CURRICULUM ALIGNMENT GUIDE

Computing programmes of study:
KEY STAGE 3 AND 4
National curriculum in England

OVERVIEW

100 Ideas for Secondary Teachers:

Outstanding Computing Lessons

INTRODUCTION

100 ideas: Outstanding Computing Lessons is a collection of 100 practical, tried-and-tested ideas for teaching computing. It is aimed at computing / ICT teachers of all levels, whether specialist or non-specialist, newly qualified or experienced.

For more information on 100 Ideas: Outstanding Computing Lessons and to find additional education resources and supporting materials, including more than 50 worksheets to accompany the activities in the book, visit: teachwithict.com/100ideas

10 sample activities can be downloaded for free at teachwithict.com/bonus

KEY STAGE 3

3.1 Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.

3.2 Understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem.

3.3 Use 2 or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions.

3.4 Understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal].

3.5 Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems.

3.6 Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits.

3.7 Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.

3.8 Create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability.

3.9 Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct, and know how to report concerns.

KEY STAGE 4

4.1 Develop their capability, creativity and knowledge in computer science, digital media and information technology.

4.2 Develop and apply their analytic, problem-solving, design, and computational thinking skills.

4.3 Understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to report a range of concerns.

PART 1: PROGRAMMING STRATEGIES

IDEA	DESCRIPTION	STANDARDS
001	**Paired programming** – A research-driven coding strategy for helping novice learners.	4.2
002	**Rubber duck debugging** – A programming strategy used to help students find bugs and in their code.	4.2
003	**Code golf** – A programming strategy to help students create more efficient code.	3.3, 4.2
004	**Game design** – Using games as a hook to encourage students to learn how to code.	4.2
005	**PRIMM** – A research-based approach to teaching coding and for reducing cognitive load.	4.2
006	**Parsons problems** – Help students learn how to code by removing some of the barriers.	4.2
007	**Use-modify-create** – Reduce anxiety while supporting growth with this simple three-stage approach to learning to code.	4.2
008	**Hour of Code** – Give your students a 'byte'-sized introduction to computer science with an hour of code.	4.2
009	**Code bug** – Build resilience and reduce anxiety when teaching children to code by purposely introducing 'bugs' early on in the learning process.	4.2
010	**Code combat** – Put your students' coding skills to the test by pitting them against each other in code combat!	4.2
011	**Teaching with robots** – Coding can often be difficult for students to grasp. Robots can provide a simpler, more tangible introduction to programming.	4.2

PART 2: COMPUTING STRATEGIES

IDEA	DESCRIPTION	STANDARDS
012	**Take your screwdrivers to work** – Students explore how computers work by taking old devices apart.	3.5
013	**DART your students** – A strategy designed to improve students' reading comprehension.	4.1
014	**Contextualise learning** – Explore strategies for making computing relevant and provide 'real-life' learning experiences for students.	3.1
015	**Go unplugged** – Teaching computing without computers.	4.1
016	**Socratic debate** – Students debate the social, ethical, and legal issues surrounding the use of computers.	4.3
017	**Peer instruction** – A research-driven approach to teaching difficult concepts that students often misunderstand.	4.1
018	**Game-based learning** – Exploring the use of games, such as Minecraft: Education Edition, to teach children how to code.	3.1, 3.3
019	**Using QR codes** – Using QR codes to teach computing theory.	3.7, 3.8
020	**Escape rooms** – Students must solve a series of binary puzzles to open physical locks and stop a simulated 'virus attack'.	4.1, 4.3
021	**Blogs and wikis** – Using blogs and wikis to teach computing theory.	3.7
022	**Flipped learning** – Reversing the traditional way of teaching to make better use of classroom time.	3.7
023	**Guided discovery** – An inductive approach to teaching and learning where students take an active role in discovering knowledge and developing understanding for themselves.	3.7, 4.1

PART 3: ICT AND DIGITAL LITERACY

IDEA	DESCRIPTION	STANDARDS
024	**Fake news** – Students learn how to identify 'fake news' articles before creating their own fake news story.	3.9, 4.3
025	**Copycat** – A fun activity that teaches students about copyright, public domain, fair use, and Creative Commons.	3.9, 4.3
026	**Mario Kart ™ spreadsheets** – An example of how to use game-based learning to teach students essential spreadsheet skills.	3.7, 3.8
027	**Fakebook** – An 'escape room' challenge which helps students understand the importance of protecting their online presence.	3.9, 4.3
028	**Database detectives** – Students test their sleuthing skills with this 'whodunnit' themed database challenge.	4.1
029	**Did you meme it?** – Students explore the purpose and ethics of memes before creating their own meme on an agreed topic.	3.9, 4.3
030	**Videography** – Students create a YouTube-style instructional video whilst also explore the importance of concise instructions (algorithms).	3.7, 3.8
031	**Infographics** – Students create infographics about their mobile phone habits.	3.7, 3.8
032	**Dragon's Den** – Students work as a team to design an innovative solution to a global problem.	3.7, 3.8
033	**Wayback Machine** – Students learn about their digital footprint and the long-lasting impact of their online actions.	3.9, 4.3

PART 4: COMPUTING ACTIVITIES

IDEA	DESCRIPTION	STANDARDS
034	**What a waste** – Students, working in teams, explore innovative ways to reduce e-waste.	2.6
035	**Role reversal** – Students take on the role of a teacher.	4.1
036	**Storage Top Trumps®** – Students explore different storage devices before creating a game of Top Trumps® based on what they have learned.	3.5
037	**Little Man Computer** – Students explore 'Little Man Computer' – a simulator that models the basic features of a modern computer that uses Von Neumann architecture.	3.6
038	**Features of a CPU (a lesson using DART)** – Students explore the main features of a CPU.	3.5
039	**Internet of things** – Students design a 'smart home' that utilises the internet of things.	3.5
040	**The great input/output QR hunt** – Students complete a QR hunt to discover facts about different input and output devices.	3.5
041	**Moral machine** – Students explore the ethics behind creating AI for self-driving vehicles.	3.9, 4.2

PART 5: COMPUTATIONAL THINKING

IDEA	DESCRIPTION	STANDARDS
042	**Making the tea algorithm** – Students explore the importance of creating precise instructions by creating an algorithm for making a cup of tea / coffee.	3.1, 4.2
043	**Teaching with magic** – Using magic to teach computational thinking skills.	3.1, 4.2
044	**Crazy characters** – Students write an algorithm for drawing a monster.	3.1, 4.2
045	**Puzzle me** – Using puzzles to practise computational thinking skills (decomposition, pattern-matching, abstraction and algorithm design).	2.1, 2.3
046	**Human robot** – Exploring algorithms through physical activities such as movement and dance.	3.1, 4.2
047	**A-maze-ing algorithms** – Students explore the importance of clear and precise instructions by writing algorithms to solve a simple maze.	3.1, 4.2
048	**20 questions** – Students explore the efficiency of different search algorithms by playing a game of '20 questions'.	4.1, 4.2, 3.2
049	**Breaking the code** – Students develop their problem-solving skills with a series of code-breaking challenges.	4.1, 4.2
050	**Origami algorithms** – Students write algorithms for folding a paper aeroplane or origami animal.	3.1, 4.2
051	**Guess the object** – Getting students to model, draw or mime a variety of different objects can help them to understand the concept of abstraction.	3.2

PART 6: UNPLUGGED ACTIVITIES

IDEA	DESCRIPTION	STANDARDS
052	**Image compression** – Students learn about lossless compression without the use of a computer.	3.6
053	**Bubble sort dance algorithm** – Students learn how a bubble sort algorithm works is via the medium of Hungarian folk dance!	3.2
054	**World Wide Web unplugged** – Students role-play what happens when a user enters an address in a web browser.	3.5
055	**Intelligent piece of paper (AI)** – Exploring artificial intelligence (AI) with a game of Tic-Tac-Toe.	4.1
056	**Envelope variables** – Demonstrate a simple program that uses variables and assignment by running them on a computer made entirely out of students.	4.1
057	**Card sort** – Students explore three common sorting algorithms (bubble, merge, and insertion) by sorting playing cards.	3.2
058	**Binary representation of images (unplugged)** – Students explore how a computer represents images using binary.	3.6
059	**How computers work** – Students take on the role of various parts of a computer and simulate the running of a simple program.	3.6
060	**Memory unplugged** – Students explore how data is transferred between different storage locations inside a computer, such as RAM, cache memory, secondary storage and virtual memory.	3.5, 3.6
061	**Network topologies** – Using string and various other household objects, students simulate the three most common network topologies.	3.5

PART 7: DATA REPRESENTATION

IDEA	DESCRIPTION	STANDARDS
062	**Binary addition** – Students learn how to add two numbers using binary.	3.4, 3.6
063	**Binary numbers** – Students learn about binary.	3.4, 3.6
064	**Binary representation of images** – Students explore how a computer represents images using binary.	3.6
065	**Binary representation of sound** – Students explore how a computer represents sound using binary.	3.6
066	**Binary bingo** – A fun strategy to test students' understanding of binary representation of numbers.	3.4, 3.6
067	**It's all about hex** – Students learn about the hex numbering system.	3.4, 3.6
068	**ASCII 'secret' agent** – Students explore how a computer represents text using binary by solving (and creating) as series of coded messages.	3.6

PART 8: EXAM PREPARATION

IDEA	DESCRIPTION	STANDARDS
069	**Padlet** – Using online curation tools, such as Padlet, to collating resources in preparation for exams.	3.7, 3.8
070	**Round-robin revision** – Make revision fun and engaging with a series of mini games.	4.1
071	**Revision podcasts** – Create revision resources that students can listen to anytime, anywhere!	3.7, 3.8
072	**PEE (point, evidence, explain)** – A simple strategy to help improve the quality of written answers to exam questions.	4.1
073	**PechaKucha** – A great way to encourage students to be more concise and a little more creative with their presentations.	3.7, 3.8
074	**Sketch-noting** – A great way to empower students and allow them to synthesise information visually.	4.1
075	**Command word bingo** – A simple starter activity that will pay dividends at exam time!	4.1
076	**BUG hunt** – A technique for helping students understand thoroughly what is expected of them during exams.	4.1
077	**Tweet IT** – A fun revision strategy that will help students to remember key information.	4.1
078	**Revision speed dating** – A fun and engaging activity that gets students talking.	4.1
079	**Match IT** – Make revision engaging and memorable by turning it into a game!	4.1

PART 9: PROGRAMMING ACTIVITIES

IDEA	DESCRIPTION	STANDARDS
080	**Magic 8-ball®** – Students create a Magic 8-ball® game using python.	3.1, 3.3, 4.2
081	**Shakespearean insult generator** – A fun way to introduce lists and file-handling in python.	3.1, 3.3, 4.2
082	**Chatting robot** – Students learn how to create a 'rule-based' chat bot using python.	3.1, 3.3, 4.2
083	**Just dance** – A lesson which uses dance as a medium for introducing key programming concepts to children.	2.1, 2.2, 2.3
084	**Adventures in text** – Students learn how to create an 80s-style text adventure game in python.	3.1, 3.3, 4.2
085	**Mad Libs®** – Students code the popular phrasal template word game in python.	3.1, 3.3, 4.2
086	**Sorting Hat** – Students create a Harry Potter-style sorting hat in python.	3.1, 3.3, 4.2
087	**Turtle power (a lesson using PRIMM)** – Students learn how to create regular polygons using the turtle library in python.	3.1, 3.3, 4.2
088	**Guess my number** – A fun programming challenge which teaches concepts such as variables, data types and selection.	3.1, 3.3, 4.2
089	**Mind-reading algorithm** – Students learn how to create a mind-reading game in python.	3.1, 3.3, 4.2
090	**Cat and mouse** – A simple cat and mouse game using Scratch.	3.1, 3.3, 4.2
091	**Reaction timer** – Students create a simple reaction timer using python.	3.1, 3.3, 4.2

PART 10: COMPUTING AND STEAM

IDEA	DESCRIPTION	STANDARDS
092	**Art attack** – Using art as a creative medium for exploring complex concepts in computing.	4.1, 4.2
093	**Lights, camera, action** – Students learn how to create light art using slow shutter speed photography and code.	3.1, 3.3, 3.7, 4.2
094	**Making music** – Students learn how to create music with code.	3.1, 3.3, 4.2
095	**Coding probability** – Students explore probability, including relative frequency, with code.	3.1, 3.3, 4.2
096	**Physical computing** – Exploring how to teach coding using physical devices.	3.1, 3.3, 4.2
097	**Turtle snowflakes** – Students learn how to code snowflakes using the turtle library in python.	3.1, 3.3, 4.2
098	**Coding the weather** – Students learn how to manipulate 'real' weather data using python and OpenWeather data.	3.1, 3.3, 4.2
099	**Rubbish robots** – Students are challenged to build a robot using general household objects and electronic components.	3.1, 3.3, 4.2
100	**Color splash** – Students learn how to create colour splash images using a free online image editor.	4.1, 4.2

PART 11: BONUS ACTIVITIES

IDEA	DESCRIPTION	STANDARDS
101	**What's your elf name?** – Students create a name generator using python.	3.1, 3.3, 4.2
102	**Guess / _ _e / word** – Students create a hangman style word game using python.	3.1, 3.3, 4.2
103	**Cards against humanities** – Students code a phrasal template word game in python.	3.1, 3.3, 4.2
104	**Shakespearean complement generator** – Coding challenge based on the Shakespearean sonnet 'Shall I compare thee to a summer's day?'	3.1, 3.3, 4.2
105	**Hacking the news** – Hacking the news with HTML.	3.1, 3.3, 3.7, 4.2
106	**Data science detectives** – Teaching computational thinking using historical data.	3.1, 4.2
107	**Code tracing** – A simple strategy for reducing cognitive load.	3.1, 4.2
108	**Make me happy** – Students create an AI powered sentiment analysis bot using Scratch.	3.1, 3.3, 4.2
109	**Worked examples** – An effective strategy for reducing cognitive load for novice learners.	3.1, 4.2
110	**Team teaching** – Tips for planning a lesson with a colleague.	4.1