Computer Science

The Ripley Academy

Curriculum Intent Overview

At Ripley Academy we offer the OCR GCSE in Computer Science at Key Stage 4. This is where our intent begins as we have plan to prepare students for an opportunity to engage in either strand come Key Stage 4.

"The intention is simple. We aim to support and offer all students the chance to become the best computational thinkers they can be. By strengthening their understanding of the links between the hardware architecture and efficiency of an algorithm, our students will aim to become creative and logical thinkers who have the ability to articulate the relationship between application use, programming and the performance of the machine, whilst also having a plethora of experience allowing them to express their creativity"

Computer Science Strand

Computer Science is firstly the study of computational thinking and logic. However, our curriculum intends to give our students a wider understanding of how the topics within this subject area connect. One's ability to think computationally can be expressed via an ability to solve problems using precise instructions. Therefore, we explore and learn computational thinking via exposing our students to different programming languages and problems. Programming is at the heart of our Key Stage 3 Computer Science strand. We study programming to help us think in a more logical way, however, to gain a wider understanding of why we need to be efficient lends us to connect the ability to link efficient solutions to the architectural design of the computer itself. Although we intend to support the development of our students as computational thinkers and logicians it is important that we continue to explore the hardware in order to gain a deeper understanding of how an efficient design impacts on performance. Exploring how we eventually lead us to how we arrived at the world we live in today. A world in which this hardware is connected via computer networks. Our learning journey continues as we explorer how these hardware connection work as well as the impact and issues that have arisen as a result of its development.

Creative Computing Strand

The development of computer science would be worthless, if it wasn't for the ability to use the technology creatively to make an impact on the world around us. Our second strand of development that flows within our curriculum is that of Creative Computing. Creative Computing allows our students to use the hardware and software to undertake creative projects via multiple applications in order to meet the needs of users. The Creative strand is the foundation to our support those students who will utilise their IT skills in future ventures which lie away from computational thinking. This strand offers students the opportunity to express their creatively and individuality. It supports their understanding of the computer as a usable system, as we expose them to different interfaces and applications, supporting their organisational and digital literacy skills. As a key element of the National Curriculum, Digital Literacy is embedded into both strands but directly focused on in the Creative Computing Strand. Student we be offered to learn and express skills in digital animation, computer graphics, audio development and interactive media. During the study of these areas, students will use the best commercially available software to design and develop solutions for different audience and purposes. Our curriculum design intends to interleave back into the creative strand the understanding of how the computer represents the data within multimedia.

We endeavour to make both sides of our curriculum as fun and as interesting as possible, coupled with a high level of challenge, in order to excite and engage our students in their learning. Our aim is to ensure that students develop a computing capability that is directly transferable, not only to other subjects but also to the KS4 curriculum and beyond.

Both strands of our curriculum map into the Secondary National Curriculum and are subject to an on-going and rigorous review process. Using our analysis of data and our continuing strive to improve teaching and learning, we endeavour to ensure that the curriculum is effective in meeting the needs of all students.

"In Computer Science students follow a meticulously planned and sequenced curriculum, designed to simply to allow them to learn more. Our strategy helps them remember more by building on previous knowledge, with the hope that their achievements will be reflected in their qualification attainment"

Year 7

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 1 – Digital Literacy	You will explore the basic foundations of digital literacy and get to grips with their new school network.	• Basic hardware understanding • Basic hardware use • Understanding of secure password and username needs	• Make use of the school network for day to day computational tasks • Awareness of basic formatting and word processing. • Make use of web browser facilities and an awareness of online safety
Unit 2 – Under the Hood	You will begin to learn about what exactly is inside of a computer and how the different parts work together to process data	• To be able to name basic input and output devices. • An understanding that the computer is a machine used to process data • An understanding of input, process and output.	• To name the components within the computer • To be able to explain how each of these devices interact • What affects the performance of the computer?
Unit 3 – Algorithms	You will drive into the world of computational thinking. An algorithm is a plan, a set of step-by-step instructions to solve a problem. If you can tie shoelaces, make a cup of tea, get dressed or prepare a meal then you already know how to follow an algorithm.	• To follow a basic instruction set • To be able to identify decisions in your everyday life • To understand the CPU and RAM and each use in a computer	• To be able to write a set of precise instructions to complete the task • To translate this into a flowchart • To understand the importance of pattern recognition and sub programs.
Unit 4 – Visual Programming	Discover the power of visual programming languages. Instead of writing code, you'll use visual elements and blocks to create programs. Learn to design intuitive user interfaces, create animations, and develop interactive applications	•	•

	by connecting blocks together. Unlock the potential of coding through a graphical approach.		
Unit 5 – P5.JS	Get hands-on with creative coding using P5.JS, a JavaScript library. You'll explore visual and interactive programming, creating animations, games, and interactive web experiences. P5.JS offers a simple syntax and powerful features, allowing you to bring your artistic and computational ideas to life in the browser.	•	•
Unit 6 – E-Safety	Explore the importance of online safety and responsible digital citizenship. Learn about potential online risks, such as cyberbullying, identity theft, and phishing. Understand how to protect personal information, navigate social media responsibly, and engage in safe online communication. Develop strategies to promote a secure and ethical online presence.	•	•

Year 8

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 7 - Visual Programming Intermediate	Take your visual programming skills to the next level. Build upon the basics and delve into more advanced concepts and techniques. Explore topics like event-driven programming, object-oriented design, and advanced user interface development. Create visually appealing and interactive applications using tools like Scratch.	•	•
Unit 8 - Graphics and Ethics	Examine the ethical considerations surrounding graphics and visual media. Explore the impact of digital manipulation, image rights, and copyright infringement. Discuss the ethical implications of using graphics in advertising, media, and entertainment. Analyse the role of graphic designers in promoting inclusive and ethical practices. Gain a deeper understanding of the ethical responsibilities in the creation and use of visual content.	•	
Unit 9 - Python Programming	Dive into the versatile world of Python programming. Learn the fundamentals of Python syntax, data types, control structures, and functions. Explore object-oriented programming concepts		•

	and modular design principles. Develop skills in file handling, exception handling, and working with libraries and modules. Gain practical experience in solving problems and building applications using the Python programming language.		
Unit 10 - Computer Logic	Discover the foundations of computer logic and digital circuits. Explore Boolean algebra, logic gates, and truth tables. Learn how to design and analyse combinational and sequential circuits. Dive into topics such as binary arithmetic, memory units, and computer organization. Gain a deeper understanding of how computers process and manipulate information at the fundamental level of logic.	•	•
Unit 11 - Python Intermediate	Take your Python programming skills to the next level. Explore more advanced topics such as data structures, loops and algorithms. Develop proficiency in handling exceptions, working with files, and interacting with databases. Enhance your problem-solving abilities through hands-on projects and coding challenges.	•	•

Year 9

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 12 - HTML and Web	Delve into the world of web development with HTML. Learn the fundamentals of HTML markup, including tags, attributes, and elements. Explore the structure and organization of web pages, creating layouts, adding multimedia content, and styling with CSS. Discover responsive design principles, accessibility considerations, and best practices for creating user-friendly websites. Gain hands-on experience in building and publishing web pages.		
Unit 13 - Cyber Security	Explore the field of cybersecurity and the measures to protect computer systems and networks from unauthorized access, attacks, and data breaches. Learn about different types of cyber threats, such as malware, phishing, and social engineering.		
Unit 14 – Sound Production	Immerse yourself in the world of sound production and audio engineering. Learn the principles of sound, including frequency and amplitude. Explore recording techniques and mixing strategies. Dive into topics such as sound editing, effects processing, and sound synthesis. Develop skills in manipulating audio to create professional-quality sound productions.		

Unit 15 – Animation	Step into the captivating realm of animation. Learn the principles of animation, including timing, spacing, and movement. Explore different animation techniques, such as traditional hand-drawn animation, stop motion, and computer-generated animation. Develop storytelling skills and create engaging animations that entertain and communicate effectively.		
Unit 16 – P5.JS Intermediate	Build upon your knowledge of P5.JS and further explore the creative possibilities of this JavaScript library. Dive deeper into advanced topics such as interactivity, animation, and responsive design. Learn to work with external data sources, create interactive visualizations, and integrate multimedia elements. Explore more complex coding concepts and techniques to enhance your ability to create engaging and interactive web experiences using P5.JS.		

GCSE Computer Science

OCR's GCSE in Computer Science is both engaging and practical, encouraging creativity and problem solving. It encourages you to develop your understanding and application of the core concepts in computer science. You will also analyse problems in computational terms and devise creative solutions by designing, writing, testing and evaluating programs. OCR's GCSE (9–1) in Computer Science consists of two compulsory components that are externally assessed.

Component 01: Computer systems
Introduces you to the central processing unit (CPU), computer memory and storage, data representation, wired and wireless networks, network topologies, system security and system software. It also looks at ethical, legal, cultural and environmental concerns associated with computer science.

- This is a compulsory component.
- It is worth 80 marks, representing 50% of the total marks for the GCSE (9–1).
- This component is an externally assessed written examination testing AO1 and AO2.
- The examination lasts 1 hour 30 minutes.
- All the questions are mandatory.
- The question paper will consist of short and medium answer questions. There will also be one 8-mark extended response question. This question will enable students to demonstrate the ability to construct and develop a sustained line of reasoning.

Component 02: Computational thinking, algorithms and programming
You apply knowledge and understanding gained in component 01. You will develop skills and understanding in computational thinking: algorithms, programming techniques, producing robust programs, computational logic and translators.

- This is a compulsory component.
- It is worth 80 marks, representing 50% of the total marks for the GCSE (9–1).
- This component is an externally assessed written examination testing AO1, AO2 and AO3.
- The examination lasts 1 hour 30 minutes and is formed of two sections.
- All the questions are mandatory.
- Section A is worth 50 marks, and assesses students' knowledge and understanding of concepts of Computer Science. Students then apply these to problems in computational terms, where they may use an algorithmic approach.
- Section B is worth 30 marks, and assesses students' Practical Programming skills and their ability to design, write, test and refine programs.

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 1 - Inside the Computer	Explore the inner workings of a computer system. Learn about the components that make up a computer, including the CPU, memory, storage, and peripherals. Understand how data is represented and processed within the computer. Dive into topics such as binary representation, Boolean logic, and the fetch-decode-execute cycle. Gain insight into the architecture and operation of computers at a fundamental level.		
Unit 2 - The workings of the CPU	Dive into the intricate workings of the Central Processing Unit (CPU). Learn about the components and functions of the CPU, including the control unit, arithmetic logic unit (ALU), and registers. Explore the fetch-decode-execute cycle and understand how instructions are executed. Gain insights into concepts such as clock speed, cache memory, and pipelining. Develop an understanding of how the CPU interacts with other components of a computer system.		
Unit 3 - Binary Logic	Explore the fundamentals of binary logic and its applications in computing. Learn about Boolean algebra and logic gates, including AND, OR, and NOT gates. Understand how these gates are combined to create more complex circuits. Explore truth tables and logical operations. Dive		

	into topics such as logic gate diagrams, Boolean expressions, and simplification techniques. Develop skills in designing and analysing digital circuits using binary logic.		
Unit 4 - Data Representation	Gain an understanding of how data is represented and stored in computer systems. Explore different number systems, including binary, decimal, and hexadecimal. Learn about data formats such as ASCII and Unicode for representing characters. Dive into topics like binary arithmetic, two's complement representation, and fixed-point and floating-point numbers. Understand the concept of data compression and its impact on storage efficiency. Develop skills in converting and manipulating data representations in computer systems.		
Unit 5 - Algorithms	Dive into the world of algorithms and computational thinking. Learn the fundamental concepts and techniques used to solve problems algorithmically. Understand algorithm design strategies such as iteration, recursion, and divide-and-conquer. Explore algorithm efficiency and analysis, including Big O notation. Learn about searching and sorting algorithms, data structures, and algorithmic problem-solving techniques. Develop skills in designing, implementing, and analysing algorithms to solve real-world problems.		

Unit 6 - Computational Thinking	Develop essential computational thinking skills. Understand the core principles and strategies used to tackle complex problems. Explore abstraction, decomposition, pattern recognition, and algorithmic thinking. Learn to break down problems into smaller, manageable parts and develop step-by-step solutions. Apply computational thinking techniques to a variety of contexts, such as data analysis, simulations, and automation. Develop critical thinking and problem-solving abilities that are applicable across various disciplines.		
Unit 7 - Defensive Design	Explore the principles of defensive design in software development. Understand the importance of designing software systems that are robust, secure, and resistant to errors and vulnerabilities. Learn techniques for handling exceptions, input validation, and error handling. Dive into topics such as defensive programming, code review, and testing strategies. Develop skills in writing resilient and secure code that can withstand unexpected situations and potential threats.		
Unit 8 - System Software	Gain insight into the role and functions of system software in computer systems. Explore operating systems, their components, and their interaction with hardware and applications. Learn about memory management, process scheduling, file systems, and device drivers. Dive into topics such as system utilities,		

	software updates, and system security. Understand the importance of system software in providing an efficient and reliable computing environment.		
Unit 9 - Networking	Explore the fundamentals of computer networking. Learn about network architectures, protocols, and technologies. Understand the layers of the TCP/IP models. Explore topics such as IP addressing, subnetting, routing, and switching. Dive into network security, including firewalls, VPNs, and intrusion detection systems. Gain hands-on experience in configuring and troubleshooting networks. Develop an understanding of network management and the importance of effective communication and collaboration in network environments.		
Unit 10 – Ethical and Moral Computing	Delve into the ethical and moral considerations related to computing and technology. Explore the impact of technology on society, privacy, and individual rights. Discuss topics such as intellectual property, copyright infringement, and digital rights management. Understand ethical issues related to data collection, surveillance, and artificial intelligence. Explore professional codes of ethics and the responsibilities of individuals working in the computing field. Develop critical thinking skills to navigate ethical dilemmas and make informed decisions in the realm of computing.		

Programming at GCSE

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 1 – The foundations	Lay the groundwork for programming in Python. Learn the basic syntax, data types, and control structures of the language. Understand how to write and execute Python programs. Explore concepts such as variables, operators, conditionals, loops, and functions. Develop problem-solving skills and gain hands-on experience in writing simple programs. Establish a solid foundation in Python programming for further learning and exploration.		
Unit 2 – Efficient Loops	Master the art of writing efficient loops in Python. Explore different loop structures, including while loops and for loops. Learn techniques to optimize loop performance, such as loop control statements and loop termination conditions. Dive into topics such as nested loops, loop algorithms, and loop design patterns. Develop skills in designing and implementing efficient loops to process large amounts of data and solve complex problems effectively.		
Unit 3 – Data structures	Dive into the world of data structures in Python. Explore fundamental data structures such as lists, tuples, dictionaries, and sets. Learn about their properties, operations,		

	and use cases. Understand how to manipulate and traverse these data structures effectively. Dive into more advanced data structures such as stacks, queues, and linked lists. Gain hands-on experience in implementing and utilizing data structures to organize and manage data efficiently in your Python programs.		
Unit 4 – Reusable Functions	Learn the power of reusable functions in Python programming. Understand the concept of functions and their role in modular programming. Explore function definition, parameter passing, and return values. Learn how to design and implement functions to perform specific tasks and solve problems. Dive into topics such as function composition, recursion, and function libraries. Gain proficiency in creating reusable functions that enhance code organisation, readability, and reusability in your Python programs.		

A-Level Computer Science

Unit Title	Unit Overview	Prior Knowledge / skills	New Learning
Unit 1 - Components of a Computer	Explore the essential components that make up a computer system. Learn about the CPU, memory, storage devices, input/output devices, and the motherboard. Understand the role of each component in data processing and system operation. Dive into topics such as CPU architecture, memory hierarchy, and peripheral connectivity. Gain insights into the interplay between hardware and software components to form a functioning computer system.		
Unit 2 - System Software	Gain a deep understanding of system software in the context of computer systems. Explore operating systems and their functionalities, including process management, memory management, and file systems. Learn about system utilities, device drivers, and virtualization. Dive into topics such as system security, software updates, and system administration. Understand the role of system software in managing and optimizing computer resources and providing a stable and secure computing environment.		
Unit 3 - Binary Logic and Arithmetic	Explore the foundations of binary logic and arithmetic in computer systems. Learn about Boolean algebra, logic gates, and their applications in digital circuits. Understand binary addition, subtraction, multiplication, and division.		

	Dive into topics such as bitwise operations, Boolean functions, and logic simplification. Explore the concept of binary representation and its significance in computer architecture. Gain proficiency in performing binary logic operations and arithmetic calculations essential for computer science and digital systems.		
Unit 4 - Networking and Web Development	Delve into the realm of networking and web development. Learn about network protocols, architectures, and topologies. Understand the layers of the TCP/IP models. Explore topics such as IP addressing, subnetting, routing, and switching. Dive into web development, including HTML, CSS, and JavaScript. Learn about client-server architecture, HTTP protocols, and web security. Gain practical skills in designing and implementing networked systems and creating dynamic web applications.		
Unit 5 - Computational Thinking	Develop advanced computational thinking skills. Explore algorithmic problem-solving, abstraction, decomposition, pattern recognition, and algorithm design. Understand different algorithmic paradigms such as divide and conquer, greedy algorithms, and dynamic programming. Dive into topics like complexity analysis, algorithm efficiency, and algorithmic problem-solving strategies. Gain hands-on experience in solving complex problems using computational thinking techniques. Apply computational thinking skills to various domains		

	and develop the ability to design efficient and scalable algorithms.		
Unit 6 - Advance Programming	Take your programming skills to the next level with advanced programming concepts and techniques. Explore object-oriented programming (OOP) principles and design patterns. Learn about inheritance, polymorphism, encapsulation, and abstraction. Dive into topics such as exception handling, file I/O, and multithreading. Understand advanced data structures and algorithms for efficient problem-solving. Gain practical experience in developing complex software applications using advanced programming techniques. Enhance your code quality, reusability, and maintainability through advanced programming practices.		
Unit 7 - Object Oriented Programming	Dive into the principles and concepts of object-oriented programming (OOP). Understand the fundamental concepts of classes, objects, inheritance, and polymorphism. Explore encapsulation and abstraction to create modular and reusable code. Learn about the importance of class design and class relationships. Dive into topics such as method overloading, method overriding, and interfaces. Gain hands-on experience in designing and implementing object-oriented solutions to complex programming problems. Develop proficiency in writing robust and scalable object-oriented programs.		

Unit 8 - Legal, Ethical and Moral	Explore the legal, ethical, and moral aspects of computer science and technology. Understand the legal framework surrounding technology, including intellectual property rights, data protection, and cybercrime legislation. Dive into ethical considerations related to privacy, security, and responsible use of technology. Discuss moral issues such as AI ethics, social impact of technology, and ethical decision-making. Explore professional codes of conduct and ethical frameworks in the field of computer science. Develop critical thinking skills to analyse and navigate the complex ethical and legal challenges in the digital world.		
Unit 9 - Data Structures	Deepen your understanding of data structures in computer science. Explore advanced data structures such as trees, graphs, and hash tables. Learn about their properties, operations, and applications. Dive into topics such as traversal algorithms, searching and sorting techniques, and graph algorithms. Understand the trade-offs between different data structures in terms of time and space complexity. Gain practical experience in implementing and utilizing complex data structures to efficiently store and manipulate large amounts of data.		
Unit 10 - Algorithms	Explore advanced algorithms and algorithmic design techniques. Dive deeper into algorithm analysis and complexity theory. Learn about algorithmic paradigms such as divide and		

	conquer, greedy algorithms, and dynamic programming. Explore advanced topics such as graph algorithms, string algorithms, and optimization algorithms. Understand algorithmic problem-solving strategies and gain proficiency in designing efficient algorithms to solve complex computational problems. Develop critical thinking and analytical skills required to analyse, evaluate, and implement advanced algorithms effectively.		
Unit 11 - Improving Performance	Focus on enhancing the performance of software systems. Explore techniques for optimizing code, improving efficiency, and reducing resource usage. Learn about profiling and benchmarking to identify performance bottlenecks. Dive into topics such as algorithmic optimisation, data structure selection, and caching strategies. Understand the impact of hardware architectures and system configurations on performance. Gain practical experience in applying performance optimization techniques to real-world software projects. Develop skills to analyse, diagnose, and improve the performance of software systems.		
Unit 12 - Number Representation	Explore the various number representations used in computer systems. Learn about binary, decimal, and hexadecimal number systems. Understand the principles of fixed-point and floating-point number representations. Dive into topics such as integer and floating-point arithmetic, rounding, and precision. Explore the representation of negative numbers, two's		

	complement, and sign-magnitude formats. Gain insights into the limitations and trade-offs of different number representations. Develop skills in converting and manipulating numbers in different representations in the context of computer science.		
Unit 13 - Big O	Delve into the concept of algorithmic complexity and analysis. Understand the Big O notation and its significance in evaluating algorithm efficiency. Learn how to analyse the time complexity and space complexity of algorithms. Explore different types of algorithmic growth rates, such as constant, logarithmic, linear, quadratic, and exponential. Dive into topics such as worst-case, best-case, and average-case analysis. Gain practical experience in comparing and evaluating the efficiency of different algorithms using Big O notation. Develop skills in selecting and designing algorithms based on their performance characteristics.		
Unit 14 - Using Data Structures	Focus on the practical implementation and utilization of data structures in computer science. Deepen your understanding of various data structures such as arrays, linked lists, stacks, queues, trees, and graphs. Explore their properties, operations, and applications. Learn how to choose the appropriate data structure for different scenarios and problem domains. Dive into topics such as data structure manipulation, traversal algorithms, and efficient data storage.		

	Gain hands-on experience in implementing and using data structures to store, retrieve, and manipulate data effectively in your programs.		
Unit 15 - Programming Project	Engage in a comprehensive programming project that allows you to apply your knowledge and skills in a practical context. Undertake a substantial programming task, such as developing a software application or system, from concept to completion. Gain experience in project planning, requirements analysis, design, implementation, testing, and documentation. Apply programming principles, algorithms, data structures, and software development methodologies to create a functioning and well-structured project. Develop critical thinking, problem-solving, and project management skills through the execution of the programming project.		