COMPUTER SCIENCE

Department Handbook 2021-2022

Table of Contents

Department Aims and Vision

working together with passion and panache

Our aims are to work effectively as a highly skilled team of staff to

- ✓ Demonstrate an **ambitious vision** for the school and **high expectation**s for what every student and member of staff can achieve, expecting excellence in every aspect of our work.
- ✓ Offer a rich, relevant, broad and balanced **curriculum within and beyond the formal timetable**, including participation in local, national and international events, competitions and visits which motivate and **inspire a thirst for knowledge**
- ✓ Constantly drive towards securing the highest standards of both **achievement and progress** for all students – meeting **FFT 50 targets** for all in the first instance.
- ✓ Enable all students to have command and control through a **mastery curriculum and the PCS Great Teaching & Learning** model of pedagogy
- ✓ Build the capacity of team members so that all colleagues are:
  - ☐ supported and challenged
  - ☐ demonstrate the highest standards of professional skill/aptitude
  - ☐ proactively contribute to the drive for continuous improvement.
- ✓ Ensure that planning for improvement and innovation is based on rigorous data analysis, quality assurance and self-evaluation.

Intent

Computing sits at the cornerstone of the modern world, affecting the way we communicate and work as it encompasses Digital Literacy, IT and Computer Science. Our curriculum offers a pathway for our students to explore the use of applications and the creation of software to solve complex real-world problems using algorithmic thinking, which consists of abstraction, decomposition, and pattern recognition. Our focus is to develop confident independent learners who can apply thinking and reasoning skills developed in computing and other subjects across the curriculum.

We aim to develop competent students who can express themselves using a variety of applications and problem-solving techniques which allows them to tackle challenging problems related to business and the communication of data. To achieve this, students must master several applications including the ability to decompose a problem into smaller more manageable tasks using abstraction to clarify the important details contained in a specification brief. To master these skills students are exposed to familiar and unfamiliar contexts which require the application of critical thinking skills to simplify computable problems. Due to the crossover between Maths and Computer Science students will be encouraged to make use of mathematical formulae when they develop software solutions as well as the use of algebra to develop and test programming concepts.

Implementation

Please refer to the relevant sections for KS3 and GCSE curriculum coverage and explanation

Curriculum Design KS3

GCSE Computer Science .

Department Roles

Team Member	Role	Stage of career/ TLR/ UPR	Developmental responsibilities	Maintenance responsibilities	Training needs
Anthony Crowther	**Head of Computer Science**	**HOD /UPR**	○ Overview of curriculum, assessment, teaching and learning, monitoring impact of revised curriculum ○ Quality assurance ○ Data analysis overview for KS3 and KS4 ○ Leading dept. Meetings ○ Performance management ○ Dept. vision and priorities ○ Overall accountability for dept. outcomes ○ Mentoring (PGCE) ○ Coaching (NQT +1) ○ SEN Ambassador	○ Overseeing department ○ Monitoring and refining online space ○ Oversight of assessment trackers	Leading improvement in Computer Science (examiner?)
Charles Wray	**Teacher of Computer Science**	**Teacher / MPS**	○ Reviewing and developing aspects of the curriculum (see plan below) ○ Leading Robotics club and other extra-curricular activities ○ Production of Dept. MCQ's (half termly 3 per year group – Add duplicate links to tracker) ○ Creation of draft 'for the love of' home learning ○ SMSC link	○ Tracking of MCQ's ○ National Centre for Computing link	Effective assessment Exam marking

Department expectations - The Basics

Lesson routines

- Levelled and progressive objectives x2 / inspiring rather than demotivating levelling
- Use of ruler/pencil for drawing, pen for writing
- Presentation via MS Office and OneNote through MS Teams:
    - Use of screenshots to present programming activities
    - Writing to explain must use structured English with full sentences and accurate grammar
    - Electronic drawings are accepted form of submission for certain activities
- Key words explored / explained in the section provided in OneNote
- Resources – attractive, quality, legible

Marking

- Expectation is that all practise work is self/peer marked but that all GCSE/multi-step problems are marked by the teacher and feedback given on:
    a) correcting misconceptions
    b) marking for mastery
    c) specifying activity to consolidate/extend for green pen/text.
- Green penning/text evident after feedback - doesn't need to take a full lesson but should be a regular element of responding to feedback
- Learning dialogue clear – specifics re how to improve skill and knowledge
- Comments - skill and knowledge focused not behaviour focused
- Teachers are expected to maintain a marksheet logging students' scores for HW and scores for classwork/exit tickets i.e., progress with multi-step questions. Tests [low stakes and MCQs] recorded via Teams Gradebook

Learning and teaching

- Question deconstruction explicitly taught [RUCSAC/BUG]
- Use of technical terminology consistently and across all year groups - including keywords with definitions in the relevant boxes in students' work. [The etymology of keywords is being built into sows and should be included wherever possible].
- Worded questions used for all areas
- Problem solving approach explicit
- Working out expected
- Student's redraft/refine to reinforce standards expected/learning [NOT a pretty copy!]
- Pupils write their own questions / problems
- Language of the exam questions explored and used: work out, describe, explain, solve, write down
- Starters used to revise
- Learning pitched to stretch no ceilings created
- A4L to address misconceptions and to inform planning for progress
- Directed questioning to hold students to account and challenge with a plenary each lesson

Curriculum Design

Coherence and flexibility

Our curriculum is structured in units. For these units to be coherent, the lessons within a unit must be taught in order. However, across a year group, the units themselves do not need to be taught in order, except for 'Programming' units, where concepts and skills rely on prior learning and experiences.

Knowledge organisation

Our curriculum uses the National Centre for Computing Education's computing taxonomy to ensure comprehensive coverage of the subject. This has been developed through a thorough review of the KS1–4 computing programme of study, and the GCSE and A level computer science specifications across all awarding bodies. All learning outcomes can be described through a high-level taxonomy of ten strands, ordered alphabetically as follows:

- **Algorithms** — Be able to comprehend, design, create, and evaluate algorithms
- **Computer networks** — Understand how networks can be used to retrieve and share information, and how they come with associated risks
- **Computer systems** — Understand what a computer is, and how its constituent parts function together as a whole
- **Creating media** — Select and create a range of media including text, images, sounds, and video
- **Data and information** — Understand how data is stored, organised, and used to represent real-world artefacts and scenarios
- **Design and development** — Understand the activities involved in planning, creating, and evaluating computing artefacts
- **Effective use of tools** — Use software tools to support computing work
- **Impact of technology** — Understand how individuals, systems, and society as a whole interact with computer systems
- **Programming** — Create software to allow computers to solve problems
- **Safety and security** — Understand risks when using technology, and how to protect individuals and systems

The taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

Core principles

Inclusive and ambitious

Our Computing Curriculum has been written to support all pupils. Each lesson is sequenced so that it builds on the learning from the previous lesson, and where appropriate, activities are scaffolded so that all pupils can succeed and thrive. Scaffolded activities provide pupils with extra resources, such as visual prompts, to reach the same learning goals as the rest of the class. Exploratory tasks foster a deeper understanding of a concept, encouraging pupils to apply their learning in different contexts and make connections with other learning experiences.

As well as scaffolded activities, embedded within the lessons are a range of pedagogical strategies which support making computing topics more accessible.

Research-informed

The subject of computing is much younger than many other subjects, and as such, there is still a lot more to learn about how to teach it effectively. To ensure that teachers are as prepared as possible, our curriculum builds on a set of pedagogical principles, which are underpinned by the latest computing research, to demonstrate effective pedagogical strategies throughout.

To remain up to date as research continues to develop, every aspect of our curriculum is reviewed each year and changes are made as necessary.

Subject Specific Pedagogy and CPD

Computing is a broad discipline, and computing teachers require a range of strategies to deliver effective lessons to their pupils. The National Centre for Computing Education's pedagogical approach consists of 12 key principles underpinned by research: each principle has been shown to contribute to effective teaching and learning in computing.

It is recommended that teachers use their professional judgement to review, select, and apply relevant strategies for their pupils.

These 12 principles are embodied by the Teach Computing Curriculum, and examples of their application can be found throughout the units of work at every key stage.

It is important for our teachers to adopt this pedagogy as it links to the CPD that has been used to develop our teachers over the last 12 months:

- Computing at Schools Master Teacher – Anthony Crowther
- National Centre for Computing Education Accelerator program - Charles Wray

Lead with concepts

Support pupils in the acquisition of knowledge, using key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding.

Glossaries, concept maps and displays, along with regular recall and revision, can support this approach.

Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson. Primarily it begins with comprehending code rather than starting with a blank slate.

Make concrete

Bring abstract concepts to life with real-world, contextual examples and a focus on interdependencies with other curriculum subjects. This can be achieved using unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in pupils' lives.

Unplug, unpack, repack

Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach, called semantic waves can help pupils develop a secure understanding of complex concepts.

Work together

Encourage collaboration, specifically using pair programming and peer instruction and structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding

Read and explore code first

When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

Create projects

Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils can consider how to develop an artefact for a particular user or function and evaluate it against a set of criteria.

Model everything

Model processes or practices — everything from debugging code to binary number conversions — using techniques such as worked examples and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

Get hands-on

Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides pupils with a creative, engaging context to explore and apply computing concepts.

Challenge misconceptions

Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

Add variety

Provide activities with different levels of direction, scaffolding, and support that promote active learning, ranging from highly structured to more exploratory tasks.
Adapting your instruction to suit different objectives will help keep all pupils engaged and encourage greater independence.

Foster program comprehension

Use a variety of activities to consolidate knowledge and understanding of the function and structure of program s including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

Structure of KS3 Units of Work and Summaries

Presented here in the order of delivery at Park.

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Year 7	**Impact of Technology** Identifying how to use online collaboration tools respectfully. An introduction to the computing lab.	**Modelling data** Sorting and filtering data and using formulas and functions in spreadsheet software.	**Networks** Recognising networking hardware and explaining how networking components are used for communication.	**Programming Part 1** Applying the programming constructs of sequence, selection, and iteration in Scratch.	**Programming Part 2** Using subroutines to decompose a problem that incorporates lists in Scratch.	**Using Media** Creating a digital product for a real-world cause.
Year 8	**Computer Systems** Exploring the fundamental elements that make up a computer system.	**Developing for the web** Using HTML and CSS to create webpages.	**Introduction to Python** Applying the programming constructs of sequence, selection, and iteration in Python.	**Vector Graphics** Creating vector graphics through objects, layering, and path manipulation.	**Mobile Apps** Using event-driven programming to create an online gaming app.	**Data Representations** Representing numbers and text using binary digits.
Year 9	**Cybersecurity** Identifying how users and organisations can protect themselves from cyberattacks.	**Data Science** Using data to investigate problems and make real-world changes.	**Animations** Creating 3D animations through object manipulation and tweaking and adjusting lighting and camera angles.	**Physical Computing** Sensing and controlling with the micro:bit.	**Python Programming** Manipulating strings and lists. Creating a programming project.	**Data Representations** Representing images and sound using binary digits.

KS3 Units Assessment Taxonomy

Objectives	7.1	7.2	7.3	7.4	7.5	7.6	8.1	8.2	8.3	8.4	8.5	8.6	9.1	9.2	9.3	9.4	9.5	9.6
Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems						T			T			T	T					T
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				T	T				T			T	T					T
Use two 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				T	T							T	T					T
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]				T	T						T							
Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems		T									T							
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								T			T	T	T			T		T
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			T			T				T					T			
Create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability	T		T	T	T		T		T	T				T				
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	T																T	

GCSE Curriculum and Assessment Taxonomy

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.1 - Systems architecture	Architecture of the CPU	common CPU components and their function: - ALU (Arithmetic Logic Unit) - CU (Control Unit) - Cache - Registers	Computer Systems															
1.1 - Systems architecture	Architecture of the CPU	The purpose of the CPU: - The fetch-execute cycle	Computer Systems															
1.1 - Systems architecture	CPU performance	how common characteristics of CPUs affect their performance: - clock speed - cache size - number of cores	Computer Systems															
1.1 - Systems architecture	Embedded systems	Examples of embedded systems	Object-oriented programming															
1.1 - Systems architecture	Embedded systems	The purpose and characteristics of embedded systems	Object-oriented programming															
1.2 - Memory and storage	Data storage	binary shifts	Data Representation															
1.2 - Memory and storage	Data storage	how an image is represented as a series of pixels, represented in binary	Data Representation															
1.2 - Memory and storage	Data storage	how sound can be sampled and stored in digital form	Data Representation															
1.2 - Memory and storage	Data storage	how to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur	Data Representation															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.2 - Memory and storage	Data storage	How to convert binary integers to their hexadecimal equivalents and vice versa	Data Representation															
1.2 - Memory and storage	Data storage	How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa	Data Representation															
1.2 - Memory and storage	Data storage	How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa	Data Representation															
1.2 - Memory and storage	Data storage	metadata	Data Representation															
1.2 - Memory and storage	Data storage	the effect of colour depth and resolution on: - the quality of the image - the size of an image file	Data Representation															
1.2 - Memory and storage	Data storage	The effect of sample rate, duration and bit depth on: - The playback quality - The size of a sound file	Data Representation															
1.2 - Memory and storage	Data storage	the relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: - ASCII - Unicode	Data Representation															
1.2 - Memory and storage	Data storage	the term ‘character-set’	Data Representation															
1.2 - Memory and storage	Data storage	the use of binary codes to represent characters	Data Representation															
1.2 - Memory and storage	Primary storage (Memory)	the difference between RAM and ROM	Computer Systems															
1.2 - Memory and storage	Primary storage (Memory)	The need for primary storage	Computer Systems															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.2 - Memory and storage	Primary storage (Memory)	the purpose of RAM in a computer system	Computer Systems															
1.2 - Memory and storage	Primary storage (Memory)	the purpose of ROM in a computer system	Computer Systems															
1.2 - Memory and storage	Secondary storage	common types of storage: - optical - magnetic - solid state	Computer Systems															
1.2 - Memory and storage	Secondary storage	the advantages and disadvantages of these, using characteristics: - capacity - speed - portability - durability - reliability - cost.	Computer Systems															
1.2 - Memory and storage	Secondary storage	the need for secondary storage	Computer Systems															
1.2 - Memory and storage	Units	How data needs to be converted into a binary format to be processed by a computer	Data Representation															
1.2 - Memory and storage	Units	The units of data storage: - Bit - Nibble (4 bits) - Byte (8 bits) - Kilobyte (1,000 bytes or 1 KB) - Megabyte (1,000 KB) - Gigabyte (1,000 MB) - Terabyte (1,000 GB) - Petabyte (1,000 TB)	Data Representation															
1.3 - Computer networks, connections and protocols	Networks and topologies	factors that affect the performance of networks	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	star and mesh network topologies	Networks															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.3 - Computer networks, connections and protocols	Networks and topologies	the different roles of computers in a client-server and a peer-to-peer network	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	the hardware needed to connect stand-alone computers into a Local Area Network: - wireless access points - routers - switches - NIC (Network Interface Controller/Card) - transmission media	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	the internet as a worldwide collection of computer networks: - DNS (Domain Name Server) - hosting - the cloud - web servers and clients	Computer Systems, Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	types of networks: - LAN (Local Area Network) - WAN (Wide Area Network)	Networks															
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Common protocols including: - TCP/IP (Transmission Control Protocol/Internet Protocol) - HTTP (Hyper Text Transfer Protocol) - HTTPS (Hyper Text Transfer Protocol Secure) - FTP (File Transfer Protocol) - POP (Post Office Protocol) - IMAP (Internet Message Access Protocol) - SMTP (Simple Mail Transfer Protocol)	Networks															
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	IP addressing and MAC addressing	Networks															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Modes of connection: - Wired - Ethernet - Wireless - Wi-Fi - Bluetooth	Networks															
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Network standards	Networks															
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	the concept of layers	Networks															
1.4 - Network security	Identifying and preventing vulnerabilities	Common prevention methods: - Penetration testing - Anti-malware software - Firewalls - User access levels - Passwords - Encryption - Physical security	Networks, Security															
1.4 - Network security	Threats to computer systems and networks	Forms of attack: - Malware - Social engineering, e.g. phishing, people as the 'weak point' - Brute-force attacks - Denial of service attacks - Data interception and theft - The concept of SQL injection	Security															
1.5 - Systems software	Utility software	the purpose and functionality of utility software	Computer Systems															
1.5 - Systems software	Utility software	utility system software: - encryption software - defragmentation - data compression	Animations															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.6 - Ethical, legal, cultural and environmental impacts of digital technology	Ethical, legal, cultural and environmental impact	Impacts of digital technology on wider society including: - Ethical issues - Legal issues - Cultural issues - Environmental issues - Privacy issues	Impacts of technology, Security															
1.6 - Ethical, legal, cultural and environmental impacts of digital technology	Ethical, legal, cultural and environmental impact	Legislation relevant to Computer Science: - The Data Protection Act 2018 - Computer Misuse Act 1990 - Copyright Designs and Patents Act 1988 - Software licences (i.e. open source and proprietary)	Impacts of technology															
2.1 - Algorithms	Computational thinking	Principals of computational thinking: - abstraction - decomposition - algorithmic thinking	Algorithms 1															
2.1 - Algorithms	Designing, creating and refining algorithms	(create, interpret, correct, complete, and refine algorithms using: - pseudocode - flowcharts - reference language / high-level programming language)	Algorithms 1, Algorithms 2, Flat-file databases, Iteration, Selection, Sequence															
2.1 - Algorithms	Designing, creating and refining algorithms	Identify common errors	Algorithms 1															
2.1 - Algorithms	Designing, creating and refining algorithms	Trace tables	Algorithms 1															
2.1 - Algorithms	Searching and sorting algorithms	standard searching algorithms: - binary search - linear search	Algorithms 2															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.1 - Algorithms	Searching and sorting algorithms	standard sorting algorithms: - bubble sort - merge sort - insertion sort	Algorithms 2															
2.2 - Programming fundamentals	Additional programming techniques	(the use of SQL to search for data)	Databases and SQL															
2.2 - Programming fundamentals	Additional programming techniques	how to use sub programs (functions and procedures) to produce structured code	Iteration, Sequence, Subroutines															
2.2 - Programming fundamentals	Additional programming techniques	Random number generation	Selection, Strings and lis															
2.2 - Programming fundamentals	Additional programming techniques	the use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays	Strings and lists															
2.2 - Programming fundamentals	Additional programming techniques	the use of basic file handling operations: - open - read - write - close	Dictionaries and datafiles Flat-file databases															
2.2 - Programming fundamentals	Additional programming techniques	the use of basic string manipulation	Strings and lists															
2.2 - Programming fundamentals	Additional programming techniques	the use of records to store data	Dictionaries and datafiles Strings and lists															
2.2 - Programming fundamentals	Data types	the use of data types: - integer - real - Boolean - character and string - casting	Sequence															
2.2 - Programming fundamentals	Programming fundamentals	the common arithmetic operators	Selection															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.2 - Programming fundamentals	Programming fundamentals	the common Boolean operators AND, OR and NOT	Selection										■					
2.2 - Programming fundamentals	Programming fundamentals	the use of the three basic programming constructs used to control the flow of a program: - sequence - selection - iteration (count- and condition-controlled loops)	Iteration, Programming I Selection, Sequence									■	■	■				
2.2 - Programming fundamentals	Programming fundamentals	the use of variables, constants, operators, inputs, outputs and assignments	Selection, Sequence, Subroutines									■	■		■			
2.3 - Producing robust programs	Defensive design	Input validation	Iteration, Sequence									■		■				
2.3 - Producing robust programs	Defensive design	Maintainability: - Use of sub programs - Naming conventions - Indentation - Commenting	Sequence									■						
2.3 - Producing robust programs	Testing	identify syntax and logic errors	Sequence									■						
2.3 - Producing robust programs	Testing	Refining algorithms	Dictionaries and datafiles														■	
2.3 - Producing robust programs	Testing	selecting and using suitable test data: - Normal - Boundary - Invalid - Erroneous	Subroutines												■			
2.3 - Producing robust programs	Testing	types of testing: - iterative - final/terminal	Dictionaries and datafiles Subroutines												■		■	

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.4 - Boolean logic	Boolean logic	applying logical operators in truth tables to solve problems	Computer Systems															
2.4 - Boolean logic	Boolean logic	combining Boolean operators using AND, OR and NOT	Computer Systems															
2.4 - Boolean logic	Boolean logic	truth tables	Computer Systems, Subroutines															
2.5 - Programming languages and Integrated Development Environments	Languages	Characteristics and purpose of different levels of programming language: - High-level languages - Low-level languages	Computer Systems, Sequence															
2.5 - Programming languages and Integrated Development Environments	Languages	the characteristics of a compiler and an interpreter	Sequence															
2.5 - Programming languages and Integrated Development Environments	Languages	the purpose of translators	Sequence															
2.5 - Programming languages and Integrated Development Environments	The Integrated Development Environment (IDE)	common tools and facilities available in an integrated development environment (IDE): - editors - error diagnostics - run-time environment - translators.	Sequence															
1.1 - Systems architecture	Architecture of the CPU	common CPU components and their function: - ALU (Arithmetic Logic Unit) - CU (Control Unit) - Cache - Registers	Computer Systems															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.1 - Systems architecture	Architecture of the CPU	The purpose of the CPU: - The fetch-execute cycle	Computer Systems															
1.1 - Systems architecture	Architecture of the CPU	Von Neumann architecture: - MAR (Memory Address Register) - MDR (Memory Data Register) - Program Counter - Accumulator	Computer Systems															
1.1 - Systems architecture	CPU performance	how common characteristics of CPUs affect their performance: - clock speed - cache size - number of cores	Computer Systems															
1.1 - Systems architecture	Embedded systems	Examples of embedded systems	Object-oriented programming															
1.1 - Systems architecture	Embedded systems	The purpose and characteristics of embedded systems	Object-oriented programming															
1.2 - Memory and storage	Compression	need for compression																
1.2 - Memory and storage	Compression	types of compression: - lossy - lossless.																
1.2 - Memory and storage	Data storage	binary shifts	Data Representation															
1.2 - Memory and storage	Data storage	how an image is represented as a series of pixels, represented in binary	Data Representation															
1.2 - Memory and storage	Data storage	how sound can be sampled and stored in digital form	Data Representation															
1.2 - Memory and storage	Data storage	how to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur	Data Representation															
1.2 - Memory and storage	Data storage	How to convert binary integers to their hexadecimal equivalents and vice versa	Data Representation															
1.2 - Memory and storage	Data storage	How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa	Data Representation															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.2 - Memory and storage	Data storage	How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa	Data Representation															
1.2 - Memory and storage	Data storage	metadata	Data Representation															
1.2 - Memory and storage	Data storage	the effect of colour depth and resolution on: - the quality of the image - the size of an image file	Data Representation															
1.2 - Memory and storage	Data storage	The effect of sample rate, duration and bit depth on: - The playback quality - The size of a sound file	Data Representation															
1.2 - Memory and storage	Data storage	the relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: - ASCII - Unicode	Data Representation															
1.2 - Memory and storage	Data storage	the term ‘character-set’	Data Representation															
1.2 - Memory and storage	Data storage	the use of binary codes to represent characters	Data Representation															
1.2 - Memory and storage	Primary storage (Memory)	the difference between RAM and ROM	Computer Systems															
1.2 - Memory and storage	Primary storage (Memory)	The need for primary storage	Computer Systems															
1.2 - Memory and storage	Primary storage (Memory)	the need for virtual memory																

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.2 - Memory and storage	Primary storage (Memory)	the purpose of RAM in a computer system	Computer Systems															
1.2 - Memory and storage	Primary storage (Memory)	the purpose of ROM in a computer system	Computer Systems															
1.2 - Memory and storage	Secondary storage	common types of storage: - optical - magnetic - solid state	Computer Systems															
1.2 - Memory and storage	Secondary storage	suitable storage devices and storage media for a given application	Computer Systems															
1.2 - Memory and storage	Secondary storage	the advantages and disadvantages of these, using characteristics: - capacity - speed - portability - durability - reliability - cost.	Computer Systems															
1.2 - Memory and storage	Secondary storage	the need for secondary storage	Computer Systems															
1.2 - Memory and storage	Units	data capacity and calculation of data capacity requirements																
1.2 - Memory and storage	Units	How data needs to be converted into a binary format to be processed by a computer	Data Representation															
1.2 - Memory and storage	Units	The units of data storage: - Bit - Nibble (4 bits) - Byte (8 bits) - Kilobyte (1,000 bytes or 1 KB) - Megabyte (1,000 KB) - Gigabyte (1,000 MB) - Terabyte (1,000 GB) - Petabyte (1,000 TB)	Data Representation															
1.3 - Computer networks,	Networks and topologies	factors that affect the performance of networks	Networks															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
connections and protocols																		
1.3 - Computer networks, connections and protocols	Networks and topologies	star and mesh network topologies	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	the different roles of computers in a client-server and a peer-to-peer network	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	the hardware needed to connect stand-alone computers into a Local Area Network: - wireless access points - routers - switches - NIC (Network Interface Controller/Card) - transmission media	Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	the internet as a worldwide collection of computer networks: - DNS (Domain Name Server) - hosting - the cloud - web servers and clients	Computer Systems, Networks															
1.3 - Computer networks, connections and protocols	Networks and topologies	types of networks: - LAN (Local Area Network) - WAN (Wide Area Network)	Networks															
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Common protocols including: - TCP/IP (Transmission Control Protocol/Internet Protocol) - HTTP (Hyper Text Transfer Protocol) - HTTPS (Hyper Text Transfer Protocol Secure) - FTP (File Transfer Protocol) - POP (Post Office Protocol) - IMAP (Internet Message Access Protocol) - SMTP (Simple Mail Transfer Protocol)	Networks															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Encryption	Security							■								
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	IP addressing and MAC addressing	Networks						■									
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Modes of connection: - Wired - Ethernet - Wireless - Wi-Fi - Bluetooth	Networks						■									
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	Network standards	Networks						■									
1.3 - Computer networks, connections and protocols	Wired and wireless networks, protocols and layers	the concept of layers	Networks						■									
1.4 - Network security	Identifying and preventing vulnerabilities	Common prevention methods: - Penetration testing - Anti-malware software - Firewalls - User access levels - Passwords - Encryption - Physical security	Networks, Security						■	■								
1.4 - Network security	Threats to computer systems and networks	Forms of attack: - Malware - Social engineering, e.g. phishing, people as the 'weak point' - Brute-force attacks - Denial of service attacks - Data interception and theft - The concept of SQL injection	Security							■								

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
1.5 - Systems software	Operating systems	The purpose and functionality of operating systems: - user interface - memory management and multitasking - peripheral management and drivers - user management - file management	Computer Systems															
1.5 - Systems software	Utility software	the purpose and functionality of utility software	Computer Systems															
1.5 - Systems software	Utility software	utility system software: - encryption software - defragmentation - data compression	Animations															
1.6 - Ethical, legal, cultural and environmental impacts of digital technology	Ethical, legal, cultural and environmental impact	Impacts of digital technology on wider society including: - Ethical issues - Legal issues - Cultural issues - Environmental issues - Privacy issues	Impacts of technology, Security															
1.6 - Ethical, legal, cultural and environmental impacts of digital technology	Ethical, legal, cultural and environmental impact	Legislation relevant to Computer Science: - The Data Protection Act 2018 - Computer Misuse Act 1990 - Copyright Designs and Patents Act 1988 - Software licences (i.e. open source and proprietary)	Impacts of technology															
2.1 - Algorithms	Computational thinking	Principals of computational thinking: - abstraction - decomposition - algorithmic thinking	Algorithms 1															
2.1 - Algorithms	Designing, creating and refining algorithms	(create, interpret, correct, complete, and refine algorithms using: - pseudocode - flowcharts - reference language / high-level programming language)	Algorithms 1, Algorithms 2, Flat file databases, Iteration, Selection, Sequence															
2.1 - Algorithms	Designing, creating and refining algorithms	Identify common errors	Algorithms 1															
2.1 - Algorithms	Designing, creating and refining algorithms	Identify the inputs, processes, and outputs for a problem	Algorithms 1															

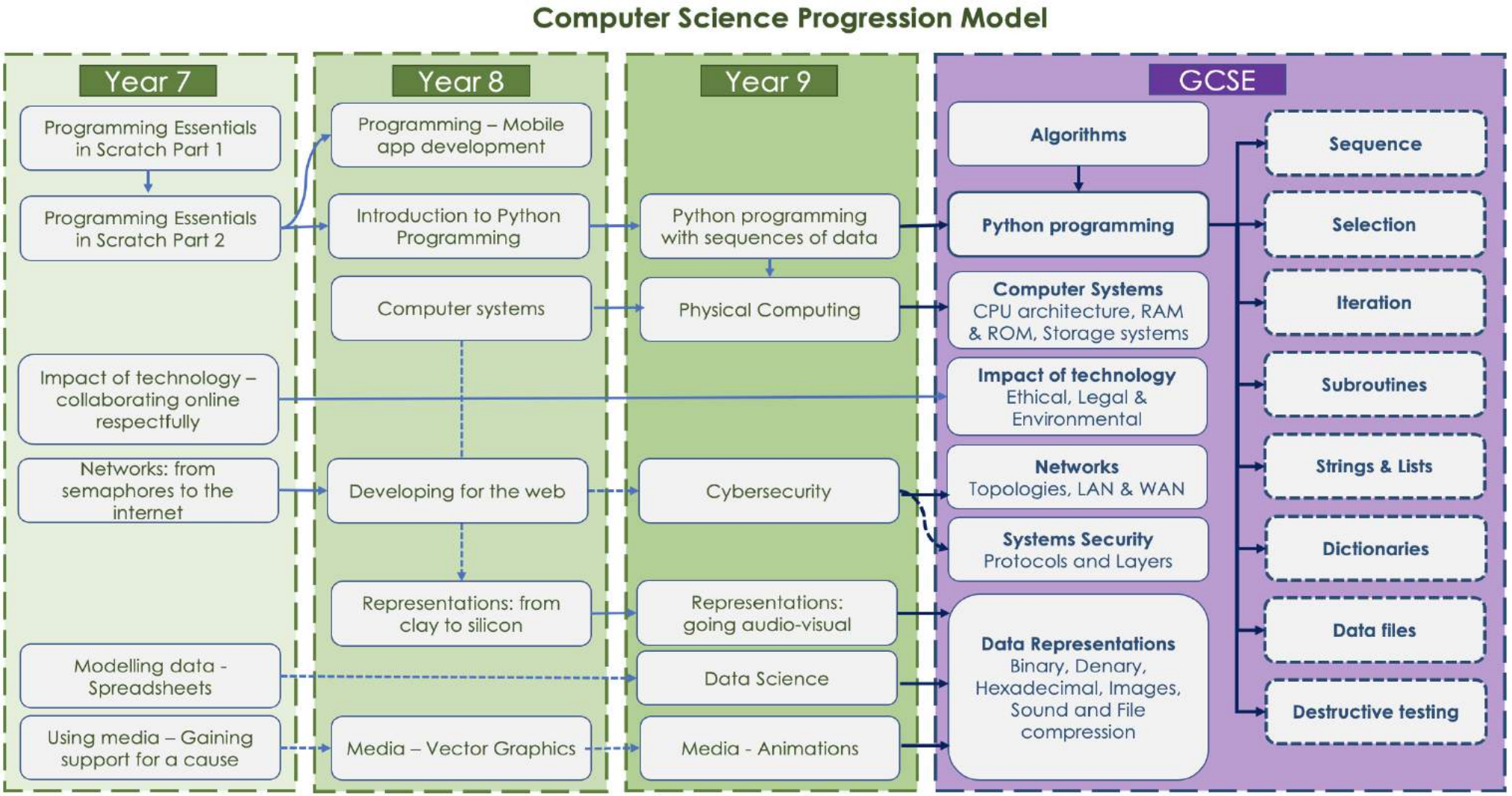
Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.1 - Algorithms	Designing, creating and refining algorithms	Structure diagrams	Subroutines															
2.1 - Algorithms	Designing, creating and refining algorithms	Trace tables	Algorithms 1															
2.1 - Algorithms	Searching and sorting algorithms	standard searching algorithms: - binary search - linear search	Algorithms 2															
2.1 - Algorithms	Searching and sorting algorithms	standard sorting algorithms: - bubble sort - merge sort - insertion sort	Algorithms 2															
2.2 - Programming fundamentals	Additional programming techniques	(the use of SQL to search for data)	Databases and SQL															
2.2 - Programming fundamentals	Additional programming techniques	how to use sub programs (functions and procedures) to produce structured code	Iteration, Sequence, Subroutines															
2.2 - Programming fundamentals	Additional programming techniques	Random number generation	Selection, Strings and lis															
2.2 - Programming fundamentals	Additional programming techniques	the use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays	Strings and lists															
2.2 - Programming fundamentals	Additional programming techniques	the use of basic file handling operations: - open - read - write - close	Dictionaries and datafiles Flat-file databases															
2.2 - Programming fundamentals	Additional programming techniques	the use of basic string manipulation	Strings and lists															
2.2 - Programming fundamentals	Additional programming techniques	the use of records to store data	Dictionaries and datafiles Strings and lists															
2.2 - Programming fundamentals	Data types	the use of data types: - integer - real - Boolean - character and string - casting	Sequence															

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.2 - Programming fundamentals	Programming fundamentals	the common arithmetic operators	Selection										■					
2.2 - Programming fundamentals	Programming fundamentals	the common Boolean operators AND, OR and NOT	Selection										■					
2.2 - Programming fundamentals	Programming fundamentals	the use of the three basic programming constructs used to control the flow of a program: - sequence - selection - iteration (count- and condition-controlled loops)	Iteration, Programming l Selection, Sequence									■	■	■				
2.2 - Programming fundamentals	Programming fundamentals	the use of variables, constants, operators, inputs, outputs and assignments	Selection, Sequence, Subroutines									■	■		■			
2.3 - Producing robust programs	Defensive design	defensive design considerations: - anticipating misuse - authentication	Dictionaries and datafiles														■	
2.3 - Producing robust programs	Defensive design	Input validation	Iteration, Sequence									■		■				
2.3 - Producing robust programs	Defensive design	Maintainability: - Use of sub programs - Naming conventions - Indentation - Commenting	Sequence									■						
2.3 - Producing robust programs	Testing	identify syntax and logic errors	Sequence									■						
2.3 - Producing robust programs	Testing	Refining algorithms	Dictionaries and datafiles														■	
2.3 - Producing robust programs	Testing	selecting and using suitable test data: - Normal - Boundary - Invalid - Erroneous	Subroutines												■			

Specification	Sub-topic	To know and understand:	Units	Algorithms 1	Computer System	Data Representati	Algorithms 2	Impacts of technolog	Networks	Security	Databases and SQ	Sequence	Selection	Iteration	Subroutines	Strings and lists	Dictionaries and datafiles	Object-oriented programming
2.3 - Producing robust programs	Testing	the purpose of testing	Subroutines															
2.3 - Producing robust programs	Testing	types of testing: - iterative - final/terminal	Dictionaries and datafiles Subroutines															
2.4 - Boolean logic	Boolean logic	applying logical operators in truth tables to solve problems	Computer Systems															
2.4 - Boolean logic	Boolean logic	combining Boolean operators using AND, OR and NOT	Computer Systems															
2.4 - Boolean logic	Boolean logic	simple logic diagrams using the operators AND, OR and NOT	Computer Systems															
2.4 - Boolean logic	Boolean logic	truth tables	Computer Systems, Subroutines															
2.5 - Programming languages and Integrated Development Environments	Languages	Characteristics and purpose of different levels of programming language: - High-level languages - Low-level languages	Computer Systems, Sequence															
2.5 - Programming languages and Integrated Development Environments	Languages	the characteristics of a compiler and an interpreter	Sequence															
2.5 - Programming languages and Integrated Development Environments	Languages	the purpose of translators	Sequence															
2.5 - Programming languages and Integrated Development Environments	The Integrated Development Environment (IDE)	common tools and facilities available in an integrated development environment (IDE): - editors - error diagnostics - run-time environment - translators.	Sequence															

Progression Model

This model below demonstrates how the Units at KS3 feed the Units at GCSE and exemplifies the importance of progression through learning foundation concepts.



Assessment Policy

The Computing assessment policy aims to ensure we have a balance between curriculum coverage and assessment, which ought to support student progress through a recognition of achievement and reflection upon areas for improvement.

Although it is important to use a range of assessment methods, a great deal of what we do in Computing is practical and therefore verbal feedback is a consistent feature of lessons.

Computational thinking and Problem solving are key features of all units of work across the department. Computational thinking can be assessed in a variety of ways; however, the clearest model is to assess the planning phase for any solution as this can highlight a weakness in sequence, selection, or iteration within an algorithm. Errors in these areas will demonstrate a weakness in logic within the solution which will lead to failures when programmed. Students can develop algorithms as electronic or analogue documents using flow diagrams or pseudocode. Assessment of these artefacts will help to improve student's success at the development stages when learning to program.

Formative assessment

Every lesson includes formative assessment opportunities for teachers to use. These opportunities are listed in the Medium-Term Plans and are included to ensure that misconceptions are recognised and addressed if they occur. They vary from teacher observation or questioning, to marked activities.

These assessments are vital to ensure that teachers are adapting their teaching to suit the needs of the pupils that they are working with. Any adaptation needs to be noted in the Medium-Term Plans.

The learning objectives are introduced in the slides or OneNote files at the beginning of every lesson. Every lesson has a starter activity and a plenary that can be used as an opportunity for formative assessment.

Summative assessment

Every unit includes an optional summative assessment framework in the form of either a multiple-choice quiz (MCQ) or a rubric. All units are designed to cover both skills and concepts from across the computing national curriculum. Units that focus more on conceptual development include an MCQ. Units that focus more on skills development end with a project and include a rubric. However, within the 'Programming' units, the assessment framework (MCQ or rubric) has been selected on a best-fit basis.

Multiple choice quiz (MCQ)

Each of the MCQ questions has been carefully chosen to represent learning that should have been achieved within the unit. In writing the MCQs, we have followed the diagnostic assessment approach to ensure that the assessment of the unit is useful to determine both how well pupils have understood the content, and what pupils have misunderstood, if they have not achieved as expected.

Each MCQ includes an answer sheet that highlights the misconceptions that pupils may have if they have chosen a wrong answer. This ensures that teachers know which areas to return to in later units.

Rubrics

The rubric is a tool to help teachers assess project-based work. Each rubric covers the application of skills that have been directly taught across the unit, and highlights to teachers whether the pupil is approaching (WT), achieving (ARE), or exceeding the expectations (AGD) for their age group. It allows teachers to assess projects that pupils have created, focussing on the appropriate application of computing skills and concepts.

Pedagogically, we want to ensure that we are assessing pupils' understanding of computing concepts and skills, as opposed to their reading and writing skills. This has been carefully considered both in how MCQs have been written (considerations such as the language used, the cultural experiences referenced, etc) and in the skills expected to be demonstrated in the rubric.

Student Accountability

Students on GCSE courses will be provided with PLC's which they must complete every time they complete a quiz. This ensures they are tracking their own progress and can identify areas which need greater focus. The PLC's also provide an area for students to write a short reflection about their learning and to describe their own next steps

Progression Pathways KS3

PP	Algorithms	Programming & Development	Data & Data Representation	Hardware & Processing	Communications & Networks	Digital: Creativity & Citizens
2	*Understands that algorithms are implemented on digital devices as programs. Designs simple algorithms using loops and selection i.e. if statements. Use logical reasoning to predict outcomes. Detects and correct errors i.e. debugging algorithms.*	*Uses arithmetic operators, if statements and loops within programs. Use logical reasoning to predict the behaviour or programs. Detects and corrects simple semantic errors i.e. debugging in programs*	*Recognises different types of data: text, number. Appreciates that programs can work with different types of data. Recognises that data can be structured in tables to make it useful.*	*Recognises that a range of digital devices can be considered a computer. Recognises and can use a range of input and output devices. Understands how programs specify the function of a general-purpose computer.*	*Navigates the WWW can carry out simple searches to collect digital content. Demonstrates sue of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online.*	*Uses technology purposefully to organise digital content. Uses a variety of software to assess, manipulate and present digital content and information. Shares their experiences of IT beyond the classroom. Talks about their work and makes improvements based on feedback.*
3	*Designs solutions (algorithms) and two-way selection i.e. IF, THEN and ELSE statements. Uses diagrams to express solutions. Uses logical reasoning to predict outputs, showing and awareness of inputs.*	*Creates programs that implement algorithms to achieve given goals. Declares and assigns variables. Uses post-tested loop e.g. until and a sequence of selection statements in programs, including an IF, THEN and ELSE statement.*	*Understand the difference between data and information. Knows why sorting data in a flat file can improve searching for information. Uses filters or can perform single criteria searches.*	*Knows that computers collect data from various input devices, including sensors and applications. Understands the difference between hardware and application software, and their roles in a computer system.*	*Understands the difference between the internet and WWW. Shows an awareness of and can use a range of internet services e.g. VOIP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services.*	*Collects and Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience. Makes appropriate improvement to solutions based on feedback and can comment on the success of the solution.*
4	*Shows an awareness of tasks best completed by humans or computers. Designs solutions by decomposing a problem and creates a sub-solution for each of these parts (decomposition). Recognises that different solutions exist for the same problem.*	*Understands IF and IF, THEN and ELSE statements. Uses a variable and relational operator within a loop. Designs, writes and debugs modular programs. Knows that a procedure can be used to hide detail within a sub-solution (procedural abstraction).*	*Performs more complex searches for information e.g. using Boolean and relational operators. Analyses and evaluates data and information and recognises that poor quality data needs leads to unreliable results and inaccurate conclusions.*	*Understands why and when computers are used. Understands the main functions of the operating system. Knows the difference between physical, wireless and mi=mobile networks.*	*Understands how to effectively use searching engines, and knows how search results are selected, including that search engines use web crawler programs. Demonstrates responsible use of technologies and online services and knows a range of ways to report concerns.*	*Makes judgements about digital content when evaluating and repurposing it for a given audience. Understands the potential of IT for collaboration when computers are networked. Uses criteria to evaluate the quality of solutions, can identify and make some refinements to the solution.*
5	*Understands that iteration is the repetition of a process such as a loop. Recognises that different algorithms exist for the same problem. Represents solutions using a structured notation. Can identify similarities and differences in situations using these to solve problems (pattern recognition).*	*Understands that programming bridges the gap between algorithmic solutions and computers. Has practical experience of a high-level language. Uses a range of operators and expressions e.g. Boolean and applies them in the context of program using appropriate data types.*	*Knows that computers use binary to represent data. Understands how bit patterns represent numbers and images. Understands the relationship between binary and file size (uncompressed). Defines data types: real numbers and Boolean. Queries data on one table using a typical query language.*	*Recognises and understands the function of the main internal parts of basic computer architecture. Understands the concepts behind the fetch- execute cycle. Knows that there is a range of operating systems and application software for some hardware.*	*Understands how search engines rank search results. Understands how to construct static web pages using HTML and CSS. Understands transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching.*	*Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. Recognises ethical issues surrounding the application of IT beyond school. Designs and uses criteria to critically evaluate the quality of solutions in order to make appropriate solutions.*
6	*Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics.*	*Uses nested selection statements. Appreciates the need for and writes custom functions using parameters. Appropriately uses procedures and functions. Understands and uses negation with operators. Uses and manipulates a one-dimensional data structure. Detects and corrects syntactical errors.*	*Understands how numbers, images, sounds and character sets use the same bit patterns. Understands the relationship between resolution and colour depth, including the effect on file size. Distinguishes between data used in simple programs (a variable) and the storage for that data.*	*Understand the Von Neumann architecture in relation to the fetch-decode-execute cycle, including how data is stored in memory. Understands the basic function and operation of location addressable memory.*	*Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, iMAP, POP, FTP, TCP/IP, associated with networking computer systems. Uses technologies and online services securely and knows how to identify and report concerns.*	*Justifies uses multiple digital devices, internet services and application software to achieve given goals. Evaluates the validity of digital content and assess the usability of design features when developing content for a given audience. Develops criteria to evaluate the quality of solutions using feedback from several sources to identify and make improvement.*
7	*Recognises that the design of an algorithm is distinct from its expression. Evaluates the effectiveness of algorithms and models for similar problems. Recognises where information can be filtered out in generalising solutions (abstraction). Uses logical reasoning to explain how an algorithm works.*	*Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from outside its function. Understands and applies parameter passing. Understands the difference between and uses both pre-tested (While) and post tested loops (Until). Applies a modular approach to error detection and correction.*	*Knows the relationship between data representation and data quality. Understands the relationship between binary and electrical circuits, including Boolean logic. Understands how and why values are data typed in many different languages when manipulated within programs.*	*Knows that processors have instruction sets and that these relate to low-level instructions carried out by a computer.*	*Knows the purpose of hardware and protocols associated with networking computer systems. Understands the client-server model including how dynamic webpages use server-side scripting and that webservers process and store data. Recognises that persistence of data on the internet requires careful protection of online identity and privacy.*	*Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. Considers the properties of media when importing them into digital artefacts, develops success criteria and Collects user feedback in order to make appropriate refinements.*

Progression Pathways KS4

Grade	Algorithms	Programming & Development	Data & Data Representation	Hardware & Processing	Communications & Networks	Digital: Creativity & Citizens
3	*Designs solutions (algorithms) and two-way selection i.e. IF, THEN and ELSE statements. Uses diagrams to express solutions. Uses logical reasoning to predict outputs, showing and awareness of inputs.*	*Creates programs that implement algorithms to achieve given goals. Declares and assigns variables. Uses post-tested loop e.g. until and a sequence of selection statements in programs, including an IF, THEN and ELSE statement.*	*Understand the difference between data and information. Knows why sorting data in a flat file can improve searching for information. Uses filters or can perform single criteria searches.*	*Knows that computers collect data from various input devices, including sensors and applications. Understands the difference between hardware and application software, and their roles in a computer.*	*Understands the difference between the internet and internet service e.g. WWW. Shows an awareness of and can use a range of internet services e.g. VOIP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services.*	*Collects and Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience. Makes appropriate improvement to solutions based on feedback and can comment on the success of the solution.*
4	*Shows an awareness of tasks best completed by humans or computers. Designs solutions by decomposing a problem and creates a sub-solution for each of these parts (decomposition). Recognises that different solutions exist for the same problem.*	*Understands IF and IF, THEN and ELSE statements. Uses a variable and relational operator within a loop. Designs, writes and debugs modular programs. Knows that a procedure can be used to hide detail within a sub-solution (procedural abstraction).*	*Performs more complex searches for information e.g. using Boolean and relational operators. Analyses and evaluates data and information and recognises that poor quality data needs lead to unreliable results and inaccurate conclusions.*	*Understands why and when computers are used. Understands the main functions of the operating system. Knows the difference between physical, wireless and mi=mobile networks.*	*Understands how to effectively use searching engines, and knows how search results are selected, including that search engines use web crawler programs. Demonstrates responsible use of technologies and online services and knows a range of ways to report concerns.*	*Makes judgements about digital content when evaluating and repurposing it for a given audience. Understands the potential of IT for collaboration when computers are networked. Uses criteria to evaluate the quality of solutions, can identify and make some refinements to the solution.*
5	*Understands that iteration is the repetition of a process such as a loop. Recognises that different algorithms exist for the same problem. Represents solutions using a structured notation. Can identify similarities and differences in situations using these to solve problems (pattern recognition).*	*Understands that programming bridges the gap between algorithmic solutions and computers. Has practical experience of a high-level language, using standard libraries. Uses a range of operators and expressions e.g. Boolean and applies them in the context of program using appropriate data types.*	*Knows that computers use binary to represent data. Understands how bit patterns represent numbers and images. Understands the relationship between binary and file size (uncompressed). Defines data types: real numbers and Boolean. Queries data on one table using a typical query language.*	*Recognises and understands the function of the main internal parts of a computer architecture. Understands the concepts behind the fetch- execute cycle. Knows that there is a range of operating systems and application software for some hardware.*	*Understands how search engines rank search results. Understands how to construct static web pages using HTML and CSS. Understands transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching.*	*Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. Recognises ethical issues surrounding the application of IT beyond school. Designs and uses criteria to critically evaluate the quality of solutions in order to make appropriate solutions.*
6	*Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics.*	*Uses nested selection statements. Appreciates the need for and writes custom functions using parameters. Appropriately uses procedures and functions. Understands and uses negation with operators. Uses and manipulates a one-dimensional data structure. Detects and corrects syntactical errors.*	*Understands how numbers, images, sounds and character sets use the same bit patterns. Understands the relationship between resolution and colour depth, including the effect on file size. Distinguishes between data used in simple programs (a variable) and the storage for that data.*	*Understand the Von Neumann architecture in relation to the fetch-decode-execute cycle, including how data is stored in memory. Understands the basic function and operation of location addressable memory.*	*Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, iMAP, POP, FTP, TCP/IP, associated with networking computer systems. Uses technologies and online services securely and knows how to identify and report concerns.*	*Justifies uses multiple digital devices, internet services and application software to achieve given goals. Evaluates the validity of digital content and the usability of design features when developing content for a given audience. Develops criteria to evaluate the quality of solutions using feedback from several sources to identify and improve.*
7	*Recognises that the design of an algorithm is distinct from its expression. Evaluates the effectiveness of algorithms and models for similar problems. Recognises where information can be filtered out (abstraction). Uses logical reasoning to explain how an algorithm works.*	*Appreciates the effect and scope of a variable e.g. a local variable can't be accessed from outside its function. Understands and applies parameter passing. Understands the difference between and uses both pre-tested (While) and post tested loops (Until). Applies a modular approach to error detection and correction.*	*Knows the relationship between data representation and data quality. Understands the relationship between binary and electrical circuits, including Boolean logic. Understands how and why values are data typed in many different languages when manipulated within programs.*	*Knows that processors have instruction sets and that these relate to low-level instructions carried out by a computer.*	*Knows the purpose of hardware and protocols associated with networking. Understands the client-server model including how dynamic webpages use server-side scripting and that webservers process and store data. Recognises that persistence of data on the internet requires careful protection of online identity and privacy.*	*Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. Considers the properties of media when importing them into digital artefacts, develops success criteria and Collects user feedback in order to make appropriate refinements.*
8	Designs a solution to a problem that depends on solutions to smaller instances of the same problem (recursion).Understands that some problems cannot be solved computationally.	Designs and writes nested modular programs that enforce reusability utilising sub-routines wherever possible.Understands the difference between 'While' loop and 'For' loop, which uses a loop counter. Understands and uses two-dimensional data structures.	Performs operations using bit patterns e.g. conversion between binary and hexadecimal, binary subtraction etc. Can explain the need for data compression. Knows what a relational database is and understands the benefits of storing data in multiple tables.	Has practical experience of a small (hypothetical) low level programming language. Understands and can explain Moore's Law. Understands and can explain multitasking by computers.	Understands the hardware associated with networking computer systems, including WANs and LANs, understands their purpose and how they work, including MAC addresses.	Understands the ethical issues surrounding the application of information technology, and the existence of legal frameworks governing its use e.g. Data Protection Act, Computer Misuse Act, Copyright etc.

Homework Policy

The Computing Department embrace homework as a useful tool for securing mastery of key skills and knowledge and to develop students' independent learning in preparation for revision and GCSE. All students are encouraged to, and re-taught how to, revise in the lead up to testing weeks and mock weeks.

Years 7, 8 and 9

In years 7, 8 and 9, students are set homework weekly. This homework is carefully designed to:

- Prepare students for future learning, such as pre-reading,
- Secure and embed learning that has taken place in the classroom,
- Develop students' ability to work independently,
- Allow students the opportunity to complete extended projects,
- Develop their cultural literacy
- Apply the skills learnt in the classroom to extended tasks, such as longer essay writing
- Prepare and revise for testing

Years 7 & 9

Every Monday Week A, students are set a multiple-choice quiz via Microsoft Forms, which will link to current and prior learning. Week B homework is 'For the love of' which investigates wider aspects of computing and helps students draw links between classwork and the real world.

Year 8

Every Monday Week B, students are set a multiple-choice quiz via Microsoft Forms, which will link to current and prior learning. Week A homework is 'For the love of' which investigates wider aspects of computing and helps students draw links between classwork and the real world.

'For the love of'

Once completed, homework is celebrated, shared or used in class. Great examples of homework are further celebrated by being displayed on walls and in assembly. Students learn, practise and revise key spellings as part of their homework to support the accuracy of their written responses and to secure their knowledge of key spelling rules.

Year 10 and 11

Every Monday Week A, students are set a multiple-choice quiz via Microsoft Forms, which will link to current and prior learning. Week B homework is tailored by the individual class teacher to suit their class's particular needs. Students in years 10 and 11 are set weekly revision challenges to support their understanding of the GCSE exam questions with a focus on extended writing. These include:

- Redrafting content produced in class in response to feedback
- Drafting and completing coursework or programming assignments
- Preparing for programming assessments
- Accessing content on GCSE Pod
- Completing learning cycles and quizzes on SENECA
- Reading critical theory linked to their GCSE texts

Students will complete homework in via MS Teams and OneNote so that this can be used in lessons and to promote the use of homework as an ongoing learning tool to support what is happening in the classroom.

Impact of homework is reviewed half termly by the homework lead to ensure it is of a high quality, is useful to the student and their learning, is valued and is being set consistently.

Period 6 Expectations

All teachers will run period 6 sessions for their KS4 classes, and invite KS3 students to period 6, on the following afternoons:
Year 11: Monday
Year 10: Wednesday
Year 7-9: Thursday

Teachers are expected to analyse their own class data after each key assessment entry and use this to inform the focus of their period 6 sessions and the students to be invited.

The year 11 students targeted for priority intervention must be targeted through period 6 sessions as well as in-class intervention where appropriate. The intention of period 6 sessions is that small numbers of students are targeted to ensure a tight focus on mastery of skills. Sessions should be structured to allow for clear, measurable progress, with the first session requiring students to complete a pre-assessment task (this may be the mock exam if students have recently completed one) and the final session requiring students to complete a final assessment task. Both of these should be marked by the period 6 teacher to provide them with a clear overview of student needs and to identify where progress has been made.

The use of Threshold Testing in year 11 will identify students requiring intervention and those not achieving 65% in the assessment will be required to attend catch-up and resit the threshold test.

Students should be focused and on task during period 6 sessions, as they would be within a typical Computing lesson. Teachers are encouraged to use the rewards and consequence system as they would in an ordinary lesson. All teachers must update the period 6 register each afternoon so that the impact of these sessions can be analysed.

Sessions may be delivered across mixed groups with individual teachers focusing on different skills for mastery. This will be decided as data trends appear throughout the year.

Red Lines (these are likely to change in view of Great Learners and will be revised)

Red Lines are whole school minimum expectations to ensure quality teaching and learning consistently take place. Do

Do Now Task: This is a task that takes place at the start of the lesson and is used to embed prior learning or introduce new learning. For KS3, this may be recapping knowledge, retrieval practice, prediction for a piece of code. For KS4, this consists of a low stakes quiz interleaving the skills and knowledge for GCSE. This may be a question and answer-based quiz, a connections map, retrieval practice.

Directed Questioning: In all cases, students are required to respond formally during directed questioning, wording their response precisely and with appropriate subject terminology. Teachers model this to students, support students in refining their responses and insist on responses being repeated appropriately.

Modelling: All assessment tasks are completed by the teacher, to the highest standard expected of the class, prior to teaching. Models are used within the scheme of learning to exemplify expectations to students. Live modelling is integral to the high-quality teaching and learning that takes place within the department. In many cases this will be delivered using screen sharing with teachers modelling metacognition, outcomes, processes and approaches to tasks, and deconstructing student and existing responses.

Oral Rehearsal: Students are expected to discuss their responses to questions before they are shared as a class to ensure responses are well-considered, developed and refined. Full sentences are expected with technical vocabulary.

Take 5: In all lessons, students independently complete the 'Summary/Questions' box on OneNote and complete a self-assessment. This may be to summarise learning, respond to key questions, ask their own questions or to complete another plenary task to demonstrate their learning and future needs. Take 5 may not necessarily take place at the end of the lesson, but at a convenient point to assess the learning so far.

Assessment for Learning (AFL): As part of the department assessment policy, AFL takes place within lessons as part of in-flight marking and feedback. Students' learning is regularly assessed through questioning, summarising and 'Take 5' tasks, live marking and conversations with students. Screen sharing is used to show student work in a live setting for teachers or students to exemplify effective features and explore how to improve assignments. Whole class feedback is used to exemplify great examples to students, highlight common errors and areas to improve, and inform next steps in teaching.

DIRT: students are expected to reflect on what they have done to improve their work and how. This can also incorporate peer assessment and feedback.

Students Read Aloud: Students are expected to read aloud to the teacher, in groups or to the whole class. Teachers use this as an opportunity to assess reading fluency.

Teachers Model Reading: Teachers model reading especially complex texts which students are expected to follow. This ought to be followed up by Direct Questioning.

Vocabulary Glossary

Core

Keyword	Definition
Systems Architecture	
CPU	*Central Processing Unit. - The "brain" of the computer*
CU	*Control Unit. - Part of the CPU that manages the functions of all other parts of the CPU*
Decoder	Part of the CU which decodes the binary instructions fetched from memory
RAM	*Random Access Memory - The main volatile memory into which programs are loaded from the hard drive*
MAR	*Memory Address Register - Small fast memory used to store the RAM address of the next instruction*
MDR	*Memory Data Register - Small, fast memory used to store the information collected from the RAM before processing*
PC	*Program Counter - Keeps track of the current instruction number of the program*
Accumulator	Small, fast memory, used to keep track of the data currently being processed
ALU	*Arithmetic and Logic Unit - Does the basic mathematics and comparisons during processing*
Bus	A physical connection between two elements of a computer system that allows the transfer of data.
Cache	Incredibly fast, but very expensive volatile memory using in the CPU
Bridge (North / South)	Junctions on a motherboard where the bus connections are controlled and routed. Northbridge deals with core functions, whilst the Southbridge deals with the peripherals, input and output devices and Secondary Storage.
von Neumann Architecture	The method used by all modern computers to allow the programming of a machine to be changed depending on the required function.
Fetch / Decode / Execute Cycle	Basis of the von Neumann architecture – the repeated process where instructions are fetched from RAM, decoded into tasks and data, then carried out.
Clock Speed	The number of FDE cycles that a CPU can carry out per second. Measured in Ghz. (1 Ghz = 10^9cycles per second or 1,000,000,000hz)

Cores	Some processors have multiple CPUs which can work in parallel, sequentially or can multitask. Dual and Quad cores are common in modern PCs
Machine Code	A program, stored in binary, that the CPU undertakes the F-D-E cycle on. All programs must be in machine code to work
Instruction	A single line of machine code, containing the command and data location on which it is to be executed. Stored in binary
Opcode	The first part of the instruction, is the command
Operand	The second part of the instruction is the data on which to carry out the command. This may be actual data stored in binary form, or a memory location reference of where to find the data
Memory	
Volatile	Memory which requires constant electrical charge. If the power is turned off, then the data is lost
Non-volatile	Memory which can retain its data when the power is turned off
RAM	*Random Access Memory*
ROM	*Read-Only Memory*
Cache	Very fast memory, on, or very close to the CPU
Virtual Memory	A section of the HDD which can be used as RAM for very memory intensive processes
Flash Memory	A type of dynamic (changeable) ROM
Boot Process	The instructions needed to start the computer and to initialize the operating system.
POST	*Power On Startup Test* A series of checks done on the hardware of the computer to ensure the machine can run.
Storage	
Secondary Storage	Primary storage is RAM. Secondary storage refers to long term, non-volatile data storage.
Non-volatile	Memory which can retain its data when the power is turned off
Magnetic	Data is stored by altering the magnetic charge (+ or -) to represent binary information
Optical	A reflective layer or dye is marked to either reflect or not reflect a laser beam. The computer reads the reflections as binary data

Solid State	Also known as *Flash Memory,* the data is stored by forcing (or flashing) electrons through a barrier into a storage layer. Here it is read as binary information
Capacity	How much data will it need to hold?
Speed	How quickly must the data be written / read?
Portability	Does the storage device need to be transported? If yes, then size, shape and weight are important. Will it require other devices to be used (eg. An optical reader).
Durability	How *robust is* the device? Can it be moved without fear of damage? Will it be used in a difficult environment? Does it need to be single use or rewritable?
Reliability	Does it need to be used repeatedly without failing, or will it receive minimal reuse? Will it need to store the information for long periods of time?
Cost	Needs to be compared with the above and considered.
Wired & Wireless Networks	
Stand Alone	A single machine, not connected to another
Network	A collection of machines which can communicate with one another
Transparent	The end-user has no need to know the specifics of a network's infrastructure
Node	A device on a network (PC or other device)
Link	The connections between nodes
LAN	Local Area Network (Single location)
WAN	Wide Area Network (Multiple connected locations)
VPN	Virtual Private Network
UTP	Unshielded Twisted Pair – a type of cable
Client	The user machines on a network
Server	The central 'controller' machine on a network, including main data storage
P2P	Peer-2-Peer. A network without a server.
WAP	Wireless Access Point
NIC	Network Interface Controller
Router	Controls the sending of data around a network

Hub	A central connection for a small network, which broadcasts all data to all clients
Switch	A smart hub for larger networks which only sends the data to the intended client
Internet	A worldwide collection of networks
DNS	Domain Name Server
Hosting	Storing a file on a web-server for access via the internet
Cloud	A service which is stored remotely
Networks & Protocols	
TCP/IP	Transmission Control Protocol / Internet Protocol. These are the standards that allows network nodes to communicate with one another on the internet
WWW	World Wide Web - Pages of content
email	Electronic mail, sent through the internet
URL	Unique Resource Location
Protocol	The rules and standards that are agreed in order to make it possible for different devices to talk to one another
IP Address	Each node on a network is given a unique 32 bit address (4x8bits) for example 192.168.0.1 There are 4 billion possible combinations.
DHCP	Dynamic Host Configuration Protocol – this protocol allows the network server to control the allocation of IP addresses
MAC Address	Media Access Control
	Unique addresses hard coded into the network interface controller. Gives the manufacturer, NIC type and unique identifying number. 48 bits displayed as Hex (eg 01-23-45-67-89-ab-cd-ef)
TCP/IP	A set of protocols that governs the transfer of data over a network
HTTP	Standards for writing webpages to display content for display
HTTPS	*Client-server protocol for requesting (client) and delivering (server) resources, such as HTML, securely*
FTP	*Used to directly send files from one node to another over the internet. Commonly used for uploading files to webservers*
POP	Used by email clients to download email from the remote email server and save it onto the users computer. More or less redundant now, and has been replaced by IMAP

IMAP	An alternative to POP, allowing more control such as the complete control of remote mailboxes
SMTP	An old standard for transmission of email. SMTP can only be used to *push* mail to client machines, whilst both POP and IMAP ae used by clients to *retrieve* mail.
Protocol	The rules and standards that are agreed in order to make it possible for different devices to talk to one another
Layering	Rules organised into a distinct order in which they need to be applied
Interoperability	The ability for different systems and software to communicate, exchange data and use the information exchanged
Encapsulation	Enclosing data inside another data structure to form a single component
De-encapsulation	Removing data from inside and encapsulated item.
Systems Security	
Hacking	Attempting to bypass a system's security features to gain unauthorised access to a computer
Malware	Malware is malicious software, loaded onto a computer with the intention to cause damage or to steal information. Viruses are a type of malware
Phishing	Phishing is a common way to try to steal information like passwords. Emails are sent, requesting the user logs into a website, but the site is a fake, and the users details are logged
Social engineering	People are the weakest point of any system. If a hacker can convince a user to give over their data, this is the easiest way into a secure system
Brute force attack	Using and algorithm to try every possible combination of characters to 'guess' the users password.
Data interception	Data interception, or *Man in the Middle attacks* are hacks that use 'packet sniffer' software to look at every piece of data being transmitted in the local area to find ones that meet the hacker's criteria. Often done by creating 'fake' wireless networks to record users details
SQL injection	Using SQL statements to trick a database management system (DBMS) into providing large amounts of data to the hacker
Denial of Service Attack	Hackers flood a network with huge amounts of fake data and requests in an attempt to overload the system so that it crashes
Penetration Testing	Employing a *white hat hacker* to try to break into a system to test how good the security is. Any problems in the security can then be fixed before they become vulnerable to real attack

Network forensics	Network procedures that capture, record and analyse all network events to discover the source of security attacks
Network Policies	Rules which govern how a network may be used – see over page
Anti-malware software	Software which analyses files, network traffic and incoming data to look for known malware such as viruses or worms. An infected file is quarantined, and either cleaned or securely deleted to prevent further infection. Needs updating very regularly to ensure that the newest malware is being checked for
Firewall	A firewall protects a system by checking all incoming and outgoing network traffic is legitimate
User level access	Limiting the access of a user by their requirements to carry out their job. An admin will have more rights than a student, for example. Often even admins do not give themselves full rights to prevent accidents and will instead have a *super-user* account that will be used only for special high level jobs.
encryption	Encoding all data with a secure private, asymmetric key system, so that if data is stolen, it cannot be read or used.
Virus	A program designed to infect a computer, then copy itself. Requires human 'help' to spread; usually through infected software being installed or spread through unsecure removable media such as usb-drives
Worm	A self-replicating program, which can run itself allowing it to spread very quickly
Trojan Horse	A program which disguises itself as legitimate software, and appears to perform one task, but is actually performing another
Ransomware	Ransomware secretly encodes a users data and files, then offers to un-encode the files if a large amount of money is paid to the hacker
Rootkit	A rootkit allows a hacker to gain full, and often repeated, control of a computer, including the host operating system, which helps the hacker avoid detection.
Network Policies	
Acceptable Use	Governs the general use of the computer system and equipment by employees. Usually limited to that which is required to carry out only the tasks that a user is employed to undertake
Passwords	Rules to ensure that passwords are strong enough to prevent guessing or brute force attack - often requiring the use of upper- and lower-case letters, numbers and special characters. Also, usually a minimum length is required. Passwords usually have to be changed on a regular basis
Email	Governs what may and may not be sent by email
Web Access	The configuration of web browsers may limit the types and categories of website that can be accessed
Mobile Use	What devices are and are not allowed to be used

Remote Access	Govern what can be accessed from outside the system, and what can only be accessed onsite
Wireless	Govern how wireless access points (WAPs) are secured, who has access, and under what circumstances
Software	Governs who can install software, and which users are able to use that software. May have different levels of access once inside the software
Server	Rules about what services are provided by the institution and who may access data stored centrally and for what purposes
Back Up	Back up policy determines how frequently backups are undertaken, and what type of back up (full, incremental, differential). It will also state where the backup media must be stored and for how long. Often a full weekly back up is required to be stored in a fire proof box in an offsite location
Incident Response Plan	Details what to do if something goes wrong, or if an attack is discovered.
Systems Software	
Operating systems (OS)	Collections of programs that tell the computer hardware what to do.
User interface	The means of communication between the user and the computer. These are typically either *command line* or *GUI.*
Command Line	The simplest form of user interface where users type commands into a prompt
Graphic User Interface (GUI)	Most modern computers have a GUI, which uses icons to represent the programs and files. The user runs the programs through a touchscreen or mouse-controlled pointer
Voice Command	Increasingly users can speak commands to devices such as Google Home and Amazon's Alexa
Memory management	The OS controls available memory, moving programs to and from secondary storage to RAM
Multitasking	Often users have more than 1 program running at once. Each CPU core can only carryout 1 task at a time, but the OS alternates between the programs to make it appear that multiple tasks are running simultaneously
Peripheral management	Computers must communicate with a range of external devices such as printers, monitors and scanners (peripherals). The OS uses *drivers*to correctly pass data to the device and ensure correct function.
Drivers	A driver is a piece of software which provides communication between the CPU and a peripherals device

User management	Multiple users can have accounts on the same computer, each with their own files, settings and applications, protected with passwords. The OS will ensure that only users who are granted permissions can use files or programs belonging to other users.
File management	Computers store files and data in hierarchical folder systems. This is efficient and allows for quick navigation
Utilities	Utility software supports the OS by performing a limited and specific task. They are used to manage specific actions of the system or undertake maintenance operations.
Encryption software	In order to keep data secure, especially against outside threats, data must be encrypted. Encryption software uses complex algorithms to encode data so it cannot be read without the private access keys.
Disk Defragmentation	Over time, through multiple updates and saves, files will become split up and distributed over the platters. It takes longer for the files to be accessed, slowing the machine down. Defragmentation reorganises the files' parts to bring them together. See fig 1.
Data Compressions	Allows files to be made smaller by removal of empty space or through compression algorithms (lossy or lossless) – see KO2.6b
Back Up	In case of hardware failure or other computer problems, data should be copied to external media so that it can be restored if lost or damaged.
Antivirus	Continually scans the system to find, quarantine, and clean any file infected with viruses.
Anti-malware	Continually scans to identify any malicious software from being introduced to the system.
Ethical, Legal, Cultural and Environmental	
Ethical	Relates to *right and wrong* but in a moral sense than a legal issue. For example, there is nothing to stop you legally from using Facebook to stalk an ex-partner, but whether it is *right* to do so, is an ethical issue
Legal	There are certain laws set by government that control how computers can be used – see box
Cultural	These issues relate to society and how technology can affect religious, or social ideas. If people spend all their time on their phones rather than talking face to face, this is a cultural issue
Environmental	How computing impacts on the global and local environments. This might be waste production, or mining to gather resources needed to make phones, or using renewable energy to charge phones, or recycling projects. Companies want to be seen to be 'green'.

Privacy	Privacy is a very important issue. A person's right to privacy is very important and there is strong law, alongside ethical guidance that govern how companies can use our data
Stakeholder	Anyone that is impacted on, in any way, by a technology. They have a vested interest
Open source	Software that is created and shared with the source-code able to be seen. Users are free to make alterations to the source-code to meet their own needs, or to improve the system for everyone
Proprietary	Software that is created but the source code is locked. This is often sold, and the company wants to protect its intellectual copyright
Legislation	Laws that relate to a certain area
Algorithms	
Algorithm	An abstracted program which completes a given task, whatever the data provided
Search	Searching is looking through data, making comparisons with a search term, until the algorithm either finds the data, or identifies that it is not present.
Sort	Putting given sets of data into specified order – usually ascending (alphabetical) or descending (reverse alphabetical)
Linear Search	A type of search where the computer checks every variable, in order, until it finds the search term. Potentially very slow.
Binary Search	A search type based on repeatedly halving the searchable data, until the search term is found
Bubble Sort	A method of sorting data which looks at pairs of variable, and swaps them around if out of order. This continues until there are no more swaps to be made
Merge Sort	Splits the data into increasingly small segments, until single data points are reached, then reassembles the data structure one item at a time.
Insertion Sort	Checks through the data until finding the first incorrectly places item. The algorithm then checks all the previous places to see where the data fits, before inserting it into this slot.
Programming Techniques	
Abstraction	Abstraction is moving a problem out of the specific in order to create a general solution that would work in similar scenarios. Ignoring the gritty details to focus on the problem
Decomposition	Breaking a problem down into smaller, computational solvable chunks
Pseudo Code	A structured way of planning code, which is 'computational' in style (uses Boolean logic, variables, comparisons and arithmetic for example) but is not tied to a strict high-level language's syntax
Flow Diagram	A diagram made using specific shaped boxes, that mocks up the flow of a program through various stages, processes and decisions.
Program Control	Using Boolean logic to guide the computer through a program based on decisions

Comparison Operators	The symbols used to look at a variable or piece of data in relation to is similarity to another piece of data or variable
Arithmetic Operators	The symbols used to show the mathematics to be carried out on a piece of data.
Robust Programs	
Defensive design	Planning a program from the very beginning to prevent accidental or purposeful misuse
Input sanitization	Removing erroneous data from a system prior to processing
Data validation	Ensuring all data is in the correct format prior to processing
Contingency planning	Having built in checks and outcomes based on what happens when things go wrong
Anticipating misuse	Building programs which do not allow a user to deliberately break the system
Authentication	Having different levels of user, and preventing everyday users from being able to significantly change a system
Maintainability	Building software which is modular to enable sections to be updated and replaced without having to write the whole program again from scratch
Code comments	Annotating code so that the person maintaining or working with your code in the future is able to understand your thought process
Indentation	Making code more readable by laying it out in a manner that keeps sections of code separate
Iterative testing	Step by step testing to ensure that small sections of the code work, before new parts are added and then retested. Important to allow *traceback* to find what caused any errors
Terminal testing	Significant testing done once a program is complete under a range of conditions and on multiple hardware – often called *Alpha Testing*
Beta Testing	Making a small release of the software to a group of tech-literate enthusiasts to broaden the usage-testing and get lots of feedback prior to full release.
syntax error	An error in the typing of the code. Missing punctuation, spacing etc
Test data	Data chosen to test the program. Testers use a specific range of data
Computational Logic	
Logic	A system designed to perform a specific task according to strict principles.

Logic Gates	The physical switches inside an electronic device which can perform the calculations a computer needs to carry out on electronic signals
Truth Table	A tabular representation of the possible inputs and outputs from a given logic gate, or collection of gates
Boolean	Mathematical *TRUE or FALSE*
Operator	A mathematical symbol in computing
+	Addition [1+2=3]
-	Subtraction [2-1=1]
/	Division [5 / 2=2.5]
*	Multiplication [2 * 2 = 4]
^	Exponentiation, raising a number to the power of… [3^3 = 3 * 3 * 3 = 27]
MOD	Modulus division. To divide a number by another, but only return the *remainder* [10 MOD 3 = 1]
DIV	Integer Division. To divide a number by another, but only return the *number of full sets.* [10 DIV 3 = 3]
Languages	
Low Level Language	A programming language which is closer to binary than English
High Level Language	An abstracted programming language which is closer to English than binary
Instruction Set	Binary code which tells the computer hardware what to do – OpCode and Operand
Machine Code	1 to 1 instruction coded in mnemonics (STO, ADD, MOD, DIV etc) which must be converted to binary to run
Abstraction	Removing a level of detail to allow focus on the problem solving rather than the specifics. *Python, and all other High-Level languages are abstracted. You do not need to know the machine code to get something to happen*
Translator	A utility to convert High Level Code into binary machine code so it can be executed
Interpreter	A utility which translates High Level code on a line-by-line basis and executes the program as it goes in a special test environment
IDE	Integrated Development Environment
Text Editor	A place to type code, focused on the content of the file, not the look of the file

Error Diagnostics	To test a program and provide feedback to the coder so that errors can be fixed
Run Time Environment	Part of an IDE which allows a piece of code to be tested without installation
Data Representation	
Denary	Base 10 number system. Uses digits 0,1,2,3,4,5,6,7,8,9
Binary	Base 2 number system. Uses digits 0 and 1 only.
Hexadecimal (Hex)	Base 16 number system. Uses characters 0-9 and A,B,C,D,E and F
BIT	Contraction of BINARY DIGIT – a single value of 0 or 1
Binary Code	Representation of values using multiple bits
Character Set	A list of unique values, stored in binary, which represent the letters, numbers and symbols a computer can show/use.
ASCII	American Standard Code for Information Interchange.
	A character set which uses 7 bits to store 128 (2^7) characters
Extended ASCII	A character set which uses 8 bits to store 256 (2^8) characters
UNICODE	A characters set which uses 16 bits to store 65,535 characters (2^{16})
INTEGER	A whole number (value written to 0 decimal places)
FLOAT	A decimal value
Conversion	Moving a value from one data type/representation to another, for example Binary to Hex
Exponent	Mathematical term which tells you how many time to multiply a BASE by itself.
Overflow Error	Where the denary value cannot be represented with the given number of bits.
Binary Shift	The method for multiplying and dividing numbers in binary. Is not necessarily mathematically correct
Most Significant Bit	The left-most bit in a binary number – it has the highest value
Least Significant Bit	The right-most bit in a binary number – it has the lowest possible value = 0 or 1
Check Digits	Bits used to ensure that the value sent digitally is not corrupted on transfer
Lossy Compression	Data is removed from the file to make it smaller. This data is lost and cannot be regained. Suitable where the loss of data is likely not to be noticed. Eg images

Lossless Compression	No data is lost, but rather rearranged to ensure a perfect version of the data can be returned. Used where exact reproduction is vital. Eg text documents
JPEG / JPG	Joint Photographic Experts Group
	Compression for images – lossy
GIF	Graphics Interchange Format
	Lossless bitmapped image format for limited colours.
PDF	Printable Document Format
	Open standard for reproducing text and graphic documents without editing permissions – lossless
MPEG	Moving Pictures Expert Group
MP3	Moving Pictures Expert Group Audio Layer 3

Programming

2D array	A static data structure that holds data both horizontally and vertically. The structure is fixed and each element has the same data type.
2D list	A dynamic data structure that holds data both horizontally and vertically. The structure can change during program execution and the data types of the elements can be different.
Algorithm	A series of instructions that end when the problem is solved.
Append	Adding to an existing data structure.
Argument	The values held in the brackets of a subroutine call. These are passed into a subroutine via the parameters.
Arithmetic expression	An expression that results in a numeric value.
Array	A fixed (static) data structure that holds items of the same data type under one name.
ASCII	Acronym for American Standard Code for Information Interchange. It is used to represent characters with a numerical value.
Assembler	An assembler translates assembly language into machine code.
Assembly language	A language that replaces machine code with mnemonics and operands to make them easier to read / write.
Assignment	Assigning a value to a variable.
Attribute	Properties or characteristics of an entity. E.g. player name, player score

BIDMAS	Acronym used to show the order of operations in an arithmetic expression. Brackets, Indices, Division, Multiplication, Addition and Subtraction. Add and subtract are interchangeable and should be read from left to right.
Boolean data type	A value that is either True or False.
Boolean expression	An expression that evaluates as True or False. Also known as a logical expression.
Boolean operator	An operator used in a Boolean expression. For example AND, OR and NOT. Also known as logical operators.
Boundary data	Data that should be accepted by a program. It tests the data right at the boundary of a range.
Caesar cipher	Named after Julius Caesar. A Caesar cipher is one of the oldest and simplest forms of encryption that involves shifting letters of the alphabet by a defined amount to create an encrypted message.
Character	A single character of string.
Comparison operators	An operator that is used to compare one operand to another. For example, < >.
Compiler	A compiler creates an executable file for a program by translating a high level language to machine readable code.
Concatenate	When two or more strings are joined together.
Condition	Used to control the flow of execution in a program. A condition contains a logical expression.
Constant	A constant is a value that cannot be changed during the execution of a program.
Control flow	The order in which instructions are executed in a program.
CSV	Acronym for comma-separated values. It is a plaintext data file where each value is separated by a single comma.
Custom built function	A function that you have created yourself and imported into other programs that you have created.
Data file	A file that can be accessed and modified by a program.
Data pairing	In a dictionary, a data pairing is when a key (the attribute identifier) is paired with the data.
Data structure	Used to store data in an organised and accessible way.
Data validation	A check performed on data input to ensure that it can be accepted by the program without causing an error.
Database	A structured and organised method for storing data. A database holds multiple records.

Decision symbol	Used on a flowchart to represent a condition.
Declaration	Declaring a variable as a specific data type.
Decomposition	Breaking down a problem into smaller sub-problems to make the more manageable.
Dictionary	A data structure that involves creating data pairings that can be located using a key.
Element	A character in a string or an item in a sequence.
Entity	An entity is a single object, place, person or thing. E.g. player
Erroneous data	Data that should not be accepted by the program or it will cause an error.
Error messages	Used for finding errors in your program. They pinpoint lines of code that contain errors and provide details about them.
Execute	Carrying out the instructions for a computer program.
Execution	Carrying out the instructions for a computer program.
Expression	An expression is a collection of operands and operators that can be evaluated.
Field	Also known as an attribute. It is the properties or characteristics of an entity.
Final testing	Testing a program at the end of its creation.
Flowchart	A visual representation of an algorithm or program.
For loop	An iterative statement that will repeat for the length of a given sequence.
Function	A subroutine that returns a value.
Function call	A statement used to execute a function.
Global variable	A global variable can be accessed and modified from anywhere in the program.
GUI	Acronym for Graphical User Interface. It is an event driven program that allows the user to interact with it in a variety of ways. For example, buttons and icons.
guizero	A third party library that can be imported into Python to create a GUI.
High level language	A human readable language written in formal, structured English.
IDE	Integrated development environment. This is a place to write programs that provides support with debugging and diagnostics.
Import	The keyword that enables a module to be brought into our programs.
Index	The location of items or elements in a list, array or string.

Initialisation	Assigning an initial value to a variable to let the compiler know that a memory location is required.
Input()	A function that prompts the user for input.
Integer	A value that is a whole number.
Integer division	In integer division there can be remainders because the resulting value will be a whole number. For example 7 ÷ 3 will calculate as 2.
Interface	A term used with subroutines to describe how it will interact with the program. It refers to the subroutine identifier, parameters, order of parameters and the return values.
Interpreter	An interpreter translates and executes code line by line. It translates the code into machine readable code.
Iteration	Repetition of code blocks. For example, a while loop.
Iterative testing	Testing a program during its creation.
Join method	The join method takes a list and joins each value in that list into one string.
Key	In a dictionary data structure, a key is used to identify each attribute held in the dictionary.
LED matrix	A group of LEDs placed in a grid structure.
Library	In Python, the library contains built-in modules that provide access to system functionality such as file i/o.
List	A dynamic data structure that holds items under one name. The items can be of varying data types.
Logic error	The program will run, but won't do what the programmer expected. These are tricky to spot as they are not picked up by the IDE.
Logical expression	An expression that evaluates as either True or False.
Logical operator	An operator used in a logical expression. For example AND, OR and NOT.
Low level language	This can be quickly executed by a computer. It is written in either machine code or assembly.
Machine code	A program written using 1s and 0s. A computer can execute this directly.
Meaningful identifiers	Naming a variable or data structure using a sensible name that can be easily recognised and remembered.
Method	A function that belongs to an object.
Mnemonic	A code to help us remember something.
MOD / Modulo	Calculates the remainder of a division. For example 7 MOD 3 will calculate as 1.

Module	In Python, a module is a file containing Python definitions and statements. The functionality of these definitions and statements is then available to be made use of.
Naming convention	A unified standard for naming things in a chosen programming language.
Naming conventions	Following the guidance in the programming language documentation about naming structures.
Nested selection	A selection block placed within another selection block.
Normal data	Data that should be accepted by a program. This is data that you would expect a user to enter.
Operand	A piece of data that can be changed
Operator	A symbol or function that performs an operation. For example +.
Parameter	Used in a subroutine to allow values to be passed into them.
Pixel	A single element of an image on a computer screen.
Procedure	A subroutine that executes a block of code when called. It does not return a value.
Pseudo-random number	This type of random number is generated using mathematical algorithms which are computer-generated and therefore highly predictable.
Pseudocode	Informal steps for an algorithm using structured English.
Pythonista	A programmer that uses Python as their desired programming language.
Real / Float	A decimal number.
Real division	In real division there are no remainders because the resulting value can be a decimal number. For example 7 ÷ 3 will be calculated as 2.3333333333333335.
Record	A collection of attributes for a single entity.
Return value	A value that is returned by a function.
Robustness	A program is robust when it does not produce any errors during execution.
Scope	The scope of a variable is the section of the program where the variable can be accessed and modified.
Selection	Controlling the flow of execution in programs using if statements.
Sense HAT	Hardware that attaches to the top of a Raspberry Pi computer and allows you to write programs that collect data from sensors. The Sense HAT also has an LED matrix for displaying output.
Sense HAT emulator	A digital representation of the physical Sense HAT device.

Sensor	A tool that collects data.
Sequence	The sequence of a program is performed from top to bottom, executing each line in turn.
Split method	The split method takes a string and splits it when it finds a defined character. The result will be held as a list.
Spreadsheet	A document where data can be arranged in rows and columns. A spreadsheet can be used to sort and perform calculations on data.
String	A value that is text. This can include numbers but they will be read as text.
String handling	Performing operations on string.
Structure chart	A top-down diagram used to design the structure of the subroutines required for completing a program.
Structured programming	A programming paradigm where sequence, selection, iteration and subroutines are used to control the flow of execution. Each block of code in a structured program has a single entry point and a single exit point.
Subroutine	A sequence of instructions to perform a specific task with an identifiable name.
Substring	Part of a string.
Success criteria	A list of key elements required in a programming solution based on the scenario or user requirements.
Syntax	In programming, the language specific code that you write in has its own syntax. The syntax is unique to that programming language.
Syntax	The formal method used to structure code in a given programming language.
Syntax error	An error where the code has been structured incorrectly and the syntax rules haven't been followed.
Tabular format	Displaying data in a grid of rows and columns.
Terminator	Oval shapes used to show the start and end of a flowchart.
Text file	A file stored on a computer that contains plain text.
Third party libraries	Code that has not been written by the Python developer or you that can be imported into your programs to save you from writing them yourself if you don't have the time or necessary skills.
Trace table	An error checking method that steps through each line of code in a program and records the state of the variables and conditions.
Translator	This executes the programs that programmers write in high level languages.

Traverse	Move through a sequence.
True-random number	This type of random number is generated using unpredictable physical means such as atmospheric noise.
Truth table	A table that lists the outputs of all possible input combinations.
Try and except	A data validation check to see if the data entered can be accepted by the program. If a defined error occurs the user will be prompted with a warning.
Variable	A value held under one name.
While loop	A loop that will continue to iterate whilst its condition evaluates as True.
XOR	A Boolean / logical operator.