Summer term Year 10

Scheme of Work

GCSE (9-1) Computer Science

Pearson Edexcel Level 1/Level 2 GCSE (9-1) in Computer Science (1CP1)

# Summer term Year 10

## Assumptions

This scheme of work is based on the assumption that the learners have been following the schemes of work for Autumn and Spring terms of Year 10.

The lessons in the various schemes of work total 150 hours altogether (see course planner); the guided learning hours for GCSE Computer Science are 120. It is expected that centres will modify the schemes of work and the course planner as required to fit their own timetabling.

The programming language used here is Python 3.0. Centres can of course use any of the other languages in the specification, namely Java, Pascal/Object Pascal, Visual Basic.NET, C-derived (C, C++, C#) and use equivalent examples from these languages.

This document is subject to change.

## Explanations

The course has been broken down into six terms. Over the course, lessons are split into the six topics in the subject content – the colours in the schemes of work mirror the colours of the topics in the course planner.

Weeks are numbered from one in each term and there are two lessons each week. This scheme of work details the content to be covered in the Summer term of Year 10. Lesson 1 covers mainly the theoretical principles of computer science. Lesson 2 covers mainly the practical application of problem solving and programming, although it should be remembered that these topics are also examined. Problem solving and programming are usually taught together to increase computational thinking skills, which will also be used in the other subject content topics.

The references to the specification represent the specification content that is covered in the lesson. During the problem solving and programming topics, many lessons cover a range of content but only the main specification reference is given. You are strongly advised to use the specification alongside this scheme of work.

The lesson summary gives an overview of the lesson content. The title in bold refers to the subject content headings given for each topic in the specification.

The lesson content gives suggestions on how this material could be taught. Example activities for each lesson are given in the activities files. The activities are numbered with the lesson number, for example Activity 1.2.3 is the third activity for Lesson 2 in Week 1. These activities are provided to give tutors ideas of activities and programming challenges they may find useful in teaching this specification. They are examples of the types of activities that can be provided and they do not exhaustively cover the specification.

A suggestion for a homework task is given at the end of each lesson. Tutors may wish to refer to the homework in the starter for the following lesson.

A list of useful resources, both online and in print, is given at the end of this document, together with teaching suggestions, starters and plenaries, homework tasks and how to get students to self-assess their programs.

**Disclaimer**: The problem solving and programming lessons are about teaching transferable computational thinking concepts using Python rather than being a Python programming course.

# Summer term Year 10

| **Week 1** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.2.2 | **Data representation:** bitmaps | Use **Summer Year 10 slides** to introduce this lesson. Explain that students will learn how binary code is used to represent photographs and other images. Zoom in on the photograph to show that a bitmap image consists of a grid of squares called ‘pixels’. Explain that each pixel can be uniquely identified by its position in the grid (x/y coordinates) and that each pixel is a single colour. Give pupils the opportunity to experiment with a bitmap image in a graphics software package. **(Activity 1.1.1)**  Ask students to research how many colours can be seen by the human eye.  Discuss: do we always need to store so many different colours (size of file, speed of loading, etc.)?  Explain that a binary code is used to represent the colour of a pixel. The number of bits used to store each pixel’s colour is known as the ‘colour depth’. The greater the colour depth, the more colours can be represented.  Explain that all colours are combinations of red, green and blue. Give pupils the opportunity to see this for themselves by experimenting with the RGB Explorer: [www.stanford.edu/class/cs101/image-rgb-explorer.html](http://www.stanford.edu/class/cs101/image-rgb-explorer.html).  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).  Point out that a computer needs to know the image size in pixels and the colour depth in order to be able to read the binary data correctly and reproduce the image. This additional information is called ‘metadata’.  Explain that it is impossible to produce an exact copy of an image as a bitmap. Bitmaps are representations of the original and some loss of detail is inevitable. Even if a pixel is very small, the colour is unlikely to be uniform across the whole pixel. The binary code represents an average for the whole pixel. The smaller the size of the pixels used, the closer the digital representation is to the original. Ask students to complete **Activity 1.1.2**.  **Homework:** Ask students to complete **Activity 1.1.3**. | Week 1  Lesson 1 activities  Summer Year 10 slides  RGB Explorer |
| 2 | 2.3.2  (extension) | Dictionaries | Ask students to explain what a dictionary does (as in a physical book type dictionary!).  A dictionary allows you to look up a word and find its definition. In Python, a dictionary is an ‘unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary)’. (From [www.docs.python.org/3/tutorial/datastructures.html - dictionaries](https://docs.python.org/3/tutorial/datastructures.html#dictionaries))  Ask students to work through **Activities 1.2.1, 1.2.2** and **1.2.3**  **Homework:** Ask students to research some use-cases of dictionaries. | Week 1  Lesson 2 activities  Python website in lesson content |

| **Week 2** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.2.2 | **Data representation:** bitmaps | Give pupils an opportunity to create some simple bitmaps of their own, using resources on CAS Online: [www.community.computingatschool.org.uk/resources/8](http://community.computingatschool.org.uk/resources/8). (**Activity 2.1.1**)  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). The higher the resolution and colour depth, the more accurately the image will be represented as a bitmap, but also the larger the file size will be.  Show students how to calculate the file size of a bitmap image, i.e. (height x resolution) x (depth x resolution) x colour depth. **(Activity 2.1.2)** Encourage pupils to examine the file sizes of bitmap images. Explain that bitmap images are often compressed to reduce their file size. (Students will be learning about compression in Year 11.)  Use **Summer Year 10 slides** to introduce the units of measurement, i.e. nibble, byte, kilobyte, megabyte, gigabyte and terabyte, and their abbreviations. Ensure that the binary values are taught, i.e. 1024 bytes in 1 Kb as opposed to 1000.  **Homework:** Ask pupils to complete the activity on measurements. **(Activity 2.1.3)** | Week 2  Lesson 1 activities  Summer Year 10 slides |
| 2 | 1.2.2  2.1.4 | **Programming**: decomposition, error checking and testing | Ask students to find out how to convert °C to °F and vice versa. **(Activity 2.2.1)** You should get something like: °C x 9/5 + 32 = °F and (°F – 32) x 5/9 = °C (from [www.manuelsweb.com/temp.htm](http://www.manuelsweb.com/temp.htm)).  Explain to students that their task is to write a converter that will take in a temperature and convert it from Celsius to Fahrenheit or the other way around. When it is initially run, the program will ask which conversion they wish to perform, then the relevant subroutine will be called and the user will enter a temperature. The answer will be calculated and displayed.  In pairs students should break the task down into manageable sections. Likely to be Menu, Convert °C to °F, Convert °F to °C **(Activity 2.2.2)**  Students should then implement and test a section at a time. The idea of stubs should be introduced or researched at this point **(Activity 2.2.3)** – a stub is a stand-in for another piece of code. So the two conversion sections could be written, but instead of actually working they simply display a message saying that the relevant subroutine has been called. Once the menu system is working correctly, each stub would be replaced and tested.  **Extension:**   1. Ensure that after a conversion is completed the menu is redisplayed 2. Find out how to have an option in the menu that exits the program   **Homework: Activity 2.2.4.** Top down design. | Week 2  Lesson 2 activities |

| **Week 3** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.2.3 | **Data representation:** sound | Using the **Summer Year 10 slides** show an image of a continuous sinusoidal waveform. Explain what is meant by the term ‘analogue’. Explain that sound is analogue and that a process of digitisation is needed to convert it to a series of binary numbers.  Describe the process of digitising sound, i.e. samples taken at evenly spaced time intervals (fractions of a second) and represented as numerical values. The sampling rate is the number of samples taken per second and is measured in hertz (Hz). A CD-quality recording has a sampling rate of 44 Khz, which means that the sound is sampled 44,000 times a second.  Explain that, in addition to the sampling rate, the more bits (bit depth) dedicated to representing the sample, the better the sound reproduction. 16 bits provide 65,536 possible levels of measurement, as compared to 8 bits, which provide only 256 levels of measurement.  Explain that sample rate and bit depth determine the quality of the reproduction. However, the higher the quality of the sound, the larger the file size. Sound files are often compressed to reduce the size. (Students will be learning more about this in Year 11.)  Ask students why file size might be important? (Relatively small amounts of memory available on mp3 players and phones, streaming issues with larger amounts of data, etc.)  Use sample sound files to show the impact of different bit depths and sampling rates: [www.music.columbia.edu/cmc/musicandcomputers/chapter2/02\_05.php](http://music.columbia.edu/cmc/musicandcomputers/chapter2/02_05.php).  This could be run as a kind of quiz to see if students can accurately identify the higher quality samples. A discussion about online music services and the quality they offer could be interesting, especially as certain providers try to differentiate themselves by the bit rate of the audio they provide.  You could ask students to research the major music providers (whether streaming or download only) and find out what file types they offer and what bit rates they provide. Would this impact on their choice of music provider?  **Homework:** Ask students to bring in three music tracks (ideally different genres) as they are going to alter the quality of the files next lesson. | Summer Year 10 slides |
| 2 | 2.1.1  1.2.1  1.2.2 | **Programming challenge 2**, part one: Maths quiz | This lesson should allow students an opportunity to solve a programing problem independently or in pairs. Students should write a program that will ask the user five simple maths questions that consist of two random numbers either added or subtracted from each other. The quiz will tell the use if each answer is correct or incorrect and give a total out of five at the end.  Students should:   * plan the program using pseudocode * implement the program.   **Activity 3.2.1** gives details of the task and additional requirements. Subheadings where pseudocode and actual code can be inserted have been provided.  Students should attempt this task independently to give you the opportunity to help those struggling in the class. To support students you could break the task down, for example write the subroutine that generates a question, then extend this to mark the input. Once this was working, use a loop to run the subroutine five times then add in the overall grade code.  Possible extensions:   1. As soon as the program is run the user should be prompted to enter his/her name. Final grade output should make use of the user’s name. 2. Extend the types of question that the quiz asks.   **Homework:** Finish the program as it will be needed in a future lesson. | Week 3  Lesson 2 activities |

| **Week 4** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.2.3 | **Data representation**: sound | Show the YouTube video ‘Bit depth and Sample Rates in Audacity’: [www.youtube.com/watch?v=s34QT6OJBEQ](http://www.youtube.com/watch?v=s34QT6OJBEQ). Audacity is free, open source, cross-platform audio software which can be downloaded here:  <http://www.audacityteam.org/download/>  Demonstrate to student how to use the basic functions of Audacity audio software <http://www.audacityteam.org/>. Selecting a section of sound, removing the rest, saving, exporting. Revise where to change bit depth and sampling rate.  Then give pupils the opportunity to use Audacity themselves to investigate the effects of altering the sample rate and bit depth. Get students to export several versions of the music they brought in from last week’s sound homework.  Students should discuss how much difference sample rate and bit depth have and compare this to file sizes.  Ask pupils to draw an analogue sound wave and explain the sampling process. **(Activity 4.1.1)**  Encourage pupils to identify other forms of analogue information that computers regularly deal with, for example temperature, light levels, pressure, rate of flow.  Explain that the process of converting from a continuous-time and continuous-amplitude analogue signal to a [discrete-time](http://en.wikipedia.org/wiki/Discrete-time) and discrete-amplitude [digital signal](http://en.wikipedia.org/wiki/Digital_signal) is always the same.  **Homework:** Ask pupils to complete the activity on sound representation. **(Activity 4.1.2)** | Week 4  Lesson 1 activities  Audacity audio software |
| 2 | 2.1.4  2.4.2 | **Programming challenge 2**, part two: Maths quiz (validation and testing) | Explain to students that in the NEA that they will be given a scenario and expected to write pseudocode, an initial test plan, implement the solution and then test it. Normally students would write tests before implementing any code.  This lesson is designed to give students experience of writing tests. Without reopening their code they should write suitable tests for the Maths quiz scenario. Remind students about valid, invalid and extreme tests. They should include all three types in the test table. Ensure that they include validation type tests. For example: What happens if no value is entered? What happens if a letter rather than a number is entered?  Get students to share and add to the test plans so that each student has a comprehensive test plan. It is likely that students will not have implemented any validation as it wasn’t specifically asked for in the requirements but students should now be starting to add this independently.  A test table following the design on page 22 of the specification can be found in **Activity 4.2.1**. A copy of the scenario and requirements is also supplied with the test table.  Once tests have been written they should use their test table to test a peer’s solution. They need to produce a brief report explaining whether the solution passed each test or not.  Students should then fix any problems with their own code. If time allows the tests could be re-run by the original tester.  **Homework:** Finish fixing any problems found in the solution. Ensure that validation is implemented | Week 4  Lesson 2 activities |

| **Week 5** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.2.1  4.2.2  4.2.3 | **Hardware**: internal components | Start the lesson asking students to find the definition of hardware. Explain that this lesson will look at the hardware components used in a digital computer. Explain that we will focus on the hardware components in a desktop computer because they are larger and easier to see but that similar components exist in all other digital computers, such as mobile phones.  Explain that computers are made up of millions of electronic components, changing state hundreds of millions of times a second. These components are reliably processing data at phenomenal speeds. We going to explore how the digital computer works and the first step is to identify the hardware components that make up a digital computer.  As a class watch these videos that identify the parts of a computer and describe their functions:   * [www.gcflearnfree.org/computerbasics/7](http://www.gcflearnfree.org/computerbasics/7) * [www.youtube.com/watch?v=yRmPTbGBqVI](http://www.youtube.com/watch?v=yRmPTbGBqVI)   Demonstrate an actual desktop computer. Take the case off and ask students to identify the processor (CPU), RAM memory, secondary storage (hard disk drive), CD and DVD drives, graphics cards and network card. Explain that the components in a digital computer need electricity, which is supplied by batteries or from a mains supply.  Ask students to name the hardware components from the functions given. **(Activity 5.1.1)**  Explain that the ways in which these hardware components relate to each other in a computer is called the ‘computer architecture’. The choice of hardware components affects the computer’s speed and power, as well as how much the computer costs and how it can be used.  Explain that the CPU is connected to the other components by wiring known as buses. There are three types of buses:   |  |  | | --- | --- | | Data bus | Transfers binary data from place to place | | Control bus | Sends and receives signals that control the CPU and other components | | Address bus | Which carries the binary address of memory locations |   Show how the data, address and control buses send electronic signals between the CPU and memory, and also to other devices such as the hard disk drive: [www.en.wikipedia.org/wiki/File:Computer\_system\_bus.svg](http://en.wikipedia.org/wiki/File:Computer_system_bus.svg). Show the pupils a bus either as cable or tracks in a printed circuit board. Get students to complete a diagram representing how the hardware components are connected using buses, and summarising what the different buses do. **(Activities** **5.1.2** and **5.1.3)**  If you have time, introduce the students to the Raspberry Pi® as another example of a digital computer: [www.youtube.com/watch?v=Jj4pjfU\_-jo](http://www.youtube.com/watch?v=Jj4pjfU_-jo)  Explain that the Raspberry Pi is an example of a cheap digital computer and includes some of the same hardware components we have just identified in the PC. Show the pupils a Raspberry Pi (or a picture of a Raspberry Pi). Explain that the CPU is an ARM processor and is a ‘System on a Chip’ where the memory is placed physically above the CPU. Get students to label the parts of the Raspberry Pi. **(Activity 5.1.4)**  If you have access to a Raspberry Pi®, you could get students to insert the SD card and power cable.  **Homework:** **Activity 5.1.5**. Students will search and label an impage of a computer and compare with someone else in the class. | Week 5  Lesson 1 activities    A desktop computer and a bus (either cable or PCB)  ‘Raspberry Pi – Tutorial 1 – An Introduction’ |
| 2 | 1.1.8 | **Problem solving: standard algorithms** – finding maximum and minimum | Give students a list of integers (not in order) such as 6,9,1,27,2,42. Which is the largest? How did you find out?  One solution is to note the first number and then compare the next number. If the next number is the same as or smaller than the noted number then move to the next number. If the next number is larger than the noted number then update the note to show the new largest number encountered so far. Repeat until entire list done. Note will contain the largest number.  Ask students to create a flowchart showing the process just described.  Next students should change the flowchart into pseudocode. Assume that the numbers will be stored within the program in a list.  Implement in Python and test.  Now students should independently create the flowchart, pseudocode and implementation for finding the minimum number from a list.  Remind students that build in methods exist for finding the minimum and maximum values in a list in Python. Discuss with students why they need to know these kinds of algorithms and when they might be useful.  Possible reasons for needing to know:   * Not all languages implement these features. * These features might not be implemented for the type of data they want to work on (for example in some languages you can easily find the minimum value in a 1 dimensional array using in built functions but not in a 2 dimensional array).   You need to understand how a process works just in case it doesn’t return the value(s) you expect as this will help with debugging.  **Homework:** Ask students to work through <https://www.khanacademy.org/math/math-for-fun-and-glory/puzzles/brain-teasers/v/finding-heavier-ball>. However to discourage just watching and not trying to problem solve, students should note down their thought processes whenever they are asked to pause and think. More detail in **Activity 5.2.1**. | Week 5  Lesson 2 activities |

| **Week 6** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.2.1  4.2.3 | **Hardware**:internal components | Start the lesson by recapping what the pupils have learnt so far about the CPU (also known as a processor or microprocessor). Explain that all computers contain a processor (PC, mobile phone, tablet, Raspberry Pi, etc.) Take a CPU out of an unused computer and hand round the class, or show an image of a CPU: [www.en.wikipedia.org/wiki/File:Intel\_80486DX2\_bottom.jpg](http://en.wikipedia.org/wiki/File:Intel_80486DX2_bottom.jpg)  Point out that the CPU has lots of ‘legs’ to provide connections to the rest of the computer and explain that they often have a heat sink to remove the excess heat.  Explain that processors are made of transistors. A transistor is an on/off switch controlled by an electric signal. Explain that if they looked inside the CPU they would see a silicon sheet containing thousands of transistors. Show an image of transistors <http://www.compoundsemiconductor.net/article/87832-imec-prepares-the-ground-for-iii-v-transistors-on-silicon.html>  The manufacture of CPUs is a fascinating process involving photographic etching on a wafer of silicon. Manufacture of CPUs is not part of the specification but, if your students are interested (and you have time), show the ‘How microchips are made’ YouTube video: [www.youtube.com/watch?v=F2KcZGwntgg](http://www.youtube.com/watch?v=F2KcZGwntgg) and see [www.newsroom.intel.com/docs/DOC-2476](http://newsroom.intel.com/docs/DOC-2476)  To help students understand the technology that makes it possible to pack millions of transistors on to silicon wafers, show the Royal Institution lecture ‘How to make a microprocessor’: [www.youtube.com/watch?v=RHAso1yM-D4](http://www.youtube.com/watch?v=RHAso1yM-D4)  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. Show the Royal Institution lecture ‘Breaking the speed limit’ which demonstrates just how fast computers are: [www.youtube.com/watch?v=FTpdAjre8LU](http://www.youtube.com/watch?v=FTpdAjre8LU) (7.00 – 11.04 minutes)  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. One hertz is one cycle per second. CPUs are so fast they are measured gigahertz (GHz). A two GHz clock sends out 2 billion pulses per second. Explain that at each clock pulse an instruction is executed in a CPU.  Draw the ‘ON’ and ‘OFF’ sequence showing the clock cycles:  ON (1)  OFF (0)  Get pupils to work out the speed of the processors (in instructions per second) in some of the devices they use. **(Activity 6.1.1)**  Show how the number of transistors on CPUs has been increasing: [www.en.wikipedia.org/wiki/Transistor\_count](http://en.wikipedia.org/wiki/Transistor_count). Explain that more transistors closer together mean that more calculations can be performed more quickly.  Point out that the switching of transistors generates heat. To help pupils understand how, run an unplugged whole class activity. **(Activity 6.1.2)**  To help pupils understand just how much heat is generated, show part of the Royal Institution lecture ‘Breaking the speed limit’ (31.04 – 35.05). Show the class the heat sinks (image <http://www.pcstats.com/articleimages/201311/LGA1156_DSCF5046.jpg>  used in different computers to remove the heat.)  **Homework:** Watch all of the Royal Institution lecture ‘Breaking the speed limit’ and then answer the questions in **Activity 6.1.3**. | Week 6  Lesson 1 activities  A CPU or an image of a CPU |
| 2 | 1.1.2  1.1.8  1.1.9 | **Problem solving: standard algorithms 2** – linear search | Start the lesson by asking student how to calculate the mean of a series of numbers.  Explain **Activity 6.2.1** and ask students to complete the exercise.  Explain what a linear search is (also known as a serial search).  A linear search is the simplest type of search to implement but it isn’t efficient. It involves going through each piece of data and checking to see if it matches the data being searched for. Show up to 1 minute 30 seconds of [www.youtube.com/watch?v=hi-lwJRQ1-s](https://www.youtube.com/watch?v=hi-lwJRQ1-s) to show how the linear search works.  You might wish to refresh what an array is and that in Python lists can be used as a kind of array.  Now ask students to complete **Activity 6.2.2**.  The next few seconds of the YouTube video above show a pseudocode version of a linear search. The students should update their own algorithm if necessary, but this is a good opportunity to show how different algorithms achieve the same thing.  **Homework:** **Activity 6.2.3**. Ask students to find out about other types of searches. What are the advantages/disadvantages of each search type they find? |  |

| **Week 7** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.2.1  4.2.2 | **Hardware**:internal components | Explain that this lesson will look at how data and instructions are stored in a computer. Remind pupils that all instructions and data are stored as bits (1 or 0). Review the roles of the address, data and control bus.  Remind pupils that the amount of information that a storage device can hold is measured in bytes (where 8 bits = 1 byte). Do a quick class review of data storage units (kilobyte, megabyte, gigabyte, terabyte).  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.  *Main memory*  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. All programs and applications that are running will have data and instructions loaded into RAM. Show RAM taken from a PC or show an image: [www.en.wikipedia.org/wiki/File:Memory\_module\_DDRAM\_20-03-2006.jpg](http://en.wikipedia.org/wiki/File:Memory_module_DDRAM_20-03-2006.jpg)  *Memory addressing*  Tell students that it is helpful to think of memory locations as adjacent ‘slots’, which can hold a piece of data or instructions. The memory is split up into 8 bit (or 1 byte) storage locations. Each storage location is given a unique address.  Ask students to store data and instructions at specified memory addresses **(Activity 7.1.1)**. You may find it useful to recap ASCII and binary numbers first.  Explain that the size of the address bus determines the maximum number of unique memory addresses that can be used.  Get students to calculate the size of the maximum addressable memory given the size of an address bus **(Activity 7.1.2)**. Explain that the width of the address bus can increase the size of memory that the processor can address.  Ask students to find out how much memory (or RAM) is in the computers they use (desktop computer, laptop, tablet, mobile phone, Raspberry Pi). They should complete the table in **Activity 7.1.3**.  Explain to students that the computer may not contain the maximum amount of addressable memory.  *Memory read*  Walk pupils through the stages involved in a memory read.   |  |  | | --- | --- | | Stage 1 | Processor places the address of the required memory location on the address bus | | Stage 2 | Processor sets the read line on the control bus | | Stage 3 | Contents of the memory location are conveyed along the data bus into the processor |     *Memory write*  Walk pupils through the stages involved in a memory write.   |  |  | | --- | --- | | Stage 1 | Processor places the address of the required memory location on the address bus | | Stage 2 | Processor places the data to be written to memory on the data bus | | Stage 3 | Processor sets the write line on the control bus | | Stage 4 | The data is conveyed along the data bus to the memory location |   Students can order the stages involved in a memory read and a memory write for themselves. **(Activity 7.1.4)**  Explain what ROM (Read Only Memory) is and how it differs from RAM. Ask students to research what ROM is needed for (initial start-up of a computer system, in embedded computers, etc.). Embedded computers will be covered in detail in Year 11.  **Homework:** Search online for companies that sell memory (such as <http://uk.crucial.com/gbr/en>). Find three further facts about memory that you can share with the class. **(Activity 7.1.5)** | Lesson 7  Week 1 Activities |
| 2 | 2.4.3 | Reading CSV files | Ask students to find out what a CSV (Comma Separated Value) file is by doing some online research.  Ask students to create a CSV file in a spreadsheet containing the data in **Activity 7.2.1**. The data is supposed to be a list of students with a score for a test out of 10. Ensure that students do save as a CSV, not as a spreadsheet file. Get students to open the CSV file in a program like Notepad to show the structure of the file.  Ask students to research why CSV files are used, note in **Activity 7.2.2**.  Students should key in the Python program in **Activity 7.2.3**, update the path and filename to match the CSV file created above.  Students should research how the code works using the python documentation. Another useful resource is [www.automatetheboringstuff.com/chapter14/](https://automatetheboringstuff.com/chapter14/).  Ask students to change the last line of code from **Activity 7.2.3** to: print(row[0]).  What has this done? How could they show the scores only? Could they show the names and scores in an easier to read format than in the original activity?  Ask students to modify the program to:   * 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.   **Homework:** Modify the program to output the names of the student(s) who scored the highest marks. | Week 7  Lesson 2 activities |

| **Week 8** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.2.1  4.2.3  4.2.2 | **Hardware**:internal components | Ask students to think back to the lessons on processors and tell them that now we are going to look at the main components that make up the CPU.  Explain that the CPU comprises three main components: the control unit, the Arithmetic and Logic Unit (ALU) and the registers.  *Control unit*  The control unit sends out signals to other parts of the computer system and is responsible for organising the fetching, decoding and execution of instructions.  *Arithmetic and logic unit (ALU)*  The ALU carries out arithmetic operations (+, -, \* , /, etc.) and logic operations (AND, NOT, etc.) It can include the ability to compare a number with zero or to test if two numbers are equal.  *Registers*  The registers are individual storage locations that hold an instruction, data or address of a memory location. Each type of processor has a number of different registers each designed for a specific function to hold a particular bit of information. Most processors have the following registers:   * The ‘instruction register’ holds the instruction that is currently being executed by the processor. * The ‘accumulator’ is a register that holds the accumulated total of results performed in the ALU. * The ‘program counter’ is a register that holds the memory address of the next instruction to be executed.   *Cache Memory*  **Note**: students may be more familiar with browser cache – make sure they realise there is a difference. This is returned to in the lesson homework.  Remind students of what RAM is and that generally it is located away from the CPU. Remind students that components within a computer system may operate at different speeds and that modern CPUs can be much faster than the RAM they are connected to.  Show students [www.youtube.com/watch?v=6JpLD3PUAZk](https://www.youtube.com/watch?v=6JpLD3PUAZk) (slow start, gets better after introduction!)  Ask students to make a note of what cache memory is, why it is used and where it is often located.  Ask pupils to summarise the information you have shared about the CPU. **(Activity 8.1.1)**  **Homework**: Students will research and document what browser cache does. | Week 8  Lesson 1 activities |
| 2 | 2.4.3 | Writing CSV files | Recap last lesson, reminding students what a CSV file is and how it is structured. Explain python can write CSV files as well as read them.  Ask students to try creating a CSV file using the code in **Activity 8.2.1**. Discuss the efficiency if a large amount of data needed to be written.  Explain that python can write a list of lists as a CSV file. Ask students to experiment with the code in **Activity 8.2.2** and add comments explaining how it works.  Students should key in the code from **Activity 8.2.3**, test it and examine the CSV file saved by the program.  Ask students to try extending the program to ask the user for a note for each item and add this to the shopping list. (The note could be something like which shop to buy from or where in the supermarket the item could be found)  **Homework:** Ask students to find and try out ways of writing CSV files without creating a list of lists. They should bring examples to the next lesson. | Week 8  Lesson 2 activities |

| **Week 9** | |
| --- | --- |
| **Lesson** | | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.1 | | **Network security**: overview | Starter: Ask students in pairs to discuss and list all the possibly confidential information that is stored on the school network  Discuss answers and build list on board. (Student personal details including address and medical, staff details, exam grades, references, etc.). Ask students about how much damage could be done if all information became public.  Discuss how might the school network or any other network be protected? Should hopefully get answers around passwords, password complexity, firewalls.  Students need to know what ‘authentication’ is. This video ([www.youtube.com/watch?v=t4kTgjQabV4](https://www.youtube.com/watch?v=t4kTgjQabV4) – you might want to stop at 6 minutes 35 seconds) introduces authentication and basic password security. Students to complete **Activity 9.1.1**.  Physical Access – students need to know what physical access is, why it is important and how it can be controlled. Watch ([www.youtube.com/watch?v=xatI10UGTRU](https://www.youtube.com/watch?v=xatI10UGTRU)) from 1 minutes 44 seconds onwards. Students should take notes using Activity 9.1.2  Firewalls – students need to know what a firewall is and what protection it offers.  Ask students to research and create a brief presentation that answers the following questions:   * What is a firewall? * Is a firewall software, hardware or both? * What does a firewall do?   You may wish to give them the following websites as a starting point:   * [www.windows.microsoft.com/en-gb/windows/what-is-firewall - 1TC=windows-7](http://windows.microsoft.com/en-gb/windows/what-is-firewall#1TC=windows-7) * [www.kaspersky.co.uk/internet-security-center/definitions/firewall](http://www.kaspersky.co.uk/internet-security-center/definitions/firewall) * [www.pctools.com/security-news/what-does-a-firewall-do/](http://www.pctools.com/security-news/what-does-a-firewall-do/)   **Homework:** Finish the presentation on firewalls. | Week 9  Lesson 1 activities |
| 2 | 6.1.3 | | **The Bigger Picture**: copyright, licensing and intellectual property | Start the lesson by showing [www.youtube.com/watch?v=eATwzWz1Dzw](https://www.youtube.com/watch?v=eATwzWz1Dzw) (copyright video).  Explain what the terms ‘copyright’, ‘patent’ and intellectual property’ mean.  Explain that computer software is intellectual property and is protected by copyright law, which gives the owner of a work certain rights over it and makes it illegal for others to copy, change or distribute the work without permission. Hardware inventions, on the other hand, are protected by patent law.  Explain that some individuals and organisations license their copyrighted software and/or patented technology to others for a fee and often earn additional royalties from the sale of products that use them (ARM is a good example of this). If someone invents a new piece of computer technology they have to apply for a patent. In contrast, copyright is automatic.  Ask students to make a poster explaining what copyright is and why everyone needs to be aware of whether the images, music, etc. they are downloading or using are covered by copyright.  Next, ask students to read *The Telegraph* articles: ‘Apple files patent for voice-based photo tagging in Siri’ and ‘British tech firms 'losing out’ in rush to patent’ and answer the questions in **Activity 9.2.1**. This activity will also require some research.  [‘Apple files patent for voice-based photo tagging in Siri’, article in *The Telegraph*, 30 December 2013 (available at [www.telegraph.co.uk/technology/apple/10542104/Apple-files-patent-for-voice-based-photo-tagging-in-Siri.html](http://www.telegraph.co.uk/technology/apple/10542104/Apple-files-patent-for-voice-based-photo-tagging-in-Siri.html))]  [‘British tech firms 'losing out’ in rush to patent’, article in *The Telegraph*, 11 March 2014 (available at <http://www.telegraph.co.uk/finance/businessclub/10688230/British-tech-firms-losing-out-in-rush-to-patent.html>)]  Explain that smartphone patent wars have been going on around the world for several years. Ask students to read the BBC news article: ‘Patent wars: Tech giants sue Samsung and Google’ and answers the questions in Activity 9.2.2.  [‘Patent wars: Tech giants sue Samsung and Google’, news article from the BBC (available at [www.bbc.co.uk/news/technology-24771421](http://www.bbc.co.uk/news/technology-24771421))]  Also worth reading:  [‘Apple’s $119.6 Million Patent Win Against Samsung Thrown Out’, news article from Bloomberg (available at [www.bloomberg.com/news/articles/2016-02-26/apple-loses-appeal-in-119-6-million-samsung-patent-case](http://www.bloomberg.com/news/articles/2016-02-26/apple-loses-appeal-in-119-6-million-samsung-patent-case) )  Show students [www.youtube.com/watch?v=8YkbeycRa2A](https://www.youtube.com/watch?v=8YkbeycRa2A) and ask them to research and discuss the Creative Commons licensing. As a class discuss why people might want to distribute their work using the Creative Commons.  **Homework:** **Activity 9.2.3**. Students should look for recent news articles regarding copyright/patents. At the time of writing, articles about why the Grooveshark website was taken down are interesting to students. | Week 9  Lesson 2 activities |

| **Week 10** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.3 | **Network security**:  cyberattacks, social engineering, phishing and shoulder surfing | Start the lesson by asking students to note what they think the term cyberattack refers to.  Discuss the student responses. One definition can be found at [www.oxforddictionaries.com/definition/english/cyberattack](http://www.oxforddictionaries.com/definition/english/cyberattack). Does it cover everything the students have said? Do they seem to understand the concept?  Get students to read the case studies in [www.cpni.gov.uk/documents/publications/2014/oxford-economics-cyber-effects-uk-companies.pdf?epslanguage=en-gb](https://www.cpni.gov.uk/documents/publications/2014/oxford-economics-cyber-effects-uk-companies.pdf?epslanguage=en-gb) (pages 44–52) and comment on whether they knew that cyberattacks might be state-sponsored and the damage/issues they can cause. The rest of the document is an interesting read for background knowledge.  Explain that social engineering is a form of cyberattack and that social engineering can be defined as ‘manipulating people to give out private or confidential information’. Students need to know about phishing and shoulder surfing.  *Phishing*  Ask students to complete **Activity 10.1.1**. They should research the answers and/or watch [www.youtube.com/watch?v=9TRR6lHviQc](https://www.youtube.com/watch?v=9TRR6lHviQc)  *Shoulder surfing*  Ask students to research and answer **Activity 10.1.2**. This should be a very quick activity.  Students also need to know that cyberattacks might be based on technical weaknesses of the systems they want to attack. They specifically need to know about unpatched software, USB devices, digital devices and eavesdropping. This can be finished in the next lesson.  Ask students to read pages 13 and 14 of  www.gov.uk/ government/uploads/system/uploads/attachment\_data/file/400106/Common\_Cyber\_Attacks-Reducing\_The\_Impact.pdf and make some brief notes  **Homework:** **Activity 10.1.3** family research on phishing. | Week 10  Lesson 1 activities |
| 2 | 6.1.3 | **The Bigger Picture**:open source and proprietary software | Start the lesson by asking students to find out the cost of buying a major piece of software such as the Microsoft Office suite as a private individual.  Discuss why the company feels justified in charging that amount of money. What needs to be paid for when software is being created? (Programmer’s salaries, research, patents, marketing updates, etc.) Does the price seem worthwhile? What would happen if a business pirated a copy of MS Office rather than purchasing it?  Ask students to read [www.zdnet.com/article/uk-firm-fined-250000-over-unlicensed-software/](http://www.zdnet.com/article/uk-firm-fined-250000-over-unlicensed-software/)  Explain that proprietary software is sometimes known as closed source software. There is an alternative: open-source software. Explain what this means and that major software like LibreOffice ([www.libreoffice.org/](https://www.libreoffice.org/)) and Linux is open-source.  Open-source has advantages and disadvantages. These YouTube videos try to explain using different analogies.   * Using Lego and skateboarding [www.youtube.com/watch?v=a8fHgx9mE5U](https://www.youtube.com/watch?v=a8fHgx9mE5U) * Using a cookie recipe [www.youtube.com/watch?v=Tyd0FO0tko8](https://www.youtube.com/watch?v=Tyd0FO0tko8)   Ask students to write a balanced argument explaining the advantages/disadvantages of proprietary and open-source software **(Activity 10.2.1)**.  **Homework:** **Activity 10.2.2**. Students should download and install Libreoffice and answer the questions in the homework activity. Discuss any potential issues they may have with doing this at home (some homes have network restrictions set up) and if it is difficult ask them to do it in school if possible. | Week 10  Lesson 2 activities  LibreOffice |

| **Week 11** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.3 | **Network security**: USB devices, digital devices and eavesdropping | Recap last lesson about cyberattacks.  *USB devices*  Ask students to read [www.pcworld.com/article/2460540/most-usb-thumb-drives-can-be-reprogrammed-to-silently-infect-computers.html](http://www.pcworld.com/article/2460540/most-usb-thumb-drives-can-be-reprogrammed-to-silently-infect-computers.html)  It would be worth mentioning that even trying to gain access to any computer system/account you aren’t authorised to is an offense under the Computer Misuse Act, and punishments range from fines to a prison sentence.  *Digital devices*  Many different types of devices exist to aid cyberattackers. You could show the USB Rubber Ducky video ([www.usbrubberducky.com/#!index.md](http://usbrubberducky.com/#!index.md)) or similar.  *Eavesdropping*  Secretly listening in to data being sent/received on a network. This can be done by plugging in a computer into a network that is running specialist software that listens for all traffic, as opposed to just the traffic sent to that particular computer.  Ask students to create a poster or presentation suitable for Key Stage 3 explaining what a cyberattack is and the types of cyberattack detailed above **(Activity 11.1.1).**  One of the potential major impacts of cyberattacks is their ability to take the powergrid down. This video is based on the USA and based around one authors book the contents are true and just as applicable to the UK powergrid. [www.youtube.com/watch?v=z2-rkZOL8C4](https://www.youtube.com/watch?v=z2-rkZOL8C4)  Discuss the potential impact around this kind of cyberattack. (Think about traffic lights, hospitals, food refrigeration, etc.)  **Homework:** **Activity 11.1.2**. Look for a news article on a recent cyberattack. Be ready to discuss in a future class. | Week 11  Lesson 1 activities |
| 2 | 1.1.8 | **Problem solving: standard**  **algorithms 3**: sorting | Begin the lesson by asking the class for examples of sorted lists of items that they are familiar with. Their suggestions may include: contact lists sorted by name, text messages and emails sorted by sender, files sorted by date, departure boards sorted by time, league tables sorted by scores.  Make sure that pupils know the difference between sorted and unsorted data. Provide them with examples of sorted and unsorted data in a variety of formats. Explain what is meant by sort ascending (1–10 and A–Z) and descending (Z–A and 10–1).  **Practical activity:** Give each student a pile of cards with numbers printed on them (some numbers should appear more than once) and ask them to sort the cards into ascending order. How do they do this? What ordering property do they use? What does ascending order mean? Does everyone use the same method of sorting?  Discuss how sorting is about re-ordering the items according to whether values are larger or smaller than each other. When sorting the number cards students compared one card with another and then, if they were out of order they swapped the cards. With numbers the ordering property is the size of the number.  Demonstrate that at each comparison in a sort there are only three possible results:   * item X is greater than item Y (X>Y) * item X is equal to item Y (X=Y) * item X is less than item Y (X<Y).   Explain that sorting is one of the most studied problems in computer science and that a large number of different sorting algorithms have been devised, including the bubble sort and the merge sort. Students need to know how both these sorting algorithms work. The most straightforward of these is the bubble sort.  Show the Hungarian dance bubble sort video (sort starts at 0:53 seconds) – draw attention to the comparisons and the swaps: [www.youtube.com/watch?v=lyZQPjUT5B4&feature=kp](http://www.youtube.com/watch?v=lyZQPjUT5B4&feature=kp)  Use the Wikipedia bubble sort animation to explain that when given a list of numbers a bubble sort algorithm iterates through the list several times: [www.en.wikipedia.org/wiki/Bubble\_sort](http://en.wikipedia.org/wiki/Bubble_sort)  Each time it ‘bubbles’ up the largest item to its correct place by comparing and swopping adjacent items. It then repeats the same for the next largest item in the list. When no further swaps are needed the list is sorted.  Using the algorithm on [www.pythonschool.net/data-structures-algorithms/bubble-sort/](http://www.pythonschool.net/data-structures-algorithms/bubble-sort/), students should implement a bubble sort in python. Use a list to store the initial data. The video shows the python implementation of the bubble sort.  **Homework:** **Activity 11.2.1**. Merge sort research and comparison with bubble sort. | Week 11  Lesson 2 activities |

| **Week 12** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.5 | Protect software from cyberattacks:  design and code reviews | Start the lesson by asking how what effects hacking could have.  Ask students to find financial / Denial of Service type hacking articles on effects of hacking.  This video shows cars being hacked and then controlled remotely ([www.youtube.com/watch?v=zurrQiETDHA](https://www.youtube.com/watch?v=zurrQiETDHA)). A Fox News section also shows the remote control of cars. The video mentions that a patch can be applied at a dealer or you can download the patch from the internet and install it yourself.([www.youtube.com/watch?v=-FqsW5IodkM](https://www.youtube.com/watch?v=-FqsW5IodkM)).  Next, ask students how we could protect software systems from cyberattacks. Give students a chance to discuss in pairs before feeding back to the class.  This question is likely to get lots of low-level type responses, such as secure password policies and locking equipment away. State that creating a secure system is hard and needs to start from the design stage. Does the stereo system in a car need to be connected to the same communication system as the brakes/accelerator, etc.?  Discuss the ‘Top 10 Secure Coding Practices’ from [www.securecoding.cert.org/confluence/display/seccode/Top+10+ Secure+Coding+Practices](https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices)  Some students may need help understanding the content but they should be able to grasp the concepts with help. The photograph towards the bottom of the webpage helps illustrate the points. Ask students to make a note of their top five secure code tips that are relevant to GCSE level coding **(Activity 12.1.1)**.  Another method to avoid vulnerabilities making it into final systems is to use code reviews. Ask students to research and write a definition of what a code review is, what a peer code review is, what an automated code review is and why they aren’t always used. **(Activity 12.1.2)**.  Explain what modular testing is and how ensuring each component part of a system works correctly helps with security.  **Homework:** **Activity 12.1.3**. Peer review a classmate’s code and make notes on whether it fits with the top five secure code tips that you wrote this lesson. | Week 12  Lesson 1 activities |
| 2 | 1.1.8  1.1.9 | **Problem solving: Standard algorithms** **4**:  binary search | Begin the lesson by showing the ‘Santa’s dirty socks’ video: [www.csunplugged.org/divideAndConquer](http://csunplugged.org/divideAndConquer)  Remind students about how the linear search worked and the problems associated with it.  Explain that the binary search algorithm uses the ‘divide and conquer’ strategy and is much more efficient than the linear search algorithm which uses ‘brute force’. Emphasise that the binary search only works if the list is sorted.  Use a set of ping pong balls, paper cups and jelly babies to demonstrate a binary search in action on a sorted list of numbers (see [www.youtube.com/watch?v=iDVH3oCTc2c](https://www.youtube.com/watch?v=iDVH3oCTc2c)).  **Practical activity:** Ask pupils to work in pairs. Give each pair a dictionary and ask them to use a ‘divide and conquer’ approach to finding the word ‘journey’. Ask them to keep a note of what they did.  **Practical activity:** Ask pupils to use the Python School website to learn about and implement a binary search algorithm in Python:  <http://www.pythonschool.net/data-structures-algorithms/binary-search/>  **Homework: Activity 12.2.1.** Ask pupils towrite an extended answer to the question: ‘A binary search (divide and conquer) is always more efficient than a linear search (brute force). Discuss.’  (Answers should include definitions of what is meant by the terms ‘efficient’, ‘brute force’ and ‘divide and conquer’; a description of how each algorithm works; an explanation of why a binary search is usually more efficient linear search algorithm and situations where this is not the case, that is when the list is unsorted list or when it is very short.) | Week 12  Lesson 2 activities |

| **Week 13** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.5 | **Protect software from cyberattacks**:  secure operating systems | **Please note – the following videos show bypassing the login password on Windows and OS X. Both of these bypasses work on standalone machines rather than networked computers. Neither technique is difficult or difficult to find (as can be seen by the videos being on YouTube). If you decide to show them you may wish to reinforce your school’s policies and that doing this on someone else’s computer without permission is illegal (Computer Misuse Act).**  Ask your IT systems department if they could send somebody along to talk to the students about how they implement operating systems security in your school.  The idea is to demonstrate that many operating systems aren’t secure and that this is an avenue for cyberattacks.  Hacking Windows 10 standalone machine [www.youtube.com/watch?v=HZmAeyNM-TM](https://www.youtube.com/watch?v=HZmAeyNM-TM)  Hacking OS X standalone machine [www.youtube.com/watch?v=-Heitu286gg](https://www.youtube.com/watch?v=-Heitu286gg)  Discuss how secure operating systems are and whether this matters. (Both Windows and OSX are considerably more secure when operated as a network, for example when Windows clients use Active Directory on a Windows Server for authentication.)  Explain that files on a networked system can be controlled so that users or groups of users can have permissions applied. The administrator can decide if someone can read, write or execute (plus some other choices) a file or a folder. As this is often restricted within schools you could show [www.youtube.com/watch?v=CHQ9LxpZJqk](https://www.youtube.com/watch?v=CHQ9LxpZJqk) to demonstrate. Using file permissions correctly can help secure an operating system.  Ensuring that networks are secure is a very important part of stopping cyberattacks from damaging a business. The National Security Agency publishes guidance on this on <https://www.nsa.gov/>. Show this site to the students and guide them through e.g. Information Assurance.  **Activity 13.1.1**. Students should to research what actions should be taken to secure a network. They could produce a list of top tips either aimed at home users or small businesses.  **Homework:** **Activity 13.2.1.** Conduct a security audit of some of the devices at home. | Week 13  Lesson 1 activities |
| 2 | 2.1.1  2.1.4  2.1.5  2.1.6 | **Programming challenge 3**: Morse code | Start the lesson by showing an introduction to Morse code [www.youtube.com/watch?v=bNoOYeS0gs0](http://www.youtube.com/watch?v=bNoOYeS0gs0) (possibly stop after 3 minutes 30 seconds)  Explain that students are going to implement an English to Morse code converter. Input will be typed into the program in capitals and the output will be dots and dashes. Give students a few minutes to discuss what they need to know to be able to do this.  As a group discuss what they need to know and how they might approach the problem. If needed remind students about the dictionary data type.  Get students to write pseudocode in pairs.  Next, students should write a test plan. Discuss and ensure that all aspects are tested by the plan, including hand tracing.  Finally, students should implement and test. The testing could be carried out as a peer assessment.  **Homework**: Use the winsound module to play the audio Morse code message. | Python winsound module |

| **Week 14** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.2.5  5.2.4 | **Protect software from cyberattacks**: audit trails  Identifying vulnerabilities | Start the lesson by stating that computers can log (record) every action taken on them. Ask what a school system might record, give students a few minutes to discuss.  Answers should include: What files being accessed, by whom (username) on which computer, what action was taken (modification, deleting, copying etc).  Discuss how much information might be generated in school if every action was recorded. Likely to be huge amounts. State that logs are also known as audit trails and can be generated by many different parts of a computer system and record many different details.  Ask students to complete **Activity 14.1.1** using [www.csrc.nist.gov/publications/nistbul/itl97-03.txt](http://csrc.nist.gov/publications/nistbul/itl97-03.txt), [www.networkworld.com/article/2193990/tech-primers/using-logs-for-forensics-after-a-data-breach.html](http://www.networkworld.com/article/2193990/tech-primers/using-logs-for-forensics-after-a-data-breach.html) and their own research to answer the questions.  Pose the question: How might a company ensure that its computer systems are as safe as possible?  Students need to know about: penetration testing (pen testing), ethical hacking, commercial analysis tools and reviewing network/user policies.  ‘Ethical hacking’ is the branch of computer science that incorporates cybersecurity and preventing cyberattacks. It is possible to get training/qualifications in Ethical Hacking that essentially teach how to hack from the point of view that knowing how to do this means that you can secure the network/computer systems.  What is ‘pen testing’? Ask students to fill in **Activity 14.1.2** whilst watching the video: <https://www.youtube.com/watch?v=b7jW9X9UqiY>  Commercial analysis tools – Many different tools are available to help a pen tester, some are open source and some are bought in commercial tools. These tools will search for common vulnerabilities (weak passwords, configuration problems, software that hasn’t been updated, etc.) and produce some kind of report. This report could be used to help fix security problems or as a starting point for a full pen test.  Network/User policies – Ask students to list what a network/user policy should cover and why they are needed. Students could then compare against the school policy and discuss additions/missing sections – why are they there/not there? Students could then work in small groups to write their own policy, perhaps with a particular client (such as a small business) in mind. Working with a scenario could be useful exam preparation.  Example policies can be found at [www.cpcstech.com/sample-network-computer-security-policies.htm](http://www.cpcstech.com/sample-network-computer-security-policies.htm) and questions to help write policies can be found in [www.watchguard.com/docs/whitepaper/securitypolicy\_wp.pdf](https://www.watchguard.com/docs/whitepaper/securitypolicy_wp.pdf)  It may be worth discussing how long the policy should be, that the technical language used should be understandable to average users and what areas it should contain.  **Homework:** Watch [www.youtube.com/watch?v=-rXdWb-8\_Xo](https://www.youtube.com/watch?v=-rXdWb-8_Xo). The video shows phone being hacked live on-air and the hacker being able to spoof calls. The idea is to remind students that it isn’t just laptops/desktop computers that can be hacked. | Week 14  Lesson 1 activities  ‘What’s a pen test?’ YouTube video |
| 2 | 2.2.1  2.1.2 | **Programming challenge 4**: FizzBuzz | Explain what ‘FizzBuzz’ is ([www.en.wikipedia.org/wiki/Fizz\_buzz](https://en.wikipedia.org/wiki/Fizz_buzz) )  FizzBuzz is a children’s game and has been used as a programming interview test by some people.  Students should plan in pseudocode how to produce a list of numbers from 1 to 100, replacing any number divisible by 3 with the word ‘Fizz’ and any number divisible by 5 with the word ‘Buzz’. Whenever a number is divisible by both 3 and 5 the word ‘FizzBuzz’ should be displayed.  Once pseudocode has been written then students should implement and test.  Allow time for students to look at other solutions, both in class and on the internet.  Some background information about the difficulty of finding programmers can be found here: [www.blog.codinghorror.com/why-cant-programmers-program/](http://blog.codinghorror.com/why-cant-programmers-program/)  Two solutions in Python can be found at [www.codereview.stackexchange.com/questions/763/two-fizzbuzz-solutions](http://codereview.stackexchange.com/questions/763/two-fizzbuzz-solutions) and solutions in many languages can be found at [www.rosettacode.org/wiki/FizzBuzz](http://rosettacode.org/wiki/FizzBuzz)  Students should discuss efficiency vs. readability:   * Which is more important? Why? * Did they come across any really difficult to read/understand solutions (such as the one line versions at [www.rosettacode.org/wiki/FizzBuzz - Python](http://rosettacode.org/wiki/FizzBuzz#Python))? * If you were hiring a programmer, what would you look for in the solution? * Can you think of a better way of finding out how good someone is at programming?   **Homework:** Look at the FizzBuzz solutions in other languages on the Rosetta code website <http://rosettacode.org/wiki/Rosetta_Code>. What do you think of other languages? Are there any you’d like to experiment with? |  |

# Suggestions for teaching this content

* Provide code as images (print screen) to force the students to type in the program code rather than just use copy and paste. They will learn more by making mistakes and having to correct them.
* Use two projectors on two screens. This allows you to display code on one screen and information or questions on the other. Use a plain wall or card on wall to make the screens.
* Divide the class into groups who work together to solve problems. Explain that they must discuss problems within the group before they ask for help. Introduce the concept of coaching other students within their groups to ensure everyone finishes within the set time. Have competitions between groups.
* Get students to ‘think, pair, share’ with problems (before asking the teacher a question) as well as using this to answer questions. Think about the problem by themselves, discuss the problem in a pair, then share it with a group around them.

## Starters and plenaries

Ideas for starters or plenaries which give students the opportunity to review computational thinking skills, reflect on their learning and build up specialist technical vocabulary.

* Give students some code and ask them what it does.
* Give students a flowchart and ask them to produce the code.
* Give students the pseudo-code and ask them to produce the code.
* Give students some code and ask them to depict as a flowchart.
* Give students some pseudo-code and ask them to depict as a flowchart.
* Give students a written description and ask them to depict as a flowchart.
* Give students a written description and ask them to produce the code.
* Give students a list of commands jumbled up and ask them to put them in the right order.
* Give students some code and ask them to find the syntax errors.
* Give students some code and ask them to produce a test plan.
* Give students some input data and ask for appropriate test data (valid, invalid, extreme).
* Ask students to summarise the lesson in 140 characters.
* Ask students to identify three ways in which a program could be improved.
* Who can find out about a topic first using Python help? (Do as a group.)
* Play videos introducing a particular aspect of Python programming.
* Ask students to produce five questions and answers on a topic.
* Online polls for formative assessment of topics.
* Peer-review of a program – how to give and receive constructive feedback.
* Self-assessment of a program.
* Invite students to ‘show and tell’ a program – what are the good features and the bad features? How could it be improved?
* ‘What would happen if I made this change to the program?’
* ‘Role play’ being a program, e.g. students act as manual debuggers and give values of variables at certain break points in a program.
* Use games, e.g. hangman with technical terms.
* Give a definition of a technical term and ask students to say what the term is.

## Suggestions for additional homework questions or extension exercises

* Ask students to summarise a topic in 140 characters or less.
* Ask students to produce a program in code/pseudo-code/flowchart given as a written description (or any combination of this).
* Ask students to write a written description of a program given as program code.
* Ask students to make a ‘topic in plain English’ video (for an example see: [www.youtube.com/watch?v=VumBNb6gcBk](http://www.youtube.com/watch?v=VumBNb6gcBk))
* Ask students to write a quiz (with answers) on a topic.
* Ask students to find the definitions of technical terms.
* Ask students to create a digital or classroom poster to explain a technical term or concept.
* Challenge students to find the five best websites to help with a particular topic.
* Ask students to search online to find the answers to some Python programming problem.
* Ask students to assess a program using an assessment checklist (give all the class the same program).

## Self-assessment of programming

By the end of the term, students should be able to assess their programs by asking the following questions.

* Does the program work? Are there any errors? How successfully does the program meet the requirements?
* How does this program work? What programming constructs have been used?
* Is the program readable? Have comments been included to explain the program? Would someone else understand the program?
* What are the good features and bad features of the program? How could it be improved?
* Did you learn anything new about programming from writing this program?
* Did you coach another student to help them solve this problem? If so, what did you learn from helping others?

Encourage students to use the correct technical terms when discussing their programs.

# Resources

## General resources

* *BCS Glossary of Computing and ICT*, 13th edition (ISBN 9781780171500)
* A range of articles on teaching coding: [www.edsurge.com/guide/teaching-kids-to-code](https://www.edsurge.com/guide/teaching-kids-to-code)

## Software

* Audacity (free open source, cross-platform software for recording and editing sounds) available to download from [www.audacity.sourceforge.net](http://audacity.sourceforge.net)
* Python (free open source software) available to download from [www.python.org/getit/](http://www.python.org/getit/)

## Python resources

* Official Python documentation (also available through help in IDLE): [www.docs.python.org/3/](http://docs.python.org/3/)
* Python summer school from Anglia Ruskin University is an excellent resource with videos and programming challenges: <http://www.pythonschool.net/>
* Python code for kids is a clearly written summary of the Python language written in accessible language: [www.pythondictionary.code-it.co.uk/](http://pythondictionary.code-it.co.uk/)
* ‘Python in 10 minutes’ is a quick run through of the basic concepts: [www.korokithakis.net/tutorials/python/](http://www.korokithakis.net/tutorials/python/)
* ‘Quintin Cutts – Too much doing, not enough understanding’ is a 20-minute video containing useful ideas and concepts on how to teach programming: [www.youtube.com/watch?v=Pim4aYfiZiY](http://www.youtube.com/watch?v=Pim4aYfiZiY)

## Free online books

* *Think Python How to Think Like a Computer Scientist* is a free online book with programming challenges at the end of each chapter: [www.greenteapress.com/thinkpython/thinkpython.pdf](http://www.greenteapress.com/thinkpython/thinkpython.pdf)
* *A Byte of Python* is an excellent online book, though it does not use IDLE as the editor: [www.swaroopch.com/notes/Python/](http://swaroopch.com/notes/Python/)

## Books

* *Python Programming for the Absolute Beginner*, M. Dawson (published Course Technology 2010) (ISBN 9781435455009) is an excellent book with clear explanations of each bit of code together with free downloads and example games built into each chapter.
* *Python for Kids: A Playful Introduction to Programming*, J. Briggs (published No Starch Press 2013) (ISBN 9781593274078).