Hobart High School Key Stage 4 Curriculum Map – Year 10

Department / Syllabus Link: Business – GCSE Computer Science (Edexcel).

	GCSE Unit Title	Knowledge & Skills Developed	Assessment	Personal Development
Autumn 1	**2.1 Introduction to Python and programming.** **2.2.1 Python IDLE and programming** **1.2.1 (1.1.2) Problem solving:** ways of describing problems (algorithms, written description, flowchart, pseudo-code). **2.5.1 Operators:** arithmetic operators, order of precedence and parenthesis.	**Knowledge/Skills:** • Introduction to Python language. • Practising the 'print' command. • Accessing online Python editors. • Decomposition of programs into input, process and output. • Program sequencing suing the print command. **Knowledge/Skills:** • Understand the interactive mode of Python (IDLE). • Use of keyboard programing shortcuts **Knowledge/Skills:** • Explain what is meant by decomposition of a problem. • Explain what an algorithm is and that algorithms can be represented as written descriptions, as pseudo-code, as a flowchart and as program code. • Emphasise the different ways of representing the same problem as an algorithmic design. **Knowledge/Skills:** • Python arithmetic operators. • Explain the use of parentheses and the order of precedence.	Student self-assessment against stored answers and solutions. Physical storage vs virtual storage article. **Teacher assessed.** Revision starters for 10. Social inclusion App design (Algorithm to provide a safe and straight forward passage to and through each groups platform). **Teacher assessed.** Problem solving task (Army recruitment algorithm). Validity of the solution will be **peer assessed.** End of topic tests. Binary cup – Student competition.	Students will gain the opportunity to develop their logic thinking and problem solving skills and apply them to any problem scenario that could have a programmable solution.

	2.1.2, 2.1.3, 2.1.5 **Developing code** **2.3.1, 2.3.4, 2.1.2, 2.4.1** **Data types, variables, input and output**	• Explain how to interpret common error messages. • Debugging programs is a normal part of programming. • Demonstrate how to include comment statements using the # symbol and explain that comments are used to make the code readable. • Explain how programs can be stored in a file using the file/new window command. **Knowledge/Skills:** • Explain and describe the four data types: **integer, real, Boolean and char.** • Explain that a variable is a named space in a computer's memory where one can store things and that can change (vary) the things that are stored in that memory. • Assigning values to variables and the conventions for selecting variable names. • Explain the use of the input function to assign input values to a variable. • Explain the use of the int function to convert a string to an integer and why it is used with the input function when entering numbers. • Explain that string method .format gives control over the formatting of output compared to using space-separated values in print. • Demonstrate numbers in the {} brackets are replaced by the objects given in the .format method. • Create a Python program that asks the user to enter three numbers, adds up the numbers and displays the answer on the screen. Link to the **input-process-output** model.		Students will be able to develop their own coding identity that supports industry requirement.

	2.2.1, 2.2.2, 2.5.2 **Selection constructs and relational operators:** if, then, else.	**Knowledge/Skills:** • How to write condition statements. • Explain that the if statement tests a condition which will be either true or false (Boolean) and always has two outcomes. • Describe the syntax of the command including the need for the colon :. • Explain that when an if else command is used one of the blocks of code will be executed depending upon the condition. • Correct indentation		
	4.2, 4.4 What is a computer? Hardware vs. software	**Knowledge/Skills:** • Define a digital computer as an automatic, programmable data processor. • Explain what digital means and what binary architecture is? • Explain what automatic, programmable and the role of a data processor is? • Explain the difference between hardware and software and list components of each.		Students will be able to make informed decisions about the component capabilities when selecting computing devices and computer components.
	4.1.1 Input-process-output model	**Knowledge/Skills:** • Explain how we can think about how computers work using the input-process-output model. • Draw and label the parts of the model.		
	3.1.1, 3.1.2, 3.1.3 Binary: introduction and representation of unsigned integers.	**Knowledge/Skills:** • Explain different data types encountered when programming in Python, i.e. ints, floats, bools and strings.		

		• Explain that a computer's electronic circuits consist of hundreds of thousands of tiny switches, called transistors. • Explain that a single binary digit is known as a bit and can have one of two values (0 or 1). Explaining that it can represent two options, e.g. true/false, yes/no, black/white. • Adding up in binary and understanding the problem of overflow errors. • Arithmetic and logical shifts. • Understand how bits in a computer are able to represent a wider range of values, e.g. with 3 bits a computer can represent eight different states. • Define bit terminology: A group of four bits is known as a nibble and a group of 8 bits is known as a byte.		

Autumn 2	**1.1.1, 1.1.2, 2.5.3 Flowcharts and Boolean operators** **2.5.3, 4.3.1 Boolean operators, logic, and truth tables** **3.1.2, 3.1.3, 3.2.4 Binary:** representation of signed and unsigned integers **3.1.1, 3.1.5 Hexadecimal:** what is it and why is it used? Converting hex-binary-hex	**Knowledge/Skills:** • Identify flowcharts and the flowchart symbols. • Demonstrate how flowcharts can represent if (decision) problems. • Explain the concept of AND, OR and NOT (Boolean operators) to combine the results of two condition statements. • Repetition construct: while loops. **Knowledge/Skills:** • Create and write programs that use AND, OR, NOT conditions. • Complete the truth tables for AND, OR and NOT. • Use IDLE's colour coding to ensure syntax is correct. **Knowledge/Skills:** • Learning how binary can represent other types of numbers, i.e. negative numbers. • Explain and identify sign and magnitude (where the most significant bit indicates whether the number is negative [0] or positive [1]). • Explain and identify two's complement (where the most significant bit is a negative number). **Knowledge/Skills:** • Explain that early programmers had to program in just 0s and 1s. Discuss the problems with that (errors, time to code, etc.).	Student self-assessment against stored answers and solutions. Revision starters for 10. End of topic tests. Binary cup – Student competition. Draw IO and Flowol software tasks – **Peer assessed.** Microbit exercises – **Peer assessed.**	

	1.1.1, 1.1.2, 2.1.1 Pseudo-code **2.3.3 Data types:** string manipulation and string methods **4.5.1 High and low level programming languages**	• Convert hex-binary-hex. **Knowledge/Skills:** • Understand that pseudo-code means 'mock' or 'pretend' code and that it is way of describing/writing a program. • Explain the reasons why flowcharts and pseudo-code are used to write algorithms. • Learn the list of the pseudo-code commands explicit with Edexcel. • Demonstrate how to translate programs written in pseudo-code into Python code. **Knowledge/Skills:** • Explain that a string is a data structure that stores a set of characters. • Explain how the position of each character in a string is called the index. • Demonstrate how to slice strings using square brackets. • Demonstrate how the len() function can be used to find the length of a string. • Demonstrate how to write programs that manipulate strings. Explain that strings are immutable, i.e. items in a string list cannot be changed. **Knowledge/Skills:** • Learn about how Programming languages run on computers by being translated into 0s and 1s. • Explain the early computers, e.g. Colossus, Manchester Baby, ENIAC. Explain how they were programmed in machine code.		Students will be able to break down a problem using decomposition and abstraction methods to create programmable solutions to problems.

	2.3.2 Data structures: lists (arrays) using Python docs	**Knowledge/Skills:** • Explain that a list (known as an array in some languages) is another data structure. • Demonstrate how to create a list and how lists are indexed. Demonstrate how to display individual elements and a range of elements in a list. • Demonstrate how to add items to a list, delete items from a list and how to check that an item is in a list. • Explain that all data types (strings, integers, floats, Boolean and other lists) can be stored in a list. • Explain that lists are mutable, i.e. the items in them can be changed. • Demonstrate the commands to change values in a list. (Tuples are like lists but are immutable; they cannot be changed).		
	4.5.2, 2.1.1, 2.1.2 Translating programming languages	**Knowledge/Skills:** • Explain the role of an assembler which is a program that takes a program in assembly language and translates it into machine code so that the processor can run it. • Explain a compiler which is a program that takes a program written in a high level language such a Python or C and translates it into assembly language or into machine code that a particular processor can understand and run.		
	2.2.2 Repetition construct: for loops, range function	**Knowledge/Skills:** • Explain that loops are an important programming construct that lets a program repeat parts of the program code.		

		• Practise the concept of repetition as a programming construct by giving the example of iterating through a list and also through a range of values. • Explain how a for loop repeats (or iterates) over the items in a list or a string. Describe the syntax of the for loop. • Explain the significance of the indent, i.e. that the loop ends when the block of code is no longer indented. • Demonstrate how the built in range() function allow a range of values to be generated.		
	4.2.1 Hardware: the function of internal components	**Knowledge/Skills:** • Explain the hardware components in a desktop computer (Computer architecture). • Explain that computers are made up of millions of electronic components, changing state hundreds of millions of times a second. • Identify the processor (CPU), RAM memory, secondary storage (hard disk drive), CD and DVD drives, graphics cards and network card. • Explain that the CPU is connected to the other components by wiring known as buses. There are three types of buses: Data, Control and Address.		
	4.4.1 Operating systems: files, hardware and the user interface.	**Knowledge/Skills:** • Define and list different operating systems in different computing devices. • Diagram to show how the operating system provides an interface between the computer's hardware and the application software used by the user.		

	2.6.1 Subprograms: random function	• Explain the main functions in an operating system: File management, Hardware management (Input/output and resource management), User management and process management. • Explain what is meant by a process. Explain that a computer program is a collection of instructions. • Explain that all application software such as a web browser, word processor, spreadsheet or apps are just computer programs and run as processes on a computer. **Knowledge/Skills:** • Explain how Python uses modules that contain pre-written functions. • Demonstrate how to import and run modules. • Explain how random numbers are used in many programs especially for writing games. • Demonstrate the use of the randint() function to generate random numbers.		
Spring 1	**5.1.1 Networks** **5.1.2, 5.1.7 LANs and WANs, Network Topologies**	**Knowledge/Skills:** • Understand what a computer network is and why computers are networked. • Explain what a computer network can do. • Identify the following features: The ability to exchange/share files and data; share physical resources, such as printers; communication (email, social media, video-conferencing); streaming content (videos, TV, music); storing data (cloud computing); accessing the internet; joining online communities. **Knowledge/Skills:**	Networking leaflet. **Teacher assessed.** Revision starters for 10. End of topic tests. Binary cup – Student competition.	Student will be able to design a network to share peripherals and improve efficiencies.

	5.1.3, 5.1.4, Network Data Speeds, Wired and Wireless Connectivity, Network Topologies **2.3.2 Data types and structures:** two dimensional arrays (lists) and nested 'for' loops	• Explain that computer networks are categorised according to their geographic coverage. • Explain the advantages and disadvantages of Local area networks (LAN) and Wide area networks (WAN). • Client-server and Peer-to-Peer networks (explaining the client and server process). • Explain how different network media effect data transmission speed (copper and fibre optic cabling). • Calculate Wi-Fi and signal range. • Explain how the term topology refers to the way in which devices are connected on a network. • Explain the advantages and disadvantages of different topologies (Bus, Star, Ring and Mesh). **Knowledge/Skills:** • Explain how network connection speed is measured by how many bits can be transferred in one second (bits per second (bps)). • Identify the different units of measurement in file size calculations. • Calculate the time required to transmit a file. **Knowledge/Skills:** • Explain how information can be stored in two-dimensional (2D) lists. • Explain that these data structures are called arrays in some programming languages. • Explain that nested loops can be used to process the information within two-dimensional lists (arrays). • Demonstrate how to create and initialise a two-dimensional list (array) in Python.		

	2.4.2 Validation **2.2.1, 2.6.1, 2.6.2, 2.6.3** **Subprograms:** Procedures, functions, return, passing parameters	• Demonstrate how information can be stored in two-dimensional lists (arrays) and how it can be accessed. • Explain that lists need to be created before they can be used. **Knowledge/Skills:** • Explain that validation is the automatic checking of entered data. • Demonstrate why validation is required when entering 'Y' or 'N' into a program and what happens when some other character is entered. • Demonstrate the 'try' command to defend a program from suspicious code that could cause a particular type of error. **Knowledge/Skills:** • Explain that as programs get more complex it is useful to divide (decompose) them into smaller parts. • Explain the 4 parts of a function: 1. A name that identifies the function 2. Parameters that allow data to be passed into a function as arguments 3. A body of commands 4. Return values to the calling program		

	2.3.4 Local and global variables **2.3.5 Constants** **5.1.5 Protocols**	**Knowledge/Skills:** • Demonstrate how variable assignments are available in parts of a program when using multiple functions and describe the concept of 'scope' of a variable. • Explain that global variables can be used anywhere in a program whereas local variables are defined for use only in a function. • Explain that local variables come into existence when a function is entered and the data they contain is lost when the execution of the function is completed. **Knowledge/Skills:** • Explain that a constant is a variable with a fixed value that cannot be changed. **Knowledge/Skills:** • Understand the different protocols for different areas of networking. • Explain the following protocols: Ethernet, Wi-Fi, TCP/IP, HTTP, HTTPS, FTP, POP3, SMTP and IMAP.		
Spring 2	**6.1.1, 6.1.2 Environmental impact of technology on society, Understanding the ethical impact of using technology (privacy, inclusion, professionalism) on society**	**Knowledge/Skills:** • Summarise the main ways in which computers are having a positive impact on people's health. • Summarise the ways in which computer manufacture, use and disposal harm the environment. • Understanding privacy settings, legislation and professionalism in computing.	News article on hazardous substances – **Teacher assessed.** Revision starters for 10. End of topic tests. Binary cup – Student competition. Social inclusion app design – **peer assessed.**	Students will be able to make conscious decisions about the impact of computing devices on the environment. They will gain an ethical awareness of the impact that technology has on an individual and how ethical awareness is

	4.3.1 Boolean Logic **5.1.6 Layered protocol stacks (TCP/IP) and packets** **2.1.3, 2.1.7, 2.1.6, 2.1.5 Errors and debugging tools in an Integrated Development Environment**	• Explaining social inclusion (technology empowered, technology excluded and the digital divide). • How cookie tracking captures data and preferences. **Knowledge/Skills:** • Explain that logic gates can be linked to each other to create complex circuits. • Explain how a Boolean value is either True or False and this can be represented using 0 (Zero) for false and 1 for true (relate to binary). In electronics this equates to part of a digital circuit having power or not. • Recognise the symbols and how the logic gates work. **Knowledge/Skills:** • Explain TCP/IP (Transmission Control Protocol/Internet Protocol) is a protocol stack or protocol suite. • Explain that a protocol stack is a layered collection of protocols. Each layer has a specific purpose. **Knowledge/Skills:** • Explain that a trace table helps to test for logical errors in a program. • Complete a trace table by listing the values assigned to a variable during the running of a specific program. • How a debugger can be used.		changing technology companies.

	2.6.1 Library sub-programs: import modules **3.1.1, 3.2.1 Data Representation**: Text **2.4.3 Reading and writing to files**	**Knowledge/Skills:** • Explain that import modules are libraries of subprograms that can be imported into programs. A module is a file that contains functions. • Explain the benefits of using imported modules, i.e. it allows programmers to re-use existing and pre-tested code that other programmers have written. **Knowledge/Skills:** • How could text be stored in a computer? • How would storing lowercase/capitals work? • How would storage of numbers work? • What else would need to be stored? • How ASCII is structured. • Define what a character set is. • Explain that python allows you to convert to/from letters to ASCII values using the functions ord and chr. So ord('a') will display 97 and chr(97) will display 'a'. **Knowledge/Skills:** • Explain that much of input/output is about being able to read information from a file and being able to write information to a file. • Describe what a file is and how files are referenced using a filename. • Demonstrate how Notepad can be used to write text files. • Explain that files need to be placed in the same folder as the Python program files that use them, or the correct pathname needs to be given. • Explain the need for caution when reading and writing to files as damage can be done to files.		

		• Explain the concept of sequential records stored in a file and how you need to open a file for reading (r) or writing (w). • Demonstrate how a program can read records from a file and write records to a file. • Demonstrate how to append records to the end of any existing file.		
Summer 1	**3.2.2 Data representation:** Images and sound **2.3.2 Dictionaries**	**Knowledge/Skills:** • Explain that the size of an image file is determined by number of pixels per inch (resolution) and the number of bits per pixel (colour depth). • Explain that all colours are combinations of red, green and blue. • Explain that any colour can be encoded as a set of three numbers. 'True colour' allocates 1 byte for red, 1 byte for green and 1 byte for blue (24-bit colour depth). • Understanding the different image compression techniques and the advantages and disadvantages of each method. • Explain that sound is analogue and that a process of digitisation is needed to convert it to a series of binary numbers. • Explain the sampling rate which is the number of samples taken per second and is measured in hertz (Hz). • Explain bit depth in sound. • Explain that sample rate and bit depth determine the quality of the reproduction and the comparable file sizes. **Knowledge/Skills:** • Explore how a dictionary allows you to look up a word and find its definition. In Python, the	Colour depth questions – **Teacher assessed.** Revision starters for 10. End of topic tests. Binary cup – Student competition. 'Name that tune' using audacity software - **Peer assessed.** Year 10 mock exam **– Teacher assessed.** How fast is your brain exercise **– student self-assess.**	

	4.2.1, 4.2.3 Hardware: internal components	dictionary is an 'unordered set of key: value pairs, with the requirement that the keys are unique. **Knowledge/Skills:** • Explain that processors are made of transistors. A transistor is an on/off switch controlled by an electric signal. • Explain that the clock speed of a CPU is how fast the transistors are switched. The faster the clock speed, the faster the computer can perform calculations. • Explain that it is the processor clock that sends out a pulse at regular intervals. The clock speed is measured in frequency (cycles per second) using the unit hertz. • Explain the energy output from transistors as heat. • Compare the increase in transistors as computing technology advances. • Explain that 'volatile' memory only retains data while it is receiving electrical power while 'non-volatile' memory retains data even in the absence of a power source. • Demonstrate how main memory (also known as Random Access Memory (RAM)) is volatile and is used to store instructions and data while programs and applications are running. • Explain that the size of the address bus determines the maximum number of unique memory addresses that can be used. • Explain to students that the computer may not contain the maximum amount of addressable memory. • Review the stages involved in a memory read and memory write. • Explain what ROM (Read Only Memory) is and how it differs from RAM.		

Summer 2	**4.2.1, 4.2.2, 4.2.3 Hardware**: internal components **2.4.3 Reading and writing CSV files** **5.2.1 Network security**	**Knowledge/Skills:** • Explain that the CPU comprises three main components: the control unit, the Arithmetic and Logic Unit (ALU) and the registers. **Knowledge/Skills:** • Creating a CSV file in a spreadsheet containing a data set. • Calculate the mean of the scores in the CSV file. • Find the maximum score in the CSV file • Find the minimum score in the CSV file. • Explain how python can write a list of lists as a CSV file. • Append code, test it and examine the CSV file saved by the program. **Knowledge/Skills:** • Explain how networks can be protected? E.G. Passwords, password complexity, firewalls. • What is the role of a firewall. • Identify and explain network threats: Cyberattacks, social engineering, phishing and shoulder surfing. • Explain how physical devices may pose a threat: USB devices, digital devices and eavesdropping. • Identify software and explain how it can protect a network from cyberattacks, e.g. Design and code reviews. • Methods to secure operating systems. • Creating audit trails and identifying vulnerabilities.	Revision starters for 10. End of topic tests. Binary cup – Student competition. Network security leaflet – **Teacher assessed.**	This is an opportunity for students to practise coding in preparation for the NEA (Practical task). Students will develop an awareness of how to protect computer systems and improve their own security measures.

	6.1.3 The Bigger Picture	**Knowledge/Skills:** • Explain the influence of copyright, licensing and intellectual property • Explain open source and proprietary software and the advantages and disadvantages of both.	Revision starters for 10. End of topic tests. Binary cup – Student competition. Legislation leaflet – **Teacher assessed.** Review of big corporation case studies **– student assessed.**	Students will be able to use information in accordance with the appropriate legislation.

Hobart High School Key Stage 4 Curriculum Map – Year 11

Department / Syllabus link: Business – GCSE Computer Science (Edexcel).

	GCSE Unit Title	Knowledge & Skills Developed	Assessment	Personal Development
Autumn 1	**3.3.2, 3.3.3, 3.3.4 Compression: lossless run-length encoding** **NEA Preparation: Requirements**	**Knowledge/Skills:** • Explain how lossless and lossy compression affects image quality and file size. • Practice encoding or decompressing colour images using RLE (Run Length Encoding). • Explain the advantages and disadvantages of applying RLE. **Knowledge/Skills:** • Analysing a problem, applying abstraction and decomposition methods. • Designing an algorithm to model a problem scenario and provide a possible solution. • Creating the initial test plan. • Programming a solution using Python software.	Student self-assessment against stored answers and solutions. NEA assessment – **Teacher assessed**	Students will be able to apply compression techniques in order to make decisions about the appropriate use of a file and the appropriate file size according to the need.
Autumn 2	**NEA Requirements** **4.2.4 Data storage:** magnetic	• Developing a coded solution using Python software. • Applying debugging tools and techniques. • Testing, refining and evaluating a coded solution. **Knowledge/Skills:** • Explain that there are three main ways in which data is stored permanently on devices (magnetic, optical and solid state). • Explain how Cloud storage provides a storage facility via the Internet.	NEA assessment – **Teacher assessed** Student self-assessment against stored answers and solutions. Revision starters for 10. End of topic tests. Binary cup – Student competition. Driverless car and the embedded systems involved – **student assessed.**	Students will have coded a real life problem, which will provide them with a set of skills that can be applied in further education or in a commercial environment.

	4.2.6 Embedded systems **5.3.1, 5.3.2 The internet, Addressing, routing, Physical routing and DNS, the World Wide Web**	• List online storage facilities, e.g. The one drive and Google drive. **Knowledge/Skills:** • Explain how embedded systems work in everyday devices. • List the devices where embedded systems are common. **Knowledge/Skills:** • Explain that the word 'internet' is short for 'interconnection of computer networks' i.e. a network of networks. Networks across the globe are connected. • The role of an ISP. • Demonstrate how to find the IP address of a computer. • What the difference between IP4 and IP6 is. • Describe the role of a **router** as a device that connects sub-networks together using the IP address. • Describe the sub-net mask (links back to Boolean logic 2.5 and hexadecimal notation 3.1.5). • Demonstrate how to use a visual trace route tool. • Explain packet switching and routers. • Explain that the role of the DNS (Domain Name Server): to match up a user-friendly domain names like to an IP address. • Explain that when a user requests a named service a message is sent to DNS to look-up the IP address of that service so that the user can then send a message to the right IP address. • Explain that the World Wide Web (WWW) refers to the huge collection of documents, written in	Packet switching exercise – **Peer assessed.**	

		HTML (Hyper Text Markup Language) that can be accessed using the internet. • What HTML documents are (text files stored on a server which are sent to your computer when you access them using web browser software). • Explain the role of a web browser. • Explain that a website is addressed using a URL (Uniform Resource Locator). • Explain that HTTP (Hypertext Transfer Protocol) defines how files on the World Wide Web are transmitted and displayed in web browsers and that HTTPS is an encrypted version of HTTP which allows data to be scrambled as it is transmitted over the internet.		
Spring 1	**3.5.1 Characteristics of structured and unstructured data, Structured databases Tables and relationships**	**Knowledge/Skills:** • Explain that a structured database is a convenient and powerful way to store and organise data. • Explain the terms 'record' and 'field'. • Create a database • Explain that a relational database helps to solve this and other problems by splitting the data into several linked or related tables. • Demonstrate why each record needs a unique identifier. • Understanding of relational database terms. • Demonstrate how keys are used to create a relationship between tables.	Mock exam – Paper 2 – **Peer assessed** Student self-assessment against stored answers and solutions. Revision starters for 10. End of topic tests. Binary cup – Student competition. Primary School database task – **Peer assessed.**	Students will be able to design and create a database which will provide them with a set of skills that can be applied in further education or in a commercial environment.
Spring 2	**3.4.2 Encryption, Caesar cipher algorithm**	**Knowledge/Skills:** • Outline the encryption process – a key is used to encrypt plain text into meaningless cipher text, the same key is used to decrypt the data back into its original form.	Mock exam – Paper 2 – **Peer assessed** Student self-assessment against stored answers and solutions. Revision starters for 10.	

	Exam skills/end	• Practise encrypting and decrypting using a Caesar cipher wheel to demonstrate a positive and negative shift. **Knowledge/Skills:** • Revision of topics and final examination skill practice.	End of topic tests. Binary cup – Student competition. Short and extended exam questions with a combination of self-assessment and teacher assessment.	
Summer 1	**Exam skills/end**	**Knowledge/Skills:** • Revision of topics and final examination skill practice.	Short and extended exam questions with a combination of self-assessment and teacher assessment.	
Summer 2	***** Computer Science Exams are set early in the exam calendar, both exam papers will have been delivered by the end of the 1st Summer Term. *****			