Computing Curriculum Intent Outline

The Joseph Whitaker Computing Curriculum aims for its students:

- To understand and use algorithms
- To provide students with a robust set of programming skills
- To support SEN & DP students with modelling of programming, differentiated targets and usage and explanation of key terminology
- To develop curiosity in how computer systems and programs work
- To create an ethical and sustainable attitude to the use of computers
- To promote safe working when online and using computers, and support student wellbeing
- To inspire creativity in their digital work
- To support SEN students with modelling of digital work, use of templates and differentiated targets
- To enable students to adapt to an ever-changing digital world
- To encourage self-reflection in the creation of digital content
- To cover as broad a scope of content as possible for Computer Science and ICT
- To enable students to develop cultural capital and understanding of British values through the use of external speakers/online opportunities
- To develop an understanding of the career options available in the industry, and how the curriculum work relates to those
- To ensure that metacognition principles are embedded in teaching to enable students to retain information
- To strengthen learning and use of vocabulary through the use of spaced retrieval and etymology of key words
- To encourage success, and see failure as a learning experience

The Joseph Whitaker Computing Department aims to deliver lessons with the following features:

- Retrieval practice at the beginning of every lesson - bringing prior knowledge to mind and improving future recall
- Questioning and oracy - improving ability to explain their thinking
- Explicit instruction of foundational knowledge, including vocabulary - essential to more advanced and independent learning
- What's the point - briefly describe the utility and application of the topic being delivered, to increase engagement
- Silent reading as an alternative to demonstration videos
- Modelling and thinking aloud - cognitive support to improve problem solving
- Low threat, successful guided practice - improve engagement and reduce misconceptions
- Formative assessment to inform future teaching, with opportunity for reflection, and to provide additional practice and reinforcement

Programme of Study	Year 7	Year 8	Year 9
design, use and evaluate computational abstractions	FLOWOL, Spreadsheets, Python	Python	Microbit Buggies, Python
understand several key algorithms that reflect computational thinking	FLOWOL, Python, Makecode Arcade	Python	Microbit Buggies, Python
use 2 or more programming languages, at least one of which is textual, to solve a variety of computational problems	FLOWOL, Python, Makecode Arcade	Python	Microbit Buggies, Python
understand simple Boolean logic and how numbers can be represented in binary	FLOWOL, Python, Makecode Arcade	Python	Microbit Buggies, Python
understand the hardware and software components that make up computer systems	CS Hardware/Software		Microbit Buggies, Y9 Final Module
understand how instructions are stored and executed within a computer system, and how data can be represented and manipulated digitally	CS Hardware/Software, FLOWOL, Python, Makecode Arcade	Python	Microbit Buggies, Python
undertake creative projects that involve selecting, using, and combining multiple applications	Spreadsheets, Makecode Arcade	Python	Microbit Buggies, Python, IT for Business
create, reuse, revise and repurpose digital artefacts for a given audience	Spreadsheets, Makecode Arcade, E-Safety		IT for Business, Cybercrime, Y9 Final Module
understand a range of ways to use technology safely, respectfully, responsibly and securely	E-Safety		Microbit Buggies, IT for Business, Cybercrime, Y9 Final Module

Year 7	Year 8	Year 9
E-Safety ● Staying safe online ● Digital Footprint ● Catphishing ● Phishing ● Illegal Downloads ● Viruses ● Hackers ● Creating a Visualisation Diagram (Rough imagery, basic descriptions) ● Evaluating a product (WWW/EBI)	**Python** ● Creating algorithms ○ Use selection (IF/ELSE) - nested statements - core programming principle ○ Use iteration - FOR/WHILE ○ Use boolean operators ○ Casting ○ Use different data types - core programming principle ○ Design 2 methods of demonstrating a solution ● Analyse algorithms ○ Problem solving - can identify how to solve a logic problem - core programming principle ○ Testing and making robust programs (against a student's own criteria) ● Evaluation - Can explain how a program's code works, suggesting any improvements	**Microbit Buggies** Use creativity and computational thinking skills to solve problems: ● pattern recognition ● algorithmic thinking ● decomposition ● abstraction Understand how to represent data when writing programs. Have repeated practical experience of writing computer programs in order to solve a variety of problems. Make appropriate use of data structures: For example, lists, tables or arrays. Design and develop modular programs that use procedures or functions.

FLOWOL Create algorithms using: ● Sequencing ● Selection ● Iteration Use success criteria to make and test programs. Describe and use computational abstractions that model the state and behaviour of real-world physical systems. Use creativity and computational thinking skills to solve problems. Identify the inputs and outputs of real-world physical systems. Have repeated practical experience of writing computer programs in order to solve problems.		**Python** ● Creating algorithms ○ Use selection (IF/ELSE) - nested, modular programs ○ Use of subroutines and functions ○ Global and local variables ○ Design 2 methods of demonstrating a solution in the most efficient way ○ Link to and build on Y7/8 learning ● Analyse algorithms ○ Problem solving - can predict where a syntax/logic problem may occur ● Evaluation ○ Can identify where problems are in their code ○ identify how to improve their code and why these would be beneficial ○ embedding principles of evaluation and what makes a good program
Spreadsheets ● Basic formula (+-*/) ● Function formula (sum/average/max etc) ● Logical formula (if/count etc) ● Charts/graphs ● Conditional formatting ● Creating a Mind Map (Min 3 nodes/min 2 Subnodes each)		**IT for Business** ● Spreadsheets ● Presentations ● Documents ● Communication ● Marketing ● Operations

CS Theory (Hardware/Software) ● Software is programmed - links to programming ● Hardware is physical ● Computers react with inputs/create outputs - links with programming ● Name hardware and understand its function		**Cyber Security** ● Understand how to communicate safely and identify threats online.Know how to protect their online identity and privacy. ● Explain how to protect against or minimise the impact of an attack. ● Understand how and why systems are attacked. ● Explain the potential impact of breaches in security. ● Describe several external and internal threats to digital systems. ● Understand how different measures can be implemented to protect digital systems ● Explain the potential impact of breaches in security.
Makecode Arcade ● Creating algorithms ○ Simple structure ○ Skeleton provided by teacher ○ Use selection (IF/ELSE) ○ Use iteration - FOR ● Analysing algorithms ○ Problem solving - can identify problems in an algorithm ○ Testing and making robust programs (against a criteria) ● Evaluation - can explain what worked and what didn't ● Create own sprites, art, and tilemaps		**Cutting Edge Computer Science** **AI** ● Limitations ● Is it "intelligent"? ● Applications ● OpenGPT etc ● Ethical considerations **Algorithms** ● Cryptocurrency ● Encryption ● Compression ● Search ● Facial recognition **Human-Computer Interface** ● Identify the different hardware and software components that make up computer systems. ● Identify different types of user interface ● Describe their use on a variety of devices, some may be new or unfamiliar technologies.

		● Understand the varying needs of an audience and how they affect the design of an interface. ● Consider user accessibility needs. ● Consider the purpose and target audience of user interfaces. ● Explore user interface design principles. ● Identify accessibility features used in an interface. ● Design and create a computer interface for a given audience, with attention to design principles and usability. ● Be able to review a user interface. ● Consider the strengths and weaknesses of an interface. ● Suggest improvements to interfaces to better meet the needs of a user. **Big Data** ● How big? Hardware and software involved ● Databases ● Applications - sport ● Personal data & privacy
Python ● Creating algorithms ○ Use selection (IF/ELSE) ○ Use iteration ○ Design one method of demonstrating a solution ● Analysing algorithms ○ Problem solving - can identify how to fix a syntax problem ● Evaluating Programs - Can explain what works and what doesn't and give a suggestion on how to overcome this		