computer software<br>**1.2b:** Enter words and numbers into the computer | **7.2a:** Create digital media<br>**7.2b:** Improve digital media for an audience<br>**7.2c:** Create a single-table data file<br>**7.2d:** Check data input for accuracy |
| **3 The nature of technology** | **1.3a:** Say what a computer is<br>**1.3b:** Say some things that can be done with a computer in school and out of school | **7.3a:** Describe how different types of data can be represented in binary digital form<br>**7.3b:** Convert between decimal and binary integers<br>**7.3c:** Perform simple binary additions |
| **4 Digital literacy** | **1.4a:** Find something out using the computer<br>**1.4b:** Be safe and polite in the computer room<br>**1.4c:** Say who can help you if you are worried | **7.4a:** Use content from online sources responsibly<br>**7.4b:** Explain risks associated with internet use<br>**7.4c:** Discuss how data may be collected when working online |

Sample from Computing Curriculum at a glance, Years 1 and 7

# Assessment framework

End of year tests and practical project papers help teachers assess students' achievement over the course of any full year.

Aligned to the requirements of the computer science examination syllabus, including OxfordAQA's International GCSEs, AS and A-levels.

Built-in projects consolidate and reflect upon learning at the end of each topic of study.

## Year 3

### Introduction

In Year 3, students can draw on developing literacy and numeracy skills to support their use of computers, so they can make more progress and take on bigger challenges.

Learning outcomes can be delivered in any order. Typically, one well-developed computing activity could provide evidence to confirm achievement against multiple outcomes. Students will learn to use computers to find and correct errors, to send and receive messages and to carry out calculations.

### Learning outcomes

These learning outcomes set out a programme of study in computing for Year 3. During the year, every student will:

- **3.1a:** Describe a simple plan for a program that changes inputs into outputs
- **3.1b:** Create a program that produces varied outputs in response to user inputs
- **3.1c:** Find and correct the errors in a program so it works the way they want
- **3.2a:** Use software to improve the appearance of a document that includes text and images
- **3.2b:** Use software to enter number data and make calculations
- **3.3a:** Describe a range of familiar digital devices
- **3.3b:** Describe tasks where computers can be helpful
- **3.4a:** Use technology to send and receive messages
- **3.4b:** Describe the parts of a message
- **3.4c:** Explain how to respond to an unsuitable communication

### Assessment criteria

The assessment criteria allow the teacher to assess the level of achievement of each student.

**3.1a:** *Describe a simple plan for a program that changes inputs into outputs*

**Developing:** The student writes a description of what they want a program to do.

**Secure:** The student makes a written plan of three or four short steps in a correct sequence.
The student makes a plan that includes inputs and outputs.

**Extended:** The student plans a program with several different inputs.

**3.1b:** *Create a program that produces varied outputs in response to user inputs*

**Developing:** The student assembles some program components in sequence.

**Secure:** The student makes a simple working program with inputs and outputs.

**Extended:** The student makes several versions of a program that work in different ways.

**3.1c:** *Find and correct the errors in a program so it works the way they want*

**Developing:** The student finds and removes at least one error from a program.

**Secure:** The student removes all errors from a program, which then works correctly.

**Extended:** The student describes how they found and fixed errors in a program.

**3.2a:** *Use software to improve the appearance of a document that includes text and images*

**Developing:** The student enters text into software such as a word-processing application.

**Secure:** The student uses software tools to format, reorganize and correct a document.

**Extended:** The student produces several versions of a document by varying formatting features.

**3.2b:** *Use software to enter number data and make calculations*

**Developing:** The student enters number values into a software application such as a spreadsheet.

**Secure:** The student enters formulas to produce a calculated result.

**Extended:** The student explains or shows the meaning of calculated results.

**3.3a:** *Describe a range of familiar digital devices*

**Developing:** The student names or indicates some digital devices such as tablets, smartphones and laptops.

**Secure:** The student describes the features and uses of familiar digital devices.

**Extended:** The student evaluates and compares familiar digital devices (for example, for usability or portability).

**3.3b:** *Describe tasks where computers can be helpful*

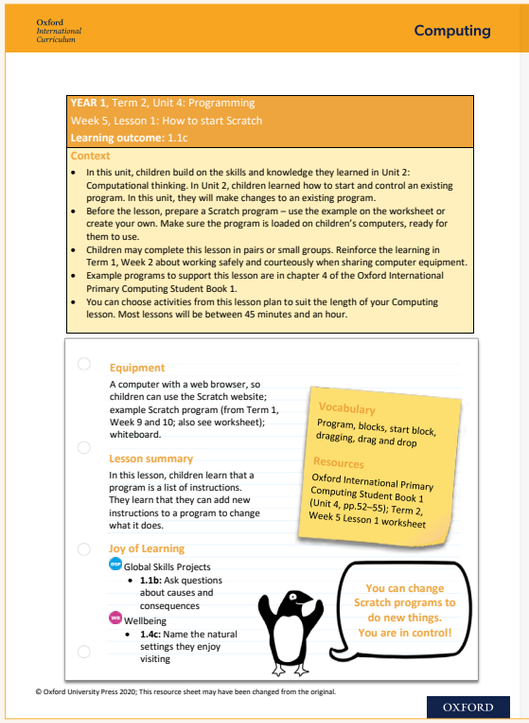
**Developing:** The student identifies at least one task where the computer has helped.

**Secure:** The student describes a range of tasks where computers can help.

**Extended:** The student explains the types of task where a computer may be helpful or less helpful.

Sample from Computing Assessment Framework, Year 3

# Lesson plans and worksheets


Oxford International Curriculum — Computing

**YEAR 1**, Term 2, Unit 4: Programming
Week 5, Lesson 1: How to start Scratch
**Learning outcome:** 1.1c

**Context**
- In this unit, children build on the skills and knowledge they learned in Unit 2: Computational thinking. In Unit 2, children learned how to start and control an existing program. In this unit, they will make changes to an existing program.
- Before the lesson, prepare a Scratch program – use the example on the worksheet or create your own. Make sure the program is loaded on children's computers, ready for them to use.
- Children may complete this lesson in pairs or small groups. Reinforce the learning in Term 1, Week 2 about working safely and courteously when sharing computer equipment.
- Example programs to support this lesson are in chapter 4 of the Oxford International Primary Computing Student Book 1.
- You can choose activities from this lesson plan to suit the length of your Computing lesson. Most lessons will be between 45 minutes and an hour.

**Equipment**
A computer with a web browser, so children can use the Scratch website; example Scratch program (from Term 1, Week 9 and 10; also see worksheet); whiteboard.

**Vocabulary**
Program, blocks, start block, dragging, drag and drop

**Resources**
Oxford International Primary Computing Student Book 1 (Unit 4, pp.52–55); Term 2, Week 5 Lesson 1 worksheet

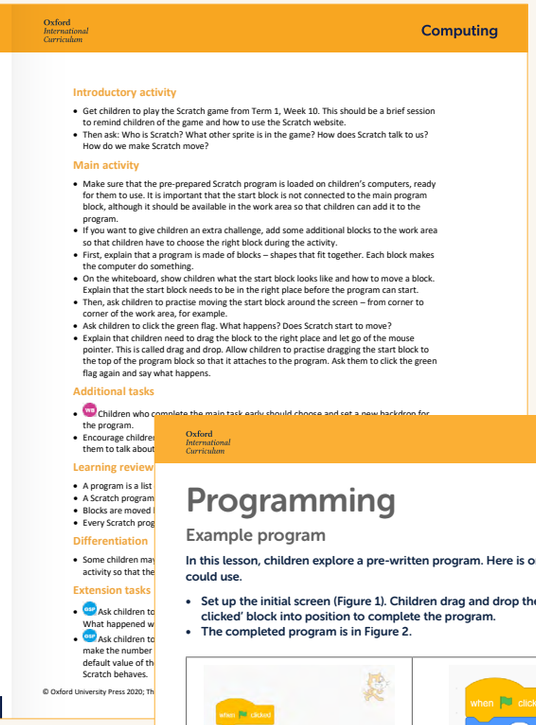
**Lesson summary**
In this lesson, children learn that a program is a list of instructions. They learn that they can add new instructions to a program to change what it does.

**Joy of Learning**
- Global Skills Projects
  - **1.1b:** Ask questions about causes and consequences
- Wellbeing
  - **1.4c:** Name the natural settings they enjoy visiting






OXFORD


Oxford International Curriculum — Computing

**Introductory activity**
- Get children to play the Scratch game from Term 1, Week 10. This should be a brief session to remind children of the game and how to use the Scratch website.
- Then ask: Who is Scratch? What other sprite is in the game? How does Scratch talk to us? How do we make Scratch move?

**Main activity**
- Make sure that the pre-prepared Scratch program is loaded on children's computers, ready for them to use. It is important that the start block is not connected to the main program block, although it should be available in the work area so that children can add it to the program.
- If you want to give children an extra challenge, add some additional blocks to the work area so that children have to choose the right block during the activity.
- First, explain that a program is made of blocks – shapes that fit together. Each block makes the computer do something.
- On the whiteboard, show children what the start block looks like and how to move a block. Explain that the start block needs to be in the right place before the program can start.
- Then, ask children to practise moving the start block around the screen – from corner to corner of the work area, for example.
- Ask children to click the green flag. What happens? Does Scratch start to move?
- Explain that children need to drag the block to the right place and let go of the mouse pointer. This is called drag and drop. Allow children to practise dragging the start block to the top of the program block so that it attaches to the program. Ask them to click the green flag again and say what happens.

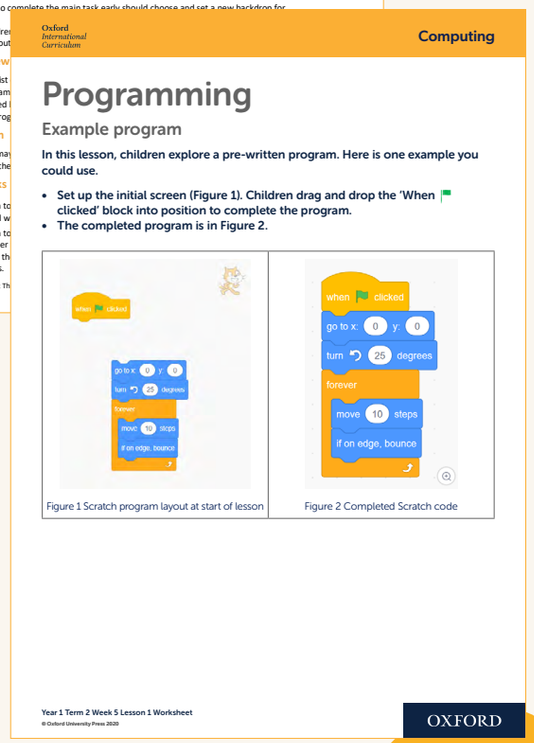
**Additional tasks**
- WB Children who complete the main task early should choose and set a new backdrop for the program.
- Encourage children to talk about

**Learning review**
- A program is a list
- A Scratch program
- Blocks are moved
- Every Scratch prog

**Differentiation**
- Some children ma activity so that the

**Extension tasks**
- ELAP Ask children to What happened w
- ELAP Ask children to make the number default value of th Scratch behaves.




Oxford International Curriculum — Computing

## Programming

### Example program

**In this lesson, children explore a pre-written program. Here is one example you could use.**
- **Set up the initial screen (Figure 1). Children drag and drop the 'When clicked' block into position to complete the program.**
- **The completed program is in Figure 2.**



Figure 1 Scratch program layout at start of lesson

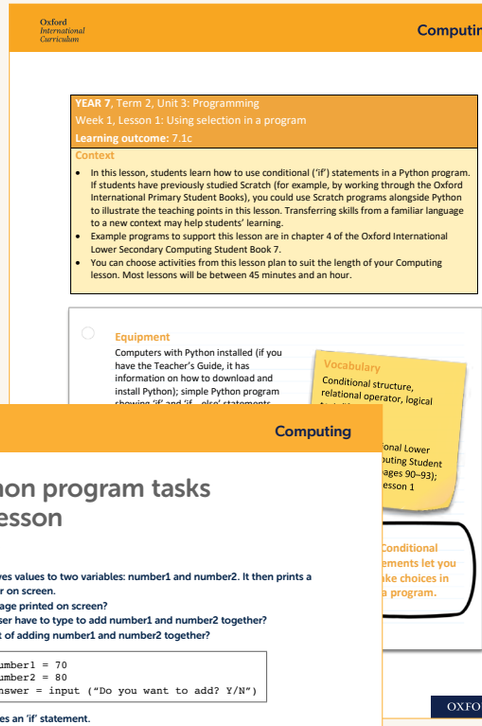


Figure 2 Completed Scratch code

Year 1 Term 2 Week 5 Lesson 1 Worksheet
© Oxford University Press 2020

OXFORD

YEAR 1, Term 2, Unit 4: Programming


Oxford International Curriculum — Computing

**YEAR 7**, Term 2, Unit 3: Programming
Week 1, Lesson 1: Using selection in a program
**Learning outcome:** 7.1c

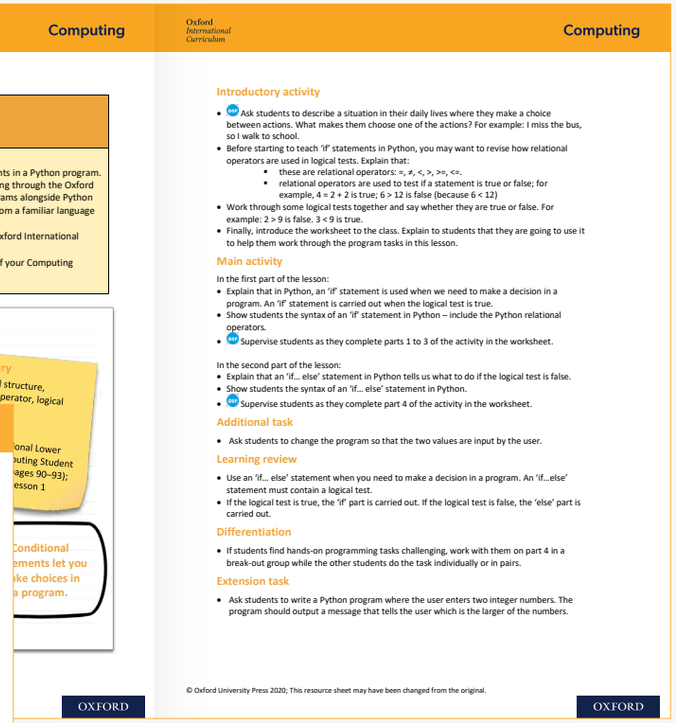
**Context**
- In this lesson, students learn how to use conditional ('if') statements in a Python program. If students have previously studied Scratch (for example, by working through the Oxford International Primary Student Books), you could use Scratch programs alongside Python to illustrate the teaching points in this lesson. Transferring skills from a familiar language to a new context may help students' learning.
- Example programs to support this lesson are in chapter 4 of the Oxford International Lower Secondary Computing Student Book 7.
- You can choose activities from this lesson plan to suit the length of your Computing lesson. Most lessons will be between 45 minutes and an hour.

**Equipment**
Computers with Python installed (if you have the Teacher's Guide, it has information on how to download and install Python); simple Python program

**Vocabulary**
Conditional structure, relational operator, logical

onal Lower puting Student ages 90–93); esson 1

Conditional ements let you ke choices in a program.


Oxford International Curriculum — Computing

**Introductory activity**
- ELAP Ask students to describe a situation in their daily lives where they make a choice between actions. What makes them choose one of the actions? For example: I miss the bus, so I walk to school.
- Before starting to teach 'if' statements in Python, you may want to revise how relational operators are used in logical tests. Explain that:
  - these are relational operators: =, ≠, <, >, >=, <=.
  - relational operators are used to test if a statement is true or false; for example, 4 = 2 + 2 is true; 6 > 12 is false (because 6 < 12)
- Work through some logical tests together and say whether they are true or false. For example: 2 > 9 is false. 3 < 9 is true.
- Finally, introduce the worksheet to the class. Explain to students that they are going to use it to help them work through the program tasks in this lesson.

**Main activity**
In the first part of the lesson:
- Explain that in Python, an 'if' statement is used when we need to make a decision in a program. An 'if' statement is carried out when the logical test is true.
- Show students the syntax of an 'if' statement in Python – include the Python relational operators.
- ELAP Supervise students as they complete parts 1 to 3 of the activity in the worksheet.

In the second part of the lesson:
- Explain that an 'if... else' statement in Python tells us what to do if the logical test is false.
- Show students the syntax of an 'if... else' statement in Python.
- ELAP Supervise students as they complete part 4 of the activity in the worksheet.

**Additional task**
- Ask students to change the program so that the two values are input by the user.

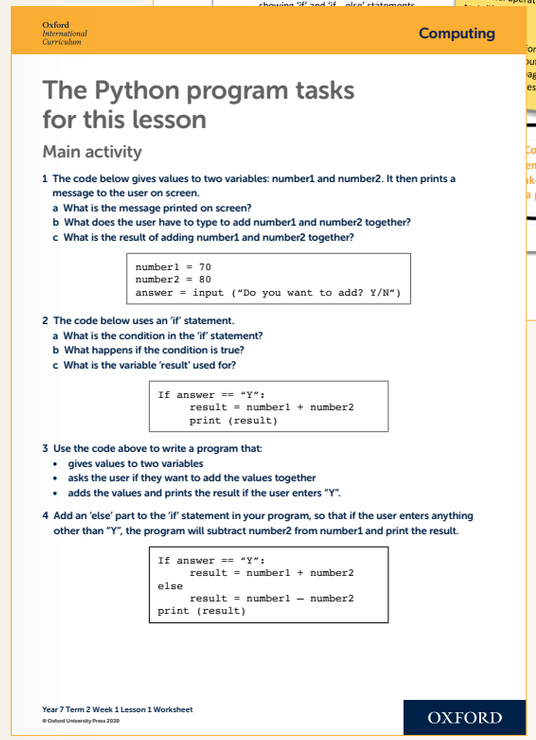
**Learning review**
- Use an 'if... else' statement when you need to make a decision in a program. An 'if...else' statement must contain a logical test.
- If the logical test is true, the 'if' part is carried out. If the logical test is false, the 'else' part is carried out.

**Differentiation**
- If students find hands-on programming tasks challenging, work with them on part 4 in a break-out group while the other students do the task individually or in pairs.

**Extension task**
- Ask students to write a Python program where the user enters two integer numbers. The program should output a message that tells the user which is the larger of the numbers.



OXFORD


Oxford International Curriculum — Computing

## The Python program tasks for this lesson

### Main activity

**1 The code below gives values to two variables: number1 and number2. It then prints a message to the user on screen.**
**a What is the message printed on screen?**
**b What does the user have to type to add number1 and number2 together?**
**c What is the result of adding number1 and number2 together?**

```
number1 = 70
number2 = 80
answer = input ("Do you want to add? Y/N")
```

**2 The code below uses an 'if' statement.**
**a What is the condition in the 'if' statement?**
**b What happens if the condition is true?**
**c What is the variable 'result' used for?**

```
If answer == "Y":
      result = number1 + number2
      print (result)
```

**3 Use the code above to write a program that:**
- **gives values to two variables**
- **asks the user if they want to add the values together**
- **adds the values and prints the result if the user enters "Y".**

**4 Add an 'else' part to the 'if' statement in your program, so that if the user enters anything other than "Y", the program will subtract number2 from number1 and print the result.**

```
If answer == "Y":
      result = number1 + number2
else
      result = number1 – number2
print (result)
```

Year 7 Term 2 Week 1 Lesson 1 Worksheet
© Oxford University Press 2020

OXFORD

YEAR 7, Term 2, Unit 3: Programming

**Every lesson highlights the learning outcomes it covers, linking back to the curriculum-at-a-glance document.**

**Step-by-step guidance navigates through the delivery of the lesson, with differentiation suggestions provided.**

**Opportunities to link to the Global Skills Projects and Wellbeing curricula are highlighted.**

**Includes links to recommended resources and worksheets where relevant.**

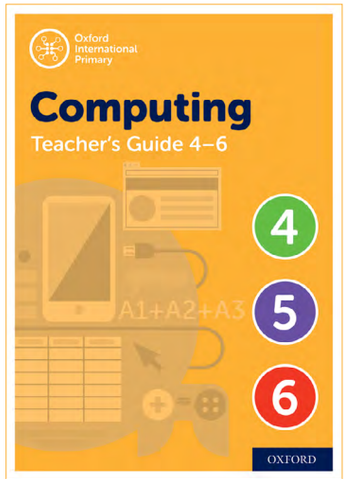
Download sample lesson plans and worksheets at
**oxfordinternationalcurriculum.com**

# Resources

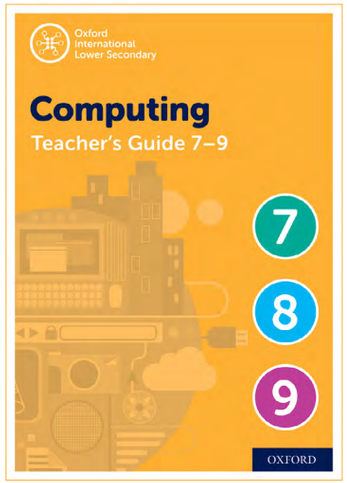
We recommend that schools use the Oxford International Primary Computing and Oxford International Lower Secondary Computing series to support the implementation of the Oxford International Curriculum for Computing.

With a structured progression and a project-based approach to learning, this computing series provides a complete and integrated computing course for Years 1–9, building digital literacy while giving students the confidence to apply their knowledge and skills to real-life situations.

The Oxford International Curriculum for Computing schemes of work and individual lesson plans are matched to the Student Book for each year group.






## Find out more at

oxfordinternationalcurriculum.com

OXFORD
UNIVERSITY PRESS

Oxford Education www.youtube.com/OxfordEducation **Blog:** educationblog.oup.com
**www.oxfordinternationalcurriculum.com**

K45806