Spring term Year 10

Scheme of Work

GCSE (9-1) Computer Science

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

# Spring term Year 10

## Assumptions

This scheme of work is based on the assumption that the learners have had previous experience of computer science or programming as part of the National Curriculum for Computing (2014). The first few programming lessons are therefore refresher lessons.

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 Spring 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.

# Spring term Year 10

| **Week 1** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.1 | **Networks** | Starter question: How useful is a standalone computer without any network connections?  Why computers are connected in a network: Ask students a series of questions to get them thinking about what a computer network is and why computers are networked.  What is a computer network? **(Activity 1.1.1)** A network is a group of computers and other devices connected together.  How many computer networks have you used today? **(Activity 1.1.2)**. Emphasise the wide availability of computer networks.  What does a computer network let you do? Ask students to create a list of the activities computer networks make possible. **(Activity 1.1.3)**  The list should include: the ability to exchange/share files and data; share physical resources, such as printers; communication (email, social media, video-conferencing); streaming content (videos, TV, music); storing data (cloud computing); accessing the internet; joining online communities.  **Homework:** Ask students to look at <https://wigle.net/> at home. Does it show the wireless network in their home? What could this data be used for? | Week 1  Lesson 1 activities |
|  | 5.1.2 | **LANs and WANs** | Explain that computer networks are categorised according to their geographic coverage.   * Local area networks (LAN) connect computers over a building or a site (small geographical area). * Wide area networks (WAN) connect computers over a larger geographical area such as a town, city or country.   Ask students to give examples of each type of network they use and to identify other examples, e.g.   * LAN: school network, home network * WAN: the Internet (a global computer network)   **Practical activity:** Set up a LAN in the classroom. (The type of computer network you set up will depend on the devices you have available. A connection to the internet is not required. The simplest approach would be to use a MiFi (3G to WiFi) to setup a WiFi hotspot in your classroom. Some students may have devices capable of setting up WiFi hotspots. You may find it useful to talk to your network manager about the possibilities available.)  If it is not possible to set up a network, get the class to plan how they would set up a LAN network. These links give some practical advice on setting up a network:  <http://www.argos.co.uk/static/BuyingGuide/bgTrail/wireless_networking2_00079.htm>  <http://windows.microsoft.com/en-gb/windows7/start-here-to-set-up-a-home-network-in-windows-7> **Homework:** Ask students to design their ‘dream’ LAN for their home. What devices would they want to connect (laptops, PCs, games consoles, phones)? How would they connect them (wired, wireless)? How would they access the internet? Whereabouts in the house would they put the router, the printer, the NAS drive and any other devices? What streaming services would they provide across the network? How would they protect their network from hackers and viruses? |  |
| 2 | 2.3.2 | **Data types and structures:** two dimensional arrays (lists) and nested ‘for’ loops | Review how to create lists, how lists are indexed using square brackets [] and how ‘for’ loops can be used to repeat an action for each item in a list **(Activity 1.2.1)**.  Explain how information can be stored in two-dimensional (2D) lists. Explain that these data structures are called arrays in some programming languages. In Python they are nested lists (lists within lists), so that the addressing is [0] [0] not [0,0] as in other programming languages.  Use the Battleships game to help students understand how two-dimensional lists (arrays) can be referenced using [row] [column]. (They are not expected to program this game.) **(Activity 1.2.2)**  Explain that nested loops can be used to process the information within  two-dimensional lists (arrays).  Demonstrate how to create and initialise a two-dimensional list (array) in Python. Explain that lists need to be created before they can be used (or the program would have to append the list).  Demonstrate how information can be stored in two-dimensional lists (arrays) and how it can be accessed. **(Activity 1.2.3)**  Work with the class to complete the two-dimensional list (array) table. **(Activity 1.2.4)**  Give students an opportunity to write programs that use two-dimensional lists. **(Activities 1.2.5 & 1.2.6)**  **Homework:** Set students a programming challenge requiring the use of a  two-dimensional list. Programs can be written in program code, pseudocode or flowcharts. | Week 1  Lesson 2 activities |

| **Week 2** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.2 | **Client-server and Peer-to-Peer** | Begin the lesson by introducing the concepts of client (someone who makes a request for something) and server (someone who responds to that request).  Explain that computers on the internet fall into two categories. Those that provide services (like Web servers or FTP servers) to other computers are servers. Those that request services are clients.  Explain what happens when you go to http://www.bbc.co.uk/news/ to get the latest news:  You are the client – you issue a request for the news page to be displayed. The BBC’s computer is the server – it responds to your request and the page appears on your screen.  Describe the two processes that take place:   * Client process: your browser sends a request to the web server for the web page. * Server process: the BBC web server accepts the request from the client, retrieves the requested web page and sends it back to your browser.   Explain that servers have to remain in a state of readiness waiting for incoming requests from clients, whereas clients can have bursts of activity and do nothing in between. One server generally supports numerous clients. The client needs to know of the existence and the address of the server, but the server does not need to know the address or even the existence of the client prior to the connection being established.  Show up to 3 minutes 30 seconds of the video ‘[Mr. Clarkson Talks About... Client-Server and Peer-Pe](http://www.youtube.com/watch?v=AWFLGFV4R4c&list=PLS47o-8L_5nwY0fYvCjM_Gi0qxmkTuzn0)er Network Models’: <http://www.youtube.com/watch?v=AWFLGFV4R4c>  Use a question and answer session to test students understanding of the  client-server model.  Show the video ‘The Client / Server model’: <http://www.youtube.com/watch?v=bW9vY0izaVA>  Ask students to write an explanation of client-server and peer-to-peer **(Activity 2.1.1)**.  **Homework:** Students to research why peer-to-peer networks are sometimes used to distribute /pirate copyrighted material (such as films, games, music) for free. | Week 2  Lesson 1 activities |
| 2 | 2.4.2 | **Validation:** try | Demonstrate why validation is required when entering ‘Y’ or ‘N’ into a program and what happens when some other character is entered.  Explain that validation is the automatic checking of entered data and that validation can only prove that the data entered is a reasonable value for the computer to accept, not that it is correct. Work with the class to complete the ‘types of validation’ table. **(Activity 2.2.1).**  Give students the opportunity to write programs with a range check, presence check, length check, type check and look-up check to test input data. **(Activities 2.2.2 – 2.2.6)**  Demonstrate the ‘try’ command to defend a program from suspicious code that could cause a particular type of error.  try:  Commands  except:  <if there is an exception execute this statement>  else:  <if there is no exception execute this statement>  Explain that the ‘try’ command allows a program to continue even if an error occurs. **(Activity 2.2.7)**  **Extension:** Regular expressions can be used as input masks to perform validation. See ‘import re’ and regular expressions.  **Homework:** Set students a programming challenge requiring validation. **(Activity 2.2.8)** | Week 2  Lesson 2 activities |

| **Week 3** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.4  5.1.3 | Network Data Speeds  Wired and Wireless Connectivity | **Network data speeds**  Explain that computer networks transmit bits (1s and 0s) from one device to another. The speed of a network connection is measured by how many bits can be transferred in one second (bits per second (bps)). Explain that the numbers are so large that they are measured in millions of bits per second (Mbps) and billions of bits per second (Gbps).  1 Mbps = 1,000,000 bits per second  1 Gbps = 1,000,000,000 bits per second  Students to note the meaning of the acronyms in **Activity 3.1.1**  **Calculating the time required to transmit a file**  Remind students that network data transmission speeds are measured in bits per second (bps) and that data storage is measured in bytes (where 1 byte = 8 bits and 1 KB = 1024 bytes).  Demonstrate how to calculate how long it would take to transmit a 5 KB file over a network with a speed of 10 Mbps.  Explain that it is useful to convert both the size of the file and the network speeds into bits then use the formula: time = size of file (in bits) / network speed (in bits).  5 KB = 1024 x 8 x 5 = 40,960 bits  10 Mbps = 10 x 1,000,000 bits per second = 10,000,000 bits per second  Time to transmit = 40,960 / 10,000,000 = 0.004096 seconds  Students to work out transmission times in **Activity 3.1.2**  Explain that this is the ‘best case’ scenario. In actual fact, various other factors are likely to increase the time it takes to transfer the file. Can students suggest any of the factors that can affect transfer speeds?  **Network media**  Outline the characteristics of different network media, i.e. copper cable, fibre optic cable and wireless. If possible, have some copper and fibre optic cables for students to look at. Explain how data is transmitted in each media, i.e. copper cable (electrical), fibre optic cable (light), wireless (radio frequency within the electro magnetic spectrum). Explain how the type of media used affects the speed of data transmission, the range over which the data can be transmitted and the security of the network.  You will find useful information at tutorialspoint: <http://www.tutorialspoint.com/data_communication_computer_network/transmission_media.htm>  **Practical activity:** Use Bluetooth on a mobile phone to transfer a file to other mobile phones in the classroom. Discuss the range (limited to metres for Bluetooth), speed (slow), security (poor) and the need for the receiving computer to understand the Bluetooth protocol.  Ask the class to rank in order of speed: a WiFi network, a school network using copper cables or a school network using optical fibre connections. (Slowest: WiFi, fastest: fibre optic).  **Practical activity:** Work with the class to produce the outline of an algorithm, which takes as its inputs the size of a file, the unit of storage and the network speed, and calculates the time needed to transmit the file. This could also be implemented if time allows.  **Homework:** Ask students to produce a presentation or video, which describes each transmission media, giving an indication of the range, bandwidth and security of each type of media. Alternatively or in addition, you could ask them to complete a table to help them revise network media basics, like the one below.   |  |  |  |  | | --- | --- | --- | --- | | Transmission media | Range | Transmission speed | Security | | Copper cable | 100s of metres | moderate | moderate | | Fibre optic cable | 1000s of metres | fast | good | | Wireless | Line of sight | slow | poor | | Week 3  Lesson 1 activities |
| 2 | 2.2.1  2.6.1  2.6.2  2.6.3 | **Subprograms:** procedures, functions, return, passing parameters | Explain that as programs get more complex it is useful to divide (decompose) them into smaller parts. Each sub-task is written as a function, which is defined using the def command.  Functions can have up to four parts:   1. A name that identifies the function 2. Parameters that allow data to be passed into a function as arguments 3. A body of commands 4. Return values to the calling program   Explain that a parameter is a special kind of variable, which refers to the data being passed into a subprogram. Arguments are the actual data values that are being passed into the subprogram. Explain how data values are passed into a subprogram. **(Activity 3.2.1)**  Explain that some programming languages make a distinction between a procedure (which executes commands and does not return any values) and a function (which returns a value). Python doesn’t do this. In Python, procedures are functions that do not return a value.  Demonstrate how to write and run functions from the IDLE shell and how to pass parameters into a function. **(Activity 3.2.2)**  Give students the opportunity to write some simple functions. **(Activities 3.2.3 – 3.2.5)**  Demonstrate the use of the return command to return a value, which is then assigned to a variable, and how to run functions from a controlling program. **(Activity 3.2.6)**  Remind students of the importance of indenting when defining functions and the need for the ‘ :’  Brainstorm the advantages of using functions. **(Activity 3.2.7**)  Describe how, in functional programming, the overall functions of the system are defined and these functions are split into smaller and smaller functions until they are small enough to be programmed.  **Homework:** Give students a program containing several functions and ask them to identify (using different colours) where the functions are called, the function definitions, any data passed as arguments and any return values that are passed back into the program. | Week 3  Lesson 2 activities |

| **Week 4** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.7 | **Network Topologies** | Begin the lesson by asking students to discuss how they think the network within your school is organised. Where are the servers? Can they spot any network hardware (e.g. switches, wireless access points etc.)?  Network topologies  Explain that the term topology refers to the way in which devices are connected on a network. Draw a diagram of each type of network topology, describe the characteristics of each (e.g. security, resilience, distance) and list the pros and cons of each.  **Practical activity:** Use string to act as a network cable and get students to create each network topology, where each student represents a computer/server/printer etc. Cutting the string can then be used to demonstrate what would happen to each network topology if a cable failed.  The table below summarises each topology, but you will find more information on Wikipedia: <http://en.wikipedia.org/wiki/Network_topology>   |  |  | | --- | --- | | Topology | Description | | bus | Single main cable to which computers are connected. | | ring | All computers are connected in a circle. | | star | All computers have their own link to a central computer. | | mesh | Where each computer can connect to other computers. |   As a class, watch the Computer Networking Topology tutorial and ask students to make notes: <http://www.youtube.com/watch?v=kfEDPQAYH4k>  **Activity 4.1.1** provides a structured way of taking notes on the different topologies.  **Practical activity:** Ask students to devise a quiz to test knowledge of network topologies.  **Homework:** Students to research why most networks are now star networks and the differences between a hub and a switch | Week 4  Lesson 1 activities |
| 2 | 2.3.4 | **Local and global variables** | Demonstrate how variable assignments are available in parts of a program when using multiple functions and describe the concept of ‘scope’ of a variable.  Explain that global variables can be used anywhere in a program whereas local variables are defined for use only in a function. **(Activity 4.2.1)**  Explain that local variables come into existence when a function is entered and the data they contain is lost when the execution of the function is completed. Local variables reduce the unplanned effect of a variable being used in another part of the program and accidentally being changed. The general rule is that global variables should only be used for data that needs to be shared between subprograms. Give students the opportunity to identify local and global variables. **(Activity 4.2.2)**  Discuss with the class why global variables need to be used with care.  Demonstrate how to write programs that use local and global variables and give students the opportunity to experiment.  **Homework:** Give students a program that uses functions, local and global variables and ask them to colour code the scope of the local and global variables. | Week 4  Lesson 2 activities |

| **Week 5** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.5 | Introduction to protocols | **Lesson preparation:** Download and familiarise yourself with the materials for the ‘Tablets of Stone’ Computing Science Inside workshop activity.The original version seems to have been taken down but CS Unplugged has this version: <http://csunplugged.org/network-protocols/> This activity will require printing and cutting out before the lesson.  Students need to understand why protocols are needed and what a protocol is.  Show students pictures of 1980s home computers and explain that if you bought a peripheral for one (a printer for a Commodore 64 for example) that it wouldn’t work with another make (a Sinclair ZX Spectrum for example). This was because they all ‘talked’ a different language in terms of what signals meant, what voltages were used, the type of error checking and the speed of data transmission etc.  Students to find a definition of what a protocol is (an agreed set of rules) and what a network protocol could specify. **(Activity 5.1.1)**  **Practical Activity:** Run the ‘Tablets of Stone’ Activity with the class.  **Homework:** Find out the difference between http and https and produce a poster explaining the difference and the advantages/disadvantages of each  **Alternative practical activity:** <http://people.cs.ubc.ca/~kla/index.php?page=Network_Routing_on_Strings>  Much more involved and contains too much for GCSE level but could be adapted.  **Homework:** Think about what protocols we follow in life and be ready to give an example next lesson. For example, how drivers respond to different coloured traffic lights is a kind of protocol. | Week 5  Lesson 1 activities  Images of 1980s computers |
| 2 | 2.3.5 | **Constants** | Explain that a constant is a variable with a fixed value that cannot be changed. Brainstorm the type of values that should be constants and why it is useful to have values that do not change. (Students may have come across examples from other subjects such as maths and physics.)  Split the students into groups to complete **Activity 5.2.1**. Each group should use flowcharts, pseudocode or written descriptions to complete the task. Ask groups to demonstrate their solutions to the class. Discuss the benefits of using these techniques. Let the students modify their designs and then program the solution.  Students can then complete **Activity 5.2.2**.  *There are no solutions for this lesson. The design is developed during the lesson and there should be many possible solutions.*  **Homework:** Compare flowcharts, pseudocode and written descriptions. What do you prefer, and why? Does this depend on the task that you are doing (design? programming?) | Week 5  Lesson 2 Activities |

| **Week 6** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 5.1.5 | **Protocols** | **Starter:** Students peer assess each other’s homework posters.  Students need to know that there are different protocols for different areas of networking. The Specification states that students need to know about the following protocols: Ethernet, Wi-Fi, TCP/IP, HTTP, HTTPS, FTP, POP3, SMTP and IMAP. This lesson covers all of these apart from TCP/IP, which is covered next lesson.  Link to previous homework and explain what the HTTP protocol is used for. Explain what the HTTPS protocol does differently from HTTP (data is encrypted on the client machine before being sent to the server).  Students should be split into small groups and given 15 to 20 minutes to research a protocol you’ve assigned out of the above list and then give a three-minute overview of that protocol. They need to state what the acronym expands to, what the protocol is used for and an example of where it is used.  Students could then make notes about each protocol in **(Activity 6.1.1)**.Alternatively students could be asked to research the protocols and find out what each protocol acronym stands for and what it is used for. **(Activity 6.1.1**) This activity could be done in pairs or small groups.  Wikipedia provides a good starting point for information on protocols. The introductory paragraphs of each protocol probably provide enough information for GCSE students. For example <https://en.wikipedia.org/wiki/Ethernet>  **Homework:** Find outwhy IMAP is now much more widely used than POP3. | Week 6  Lesson 1 Activities |
| 2 | 6.1.1 | **Environmental impact of technology**  (health, energy use, resources) on society | Starter: Get students to discuss the statement ‘Computers damage the environment’, thinking about the particular types of damage being done (health, energy use, waste/lack of recycling).  Emphasise that, whilst computers undoubtedly enrich our lives, there is a downside. One serious problem is their negative impact on the environment. Summarise the ways in which computer manufacture, use and disposal harm the environment. Ask students to suggest ways in which the damage computers cause to the environment can be reduced **(Activity 6.2.1)**.  Point out that it’s not all bad – computers are also helping to safeguard the environment, for example by reducing traffic congestion and air pollution in cities.  Ask students to watch the CSTA video: ‘Computer Science and the Environment’ and find examples of how computers are helping to safeguard the environment <https://www.youtube.com/watch?v=YQdmj6njzzw> . **(Activity 6.2.2)**  Display the statement. ‘Computers are the environment’s number one enemy.’ Get students to write an essay that gives a balanced argument on the ways in which computers harm the environment and the positive impact computers have on the environment.  Show students how to construct a balanced response to the statement, considering the benefits as well as the downsides, and taking into account what can be done to reduce the latter.  Point out that, as budding computer scientists, they are expected to have an awareness of the impact computers have on individuals, society and the environment and be able to express an informed opinion. There may well be an extended question in the exam on some aspect of this topic. They are easy to spot because questions like this usually start with the command word ‘Discuss’! | Week 6  Lesson 2 Activities |

| **Week 7** | |
| --- | --- |
| **Lesson** | | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | | 5.1.6 | **Layered protocol stacks (TCP/IP) and packets** | Recap what a protocol is and why it is needed. Students should answer this on their own as a starter activity in **Activity 7.1.1** and then work as a class to write an improved class agreed answer in **Activity 7.1.2**  Link to previous homework and discuss why IMAP is used more than POP3 now. The likely reason is that people use multiple devices to manage their email (computer, phone, tablet etc.) and IMAP is much better at synchronising changes, whereas POP3 was used to download messages from a server to an email client. Often the email client would delete the messages off the server as they only provided a limited amount of storage space.  Introduce the idea of protocols needing to be used together. For example, a computer might use Ethernet to connect to the school network and HTTP for transferring webpages.  TCP/IP (Transmission Control Protocol/Internet Protocol) is a protocol stack or protocol suite. A protocol stack is a layered collection of protocols. Each layer has a specific purpose. Students can write a definition in **Activity 7.1.3**.  The number of layers and the names of each layer vary depending upon source. RFC1122 specifies 4 layers and is used for the names below. Students will need to be guided about this, see layer names table for details on <https://en.wikipedia.org/wiki/Internet_protocol_suite>  The TCP/IP stack has four layers:   * Application layer * Transport layer * Internet layer * Link layer.   Set students the task of finding out what each layer does and what protocols are used in each layer (TCP/IP has a wide number of different protocols that can be used at each layer). This could be represented as a table. **(Activity 7.1.4**) A useful diagram can be found at <https://technet.microsoft.com/en-us/library/bb726993.aspx>  Introduce the concept of packets in the form that messages are broken up into small pieces (packets). Packets don’t have to follow the same route on a network, so packets might not arrive in order. A packet contains some data and a header that contains details to ensure that packets get to the correct computer on the network and can be put back in order.  Students should research what a packet header for IP contains and what each piece of data in the header is for. Findings could be recorded in **Activity 7.1.5**. Students should gain knowledge of what an IP address is and why they are needed. The teacher will probably need to explain about the differences between IPv4 and IPv6.  **Homework**: Students could be asked to create a revision sheet about protocols or research what other protocols are available and what they are used for. | Week 7 Lesson 1 Activities |
| 2 | 6.1.1 | | **Environmental impact of technology on society**  (health, energy use, resources) | Show the CSTA video ‘Computer Science and Medicine’ (first 90 secs). <https://www.youtube.com/watch?v=J_SMo-o2on8>  Summarise the main ways in which computers are having a positive impact on people’s health (monitoring the impact of new drugs, medical record keeping, robotic surgery, life support and monitoring systems, diagnosis by expert systems, MRI scanning, 3D printing of body parts, web-connected biometric devices, health promotion campaigns)  Ask learners to work in groups to research and identify health concerns associated with computers.  Split the students into groups and get them to create a presentation on the pros and cons of the environmental impact of technology on health.  **Homework:** Ask learners to find a recent news article about the environmental impact of technology and be able to summarise it for the class next lesson. The Guardian is a good source of these types of news article. |  |

| **Week 8** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.3.1 | **Boolean Logic** | Students should find out who George Boole was and complete **Activity 8.1.1** to give the historic background to Boolean values.  Explain how a Boolean value is either True or False and this can be easily represented using 0 (Zero) for false and 1 for true (relate to binary). In electronics this equates to part of a digital circuit having power or not.  Introduce idea of logic gates. A logic gate is a building block of electronic circuits. Most logic gates will take an input or two and then provide an output. These gates can be represented in a diagram using standard symbols and truth tables to show all possible inputs and associated output.  The specification requires students to know about AND, OR and NOT gates.  Use <http://www.neuroproductions.be/logic-lab/> and demonstrate wiring an AND gate to two switches for input and a bulb for output. Students should then use the logic lab to complete **Activity 8.1.2** to show the AND logic gate symbol and the associated truth table.  Students should then do **Activity 8.1.3** for the OR gate.  **Homework:** Students should be asked to watch <https://www.youtube.com/watch?v=_76g8EM4DVU> which covers all three gates, truth tables and combining logic gates in preparation for the next Boolean Logic lesson | Week 8  Lesson 1 Activities  Online logic lab |
| 2 | 2.1.3  2.1.7  2.1.6  2.1.5 | Errors and debugging tools in an Integrated Development Environment | Recap that a bug is a fault in a program that causes errors.  Describe and show examples of the three different types of errors: syntax, runtime and logic. **(Activity 8.2.1)**  Explain that a trace table helps to test for logical errors in a program. A trace table is completed by listing the values assigned to a variable during the running of a program. Demonstrate how to complete a trace table. **(Activity 8.2.2)**  Explain that some bugs in programs can be difficult to find. A debugger can be used to help find the bug. A debugger is a diagnostic program that helps in the detection, location and corrections of faults (or bugs) in a program.  Show learners a program that is not working and demonstrate how the debugger can be used to help find the problem.  The integrated development environment (IDLE) includes a debugger which provides the following tools:   * **Single-step** – where the program is executed one statement at a time under user control. This allows the user to observe the effects of each statement by performing a **step-through** of the program. * **Set break points –** which are positions in the program when the program is halted and the programmer can investigate the values of variables to help locate errors. * **Watchers –** to allow the programmer to watch for certain events such as variables changing.   Show learners how to role-play being a debugger by single-stepping through a program written on the board, ‘watching’ the variable values and stopping at break points. Have different learners displaying the command that is to be executed and showing the current variable values. Set a break point and get the learners to ‘interrupt’ when the break point is reached.  Explain that the IDLE debugger can be started by selecting Debug/Debugger in the Python shell. The words [DEBUG ON] will be displayed. Give learners the opportunity to run the debugger on programs that they have written. **(Activity 8.2.3)**  **Homework:** Ask learners to produce a video (using a mobile phone or tablet) explaining ‘How to use the IDLE debugger in plain English’ in the style of the YouTube video ‘Twitter in Plain English’: <http://www.youtube.com/watch?v=ddO9idmax0o> | Week 8  Lesson 2 Activities |

| **Week 9** |
| --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.3.1 | **Boolean Logic** | Recap what a logic gate is and what a truth table is for.  Get students to go back onto <http://www.neuroproductions.be/logic-lab/>.  Students should complete **Activity 9.1.1** to create the truth table for a NOT gate.  Explain that logic gates can be linked to each other to create complex circuits. Student should now complete **Activity 9.1.2** – answers could be written in exercise books or on a computer. This activity should ensure students can recognise the symbols and how the logic gates work. Students could self-assess their work by constructing the circuits in the online logic lab and check their answers.  Students should then be asked to draw solutions to problems such as the following:  Draw the diagram and the truth table for a greenhouse monitoring system. An alarm should sound if the temperature goes above 40 °C and the windows of the greenhouse are closed. You can assume that a sensor will output a 1 (on) when the temperature goes above 40 °C and that an alarm exists that will sound when a 1 (on) signal is sent to it.  Images of logic gate symbols can be taken from <http://www.clker.com/clipart-digital-logic-gates.html>, where they are declared as in the Public Domain without any restrictions for use.  **Homework:** Students should look for examples of Boolean logic around them. For example, a microwave won’t operate unless the door is closed and time has been set. These could be discussed next lesson. | Week 9  Lesson 1Activities  Online logic lab |
| 2 | 2.6.1 | Library sub-programs: import modules | Explain that import modules are libraries of subprograms that can be imported into programs. A module is a file that contains functions.  Explain the benefits of using imported modules, i.e. it allows programmers to re-use existing and pre-tested code that other programmers have written.  Show learners how they can import modules and use the dir(module\_name) to list the functions in that module. Examples of modules include:   * Import web access library: import webbrowser * Import sound library: import winsound * Import maths library: import math   Demonstrate how to import these modules and how to use the functions they provide. **(Activity 9.2.1)**  Give learners the opportunity to explore other modules. **(Activities 9.2.2 – 9.2.3)**  **Homework:** Ask learners to research other useful Python modules that can be imported and what they can be used for.  Organise a follow-up class vote or discussion on the most useful module. | Week 9  Lesson 2 Activities |

| **Week 10** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 4.3.2 | **Boolean Logic** | Show students the truth tables in **Activity 10.1.1** and ask them to note down which logic gate each represents to check understanding of truth tables.  Show students the logic gates in **Activity 10.1.2** and ask students to note down the names of the relevant logic gates to check that they can recognise the logic gate symbols.  Introduce students to the idea of a logic statement and how they are constructed. Students could write the logic statement for the washing machine problem from Week 9, Lesson 1.  Exam questions on Boolean logic can take several forms, including drawing a truth table or writing a Boolean expression that will explain how a particular system behaves. **Activity 10.1.3** can be used to show the kind of problem that might be given and then talk through the construction of the answer. Answer would be ‘S AND D AND W’ – obviously the order of the letters could be rearranged in this example without causing the output to be wrong. More complicated expressions may need to use ( ) to ensure that they are readable and are equated correctly. (The precedence order for operations are first NOT, second AND, third OR. Expressions inside brackets are evaluated first and this overrides precedence.)  Students should answer **Activity 10.1.4** individually and then the answer can be discussed.  Students should create a revision sheet/poster for the Boolean logic section of the course. It would need to contain:   * details of logic gates (AND, OR, NOT) * truth tables for each logic gate * the symbol for each logic gate * example questions and answers showing how to construct a logic statement.   Minecraft has logic gates and they are explained in this video: <https://www.youtube.com/watch?v=9EY_XoEImjM> ; however it should be noted that traditional logic gates only take one or two inputs, not more than that as Minecraft do.  **Homework:** Finish the Boolean logic revision sheet. | Week 10  Lesson 1 Activities |
| 2 | 6.1.2 | **Understanding the ethical impact of using technology (privacy, inclusion, professionalism) on society** | Begin the lesson by reminding students that, while computers are helping to improve our lives, this comes at a cost. Ask them if they would be happy if their parents could track their every move – wherever they went, whatever they did? How would they feel about you (their tutor) being able to do the same?  Ask learners to read Stephen Waddington’s blog post: ‘My mobile phone knows more about me than my family’ and answer the questions in **Activity 10.2.1**.  Stephen highlights ‘the important ongoing conversation about the benefit of technology versus privacy’. Ask students to define what privacy is and why it is important to protect it. How much privacy would they be prepared to give up in order to make life more convenient?  Point out that every time students post a comment on Facebook, tweet, download an app, search for information on Google, send an email, give permission for an app to use their location, they are giving away personal information about themselves. Also discuss that once something is put on the internet it may always be available and impossible to remove. You could talk about people downloading webpages and sites like the wayback machine.  Show the CIR video ‘Hot on Your Trail: Privacy, Your Data, and Who Has Access to Information IT’.  Ask students to read James Ball’s article ‘Me and my data: how much do the internet giants really know?’ and answer the questions in **Activity 10.2.2**.  **Homework:** Ask students to check all their privacy settings on Facebook, Twitter, Instagram etc. and update if necessary. | Week 10  Lesson 2 Activities |

| **Week 11** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.1.1  3.2.1 | **Data Representation**: Text | Remind students that computers store everything as binary. Pose the question how could text be stored? Discussion should be moved towards each letter having its own unique number. Students should then discuss with a partner the questions in **Activity 11.1.1**.  Questions in the activity:   * How would storing lowercase/capitals work? * How would storage of numbers work? * Other than letters and numbers, what else would need to be stored?   Bring class back together and work through answers Students should work out that lowercase/uppercase letters would have their own unique codes; that numbers 0 to 9 would need to have a unique code and that various other formatting codes (like CR) would need to be stored and therefore have their own unique codes.  Students should then discuss **Activity 11.1.2**. They may need prompting to think about file from Computer 1 being opened on Computer 2 or sending email from Computer 1 to Computer 2. This activity should make them think about the need for a standard way to store text using binary/decimal within a computer system.  Students could then fill in **Activity 11.1.3** on their own, so they have a reference to how ASCII is structured.  **Homework:** Research what the term ‘character set’ refers to. This feeds into the next lesson. | Week 11  Lesson 1 Activities |
| 2 | 6.1.2 | **Understanding the ethical impact of using technology (privacy, inclusion, professionalism) on society** | Explain that websites like Googleuse cookies to track which sites we visitso that they can target advertisements at us.Outline what a cookie is, what it is used for, and how the EU Cookie Law (e-Privacy Directive) regulates cookie use.  Refresh James’ Ball’s article.  In his article James mentions Europe-wide data protection rules, which are designed to safeguard privacy. Explain how the UK Data Protection Act (DPA) protects citizens against misuse of their personal data.  Ask learners to work in groups to investigate security threats to data and what preventative measures organisations can take. **(Activity 11.2.1)**  Explain that the Computer Misuse Act makes hacking into other people’s computer systems illegal. Ask students to use the wiseGEEK website to find out more about the Act.  **Homework**: Ask students to summarise the legal impact of using technology on society, in an essay of about 300 words in length. | Week 11  Lesson 2 Activities |

| **Week 12** | |
| --- | --- |
| **Lesson** | **Spec ref** | **Lesson summary** | **Lesson content** | **Lesson resources** |
| 1 | 3.2.1 | **Data Representation**: Text | Ask students to define what a character set is. Students could discuss in pairs and then fours to write a definition before sharing with the class.  Recap that each letter/number/symbol is stored as a unique code in a computer and why standards such as ASCII are used.  You could show students examples of ASCII art and explain why it was used (lack of graphics displays on old computer systems). Examples can be found at <http://www.chris.com/ascii/> and students could generate ASCII writing art of their name on <http://patorjk.com/software/taag/> ASCII art can be displayed by python programs if a heading/title is needed.  Explain that python allows you to convert to/from letters to ASCII values using the functions ord and chr. So ord(‘a’) will display 97 and chr(97) will display ‘a’. These examples will work in the python shell if you wish to demonstrate them.  Students should write a program that will display the ASCII values for a letter that is input. This could then be extended to display the codes for each letter in an input word.  Students could also write a program that reads in a letter or word and then converts it from lowercase to uppercase by changing to the ASCII code and subtracting the difference between uppercase and lowercase. This could be demonstrated by typing ord(‘A’) in the shell and ord(‘a’) in the shell. If the students subtract 32 from the ASCII value of each letter then display using chr then the uppercase equivalent letter will be displayed.  Students could be asked to implement a Caesar Cipher encryption/decryption program. This might be a good task to do in pairs and get students to write the pseudocode before implementing. (<https://en.wikipedia.org/wiki/Caesar_cipher>)  **Homework:** Students should research why other standards such as EBCDIC and Unicode exist. What is the advantage of Unicode over ASCII? |  |
| 2 | 2.4.3 | Reading and writing to files | Explain that much of input/output is about being able to read information from a file and being able to write information to a file.  Describe what a file is and how files are referenced using a filename.  Demonstrate how Notepad can be used to write text files.  Explain that files need to be placed in the same folder as the Python program files that use them, or the correct pathname needs to be given. (An explanation of pathnames may be needed.)  Explain the need for caution when reading and writing to files as damage can be done to files.  Explain the concept of sequential records stored in a file and how you need to open a file for reading (r) or writing (w).  Demonstrate how a program can read records from a file and write records to a file. Demonstrate how to append records to the end of any existing file.  Give learners the opportunity to write programs to read, write and append to files. **(Activity 12.2.1)**  *There are no solutions for this lesson. The Activity has short programming sections which students will develop during the lesson.*  **Homework:** Ask learners to create a one-minute video entitled ‘Using files in Python in plain English’. | Week 12  Lesson 2 Activities |

# 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).