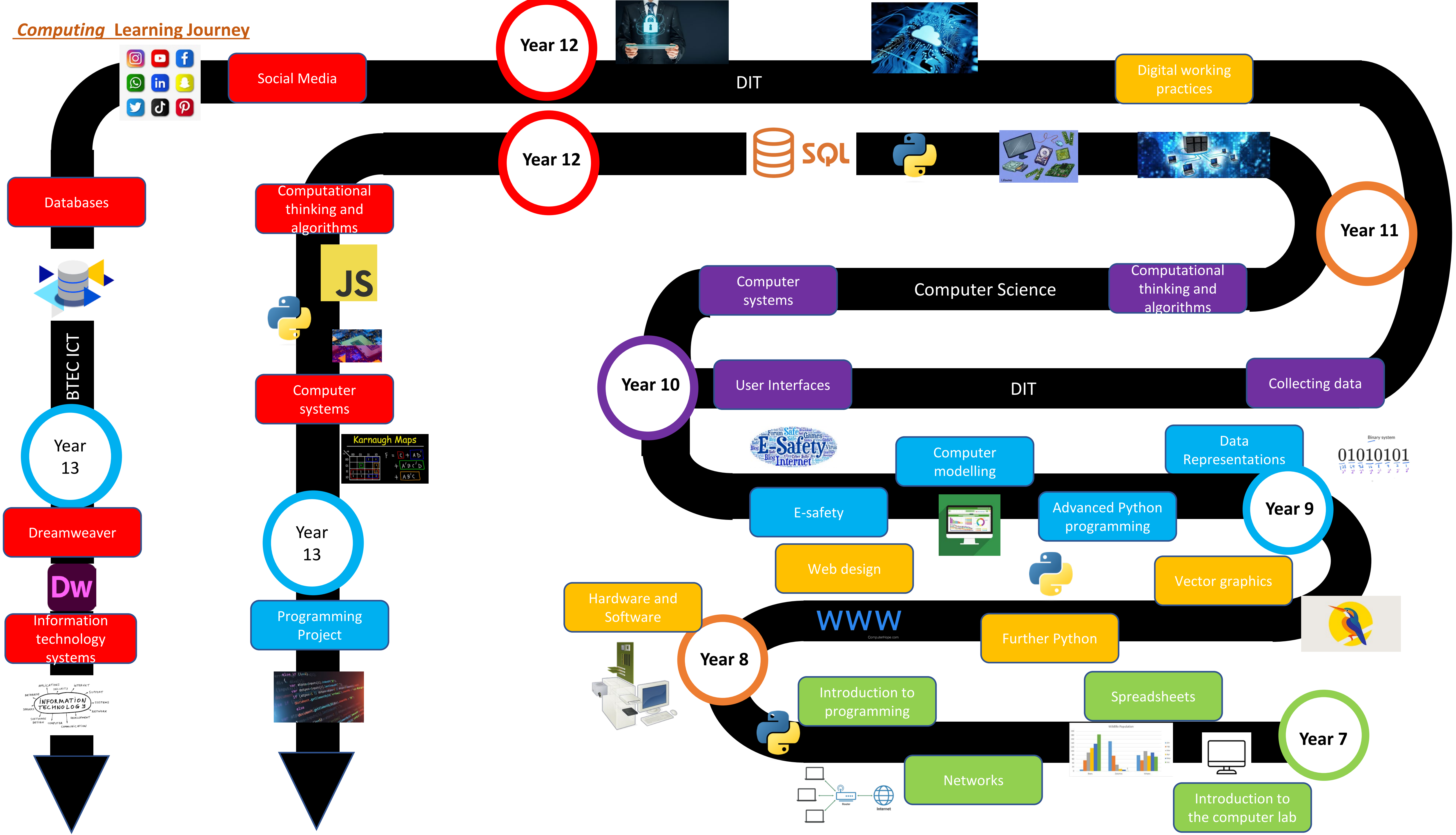
# IDSALL SCHOOL

## Computer Science Curriculum

**Our vision for Computer Science GCSE:**

- Computer Science is a dynamic and rapidly growing field of study and has quickly become an integral part of modern society.
- Through our curriculum we aim to develop students' computational thinking and encourage their creativity, enabling them to decompose complex problems and find efficient solutions.
- We equip students with a deep understanding of computational theory and emerging technologies and provide them with a rich experience in practical programming so that they can bring their ideas to life.
- Our students leave us as digitally literate individuals who are well-prepared to engage with and thrive in, an ever-changing technological world.

## Computing Learning Journey



**YEAR 10 CURRIULUM OVERVIEW – *Computer Science***

**The Big Picture- Intent:**

Year 10 Computer Science has been designed to maximise progression. Topics covered revisit prior learning, as well as enabling students to deepen their understanding of the core concepts of computer science. Topics will be taught from both papers concurrently, maximising chances to revisit more challenging content in retrieval practice. New content and topics are also introduced throughout year 10. Computer programming is taught throughout year 10 to overcome the forgetting curve when students have lengthy periods of time without utilising these skills.

All students will be able to access the main content of all lessons and all students will be taught to the top with scaffolding, adaptive teaching and stretch and challenge provided where necessary.

**Implementation**:
When delivering programming content, teachers will model coding where the core skills are introduced. Students will then be given a similar task to complete after the demonstration and may be given example code to copy and then edit. During Students will then be given a real world concept scenario that require them to apply the skills learnt. This will help solidify this knowledge, and also help students understand when and where to use programming techniques in solving computational problems.

Lesson sequences have been carefully chosen to ensure that students have the required background knowledge to fully understand and apply skills in relation to the topic. Therefore the lesson sequence may not match that of the exam specification, however all content is covered.
This is especially important for paper 2, where students need concrete understanding of the key programming techniques before applying these to producing GCSE Algorithms.

Lessons follow a consistent format beginning with a retrieval practice activity in the form of Revise, Recap, Review. This will normally involve students answering 3 questions from last lesson, followed by 2 questions from previous study and one more challenging question. Each activity will involve students being posed questions interleaved over multiple units delivered throughout the year. Students are encouraged to work independently through the provision of scaffolding where required. Computing lessons often involve the application or practical/technical skills. These will be modelled to students using the I do, we do, you do approach. Students will be assessed at the end of each unit. Following assessment, students will complete a follow up activity based upon the individual areas to be improved that have been identified.

**Key Summative Assessments:**

End of topic tests upon completion of each distinct topic. Some questions on paper are interleaved from previous topics/across papers

End of year/mock exams

Retrieval homework.

Live marking and low stakes quizzing

**Autumn Term:**
1.1 systems architecture
2.1 computational thinking
1.2 memory and storage
2.2 algorithms

**Spring term:**
2.2 algorithms
1.3 networks
2.3 producing robust programs
1.4 network security

**Summer Term:**
2.3 producing robust programs
Mocks

**Impact:**
Students will have deepened their understanding of the computer system and know how instructions are fetched and executed by the CPU. Students will be more confident in their problem solving abilities and will be able to apply computational thinking strategies to solve problems. The year 10 curriculum will foster students intellectual curiosity around topics delivered

| Content | **Disciplinary Knowledge (Skills)**<br>This is the actions taken within a topic to gain substantive knowledge | **Substantive Knowledge**<br>This is the specific, factual content for the topic, which is connected into a careful sequence of learning. | **Prior Learning (Y7/8/9)** | **Future learning (Y11)** |
|---|---|---|---|---|
| **Paper 1** | • Illustrate the FDE cycle<br>• Understand the relationship between memory and the CPU<br>• Understand factors that affect system performance and justify system specification for specific scenarios<br>• Critically compare secondary storage devices for given scenarios<br>• Know the difference between LANs and WANs<br>• Understand the hardware required to create both LAN and WAN networks<br>• Know the advantages and disadvantages of CS and P2P networks and where each could be used<br>• To be able to recommend topologies for different networking requirements<br>• To know how the internet VPNs and VLANs work in unity<br><br>**Numeracy:**<br>• Calculate storage requirements for a given scenario<br>• Convert between binary, decimal and hexadecimal number systems<br>• Perform binary addition and shifting<br>• Show an understanding of how sound, text and images can be represented using binary | • CPU FDE cycle<br>• Memory<br>• CPU Performance<br>• Secondary storage<br>• Calculate storage requirements<br>• Number systems<br>• Data representation<br>• LANs and WANs<br>• Network Hardware<br>• Client server and peer-to-peer<br>• Topologies<br>• The internet, VPNs and VLANs | Year 8 under the hood<br>Year 7 networks<br>Year 8 data representation | • Protocols and IP addresses<br>• Packet switching<br>• Malware<br>• Preventing Vulnerabilities<br>• The OS<br>• Utility and application software<br>• Ethical, Legal Moral issues<br>• Legislation |

| Content | Disciplinary Knowledge (Skills)<br>This is the actions taken within a topic to gain substantive knowledge | Substantive Knowledge<br>This is the specific, factual con tent for the topic, which is connect ed into a careful sequence of lea rning. | Prior Learning (Y7/8/9) | Future learning (Y11) |
|---|---|---|---|---|
| Paper 2 | • Apply computational thinking techniques to solve computational problems<br>• Confidently use selection and nested selection to solve computational problems including the use of switch case statements<br>• Students can confidently use iteration to good effect in programs. Students know when to select count controlled or condition controlled iteration<br>• Students can represent problems using flow charts and interpret flow charts<br>• Students understand how linear and binary search algorithms work and can articulate how the algorithm would find data in an example data set<br>• Students can create trace tables to debug programs<br>• Understand how bubble, merge and insertion sorts work. Articulate how data is sorted on a given data set<br>• Students can connect external files to python programs<br>• Create SQL statements to select specific data. Interpret data returned from a given SQL statement<br>• Be able to use lists/arrays of both 1 and 2 dimensions<br>• Create functions and procedures and know the difference between them<br>• Understand the importance of anticipating misuse and know how to use iterative and final testing<br>**Numeracy:**<br>• Create truth tables for AND, OR and NOT gates. Use Boolean algebra to represent logic circuits<br>• Revision of operators and how these act upon arguments<br>• Use variables and constants in pseudocode<br>• Be able to use string manipulation techniques to solve a problem | • Computational thinking<br>• Arguments and operators<br>• Variables and constants<br>• String manipulation<br>• Selection<br>• Iteration<br>• Flow charts<br>• Searching algorithms<br>• Trace tables<br>• Sorting algorithms<br>• File handling<br>• Databases<br>• Arrays<br>• Sub programs<br>• Robust programs and testing<br>• Logic gates and Boolean algebra | Year 8/9 python programming | • Translators, compilers and assemblers<br><br>• IDEs |

## Year 10 Computer Science Learning Journey

End of year 10…..

Year 11

Threats and defence

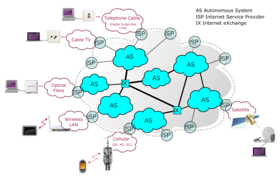
Network Security

Lists and Arrays
Functions and Procedures

Arrays and sub programs

SQL
Records
Fields

Databases



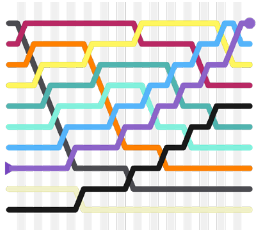
LAN/WAN
Star and mesh topologies

Networks and Topologies

File handling

Algorithms

Searching and sorting



RAM/ROM
Secondary Storage
Data representation

Memory and storage

Programming Fundamentals

Selection, sequence and iteration

Abstraction, decomposition and algorithmic thinking

Computational thinking

Start

Systems Architecture

Cpu, architectures and registers

YEAR 11 CURRIULUM OVERVIEW – *Computer Science*

**The Big Picture- Intent:**

Year 11 Computer Science has been designed to maximise progression.
All students will be able to access the main content of all lessons and all students will be taught to the top with scaffolding, adaptive teaching and stretch and challenge provided where necessary.

**Implementation**:
When delivering programming content, teachers will model coding where the core skills are introduced. Students will then be given a similar task to complete after the demonstration and may be given example code to copy and then edit. During Students will then be given a real world concept scenario that require them to apply the skills learnt. This will help solidify this knowledge, and also help students understand when and where to use programming techniques in solving computational problems.

Lesson sequences have been carefully chosen to ensure that students have the required background knowledge to fully understand and apply skills in relation to the topic. Therefore the lesson sequence may not match that of the exam specification, however all content is covered.
This is especially important for paper 2, where students need concrete understanding of the key programming techniques before applying these to producing GCSE Algorithms.

Lessons follow a consistent format beginning with a retrieval practice activity in the form of Revise, Recap, Review. This will normally involve students answering 3 questions from last lesson, followed by 2 questions from previous study and one more challenging question. Each activity will involve students being posed questions interleaved over multiple units delivered throughout the year. Students are encouraged to work independently through the provision of scaffolding where required. Computing lessons often involve the application or practical/technical skills. These will be modelled to students using the I do, we do, you do approach. Students will be assessed at the end of each unit. Following assessment, students will complete a follow up activity based upon the individual areas to be improved that have been identified.

**Key Summative Assessments:**

End of topic tests upon completion of each distinct topic. Some questions on paper are interleaved from previous topics/across papers

End of year/mock exams

Retrieval homework.

Live marking and low stakes quizzing

**Autumn Term:**

Programming fundamentals

Algorithms

**Spring term:**
Producing robust programs
Boolean Logic

**Summer Term:**

Languages and IDEs
Exam preparation

**Impact:** Students will have deepened their understanding of the computer system and know how instructions are fetched and executed by the CPU. Students will be more confident in their problem solving abilities and will be able to apply computational thinking strategies to solve problems. The year 10 curriculum will foster students intellectual curiosity around topics delivered

| Content | Disciplinary Knowledge (Skills)<br>This is the actions taken within a topic to gain substantive knowledge | Substantive Knowledge<br>This is the specific, factual con tent for the topic, which is connect ed into a careful sequence of lea rning. | Prior Learning (Y7/8/9) | Future learning (Y12) |
|---|---|---|---|---|
| **Paper 2** | • Apply computational thinking techniques to solve computational problems<br>• Use variables and constants in pseudocode<br>• Be able to use string manipulation techniques to solve a problem<br>• Confidently use selection and nested selection to solve computational problems including the use of switch case statements<br>• Students can confidently use iteration to good effect in programs. Students know when to select count controlled or condition controlled iteration<br>• Students can represent problems using flow charts and interpret flow charts<br>• Students understand how linear and binary search algorithms work and can articulate how the algorithm would find data in an example data set<br>• Students can create trace tables to debug programs<br>• Understand how bubble, merge and insertion sorts work. Articulate how data is sorted on a given data set<br>• Students can connect external files to python programs<br>• Create SQL statements to select specific data. Interpret data returned from a given SQL statement<br>• Be able to use lists/arrays of both 1 and 2 dimensions<br>• Create functions and procedures and know the difference between them<br>• Understand the importance of anticipating misuse and know how to use iterative and final testing<br>• Know how programming language is translated into machine code<br>• Understand when different types of translators would be used<br>• To know the features of an IDE and the advantages that the use of an IDE has when developing solutions<br>**Numeracy:**<br>• Revision of operators and how these act upon arguments<br>• Create truth tables for AND, OR and NOT gates. Use Boolean algebra to represent logic circuits | • Computational thinking<br>• Arguments and operators<br>• Variables and constants<br>• String manipulation<br>• Selection<br>• Iteration<br>• Flow charts<br>• Searching algorithms<br>• Trace tables<br>• Sorting algorithms<br>• File handling<br>• Databases<br>• Arrays<br>• Sub programs<br>• Robust programs and testing<br>• Logic gates and Boolean algebra<br>• Translators, compilers and assemblers<br>• IDEs | Year 8/9 python programming | H446 Computer Science |

# Year 11 Computer Science Learning Journey

Start

Operating systems

Functions of the OS

Defensive Design

Anticipating misuse

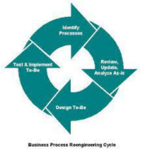
Abstraction, decomposition and algorithmic thinking

Utilities

Backups
Defragmentation
Anti-virus


Testing

Final and iterative testing

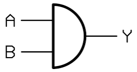


Legal Ethical and Moral issues

Ways in which computing impacts society
Moral issues
Automation and AI

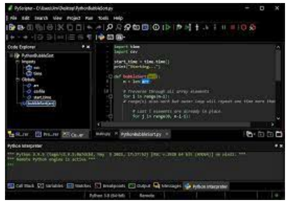

Boolean logic

Logic circuits
AND, OR NOT




IDEs

Compilers, interpreters and assemblers



Exam preparation

End of year 101.....

KS5