Subject: ICT

Year Group: 7

Week beginning	Subject Topic	Key Learning points/big questions	Independent/Home learning	Linked Assessment	Resources
7/9	E-Safety Overview, Expectations			*https://eggbuckland.sharepoint.com/:w:/g/ict/EbIc2wOy9WZNmVdUVtmr3MQBmDjxjYTOlO6-YCzg2byvsg?e=cnfi78*	*https://eggbuckland.sharepoint.com/:f:/g/ict/Eg5-BJHZZDBIsNnnYH9i_gAB_giOpVsFdaJnbUAbuxdivQ?e=7JP3yQ*
14/9	Online safety 1 Viruses, password security and digital footprints	Students will understand how computer viruses are transmitted, how to recognise them and how they can reduce their risks of downloading them.	Students will understand how computer viruses are transmitted, how to recognise them and how they can reduce their risks of downloading them		
21/9	Online safety 1 Viruses, password security and digital footprints	Students will understand the importance of secure passwords. They will learn a technique to help them create strong and memorable passwords to protect their school work and online privacy.	Create and use a set of memorable but secure passwords for all of their accounts.		
28/9	Online safety 2 Cyberbullying and grooming	I will understand the effects that cyberbullying can have on somebody else I will know what to do if I am ever cyberbullied by somebody.	Research online to find out a recent news story about cyber bullying. Think		

		I will think about my own behaviours online and think about the impact that they might have on someone else	about the following: • Who was being bullied? • How did the bullying happen i.e. technology used? • What effect did it have on the person being bullied? Did they do anything to try to stop the bullying? If so, what was the outcome?		
5/10	Online safety 2 Cyberbullying and grooming	I will understand how to recognise online grooming I will know what I need to do to reduce the risks of me becoming a victim of online grooming I will know what I need to do should I experience online grooming	https://eggbuckland.sharepoint.com/:w:/g/ict/EZNhQMk_nWRMr53kx6vrO8oB0J_73XTXEfzQUv3p0KrIZQ?e=ORm5Ad		
12/10	Online safety 3 Sexting and selfies	I will understand what is meant by sexting I will understand the consequences of sexting	https://eggbuckland.sharepoint.com/:w:/g/ict/EXZ27bvtVW9Ii3Nz-prHhTEBJe0xtrHgO6		

		I will know where to find help and advice about sexting	K3BT5jMewy7g?e=U16bfR		
19/10	Online safety 3 Sexting and selfies	I will understand how and why selfies are used I will be undertake my own research about selfies I will understand some of the negative effects and issues of taking and using selfies	Find an interesting news article about selfies. Write a paragraph to explain what the news story was about and any lessons learned.		
Half term					

Subject: ICT

Year Group: 8

Week beginning	**Subject Topic**	**Key Learning points/big questions**	**Independent/Home learning**	**Linked Assessment**	**Resources**
7/9	Python 2 - simple program and variables	Students will learn to write simple text based programs using the Python programming language. They will understand the concepts of variables and values.	a) Find out the history about 'Hello World' and why almost everyone uses as their first output b) Python is one of many programming	*https://eggbuckland.sharepoint.com/:w:/g/ict/EShtiHsI6UJLmYjwJRhRv9MBYHifXNYDw44WeoXg6i2Avg?e=GLhAhK*	*https://www.teach-ict.com/2016/ks3/sows/sow1/sow_menu.html* *username: pl65yb*

			languages. Find out about 4 other programming languages, what they are used for and how they differ to Python.		*password: kilobyte1*
14/9	Python 2 - inputs and concatination	During this lesson students will learn to use the Input Function to enable them to write programs which expect a response. They will then make use of the input data to provide a tailored response back to the user. They will begin to see the link between what they are doing and a programmable chatbot.	Find out about the Turing Test.		
21/9	Python 2 - sequence and selection	*Know:* the purpose of an If .. Else condition *Understand:* the purpose of == and indentation within the code *Be able to:* correctly order the code required for If … Else conditions	Now have a go at writing your own Elif program		
28/9	Python 2 - plan chatbot code	*Know:* how to construct the structure of a simple chatbot program *Understand:* the importance of planning before beginning to write programming code	Students should spend at least another 20 minutes working on their chatbot programs.		

		Be able to: write a plan for their chatbox logic, gain feedback and act upon that feedback in order to improve their plan.			
5/10	Python 2 - chatbot - peer assessment	Students will complete their code for their chatbot program. They will assess programs written by other students and provide constructive feedback. They will take on board feedback provided to them from other students and consider how they might improve their own work.	Students should spend at least another 20 minutes working on their chatbot program improvements.		
12/10	Python 2 - chatbot improve/develop code				
19/10	Assessment	*https://eggbuckland.sharepoint.com/:w:/g/ict/EShtiHsI6UJLmYjwJRhRv9MBYHifXNYDw44WeoXg6i2Avg?e=GLhAhK*			
Half term					

Subject: ICT

Year Group: 9

Week beginning	Subject Topic	Key Learning points/big questions	Independent/Home learning	Linked Assessment	Resources
7/9	Introduction to User Interfaces	Introduction to user interfaces: hardware features, software features and human facilitation and example uses Basic user interface: text-based and menu-based	Open your 'Knowledge Organiser'. Turn to the ICT (Computing) page. Type 3 questions and answers based upon information on that page. Save/update file 'User Interface Key Terms and Definitions'.	*https://eggbuckland.sharepoint.com/:w:/g/ict/EXA232227-dDnD6vRvULfRqB_nvurFTOJO9Ln6L7F9-LLA?e=dRcKcm*	User Interface mini-web https://www.teach-ict.com/gcse_new/computer%20systems/user_interface/miniweb/pg6.htm ICT Knowledge Organiser Share-point Folder https://eggbuckland.sharepoint.com/:b:/g/ict/EfJuukhAZChHmHfBkV-e2bkBwO2fdLKg6EL-
14/9	Introduction to User Interfaces – project requirements	Defining the project requirements: user requirements, output requirements, input requirements and user accessibility requirements	Students identify which requirements from a project brief are user requirements, input requirements, output requirements and accessibility requirements.		
21/9	Introduction to User Interfaces – creating an initial design	Designing the visuals: sketches and storyboarding	Print screen of two of your interfaces and annotate them below to show how they have been		

			made easier to use and the design principles you have demonstrated.		GpUZ8cweyg?e=YypsoU 'E-textbook Exploring User Interface Design' https://eggbuckland.sharepoint.com/:b:/g/ict/EfJuukhAZChHmHfBkV-e2bkBwO2fdLKg6EL-GpUZ8cweyg?e=wonYTJ 'History of the Graphical User Interface' video https://eggbuckland.sharepoint.com/:f:/g/ict/EiRImp8266
28/9	Introduction to User Interfaces – developing an initial design	Developing a functional user interface: showing the outputs, inputs and the navigational methods	Identify three possible improvements that you could make to your interface.		
5/10	Introduction to User Interfaces – Reviewing user interface initial design	Reviewing the user interface and what areas could be developed further	Finally, you should review your user interface designs. Explain how your deisgn(s) meet the project requirements you set out in week/lesson 2.		
Half term					

					pKiyQTAe0n38gB1lOM2dhs0tSUqusj1etEsQ?e=TVvQMC

Subject: ICT

Year Group: 10

Week beginning	Subject Topic	Key Learning points/big questions	Independent/Home learning	Linked Assessment	Resources
7/9	A: Investigate user interface design for individuals and organisations A1 What is a user interface?	Introduction to user interfaces: hardware features, software features and human facilitation and example uses Basic user interface: text-based and menu-based	Students confirm their understanding of 'user interface', by describing a user interface giving features of – and example interactions with – a user interface. Explain which type of interface may be used on a	Coursework practice case study: https://eggbuckland.sharepoint.com/:w:/g/ict/EX5AEVgwdWpCohT7GOwM2JcBeyeTwek0l1P0XRlCRaWd5A?e=peqAUf	User Interface mini-web https://www.teach-ict.com/gcse_new/computer%20systems/user_interface/miniweb/pg6.htm

			smartwatch and justify your choice.		ICT Knowledge Organiser Share-point Folder https://eggbuckland.sharepoint.com/:b:/g/ict/EfJuukhAZChHmHfBkV-e2bkBwO2fdLKg6EL-GpUZ8cweyg?e=YypsoU 'E-textbook Exploring User Interface Design' https://eggbuckland.sharepoint.com/:b:/g/ict/EfJuukhAZChHmHfBkV-
14/9	A: Investigate user interface design for individuals and organisations A1 What is a user interface?	Complex user interfaces: speech/natural language-based, GUI/WIMPs and sensor-based Choosing a user interface: performance/response time, ease of use, user requirements, user experience, accessibility and storage space	Students describe two ways in which supermarket self-checkouts could make use of graphical-, sensor- and speech-based interfaces		
21/9	A: Investigate user interface design for individuals and organisations A1 What is a user interface? A2 Audience needs	How hardware and software affects user interfaces: operating systems/platforms, screen type/size, types of user input, hardware resources available and emerging technologies User accessibility needs: visual, hearing, speech, motor and cognitive needs	Students to research the typical user interfaces found in fast food chain restaurant. Find out specific information about the components and consider how these components impact the user interface.		
28/9	A: Investigate user interface design for individuals and organisations A2 Audience needs A3 Design principles	User skills: expert, regular, occasional and novice user skills and demographics: age, beliefs/values, culture and past experiences Design principles: visual elements: colour and font style/size	Students consider how a user interface for a local college application form can be adapted for users with visual, hearing and speech accessibility needs.		

5/10	A: Investigate user interface design for individuals and organisations A3 Design principles	Design principles: text elements: language and amount of information Design principles: layout: consistency, placement of items, user expectations, grouping related items, navigational components and input controls	Students write a list of dos and don'ts when using colours and fonts in a user interface. Students to evaluate why the layout of items on a user interface is important. What factors influence the layout of items on the screen?		e2bkBwO2fdLKg6EL-GpUZ8cweyg?e=wonYTJ 'History of the Graphical User Interface' video https://eggbuckland.sharepoint.com/:f:/g/ict/EiRImp8266pKiyQTAe0n38gB1lOM2dhs0tSUqusj1etEsQ?e=TVvQMC
12/10	A: Investigate user interface design for individuals and organisations A3 Design principles	Design principles: user expectations: colour, sound, symbols, visuals Design principles: keeping the user engaged: uncluttered screens, tip text, labels, default values and autofill	Students to think of what immediately comes into their mind when they hear positive high-pitched sounds and negative low-pitched sounds. Give students an example of audio that may sound when a file is deleted. Students to discuss their immediate reaction to the sound.		

			Students to think of three other situations of when a positive high-pitched sound and sharp negative low-pitched sound may be played while using devices.		
19/10	A: Investigate user interface design for individuals and organisations A3 Design principles A4 Designing an efficient user interface	Design principles: intuitive design: graphics denoting actions, helpful messages, easy reversal of actions, help features and consistency Improving the speed of user interfaces: keyboard shortcuts, reversal of actions, informative feedback and distinguishable objects	Students list three methods that can be used to sustain the attention of a user while they are using a user interface. Students confirm their understanding about what the word 'intuitive' means. Students explain how to achieve an intuitive design and why intuitive design is important in a user interface.		
Half term					

Subject: Computer Science

Year Group: 11

Week beginning	Subject Topic	Key Learning points/big questions	Independent/Home learning	Linked Assessment	Resources
7/9	2.5 Translators and Facilities	• To be able to describe the different generations of programming language • To be able to describe the differences between Low Level and High Level Languages • To evaluate the benefits of programming in both Low and High Level languages • To state which translator is needed for each and why • To be able to describe how an Assembler works • To be able to describe the differences in operation between a Compiler and Interpreter • To be able to describe the common tools and facilities in an Integrated Development Environment (IDE)	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.	Complete Learning Grid for each paper 2 topic https://eggbuckland.sharepoint.com/:f:/g/ict/EpwM1ZGpfF1NuQS_SwVjyvwB9-cqQIcoSTbs-E0XzW1zbw?e=M0Q0bp Paper 2 practice questions	Resources for all paper 2 units https://eggbuckland.sharepoint.com/:f:/g/ict/Ep0_UrVWpUFKvF_JYRKBN0MB_UTSLqRN8c-kjR8YEL-fCA?e=T4QJrW

				https://eggbuckland.sharepoint.com/:b:/g/ict/EQ-MdGmKFLxLg_EgSkpM7JMB2R-C_dSLH-HagZy9g3luUA?e=atX9Q9	Learning Grid https://eggbuckland.sharepoint.com/:f:/g/ict/EpwM1ZGpfF1NuQS_SwVjyvwB9-cqQIcoSTbs-E0XzW1zbw?e=M0Q0bp
14/9	2.4 Computational Logic	1. Explain why data needs to be in binary form 2. Draw diagrams for the AND, OR and NOT gates 3. Create a Truth Table for AND, OR and NOT gates 4. Draw Logic Circuits and Truth Tables for 2nd Level Logic Circuits	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		
21/9	2.1 Algorithms	1. Understand the term and processes in computational thinking. 2. Be able to use the skills of abstraction, decomposition and algorithmic thinking. 3. Be able to use a linear search to find data. 4. Be able to use a binary search to find data. 5. Understand the differences between a linear and a binary search. 6. Understand the principles of a bubble sort. 7. Be able to perform a bubble sort on a set of data. 8. Understand how the number of comparisons increases in a bubble sort. 9. Understand the principles of a merge sort. 10. Be able to perform a merge sort on a set of data.	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		

		11. Understand the principles of an insertion sort. 12. Be able to perform an insertion sort on a set of data. 13. Be able to produce an algorithm using a flowchart. 14. Be able to produce an algorithm using pseudocode. 15. Be able to find and correct errors in algorithms. 16. Be able to complete algorithms where code is missing.			
28/9	2.3 Producing Robust Programs	1. To understand the elements of defensive program design 2. Know how comments and indentation can support maintainability 3. Describe the role of testing, including how to identify errors and select appropriate test data. 4. Understand the purpose of testing 5. Identify different types of program errors 6. Know the difference between iterative and terminal testing	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		

		7. Be able to select suitable test data			
5/10	2.6 Data Representation	• Define the units bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte. • Know how data needs to be converted into a binary format to be processed by a computer. • Know how to convert positive denary whole numbers (0-255) into 8 bit binary numbers and vice versa. • explain the use of binary codes to represent characters • explain the term character set • describe with examples (for example ASCII and Unicode) the relationship between the number of bits per character in a character set and the number of characters which can be represented. • **Understand how a digital image is made up and be able to recognise the affect changing the resolution has on an image.** • **Understand how a computer displays coloured images using binary and rgb values.** • Understand how sound is stored into binary values	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		

		• Understand the factors that affect how sound is stored and how this affects the memory needed for storage. • Understand and be able to explain why the factors affect memory storage and how this can be overcome through file compression. • **Understand** the need for compression. • **Know** the difference between lossy and lossless compression. • **Analyse** when lossy and lossless would be used.			
12/10	2.2 Programming Techniques Recap	• Identify and name variables and constants in code, pseudocode or flowcharts • Recognise the need for descriptive identifiers like variable names • Recognise the need for constants • Be able to assign input to a variable and output it • Use the most appropriate arithmetic, Boolean and string operators to produce the solution involving variables and constants • Link the use of Boolean operators and variables to selection and iteration • Count the length of a string	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		

		• Concatenate (join) strings • Search for a substring (presence of a character in a string) • Count instances of a substring in a string • Extract substrings (part of a string): from the Left, Mid, Right, or by a delimiter (e.g. a comma) • Change case of a string's characters: alternate between lowercase, UPPERCASE and Titlecase. • Finding out ASCII/Unicode position for a character • Escape and whitespace characters • Be able to break a complex task into a sequence of simple steps which would each require one line of pseudocode and/or one block in a flowchart • Understand that the order of steps in algorithms matters • Understand when to use selection, including nested selection • Recognise the two common types of selection: IF-ELSE IF-ELSE and SELECT CASE/SWITCH • Be able to use Boolean operators to narrow and/or expand selection • Understand when to use iteration, including nested iteration, stepping (where a counter is incremented by a			

		number other than 1), and backward-flowing iteration (where a counter is decremented) • Recognise the two common types of iteration: WHILE (condition-controlled) and FOR COUNTER (counter-controlled) and apply the most suitable for the problem. • Be able to rewrite any FOR loop as a WHILE loop, and some WHILE loops as FOR loops • Be able, given a path/file name open a serial/text file (e.g. txt, csv, anything that can be read as plain text in a program like Notepad) and store its contents in a variable or an array • Be able to get the list of files in a folder/directory and recognise which files are text files • Recognise that text files are stored in string format and string manipulation and casting is often needed to make it usable in a solution • Understand that the two most common ways to write to a text file are: append text and write text (destroys all prior text in a file). • Be able to serialise an array up to two dimensions, converting/casting all array elements to string and			

		concatenating with delimiters for rows/columns. • Be able to search/make/write changes to a particular item of data stored in a text file. • Understand the need to close an open file to release memory and save changes made to file if written to. •			
19/10	2.2 Programming Techniques Recap	• Be able, given a path/file name open a serial/text file (e.g. txt, csv, anything that can be read as plain text in a program like Notepad) and store its contents in a variable or an array • Be able to get the list of files in a folder/directory and recognise which files are text files • Recognise that text files are stored in string format and string manipulation and casting is often needed to make it usable in a solution • Understand that the two most common ways to write to a text file are: append text and write text (destroys all prior text in a file). • Be able to serialise an array up to two dimensions, converting/casting all array elements to string and concatenating with delimiters for rows/columns.	Smart Revise – spend 1 hour answering questions on topic. Time can be split into blocks i.e. 2 x 30 minutes or 3 x 20 minutes, to suit.		

		• Be able to search/make/write changes to a particular item of data stored in a text file. • Understand the need to close an open file to release memory and save changes made to file if written to. • Recognise arrays in code • Understand the need and justify the use of an array in a solution • Understand and be able to use an index for a one-dimensional array and a double index (a tuple) for a two-dimensional array • Understand the connection between the number of dimensions in an array and the composition of its index • Identify the data type of an array and understand that traditional arrays must have all array elements of the same data type • Be able to find out the length of an array • Be able to read a value of an array element and overwrite it if needed • Be able to resize and copy arrays • Identify common errors relating to the arrays, such as “out of index” error • Be able to convert a string àstring array, string array àstring with SPLIT() and concatenation			

		• Having broken the solution into a sequence of steps, a learner can identify sub-sequences – repeating series of steps and chunk them into subprograms • Be able to use the top-down approach to plan a program's processing stages resulting in structured code • Understand how the use of subprograms increases the efficiency of coding through the reduce-reuse of code • Understand the difference between global and local variables (variable scope) • Be able to identify and use parameter passing to make a sub-program more abstract/generic and reusable in even more cases • Recognise the difference between procedures and functions • Understand the need for various data types • Choose the best data type for the problem among integer, real, Boolean, character and string • Be able to convert (cast) a variable from any of these types to any other, if possible			

Half term					