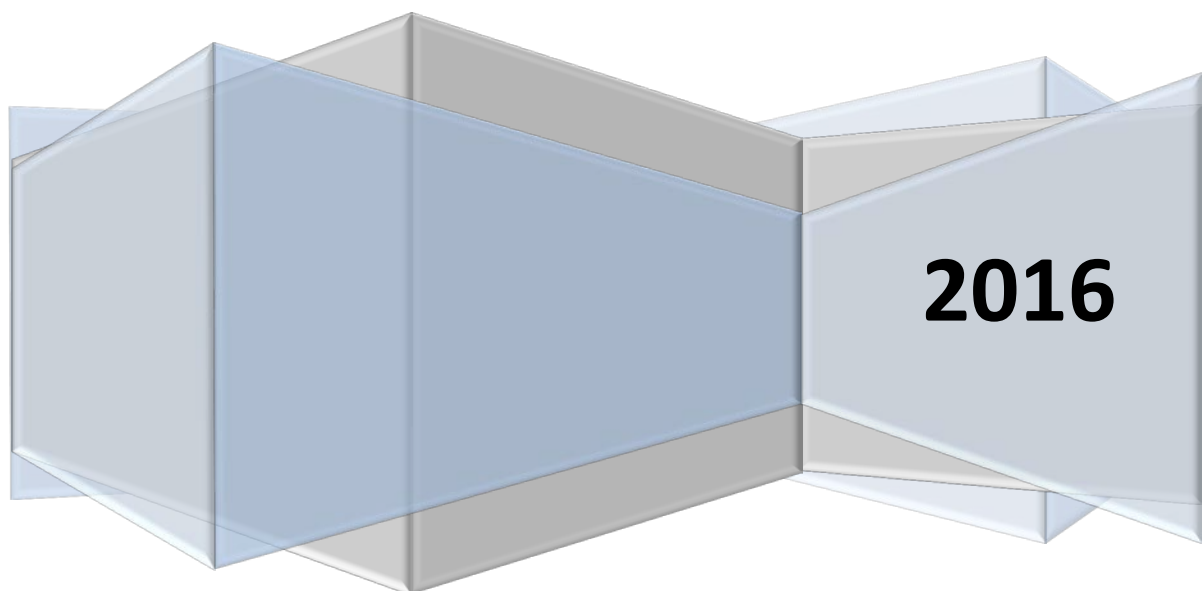


# GCSE Computer Science



Geraint D. Jones

Jen M. Gillies

Mark D. Thomas

Martin C. Gillies

# Disclaimer

This resource is provided to support the teaching and learning of GCSE Computer Science. The materials provide an introduction to the main concepts of the theory of the subject and should be used in conjunction with other resources and sound classroom teaching.

It is intended that the resource will be updated periodically due to the evolving nature of the subject. Suggestions as to how the materials may be developed further are welcome through our feedback system or by sending an e-mail to: [resources@wjec.co.uk](mailto:resources@wjec.co.uk)

# Contents

## Component 1

1. Hardware
2. Logical operations
3. Communication
4. Organisation and structure of data
5. Operating systems
6. Principles of programming
7. Software engineering
8. Program construction
9. Security and data management
10. Ethical, legal and environmental impacts of digital technology

## Component 2

1. Problem solving
2. Algorithms and programming constructs
3. Programming languages
4. Data structures and data types
5. Security and authentication

# 1. Hardware

## CPU Computer processor unit

To a computer, the world consists of zeros and ones. Inside a processor, we can store zeros and ones using transistors. These are microscopic switches that control the flow of electricity depending on whether the switch is on or off. So the transistor contains binary information: a one if a current passes through and a zero if a current does not pass through.

Transistors are located on a very thin slice of silicon. A single silicon chip can contain thousands of transistors. A single CPU contains a large number of chips. Combined, these only cover a couple of square centimetres, but can hold several million transistors and process hundreds of millions of instructions per second.

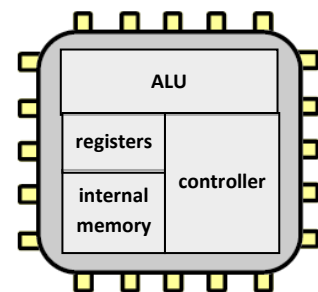
## CPU (Central Processing Unit) Architecture

CPU architecture is a term that refers to the design of a Microprocessor, an integrated circuit, where the components of the CPU are combined as a single unit.

## Components of the CPU

The **CPU** is the main component in a computer for processing data and instructions. It could be considered as the computing equivalent of the human brain. It is a hardware device that is made up of many sub components:

- controller
- ALU: arithmetic and logic unit
- registers
- internal memory



These components have specific functions.

## Controller or control unit

The controller sends and receives signals from all parts of the computer. It ensures that all processes take place at the right time and in the correct order.

## Arithmetic and logic unit (ALU)

The ALU is the part of the CPU that processes and manipulates data. It performs simple calculations on the data that is temporarily stored in the registers.

The ALU is also able to perform comparisons on data. It is these comparisons that allow programs to make use of *choice* – e.g. in a high-level language an IF statement.

## Registers

A register is a storage location found on the CPU where data or control information is temporarily stored. Registers are usually much faster to access than internal memory, since they have to be accessed so often.

An accumulator is a common example of a register. This is the register used by the ALU to store the results of its calculations.

### SUMMARY DEFINITIONS

**Controller:** manages the execution of instructions

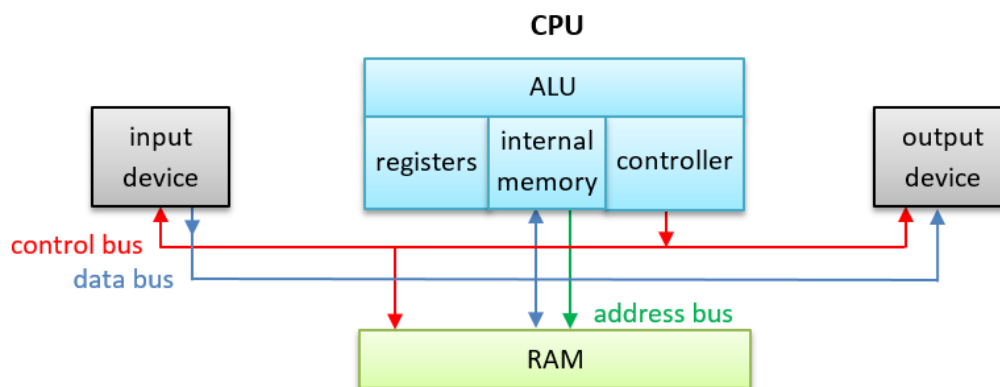
**ALU:** processes and manipulates data

**Register:** fast access temporary data store

**Internal memory:** fast access memory on the CPU

## Internal memory

Internal memory (*sometimes called level 1 cache memory*) is fast access temporary storage on the CPU. Data is moved from the registers to the internal memory when it is not being actively used. Data from internal memory can then either be written to RAM or called back into the registers for further processing.



## Buses

Buses allow data to be transferred to different parts of the computer. There are three main buses used by the CPU.

1. Address bus. When data is saved or loaded from memory, the address at which it is to be stored or loaded from must be sent. The storage address of data always travels along an address bus.
2. Data bus. Data will then need to be moved between several parts of a computer. The path along which data travels is called a data bus.
3. Control bus. The controller uses the control bus to send control signals to different parts of the computer.

## Von Neumann

Early computers were generally designed to carry out a specific task or calculation. Re-programming these custom built computers was very difficult and could even involve some re-wiring. In 1945 John von Neumann proposed storing the program instructions in the same memory as the data. This idea of a **stored program**, which is often referred to as 'Von Neumann' architecture, resulted in computers that could be more easily re-programmed and is the basis for the fetch-decode-execute cycle, fundamental to modern computer processing.

### The fetch-decode-execute cycle

There are **three** steps to processing instructions given by a currently running program:

1. The fetch cycle takes the address required from memory, stores it in the instruction register, and moves the program counter on one so that it points at the next instruction.
2. The control unit checks the instruction in the instruction register. The instruction is decoded to determine the action that needs to be carried out.
3. The actual actions that happen during the execution cycle depend on the instruction itself.

## CPU Performance and specification

When we use a computer, we want the instructions to be carried out very fast. As the instructions become more complicated (for example, creating a 3D animation or editing a video file), we demand more from the CPU. The technological advances we have seen in processor technology have largely been driven by the need for speed.

### CPU Cache memory

Cache memory is a fast access type of memory that is very expensive. Due to its cost, only small amounts of cache memory are present in most computer systems. Cache memory improves the performance of the CPU as it is able to provide instructions and data to the CPU at a much faster rate than other system memory such as RAM. The more cache memory your system has, the better its performance is likely to be.

### Clock speed

The speed at which a processor operates is called the clock speed. The faster the clock speed, the faster the computer is able to run the fetch-decode-execute cycle and therefore process more instructions.

The speed of the processor is measured in Hertz (Hz). One clock tick per second would be measured as 1 Hz. Therefore a processor that operates at 1,000 clock ticks per second would be a 1,000 Hz processor, also known as a 1 kHz processor.

At this stage, it is a good time to introduce *prefix multipliers* for clock speeds:

Prefix	Symbol	Multiplier	Power of 10
Kilo	k	1,000	$10^3$
Mega	M	1,000,000	$10^6$
Giga	G	1,000,000,000	$10^9$

A typical modern day home computer would have a 2.5 GHz processor. This means the clock speed of the processor runs at 2,500,000,000 Hz or clock ticks per second.

The clock speed inside the CPU can sometimes be changed. A processor can be set to run faster than its original design. By doing this however, it uses more energy and produces more heat. If this heat is not removed through cooling, the CPU can overheat, which will damage the CPU and shorten its lifespan. This is called *overclocking*.

#### INTERESTING FACT

Most CPUs are cooled using a fan mounted on a metal heat sync. Liquid cooling systems have been found to be more effective at cooling CPUs, although the water used in these systems conducts electricity and can therefore be dangerous. A computer designer, Seymour Cray, designed a computer cooling system that used artificial human blood to cool the CPU, as it does not conduct electricity.

Some computer systems, especially mobile devices, set the clock speed of the CPU lower than its original design. This results in less power consumption, less heat being produced and will therefore increase the battery life of the device. This is called *underclocking*.

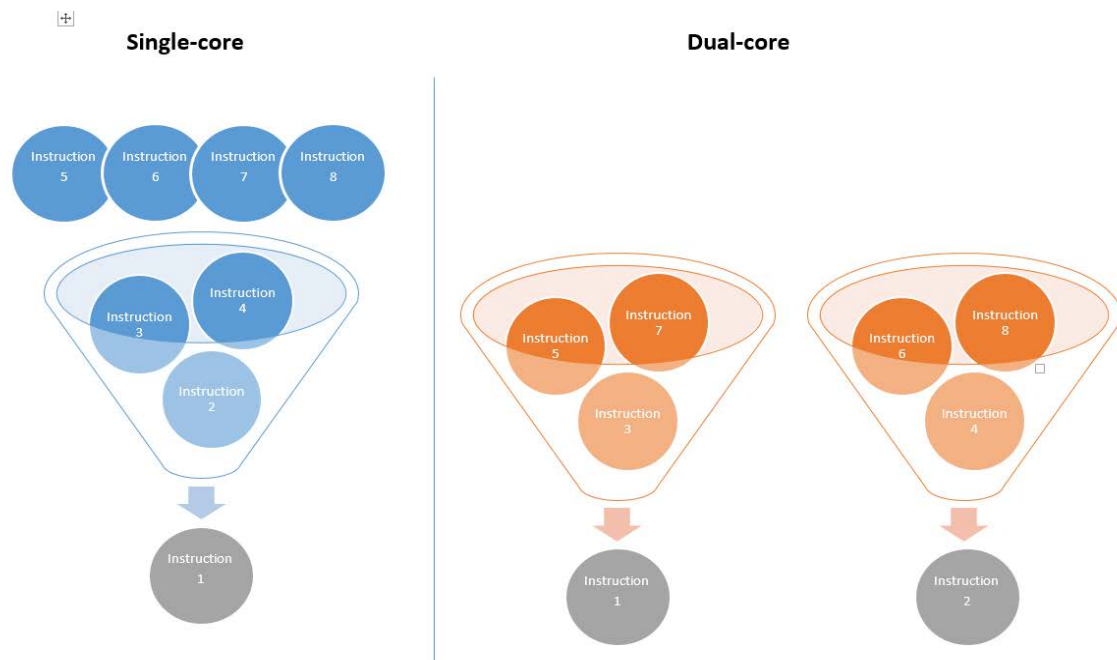
Some devices are able to change their own clock speed dynamically. For example, when your computer is idle, the clock speed may be set at a lower rate than if you were running a CPU intensive program, such as a computer game.

### Number of cores

We have assumed that each CPU has only one core. However, this isn't always the case, as some CPU's have multiple cores. You may be familiar with the terms 'dual-core' and 'quad-core', so what exactly do these terms mean?

A core is the term used to describe the processing components within the CPU. Multi-core processors therefore have many processing components within the same CPU. Below is a

diagram that illustrates a number of instructions waiting to be processed in single-core and dual-core CPUs.



In a single-core CPU each instruction is processed one after the other, whereas in a dual-core CPU, two instructions may be processed at the same time. In theory, a dual-core CPU should be able to process instructions twice as fast as a single-core CPU. However, this isn't always the case, as sometimes *Instruction 2* may need the result of *Instruction 1* before it can be processed.

In general, a computer running many programs at the same time will run faster on a multi-core processor than on a single-core processor.

#### INTERESTING FACT

Many high-end gaming consoles include CPUs with multiple cores. The Sony *Playstation 3* has an 8 core CPU.

### RISC and CISC processors



There are two main types of processor, namely **Reduced Instruction Set Computer (RISC)** and **Complex Instruction Set Computer (CISC)**.

RISC processors can process a limited number of relatively simple instructions. To carry out more complex commands the problem is broken down into a longer list of simpler instructions. The advantage of this is that a RISC processor is able to process these simpler instructions quickly. Processing simpler instructions also requires less circuitry to decode and execute these instructions, which in turn means less power consumption and therefore less heat being generated.



CISC processors can process a large number of complex instructions. This allows the processor to understand and carry out complex tasks with only a few instructions. The advantage of this is that a CISC processor is able to process complex instructions, without having to break them down into many simpler instructions. Processing complex instructions however requires more circuitry to decode and execute these instructions, which in turn means more power consumption and therefore more heat being generated.

## Input and Output

<b>Input devices</b> An input device allows data, such as text, images, video or sound, to be entered into a computer system.	<b>Output devices</b> There are many outputs created by a computer system. These include printed documents, on-screen data and sound.
 <p style="text-align: center;"><i>Common hardware input devices</i></p>	 <p style="text-align: center;"><i>Common hardware output devices</i></p>

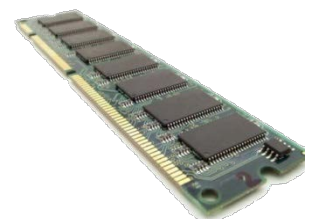
### INTERESTING FACT

Doug Engelbart made the first computer mouse in 1964.  
It was made out of wood!!

## Primary storage

### Random Access Memory (RAM)

**RAM** is used for the temporary storage of currently running programs and data. It consists of a large number of store locations each of which is identified by a unique address. The data in each store location can be changed. RAM is volatile – data is lost when the power is switched off.



Example: When you are working on a word-processed document, the program you are using and the data within the document are both stored in RAM.

## Read-only Memory (ROM)

**ROM** is used for the permanent storage of data. The data in each store location cannot be changed. ROM is permanent – data is not lost when the power is switched off.

Example: ROM can be used for storing the programs such as the BIOS. The disadvantage of using ROM to store the BIOS is that it cannot be upgraded.

### KEY INFORMATION

**Basic Input Output System (BIOS):** A low-level program that handles input and output operations relating to the keyboard and screen of the system. It provides an interface between the hardware and the operating system. One of its primary functions is loading and executing the *bootstrap loader* – the program that loads the operating system.

## Flash memory

**Flash memory** is used for the permanent storage of data. However, the data stored in flash memory can be changed. Flash memory is permanent – data is not lost when the power is switched off.

Example: Flash memory can be used for storing the programs such as the BIOS, which is advantageous as the BIOS can be upgraded.

## RAM Cache memory

**RAM Cache memory** is used for the temporary storage of frequently accessed data and instructions. It consists of a small number of store locations that can be accessed very quickly by the CPU; it is quicker than RAM. Cache memory is volatile – this means that data is lost when the power is switched off.

## Summary of different types of memory

	Permanent	Volatile	Data can be changed	Speed
Cache memory		✓	✓	★★★★★
ROM	✓			★★★★
RAM		✓	✓	★★★
Flash memory	✓		✓	★

## Secondary storage

**Secondary storage** is also known as backing storage.

Data is written from memory to secondary storage when data is no longer being actively used, for retrieval at a later time.

### INTERESTING FACT





The first commercial hard disk drives had the capacity to store approximately 5 MB and were the size of a dining room table. They were also called a *Winchester Drive*.

The time a computer takes to access data stored on secondary storage is **longer** than the time taken accessing data from memory.

## Data Capacity

**Data capacity** is the amount of data a storage device can hold measured in Kilobytes (Kb), Megabytes (MB), Gigabytes (GB) and Terabytes (TB).

The most frequently used backing storage media are:

Media	Suitability	Typical capacity	Durability	Portability	Speed
 Flash drive	Moving relatively small files from work to home	2 GB – 64 GB	★★★★	✓	★★★★
 External hard drive	Backing up a home computer system	500 MB – 4 TB	★	✓	★★★
 CD/DVD/Blu-ray disk	Storing multimedia files	650 MB (CD) 9 GB (DVD) 50 GB (Blu-ray)	★★★	✓	★★
 Magnetic tape	Backing large commercial servers on multiple tapes	200 GB – 400 GB	★★	✓	★

Different types of data can create files that vary in size. In general, text based files are relatively small but audio and video files are much larger. Here are some typical file sizes.

File size	Typical contents
1 B	A single key stroke or a number from 0 to 255
70 B	One line of text
1 KB	A third of a page of text or a short email
8 KB	A school logo
30 KB	A basic web page
100KB	Maximum size for all elements of a web page
500 KB	A five page word processed document or a PDF for downloading
1 MB	One minute of audio when stored as an MP3
5 MB	DVD Movie
10 MB	HD Movie
25 MB	Blu-ray Movie
700 MB	Maximum amount of data on one CD-ROM

It is important to consider the type of data that is going to be stored when you are choosing a storage device.

## Technologies such as: optical, magnetic, solid state, storage in the cloud

### Optical



Optical storage media uses technology such as lasers. Laser beams are projected onto a CD/DVD or Blu-ray disk and if light is reflected back, then data is read as a 1 and if light is not reflected back, data is read as a 0. Lasers are used to read and write information to a disk.

#### KEY INFORMATION

**Binary digiT (BIT):** In computer systems, data is represented by either a 1 or a 0.

### Magnetic

This technology is used in hard disks and tapes. Data is stored on a magnetic medium, which can be a disk or a tape, by writing data using a write-head. Data can then be read by the read-head.

### Solid state

Solid state technology is used in storage media such as USB flash memory sticks. The technology is called solid state as it does not have any moving parts, such as a read-head in magnetic storage. Solid state storage technology

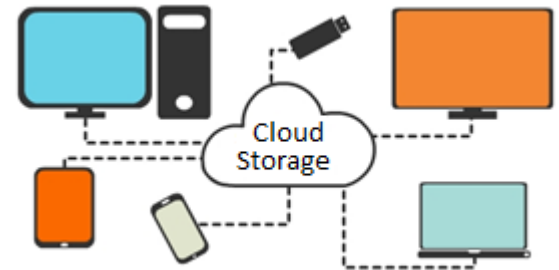


is increasingly used to replace both magnetic and optical storage, especially in mobile devices, where its low power consumption and high speed access is advantageous.

## Storage in the cloud

Storage in the cloud is a contemporary data storage facility that allows users to store their data on third-party servers. They can then access that data from many computing devices.

There are many **advantages** to this, such as maintenance tasks, backup and data replication. Purchasing additional storage devices becomes the responsibility of the cloud storage service provider.



A **disadvantage** of storage in the cloud is that an Internet connection is required to access the data. Some other disadvantages include the concern for some organisations that personal data held on a third-party server could be physically stored in a country where adequate data protection legislation does not exist. Another disadvantage is that users are solely reliant on the cloud storage provider when it comes to ensuring that their data is stored safely and can be retrieved at a later date.

## Storage requirements

Computer systems can only store and process Binary digits, also known as BITS. A BIT is either a 1 or 0. When 8-bits are stored as a binary number, they are collectively called a byte.

	Symbol	Value
Byte	B	8 bits
Kilobyte	Kb	1024 bytes
Megabyte	MB	1024 Kb
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

**INTERESTING FACT**  
Half a byte (4 bits) is called a nybble.

## **Additional hardware components**

### **Motherboard**

The motherboard is the main circuit board of a computer. The CPU and ROM will be mounted on the motherboard, which also provides RAM expansion slots, USB ports, PCI slots for expansion cards and controllers for devices such as the hard drive, DVD drive, keyboard, and mouse.

### **Graphics Processor Unit GPU**

A GPU is a microprocessor that performs the calculations needed to produce graphic images on screen. Initially the CPU performed these calculations, but as more complex applications were developed, such as 3D graphics and video quality animations, the GPU was introduced to offload those tasks from the CPU.

GPUs can be integrated within the circuitry of the motherboard, or provided on a dedicated graphics card.

### **Integrated GPU**

An integrated GPU uses the computer's RAM. An integrated unit is cheaper than installing a dedicated GPU, it generates less heat and uses less power. They are perfect for general graphics processing such as watching or editing videos and word processing.

### **Dedicated GPU**

A dedicated GPU has its own video memory. Dedicated cards provide the best visual experience and are used by people such as professional graphic designers and serious gamers, but they use more power and require a good cooling system.

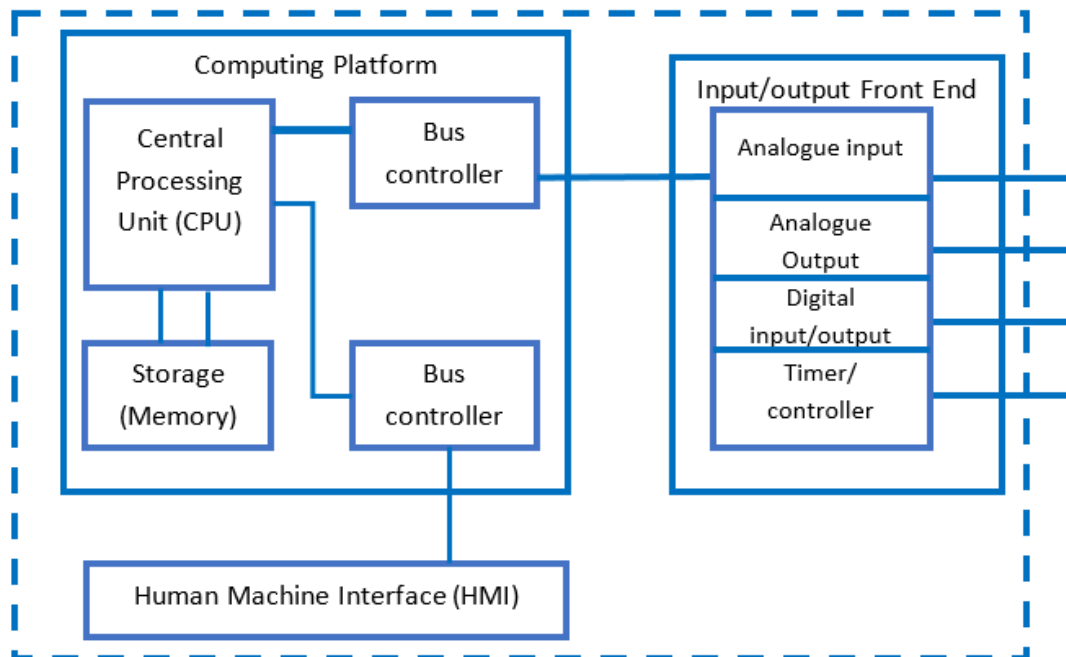
### **Sound cards**

Sound cards may be on board (on the motherboard) or designed to fit a PCI slot. They enable the computer to output sound through speakers, to record sound from a microphone and to manipulate sound stored on a disk. Sound cards convert analogue input signals into digital data and reverse this process for audio output.

### **Embedded Systems**

An embedded system is a combination of software and hardware that performs a specific task rather than a general-purpose computer that is designed to carry out multiple tasks.

Embedded systems are included as a part of a complete device often with hardware and mechanical parts. As the systems carry out specific tasks they can be designed to be small and have a low cost. Mass-production of embedded systems can save large amounts of money.



The software written for an embedded system is known as firmware. The instructions are stored in read-only memory or in Flash memory. The software runs with limited computer hardware resources, little memory and no peripherals.

Most embedded systems are reactive - they react to conditions such as temperature, weight, vibration and air quality. These systems detect external conditions and react to them by recording data, turning motors on or off, sounding an alarm or sending a message to another processor.

Reactive embedded systems often control real time events so must be completely reliable. For example, drivers rely of the anti-lock braking system of their car working correctly to avoid accidents on the road.

When an embedded system performs operations at high speed, and if it is very reliable, it can be used for real -time applications. If the size of the embedded system is very small and power consumption very low, then the system can be easily adapted for different situations.

**INTERESTING FACT**  
98% of the microprocessors manufactured go into embedded systems.

Some examples of devices that incorporate embedded systems:

Electronics	Mobile phones, games consoles, printers, televisions, digital cameras
In the home	Washing machines, microwave ovens, refrigerators, dishwashers, air conditioners
Medical equipment	CT Scanners, Electrocardiogram (ECG), MRI Scanners, blood pressure monitors, heartbeat monitors
Cars	Electronic fuel injection systems, anti-lock braking systems, air-conditioner controls.





## 2. Logical Operations

### Using logical operators: NOT, AND, OR, and XOR

Logical operators are used in programming. It is not unusual to find a code such as:

- IF A = 1 **AND** B = 1 THEN
- IF A = 1 **OR** B = 1 THEN

The rules for combining logical expressions are usually written down as tables that list all the possible outcomes. These tables are known as truth tables.

#### NOT

The NOT logical operator has only one input and one output. The output is the opposite of the input.

Input (A)	Output (NOT A)
0	1
1	0

#### AND

The AND logical operator has two inputs and one output. The output is 1 only if A and B are both 1.

Input (A)	Input (B)	Output (A AND B)
0	0	0
0	1	0
1	0	0
1	1	1

#### OR

The OR logical operator has two inputs and one output. The output is 1 if either A or B is 1.

Input (A)	Input (B)	Output (A OR B)
0	0	0
0	1	1
1	0	1
1	1	1

#### XOR

The XOR logical operator has two inputs and one output. The output is 1 only if A and B are different.

Input (A)	Input (B)	Output (A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

## Logical operations

Logical operations can be used in control systems. For example, a control system that is required to close the windows on a commercial greenhouse when at least one of the following conditions is true:

- the wind speed rises above 12 km per hour.
- it is raining.

would use the logical operator **OR**.

A control system that is required to turn on a sprinkler system in a field when both of the following conditions are true:

- the temperature rises above 25° Celsius
- it has not rained in the last five days

would use the logical operator **AND**.

## Boolean algebra

In addition to truth tables, Boolean algebra can be used to describe logical expressions. Logical expressions can become complex and Boolean algebra is a method that can be used to simplify these expressions.

### Notation

The logical operators that are going to be used are NOT, AND, OR and XOR. Boolean algebra uses the following notation to represent the input variables and operators:

- a bar on top of and input variable represents the NOT operator. NOT A =  $\bar{A}$
- a dot “.” represents the AND function. A and B =  $A.B$
- a plus sign “+” represents the OR function. A or B =  $A + B$
- a plus sign with a circle represents the XOR function. A exclusive or B =  $A \oplus B$

### Laws of Boolean Algebra

Boolean Algebra is a system of mathematics based on logic that has its own set of rules or laws that can be used to simplify Boolean expressions.

#### Annulment Law

- $A.0 = 0$       A variable AND 0 is always equal to 0  
 $A + 1 = 1$       A variable OR 1 is always equal to 1

### Identity Law

$A + 0 = A$       A variable OR 0 is always equal to the variable  
 $A.1 = A$       A variable AND 1 is always equal to the variable

### Idempotent Law

$A + A = A$       A variable OR itself is always equal to the variable  
 $A.A = A$       A variable AND itself is always equal to the variable

### Complement Law

$A.\bar{A} = 0$       A variable AND its complement is always equal to 0  
 $A + \bar{A} = 1$       A variable OR its complement is always equal to 1

### Commutative Law

$A.B = B.A$       The order of two variables with AND makes no difference  
 $A + B = B + A$       The order of two variables with OR makes no difference

### Double Complement Law

$\overline{\bar{A}} = A$       A double complement of a variable is always equal to the variable

### Distributive Law

$A(B + C) = A.B + A.C$       (OR Distributive law)  
 $A + (B.C) = (A + B).(A + C)$       (AND Distributive law)

### Absorptive law

$A + (A.B) = A$       (OR Absorption law)  
 $A(A + B) = A$       (AND Absorption law)

### Associative Law

$A + (B + C) = (A + B) + C = A + B + C$       (OR Associative law)  
 $A(B.C) = (A.B)C = A.B.C$       (AND Associative law)

### **Examples - using the rules.**

1. Simplify the Boolean expression:

$$\begin{aligned}
 X &= A.B + A.\bar{B} \\
 X &= A.(B + \bar{B}) && \text{Distributive law} \\
 X &= A.1 && \text{Inverse law} \\
 X &= A && \text{Identity law}
 \end{aligned}$$

2. Prove that  $A + \bar{A}.B = A + B$

$A + \bar{A}.B = A.1 + \bar{A}.B$	Identity law
$= A.(1 + B) + \bar{A}.B$	Annulment law
$= A + A.B + \bar{A}.B$	Distributive law
$= A + B.(A + \bar{A})$	Inverse law
$= A + B$	

# 3. Communication

## Networks

A network consists of a number of computer systems connected together. There are many advantages and disadvantages of using a computer network over a stand-alone computer.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Share hardware</li> <li>• Share software</li> <li>• Share data/files</li> <li>• Easier for internal communication/can send email</li> <li>• Central backup</li> <li>• Easier to monitor network activity</li> <li>• Centrally controlled security</li> <li>• Can access data from any computer</li> </ul>	<ul style="list-style-type: none"> <li>• A network manager may need to be employed – expensive</li> <li>• Security problems – files sent between computers could spread a virus</li> <li>• Hackers can gain access to data more easily</li> <li>• If the server is down, all workstations on the network are affected</li> <li>• Initial cost of servers, communication devices, etc. can be expensive</li> </ul>

There are two main types of network, namely a **Local Area Network (LAN)** and a **Wide Area Network (WAN)**.

A LAN is a network in which the computer systems are all located relatively close to each other, for example, in the same building or on the same site, such as a school.

A WAN is a network, in which the computers systems are all located relatively distant from each other, for example, in different buildings all over the country or in different countries. The Internet is an example of a WAN. You will note that many LANs could be linked using a WAN.

Computer networks use agreed upon protocols to communicate, i.e. common methods of sending data and consistent data formats. If they did not agree on the protocols to be used, the individual computer systems would not be able to communicate with each other.

### Network topologies

A network topology is the theoretical layout of computer systems on a network. There are a number of different network topologies. Common network topologies include:

- bus network
- ring network
- star network

## Bus network

The computer systems, also called *nodes* of the network, are each connected to a single cable on which data can be sent, called the bus. A bus network has terminators on each end, which is needed to ensure that the network functions correctly.



The bus carries packets along the cable. As the packets arrive at each computer system, it checks the destination address contained in the packet to see if it matches its own. If the address does not match, the computer system ignores the packet. If the address of the computer system matches that contained in the packet, it processes the data.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Easy to implement and add more computer systems to the network</li> <li>• Quick to set up – well suited for temporary networks</li> <li>• Cost-effective – less cabling</li> </ul>	<ul style="list-style-type: none"> <li>• It is difficult to troubleshoot the bus</li> <li>• Limited cable length and number of stations – performance degrades as additional computers are added</li> <li>• If there is a problem with the main cable or connection, the entire network goes down</li> <li>• Low security – all computers on the bus can see all data transmissions</li> <li>• Proper termination is required</li> <li>• Data collisions are more likely, which causes the network to slow down. A collision is when two computers try to send a packet at the same time</li> </ul>

## Ring network



In a ring network, computer systems are connected in a ring or a loop. Packets are sent around the ring, being passed from one computer system to the next until they arrive at their destination.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Data is quickly transferred without a bottleneck – consistent data transfer speeds</li> <li>• The transmission of data is relatively simple as packets travel in one direction only</li> <li>• Adding additional nodes has very little impact on bandwidth</li> <li>• It prevents network collisions.</li> </ul>	<ul style="list-style-type: none"> <li>• If any of the computer systems fail, the ring is broken and data cannot be transmitted efficiently</li> <li>• If there is a problem with the main cable or connection, the entire network goes down</li> <li>• It is difficult to troubleshoot the ring</li> <li>• Because all nodes are wired together, to add another you must temporarily shut down the network</li> </ul>

### Star network



In a star network, each computer system is connected to a central node, also known as a hub.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Good performance/fast network speed</li> <li>• Easy to set up</li> <li>• Possible to add more computer systems without taking the network down</li> <li>• Any non-centralised failure will have very little effect on the network</li> <li>• Minimal network collisions</li> <li>• Better security</li> </ul>	<ul style="list-style-type: none"> <li>• Expensive to install – more cabling required</li> <li>• Extra hardware required, such as a hub</li> </ul>

### Connectivity

To connect a computer system to a network, a Network Interface Card (NIC) is required. A physical hardware port allowing a cable to connect your computer system to the network provides one method of connection. The second method is to connect a computer system using a wireless connection, called a Wi-Fi.

## Typical network speeds

A physical connection may be made using:

- a copper cable, with typical data transfer speeds of between 100 Megabits per second (Mbps) and 1 Gigabit per second (Gbps)
- a fibre-optic connection which has typical data transfer speed of between 1-10 Gbps

Wi-Fi connections have typical data transfer rates of 54 – 108 Mbps. However, this can be severely affected by the distance between the device providing the Wi-Fi connection and computer systems. The data transfer rates can also be severely affected by atmospheric conditions, in particular heavy rain.

## Circuit switching

**Circuit switching** is a networking technology that provides a temporary but dedicated link between two stations or nodes regardless of the number of switching devices through which the data has to travel. During the connection no other data can be transmitted along the same route. The landline telephone system is an example of a circuit switched network. When you phone someone and they answer a circuit connection is made and you can pass data along the connection until you put down the telephone to end the connection.

The main advantage of circuit switching is that it is reliable and once the connection is established it is fast and generally error free. However, it takes time to establish the connection. Should anywhere on the route fail then the connection will be broken.

To overcome the problems with circuit switching, packet switching was developed. Rather than relying on a dedicated connection packet switching breaks the data down into small packets that can be sent by more than one route.

## Packet

A **packet** is a collection of data that is transmitted over a **packet-switched network**. Packets are provided to a network for delivery to a specified destination. Each packet of data is redirected by a computer system along the network, until it arrives at its destination. Data may be split up into a number of packets. These packets are transmitted over a network and may take different routes to its destination. When all the packets have arrived the data is reassembled. The Internet is an example of a packet-switching network.



Here is a simplified diagram showing what a packet will typically contain:

The source address	The destination address
Information which enables the data to be reassembled into its original form	
Other tracking information	
The data itself	A checksum that checks that the data has not been corrupted

#### INTERESTING FACT

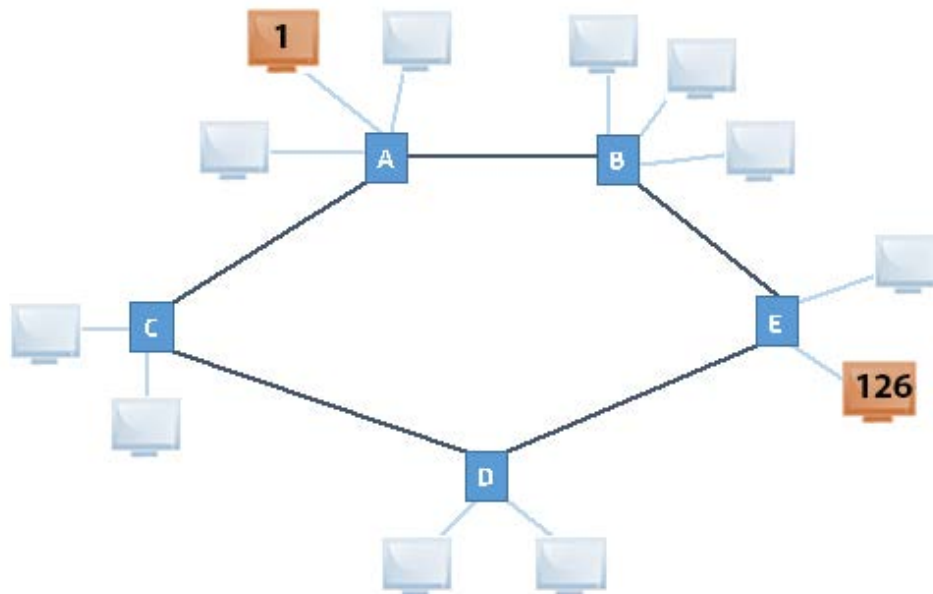
When packets are transmitted over a Wi-Fi network, they can be intercepted by any device, this is called *packet sniffing*. If you perform thorough analysis on a large number of packets, you can often break the encryption. This is why security services, such as National Security Agency, do not allow any Wi-Fi devices on their network.

## Packet switching (including data redundancy)

Packet switching is the process of delivering packets from one computer system to another using a designated device, such as a *switch* or a *router*. Packets are provided to a network for delivery to a specified destination. Each packet of data is redirected by a computer system along the network, until it arrives at its destination. Data may be split up into a number of packets. These packets are transmitted over a network and may take different routes to its destination. When all the packets have arrived the data is reassembled. The Internet is an example of a packet-switching network.

## Routing

Routing is the name given to the method of selecting paths along which packets are sent on a computer network. Specialist computer systems such as routers, switches, bridges, firewalls and gateways construct in their memory a *routing table*, which stores a number of paths along which it is best to send packets to reach a specific destination. Maintaining accurate routing tables is essential for ensuring that packets are delivered as quickly as possible.



In the example shown, computer system 1 is sending a packet to computer system 126. Clearly, the quickest route for the packet to arrive at its destination is to be sent from router A, on to router B followed by router E for delivery to computer system 126. This path would be determined by routing, using a routing table. A poorly constructed routing table may choose to send the packet from router A, on to router C followed by router D and then router E, for delivery to computer system 126. This would take longer and is not a good use of network resources.

Most routers use only one network path at a time, such as the preferred route above (Computer system 1 > Router A > Router B > Router E > Computer system 126). Some multipath routing techniques enable the same packets to be sent using multiple alternative paths at the same time. This means that in the event of Router B failing in the transmission above, the same packet would also have been sent via the alternative longer route set out above (Computer system 1 > Router A > Router C > Router D > Router E > Computer system 126), to ensure that the packet arrives at its destination.

## MAC addresses

A MAC address (media access control address), also known as a physical address or a hardware address, is a unique hexadecimal number given to any communication device, such as a network interface card. An example of a MAC address is 74:E1:B6:8E:18:77. The address is usually stored in a communication devices' ROM. Hexadecimal notation is used as it allows for over 281 trillion different combinations of MAC address.

### INTERESTING FACT

Although MAC addresses are designed to be unique and unchangeable, some devices or specialised software allow you to change your own MAC address. This is called *MAC address spoofing* and can be used by hackers to trick computer systems into providing data.

Routing tables store the MAC address of communication devices on its network, as the address is permanent and does not change like an IP address. A computer system can have multiple network interface cards, each with its own unique MAC address.

## **IP addresses**

An IP address is an address, which is allocated to a computer system on a network, usually by a DHCP (Dynamic Host Configuration Protocol) server. Alternatively, you may assign your own IP address if you do not wish to rely on the services of a DHCP server. An example of an IP address is 195.10.213.120.

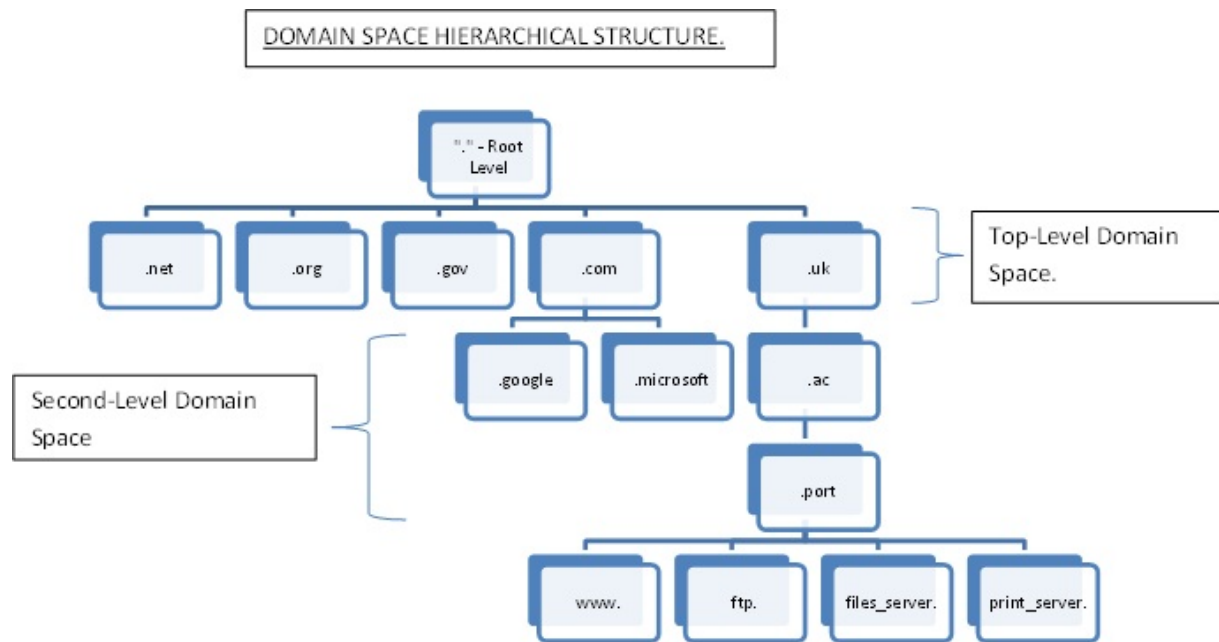
It is used to uniquely identify computer systems on a network, thus allowing communication between them. In routing tables, the corresponding IP address of a unique MAC address is stored and updated as necessary.

## **Internet Domain Name System (DNS)**

A Domain Name System (DNS) is a distributed database that matches IP addresses to computer system resources.

One example of this is to match an IP address to a human friendly domain name. For example, if you wanted to visit the Google search engine, the computer system on which the website is stored has an IP address assigned to it; 173.194.34.191. Try typing this into the address bar of your web browser; you should be able to view the website that you would be more familiar with when accessing the domain name [www.google.co.uk](http://www.google.co.uk). Here your computer system sent a request to its DNS server for the IP address that is mapped to the domain name [www.google.co.uk](http://www.google.co.uk). The DNS server returned the IP address 173.194.34.191, which allowed your computer system to communicate with the computer system where the Google search engine is stored.

Of course there are many different DNS servers located across the world. If your local DNS server does not store the address of the resource you are requesting, it will pass the request along to another higher level DNS server, such as your Internet Service Provider's (ISP) DNS server. If again the address is not found, your ISP's DNS server will pass the request on to a higher level DNS server which may be the DNS server responsible for an entire zone, such as the *.co.uk zone*. This continues until the address is found or the DNS query fails.



Another example where a DNS server is used is where a computer system, on joining a network, would query the DNS server for the IP address of other useful computer systems, such as the logon server, which stores the details of all usernames and passwords.

#### INTERESTING FACT

In 2015 users of YouTube uploaded 400 hours of video every minute. That means that 1000 days of video are uploaded every hour of the day.

## Protocols

A **protocol** is an agreed format, which allows two devices to communicate. The protocol, put simply, is a set of rules. These rules can include the following:

- handshaking, where two devices establish their readiness to communicate
- how the sending device will indicate that it has finished sending a message
- how the receiving device will indicate that it has received a message
- the type of error checking to be used
- agreement on the data compression method to be used

There are many standard protocols used with computer systems. The table illustrates the protocols with which you need to be familiar:

Protocol	Description
TCP/IP (Transmission Control Protocol/Internet Protocol)	Two protocols that combine to allow communication between computer systems on a network. IP is a protocol that sets out the format of packets and an addressing system. TCP is a protocol that allows packets to be sent and received between computer systems
HTTP (Hypertext Transfer Protocol)	HTTP is a protocol than can be used to transfer multimedia web pages over the Internet.
FTP (File Transfer Protocol)	FTP is a protocol that can be used when copying a file from one location to another via a network or the Internet. It is typically used for the transfer of large files, as it allows broken communications to resume transferring a file rather than having to restart.

A **protocol stack** is a set of protocols that work together to provide networking capabilities. It is called a stack because it is designed as a hierarchy of layers, each supporting the one above it and using those below it. The use of a layered approach enables different protocols to be substituted for each other to allow for e.g. new protocols and different network architectures. The number of layers varies according to the particular protocol stack. However, the lowest layer will deal with physical interaction of the hardware, with each higher layer adding additional features, and user applications interacting with the top layer.

#### INTERESTING FACT

Although the Bluetooth protocol has been agreed, the protocol stack varies considerably from device to device.  
Try sending a photograph via Bluetooth from one smartphone to another.

### TCP/IP 5 layer protocol stack model

TCP stands for *Transmission Control Protocol* and IP stands for *Internet Protocol*. There are five layers to this model:

- Physical layer
- Data link layer
- Network layer
- Transport layer
- Application layer

### **Physical layer**

The physical layer transmits the raw data. It consists of hardware such as switches and routers. The layer deals with all aspects of setting up and maintaining a link between the communicating computers.

### **Data Link Layer**

The data link layer sends data from the network layer to the physical layer. It divides the data to be sent into data frames. A data frame consists of a link layer header followed by a packet. The data link layer handles the acknowledgements sent from the receiver and ensures that incoming data has been received correctly by analysing bit patterns in the frames.

### **Network layer**

The network layer is responsible for the addressing and routing of data. Routers belong to the network layer as they use logical addresses to direct the data from the sender to the receiver. A router determines the path the data should take based on network conditions. Routers manage traffic problems on the network such as the routing of packets to minimise congestion of data.

### **Transport layer**

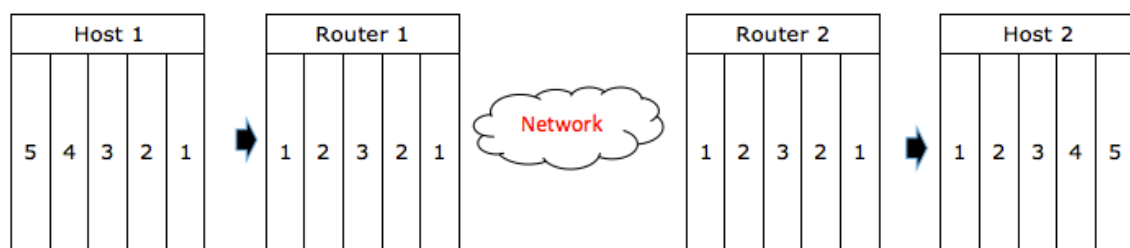
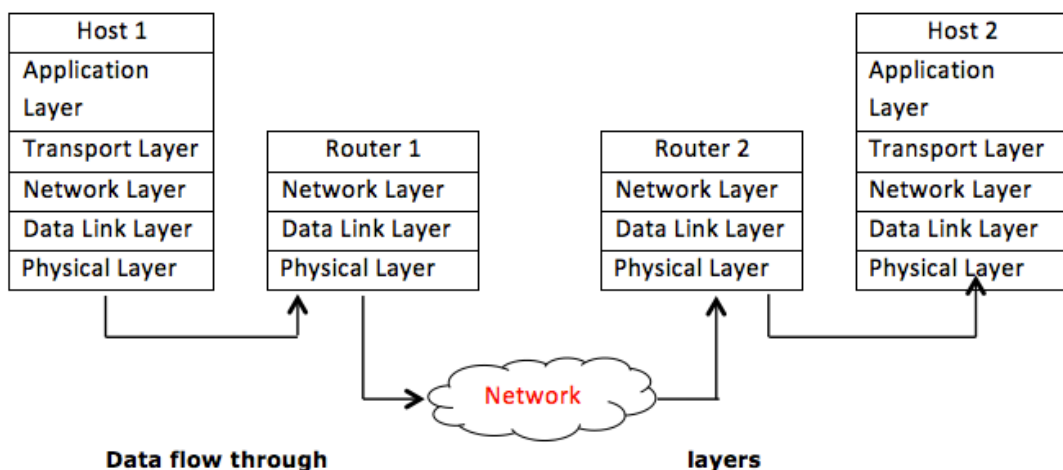
The transport layer ensures that data is transferred from one point to another reliably and without errors. The transport layer is responsible for making sure that data is sent and received in the correct order. The transport layer is implemented in the sending and receiving computers but not in the routers on the path between them. It acts as an interface between the communicating computers and the network.

### **Application layer**

The application layer provides interfaces to the software to allow it to use the network. Examples of software include email, file transfer protocol (FTP) and the World Wide Web (WWW).

### **Sending data from Host 1 to Host**

The diagram shows Host 1 sending a message to Host 2. From Host 1 the data flows down through the 5 layers of protocols and then to Router 1. Router 1 is the gateway to the operating area of Host 1 and therefore only the network, data link and physical layers are involved. Similarly, with Router 2 only the three layers are involved as the data is passed into the operating region of Host 2. Finally the data passes up through the layers to Host 2.



## Layers and protocols

Layer	Protocol
Application layer	Hypertext Transfer Protocol (HTTP) Simple Mail Transport Protocol (SMTP) File Transfer Protocol (FTP)
Transport layer	Transmission Control Protocol (TCP)
Network layer	Internet Protocol (IP)
Data Link Layer	Ethernet Protocol
Physical Layer	Physical connection using a NIC or router to connect to the Internet

## Ethernet protocol

At the data link layer Ethernet protocols describe how network devices can format data for transmission using frames and packets. Ethernet protocols are also used to define standards for types of network cabling used at the physical layer and the corresponding transmission speeds.

## **Wi-Fi protocol**

Wi-Fi is a term for certain types of wireless networks that use 802.11 wireless protocols for transmitting data using electromagnetic waves in place of cables. 802.11 wireless networks use security protocols, such as Wi-Fi Protected Access (WPA), to provide a level of security and privacy comparable to that of a wired network. Bluetooth is another example of a wireless protocol and WAP (Wireless Application Protocol) are protocols to standardise the way that wireless devices can be used for Internet access.

## **Email protocols**

To use email you must have an email client on your computer that has access to a mail server. Your Internet Service Provider (ISP) often supplies this server. The mail client and the mail server exchange information with each other using email protocols to transmit information.

### **1. IMAP protocol**

Internet Messaging Access Protocol (IMAP) is an email protocol that stores email messages on a mail server. It allows the email user to read and handle email messages as though they were stored locally on their own computer. The user can manage their email with facilities such as the ability to create folders to organise their messages, store draft messages in the server and delete unwanted messages.

### **2. POP3**

Post Office Protocol 3 (POP3) is the third version of a protocol for receiving email. POP3 receives email for a client and stores it in a single file on the mail server. When the email client logs onto the mail server the email is transfer to the users computer. There are no copies of the email stored permanently on the server after they have been downloaded.

### **3. SMTP**

The Simple Mail Transfer Protocol (SMTP) is used to deliver email from the sender to an email server or when email is delivered from one email server to another. SMTP can only be used to send emails but not to receive them.

#### **INTERESTING FACT**

email existed before the World Wide Web. Early email was very simple – it just put a message in another user's directory in a location they could see when they logged on



## 4. Organisation and structure of data

### Organisation and structure of data

#### Representation of numbers

##### The nature of data

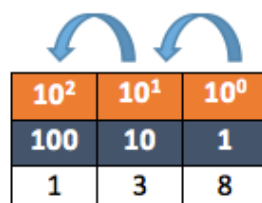
Data is made up of raw facts and figures and can be represented in many different forms including text, numbers, pictures, sounds and video clips. Information can be derived from data when it is processed.

##### Why data needs to be converted into binary format

You will need to be familiar with three different counting systems. These are denary, binary and hexadecimal.

##### Denary

The first counting system that you need to be familiar with is the **denary** counting system, also known as the **Base 10** or decimal counting system. In the denary counting system, the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are used to represent numbers. The number *138* for example, actually means 1 ‘hundred’, 3 ‘tens’ and 8 ‘units’. This gives the total one hundred and thirty-eight:



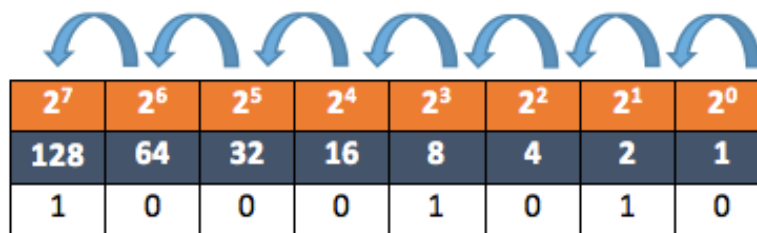
$10^2$	$10^1$	$10^0$
100	10	1
1	3	8

##### Binary

The second counting system that you need to be familiar with is the **binary** counting systems, also known as the **Base 2** counting system. In order for data to be processed by a computer system, it must be converted into binary format. This is because computer systems can only store and process **Binary digITs**, also known as **BITs**. A **BIT** is either a 1 or 0. You may think of this as a light switch, where the switch is either **ON** or **OFF**:

- If the switch is ON it is stored as the digit 1.
- If the switch is OFF it is stored as the digit 0.

A binary number is a string of BITS, for example 10001010.

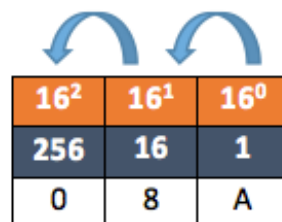


$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
128	64	32	16	8	4	2	1
1	0	0	0	1	0	1	0

The binary number 10001010 is therefore a binary representation of the denary number 138 ( $128 + 8 + 2$ ).

### Hexadecimal numbers representing binary numbers

The third counting system that you need to be familiar with is the **hexadecimal** counting system, also known as the **Base 16** counting system. In the hexadecimal counting system, the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are used to represent 1–9 and then the characters A, B, C, D, E and F are used to represent 10–15. The hexadecimal number 8A for example:



$16^2$	$16^1$	$16^0$
256	16	1
0	8	A

The hexadecimal number 8A therefore represents 8 ‘sixteens’ and 10 ‘units’. This gives the total one hundred and thirty-eight. Remember that A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

#### INTERESTING FACT

Up until the late 20<sup>th</sup> century, traditional Chinese weights and measurements used in the marketplace used the hexadecimal counting system. Other cultures used different base counting systems, e.g. the ancient Babylonians used a **Base 60** counting system.

Hexadecimal is widely used as binary numbers can be quickly converted into hexadecimal numbers that are more convenient for humans to use. For example, a telephone conversation where you might read out the binary number

10001010 could cause confusion. It would be easier to simply say 8A and mistakes are less likely to be made.

## Denary to binary and hexadecimal, binary to denary and hexadecimal, hexadecimal to binary and denary

In this section, we will discuss how to convert between different number systems.

### Denary to binary

One way of converting a denary number to a binary number is by drawing a Base 2 table from the right to the left.

128	64	32	16	8	4	2	1

In this example, we will convert the denary number 198 into a binary number. Take 198 and see if it is more than the first number on the left. In this case, 128 is the number on the left and so we write a 1 under the heading 128.

128	64	32	16	8	4	2	1
1							

We now deduct 128 from our original denary number, which leaves 70. The next number in our Base 2 table is 64. If the number remaining, 70, is more than the next number on the left, 64, write the number 1 under the heading 64.

128	64	32	16	8	4	2	1
1	1						

We now repeat the process again and deduct 64 from 70, which leaves 6. The next number in our Base 2 table is 32. If the number remaining, 6, is more than the next number on the left, 32, write the number 1 under the heading 32. However, in this case the number remaining is less than the next number on the left, so we write a 0 under the heading 32.

128	64	32	16	8	4	2	1
1	1	0					

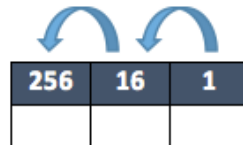
This process is repeated until you reach the final heading and the binary number for the denary number 198 is found:

128	64	32	16	8	4	2	1
1	1	0	0	0	1	1	0

The binary number for the denary number 198 is therefore 11000110 (128 + 64 + 4 + 2).

## Denary to hexadecimal

You may wish to convert a denary number into a hexadecimal number. To do this, take the number 198 from our previous example and draw a **Base 16** table, from right to left, as before.



256	16	1

Take 198 and see if it is more than the first number on the left. In this case, 256 is the number on the left and so we write a 0 under the heading 256.

256	16	1
0		

The next number in our **Base 16** table is 16. If the number remaining, 198, is more than the next number on the left, 16, work out how many 16s are needed without going over the number remaining. In this case it is C ( $C = 12$ ,  $12 \times 16 = 192$ ).

Remember that A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

256	16	1
0	C	

We now deduct 192 from our remaining denary number, 198, which leaves 6. The next number in our **Base 16** table is 1. If the number remaining, 6, is more than the next number on the left, 1, work out how many 1s are needed without going over the number remaining. In this case it is 6.

256	16	1
0	C	6

The hexadecimal number for the denary number 198 is therefore C6.

## Hint

You may find it easier to convert a denary number into a binary number first and then into a hexadecimal number. See the example *binary to hexadecimal* below.

## Binary to denary

To convert a binary number into a denary number, draw a Base 2 table from right to left and populate the table with the binary number you are converting. In this case we will use 00100011.

128	64	32	16	8	4	2	1
0	0	1	0	0	0	1	1

Simply convert the binary number into a denary number by adding the headings with a 1 under them:  $32 + 2 + 1 = 35$ . The denary number for the binary number 00100011 is therefore 35.

## Binary to hexadecimal

To convert a binary number into a hexadecimal number, there is a shortcut that you can use by drawing a Base 2 table from right to left and then populating the table with the binary number you are converting. In this case we will use 00101011.

128	64	32	16	8	4	2	1
0	0	1	0	1	0	1	1

Now split the Base 2 table into two smaller 4-bit Base 2 tables.

128	64	32	16
0	0	1	0

8	4	2	1
1	0	1	1

Now change the headings of the left 4-bit table.

8	4	2	1
0	0	1	0

8	4	2	1
1	0	1	1

Remember that A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

**2**

**B**

Now take the hexadecimal number of each 4-bit table and this is the converted hexadecimal number.

256	16	1
0	2	B

The hexadecimal number for the binary number 00101011 is therefore 2B.

### Hexadecimal to denary

You may wish to convert a hexadecimal number into a denary number. To do this you may take the number C6 and draw a Base 16 table, from right to left as before.

256	16	1
0	C	6

Now multiply each heading to obtain the denary converted number.

$$\begin{array}{r}
 C(12) \times 16 = 192 \\
 6 \times 1 = \quad 6 \quad + \\
 \hline
 198
 \end{array}$$

The denary number for the hexadecimal number C6 is therefore 198.

### Hexadecimal to binary

To convert a hexadecimal number into a binary number, there is a shortcut that you can use similar to the one above by drawing two 4-bit Base 2 tables from right to left.

8	4	2	1

8	4	2	1

In this example, we will convert the hexadecimal number 2B into a binary number. First start by representing the first number, 2, in the left table.

8	4	2	1
0	0	1	0

8	4	2	1

Then complete the second table by representing B in the right table, remembering that B = 11.

8	4	2	1
0	0	1	0

8	4	2	1
1	0	1	1

Now re-label the headings in the left table as shown below and join the two 4-bit tables together to make one 8-bit table.

128	64	32	16
0	0	1	0

↔

8	4	2	1
1	0	1	1

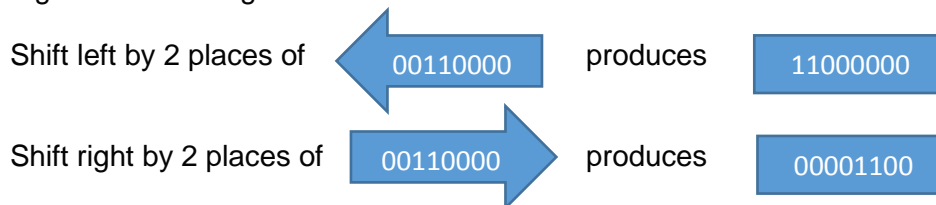
128	64	32	16	8	4	2	1
0	0	1	0	1	0	1	1

And so, the hexadecimal number 2B can be represented as 00101011 in binary number form.

### Arithmetic shift functions.

Shifts are manipulations of bit patterns. A shift involves moving the bits in a specified direction, either left or right, by a specified number of places

e.g. for an 8 bit register

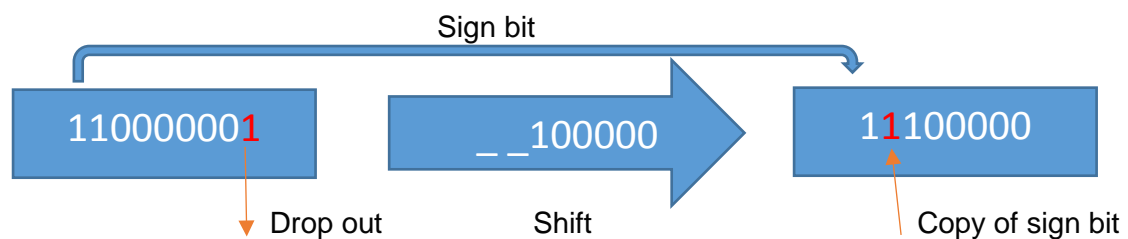


Arithmetic shifts can be used for division and multiplication.

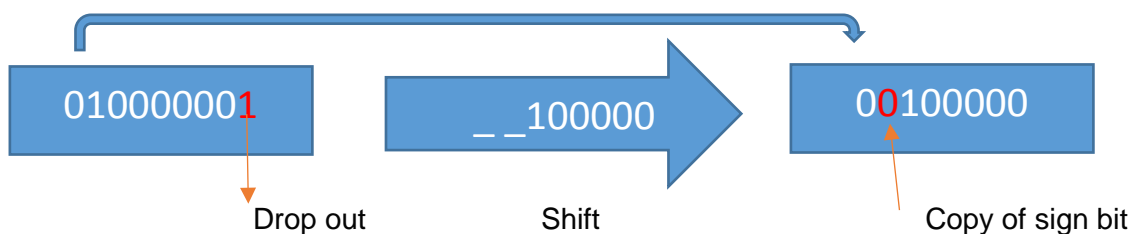
### Arithmetic shift right

This operation preserves the sign of a number and will divide a binary number by 2 at each shift. It will work for positive and negative numbers.

At each shift the right hand bit is lost and a copy of the sign bit is inserted to the left.



Negative integer in Two's complement: - 63 right shift 1 place = - 32



Positive integer in Two's complement: 65 right shift 1 place = 32

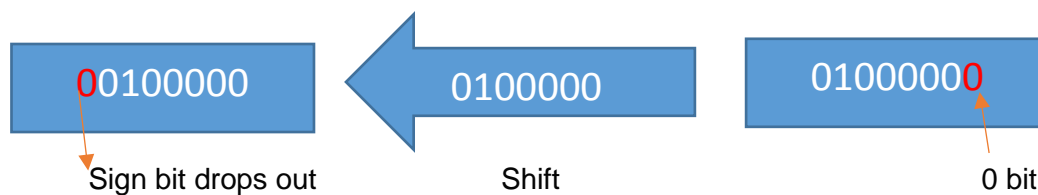
Note when the right bit has value i.e. when the shift is applied to an odd number the result is rounded down to the next even integer.

## Arithmetic shift left

Similar, but at each shift the sign bit is lost and a 0 bit is moved in to the right. The effect of each shift is to multiply the integer  $\times 2$ . The process can be repeated until the sign bit is changed, at which point overflow occurs.



Negative integer in Two's complement: - 64 left shift 1 place = - 128



Positive integer in Two's complement: 32 left shift 1 place = 64



Positive integer in Two's complement: 64 left shift 1 place = overflow

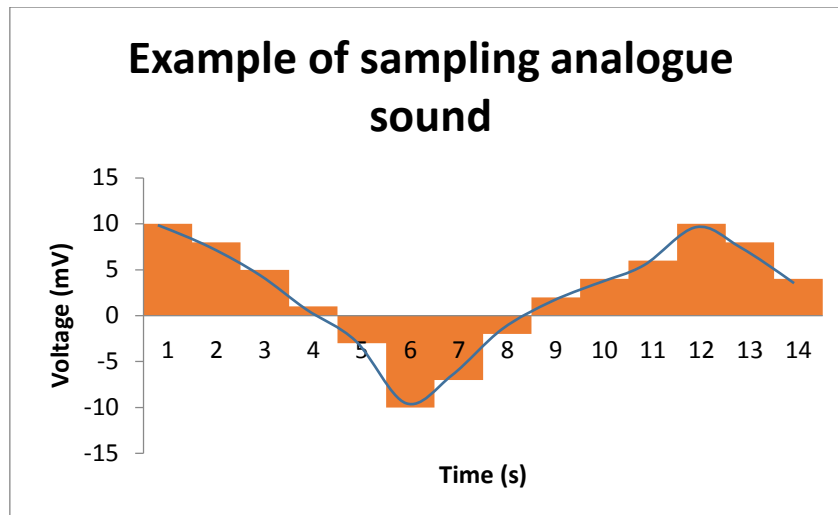
The shift changes the sign bit and overflow occurs.

## How sound can be sampled and stored digitally

As we have already established, a computer system is only able to store and process binary digits, as it is a digital device. Since this is the case, how can sound be stored as it is an analogue signal not digital? If an analogue signal, such as sound, is sent to a computer system, it has to be converted into a digital signal before it can be processed.

Sound is converted into a digital signal by a process called *sampling*. Sampling is where hardware, such as a microphone, measures the level of sound many times per second and records this as binary digits.





The number of times that the sound level is sampled per second is called the *sampling frequency*. The higher the sampling frequency, the better the quality of the sound recorded. A typical sampling frequency is 44,000 times per second, also known as 44 kHz. This is the sampling frequency used on most audio CDs.

Sound sampled at 44 kHz in stereo will produce a large amount of data and as such, this data may need to be compressed. When sound files are compressed, data is removed to reduce the size. This reduction in size means that

DIGITAL AUDIO QUALITY

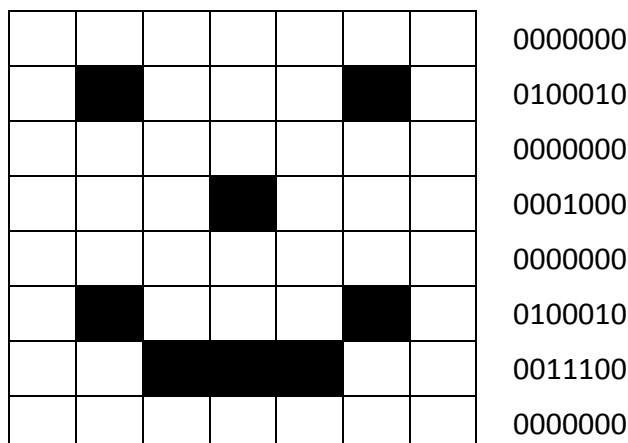
**Sample rate** – the number of audio samples captured every second

**Bit depth** – the number of bits available for each clip

**Bit rate** – the number of bits used per second of audio

### How an image is represented by pixels in binary format

Images on a computer system are made up of thousands of small coloured dots, known as pixels (short for picture elements). Bitmap images are stored as an array of pixels. A black and white bitmap image will store a 1 for a black pixel and 0 for a white pixel.



0000000  
 0100010  
 0000000  
 0001000  
 0000000  
 0100010  
 0011100  
 0000000

This bitmap image can be represented using a 56 bits (or 7 bytes).

A colour bitmap image is stored by replacing the 1s and 0s above with a longer number that represents how much red, green and blue (RGB) is required in the colour of each pixel; this is known as *colour depth*. In a 256-colour palette, the image would require 1 byte of storage per pixel – so we would need 448 bits (or 56 bytes) to store the image above in colour. There are other colour depths available, which can store more information about the colours in each pixel of an image. The more information stored about the colour of each pixel, the larger the file size becomes.

You may also have heard of vector images. These images do not store the data by pixels, but are a set of instructions for drawing a geometric shape. The advantages of a vector image are that they can be scaled without loss of quality (pixilation etc.) and use less storage space.

Images require a large amount of storage space and as such, may need to be compressed.

### **Why metadata needs to be included in an image file (including height, width, colour depth)**

The term metadata refers to 'data about data'. Key properties that are needed to display an image correctly are stored as metadata. Data such as an image's height, width and colour depth are typical examples of data stored in the metadata about an image. Without metadata, a computer system may render an image incorrectly on screen, such as displaying all pixels in one row.

Other data may also be stored in the metadata of an image file, such as the date the image was made, the geographical location of a photograph.

## Binary numbers representing characters

A **character** can be a letter, digit, space, punctuation mark or various other symbols. When characters are stored on a computer system, they are stored as a binary number.

It is important that computer systems recognise that characters can be represented differently by other computer systems; otherwise data could not be exchanged between computers.

## The terms 'character set', Unicode and American Standard Code for Information Interchange (ASCII)

In order to allow for data exchange between computer systems, **character sets** were devised. A character set is a table that maps a character with a unique binary number.

One such character set is the 7-bit American Standard Code for Information Interchange (ASCII). Part of the ASCII character set, that includes printable characters only, can be seen in the table overleaf.

### INTERESTING FACT

Before the widespread adoption of graphical user interfaces, programmers used the ASCII character set to design simple interfaces. Try searching for some on the Internet.

Denary	Binary	Hex	Character
32	100000	20	space
33	100001	21	!
34	100010	22	"
35	100011	23	#
36	100100	24	\$
37	100101	25	%
38	100110	26	&
39	100111	27	'
40	101000	28	(
41	101001	29	)
42	101010	2A	*
43	101011	2B	+
44	101100	2C	,
45	101101	2D	-
46	101110	2E	.
47	101111	2F	/
48	110000	30	0
49	110001	31	1
50	110010	32	2
51	110011	33	3
52	110100	34	4
53	110101	35	5
54	110110	36	6
55	110111	37	7
56	111000	38	8
57	111001	39	9
58	111010	3A	:
59	111011	3B	;
60	111100	3C	<
61	111101	3D	=
62	111110	3E	>
63	111111	3F	?

Denary	Binary	Hex	Character
64	1000000	40	@
65	1000001	41	A
66	1000010	42	B
67	1000011	43	C
68	1000100	44	D
69	1000101	45	E
70	1000110	46	F
71	1000111	47	G
72	1001000	48	H
73	1001001	49	I
74	1001010	4A	J
75	1001011	4B	K
76	1001100	4C	L
77	1001101	4D	M
78	1001110	4E	N
79	1001111	4F	O
80	1010000	50	P
81	1010001	51	Q
82	1010010	52	R
83	1010011	53	S
84	1010100	54	T
85	1010101	55	U
86	1010110	56	V
87	1010111	57	W
88	1011000	58	X
89	1011001	59	Y
90	1011010	5A	Z
91	1011011	5B	[
92	1011100	5C	\
93	1011101	5D	]
94	1011110	5E	^
95	1011111	5F	_

Denary	Binary	Hex	Character
96	1100000	60	`
97	1100001	61	a
98	1100010	62	b
99	1100011	63	c
100	1100100	64	d
101	1100101	65	e
102	1100110	66	f
103	1100111	67	g
104	1101000	68	h
105	1101001	69	i
106	1101010	6A	j
107	1101011	6B	k
108	1101100	6C	l
109	1101101	6D	m
110	1101110	6E	n
111	1101111	6F	o
112	1110000	70	p
113	1110001	71	q
114	1110010	72	r
115	1110011	73	s
116	1110100	74	t
117	1110101	75	u
118	1110110	76	v
119	1110111	77	w
120	1111000	78	x
121	1111001	79	y
122	1111010	7A	z
123	1111011	7B	{
124	1111100	7C	
125	1111101	7D	}
126	1111110	7E	~

Using the ASCII character set, the character A would be stored as the binary number 1000001.

The problem with using this ASCII character set is that it is only able to represent 128 different characters and computer systems need to be able to store more characters than this. For example, you may have noticed that the £ character is missing from the table above. As a result, other character sets were developed and used to allow computer systems to store more characters.

**Unicode** is a standard character set that has combined and replaced many others. It was originally an extension to the ASCII character set and it contains many of the characters used around the world.

## Data types such as: integer, real, Boolean, character, string

Many different data types can be stored on a computer system. The data types that are commonly used are as follows:

Data type	Description	Examples
<b>Integer</b>	Whole numbers, positive or negative	<b>42, -11, 0</b>
<b>Real</b>	Numbers, including fractions or decimal points	<b>12.9, -17.50, 28.0</b>
<b>Boolean</b>	True or false	<b>1 or 0</b>
<b>Character</b>	Letter, digit, space, punctuation mark or various other symbols	<b>'A', 'b', '7', '?'</b>
<b>String</b>	A sequence of characters	<b>'Computer science' 'The cat sat on the mat'</b>

## Data structures

A data structure is a specific way of organising data within memory so that it can be processed efficiently. There will be a relationship between the data items that will vary according to the type of data structure being used.

### Static data structure

A static data structure is designed to store a known number of data items. The values of the data can be changed but the memory size is fixed. An array is an example of a static data structure; we can change the values of the elements in the array but we cannot alter the memory size allocated to the array. Memory is allocated at compile time.

As static data structures store a fixed number of data items they are easier to program, as there is no need to check on the size of the data structure or the number of items stored.

## Dynamic data structure

Dynamic data structures are designed to allow the data structure to grow or shrink at runtime. It is possible to add new elements or remove existing elements without having to consider memory space. Memory is allocated at runtime.

Dynamic data structures make the most efficient use of memory but are more difficult to program, as you have to check the size of the data structure and the location of the data items each time you use the data.

### List

A list is a data structure that has the data items stored in the order they were originally added to memory. If the list is made up of a set number of data items, then it can be a static data structure. If the list can vary in the number of data items, then it will be a dynamic data structure.

### Array

An array is a data structure that can hold a fixed number of data items, which must be of the same data type i.e. real, integer, string etc. The data items in an array are known as elements. An array is an example of a static list.

The elements in an array are identified by a number that indicates their position in the array. This number is known as the index. The first element in an array usually has an index of 0.

Elements	37	11	42	6	26	56	4	76
Index	0	1	2	3	4	5	6	7

- There are 8 elements in this array.
- The index always starts at position 0.
- Each element can be accessed using its index. The element at index 5 is 56.
- This type of array is known as a one-dimensional array.

### Using one-dimensional arrays

A one-dimensional array can be used to store a list of data in memory that can be used by a program at runtime. There are basic operations that can be carried out on data in a one-dimensional array.

## Traversing

Traversing an array simply means using a loop to use each element of the array in a section of a program. If you wanted to print out the contents of an array called myArray that has 10 elements you would use a 'for . . . next' loop. Like this:

```

1
2 myArray[10]           'dimensions the array
3
4 for i = 0 to 9       'sets the loop
5     print myArray[i] 'prints each element in sequence
6 next in              'end of loop
7
8

```

## Insertion

You can add an element to an array at a given index.

```
myArray[3] = 27
```

This would store the value 27 at the index 3 of the array.

## Deletion

You can delete an element from an array.

```
myArray[6] = ""
```

This would leave the memory at index 6 blank.

## Search

Arrays can be search using the index or the value stored at the index.

## Two-dimensional arrays

Often the data we want to process comes in the form of a table. The data in a two-dimensional array must all be of the same data type.

For example, your teacher may have a spreadsheet of your class' test results.

Pupil	Test 1	Test 2	Test 3	Test 4	Test 5
Sam	23	5	34	4	23
Maisie	25	7	12	6	12
Elsie	34	8	45	8	32
Frank	9	4	43	9	26

Elements in a two-dimensional array are indexed by two numbers – one for the row it is in and one for the column.

		0	1	2	3	4
	Pupil	Test 1	Test 2	Test 3	Test 4	Test 5
0	Sam	23	5	34	4	23
1	Maisie	25	7	12	6	12
2	Elsie	34	8	45	8	32
3	Frank	9	4	43	9	26

If the two-dimensional array is called testMark then the command to declare this array would be:

```
testMark [4,5]
```

In this declaration, the 4 refers to the number of pupils (rows) and the 5 to the number of tests (columns). The index for Sam's marks for Test 2 would be [0,1].

To print out the pupils test marks you would need to use one loop inside another loop:

```

1
2 testMark[4,5]
3
4 for i = 0 to 3
5     for j = 0 to 4
6         print testMark[i, j]
7     next j
8 next i
9
10
```

## Records

Arrays can only hold data if it is all of the same data type. If you need to hold related data of different data types you will need to use a data structure called a record. A record will be made up of information about one person or thing. Each piece of information in the record is called a field.

For example, an after school club wants to store data about its members' emergency contact information.



## Record Structure

Field Name	Field Type	Example data
Membership Number	Integer	1074
First name	String	Sara
Surname	String	Davies
Date of Birth	Date	12/07/2004
Contact Name	String	Mrs Davies
Contact phone number	String	07564 191919

### Key field

Each record in a file should have a key field. That is an item of data that is unique and can be used to identify the individual record. In this example the membership number would be the key field.

### Files

For a computer to function it must have data flowing through the central processing unit (CPU) under the control of a program. More often than not this data will have come from a stored file.

A program will load the file from secondary storage, such as a hard disc, into the computer's memory. The data will be manipulated by the CPU and then output. The output could be another data file, screen images or a document.

Data stored in a file will have a structure and organisation known as the file format. A data file will be made up of records. It would be most efficient for the fields in a record to be stored next to each other so that the data can be read into the record data structure in memory for processing by the CPU.

In summary – files are made of records of the same structure and records are made up of fields containing information about one person or item.

## Validation

Validation is a process to check that input data is reasonable or sensible. Frequently used validation algorithms include;

Presence checks	Used to prevent further progress if a required field is left blank.
Format checks	Used to ensure data matches a specific pattern, such as dd/mm/yyyy for a date. Input masks are often used to create format checks on database forms.
Length checks	Used to ensure an input data string is a sensible length e.g. number of characters in 'firstName' to be between 3 and 16
Type checks	Used to ensure input data is a particular data type e.g. quantity ordered to be integer or cost to be real.
Range checks	Used to ensure input data lies within a specified range e.g. over time hours to be > 0 and < 15.

Look up checks can also be used to ensure that input data matches an item in a list of valid entries e.g. input look up "none"; "vegetarian"; "vegan" will limit the acceptable input to one of three entries.

## Verification

Verification is a process for checking data is correct. It can be carried out as a user enters data, such as via a keyboard and also when data is copied from one part of a system to another. Copying should not change the data.

Examples of verification of user input include double entry and screen based verification.

Double entry involves comparing two versions of data input. e.g. "re-enter your email address". A verification algorithm will compare the two versions and inform the user if they are not identical.

Screen based verification requires the user to check a display of input data and confirm that it is correct.

More sophisticated verification algorithms apply calculations to input data. e.g. to produce the check digits of bar codes. Repeating the calculations and checking that the result is the same can verify the data.

Similar verification algorithms, including parity checks and checksums, can be used when sending data between computers to check that the data has not been corrupted during transmission.

### Algorithm design - examples

- A. Design an algorithm that will sensibly validate the input data for a diving competition, where entrants must be between 16 and 19 years of age. The competition is to involve 5 dives, with each dive being awarded a score of between 0 and 20.

The algorithm should verify the entrants date of birth, check that they are eligible for competition and ensure that sensible scores are recorded.

- B. The table includes input data for a payroll system. Some of the data can be sensibly validated.

Surname	String
National Insurance (NI) number	Standard format LL123456L
Job title	Apprentice, semi-skilled, skilled, supervisor
Week no.	Integer
Full time	Y or N, Full time = 38 hours per week
Hours worked	Integer, hours worked in current week, maximum 10 hours overtime in one week.  Overtime rate = 1.5 x pay rate
Pay rate	Real, hourly pay rate, max £15.00 / hour

Write validation checks that will help to ensure that input is sensible.

# 5. Operating Systems

## Managing resources

An operating system is software that manages a computer system. The operating system is loaded by the bootstrap loader. For more information see page 13. One of its primary functions is to manage resources. Here are some examples of how the operating system manages the computer systems resources:

### Manages peripherals such as input and output devices

- Communicates with and sends data output to a printer/monitor/other valid output device
- Communicates with and receives data input to a keyboard/mouse/other valid input device

### Manages printing using spooling

- Data is stored on hard disc/in memory/stored in a queue
- Document is printed when printer is free/in correct order
- Benefit of spooling – user can carry on working/log off when waiting for job to print

### Manages backing store

- Ensures that data is stored and can be retrieved correctly from any disk drive
- Creates and maintains Filing system such as FAT or NTFS (accepted but not expected)
- Organise files in a hierarchical directory structure.

### Manages memory (RAM)

- Ensures that programs/data do not corrupt each other
- Ensures that all programs and data including itself is stored in correct memory locations

### Manages processes

- Ensures that different processes can utilise the CPU and do not interfere with each other or crash
- On a multi-tasking O/S ensure that all tasks appear to run simultaneously

### Manages security

- Allows creation and deletion of user accounts
- Allows users to logon and change passwords

## Providing user interface

Another function of the operating system is to provide a user interface. Here are some examples of how the operating system provides a user interface:

- allows copying/deleting/moving/sorting/searching of file or folders
- allows access to system settings such as hardware
- provides a command line interface
- allows users to have more than one window open
- provides a graphical user interface (Windows, Icons, Menus, Pointers)
- provides user with errors/help messages
- allows customisation of interface, e.g. change desktop background/layout
- allows user to switch between tasks (programs/windows)

**Human-computer interaction (HCI)** is the term used to describe the communication between people and computer systems. To allow a person and a computer system to communicate, an interface is required, often called a **human-computer interface**.

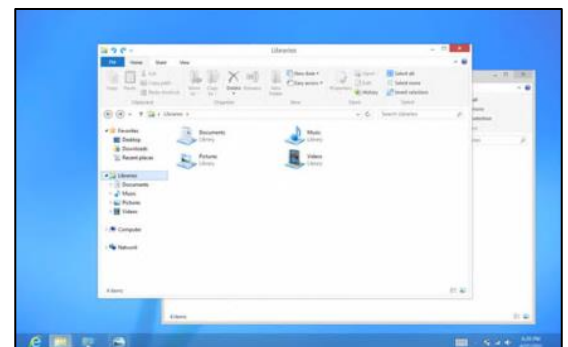
**INTERESTING FACT**  
The original name for Microsoft's Windows GUI was **Interface Manager**.

Different interfaces are provided by the operating system and can be identified by the style of communication they use. Some are entirely text-based whereas others use images to represent different commands.

## Graphical User Interface (GUI)

A GUI is a type of interface that allows users to interact with a computer system through graphical icons.

GUIs were introduced to help users, especially beginners, to use a computer systems as Command Line Interfaces (CLIs) were found to be difficult.



There are many different **features** of a graphical user interface. These include:

- windows
- icons
- menus
- pointers
- assistants/help files/tutorials
- favourite settings/change environment/customisation
- shortcuts/hot keys
- task bar/ribbon bar/tabs/customised toolbar

Here are some of the benefits and drawbacks of a graphical user interface:

Benefits	Drawbacks
<ul style="list-style-type: none"> <li>• Intuitive</li> <li>• Easy to navigate</li> <li>• Uses Windows, Icons, Menus and Pointers</li> <li>• No complicated commands</li> <li>• Data between different software applications is easily exchanged</li> </ul>	<ul style="list-style-type: none"> <li>• Requires a large amount of memory</li> <li>• Is relatively processor intensive</li> <li>• Computing experts may find a GUI slower than a command line interface</li> <li>• GUIs take up a much larger amount of hard disk space than other interfaces</li> </ul>

### Menu driven



This type of interface allows people to interact with a computer system by presenting the user and allowing them to work through a series of menus. The *iPod Classic* is a perfect example of a device that uses a menu driven interface as users are presented with a menu that contains a list of artists. Having chosen an artist, another menu appears with a list of albums belonging to that artist.

Following this, another menu is presented with a list of songs belonging to the chosen album.

Here are some of the benefits and drawbacks of a menu driven interface:

Benefits	Drawbacks
<ul style="list-style-type: none"> <li>• No need to learn a lot of commands</li> <li>• Intuitive/easy to understand</li> <li>• Easy to navigate</li> <li>• Ideal for beginners – everything is in a logical place/order</li> <li>• No need of expert language to learn</li> <li>• Little processing power needed</li> </ul>	<ul style="list-style-type: none"> <li>• Irritating if there are too many menu screens to work through – users get annoyed or bored if it takes too long</li> <li>• Navigating can be a long process</li> </ul>

### Voice-driven

Voice driven interfaces, also called voice recognition, can be used to issue commands to a computer system and enter data into it. Voice-driven interface is a popular interface as it is natural for people to communicate in this way.



Here are some of the benefits and drawbacks of a voice-driven interface:

Benefits	Drawbacks
<ul style="list-style-type: none"> <li>• Speech input is much faster than keyboard input</li> <li>• No need to learn to type</li> <li>• Less danger of RSI</li> <li>• Reduces typing mistakes such as spelling/hitting wrong key</li> <li>• Keyboard takes up room on the desk</li> <li>• Users with a disability that prevents typing can use speech input</li> <li>• Hands-free advantages – can multitask</li> <li>• Users find talking more natural than typing.</li> </ul>	<ul style="list-style-type: none"> <li>• Background noise interferes with speech recognition</li> <li>• User when they have a speech impediment, sore throat, cold or a strong accent will not be understood</li> <li>• Users with a disability that prevents speech would need to find a different method for input</li> <li>• Difficult to keep data input private as people can hear what you are saying</li> <li>• Words that sound the same, such as ‘too’ and ‘two’ may not be recognised</li> </ul>

## Command Line Interface (CLI)



A Command Line Interface is an entirely text-based interface that allows a user to communicate with a computer system by typing in commands. However, computer systems will only execute specific commands that are predefined.

Before GUIs were developed, command line interfaces were the most widely used interface.

Here are some of the benefits and drawbacks of a command line interface:

Benefits	Drawbacks
<ul style="list-style-type: none"> <li>• Quicker to type commands</li> <li>• Quicker to input commands as shortcut keys can be used</li> <li>• Little memory and processing power needed compared with other interfaces</li> <li>• Little storage space is required (no graphical images to store)</li> <li>• Experts who have memorised the commands find it very fast to use</li> </ul>	<ul style="list-style-type: none"> <li>• Very confusing for someone who has never used a command line interface</li> <li>• Commands have to be typed precisely. If there is a spelling error the command will fail</li> <li>• A large number of commands need to be learned</li> <li>• Instructions cannot be guessed</li> <li>• Not suitable for a novice</li> </ul>

## Touch Sensitive Interface

Touch sensitive interfaces are becoming more popular and are extensively used in mobile computing devices. Commands are issued or data is input by touching the screen with your finger or a stylus pen. As well as tapping the touch sensitive screen, the screen can interpret other actions made by the user, such as pinching and swiping.



Here are some of the benefits and drawbacks of a touch sensitive interface:

Benefits	Drawbacks
<ul style="list-style-type: none"> <li>• Very intuitive</li> <li>• Easier to use as the user simply touches what is seen on the display</li> <li>• No keyboard or mouse is required</li> <li>• Touching a visual display of choices requires little thinking and is a form of direct manipulation that is easy to learn</li> <li>• Easier hand-eye coordination than mice or keyboards</li> </ul>	<ul style="list-style-type: none"> <li>• Screen can be easily damaged/scratched</li> <li>• Dirty screens are difficult to read</li> <li>• Users must be within arm's reach of the display</li> <li>• It is difficult to select small items</li> <li>• User's hand may obscure the screen</li> <li>• Screens need to be installed at a lower position and tilted to reduce arm fatigue</li> <li>• Some reduction in image brightness may occur</li> </ul>

## Disk organisation such as: file transfer, formatting, compression

### File transfer

File transfer is the ability to transfer data from one location to another. This can be done by simply copying a file from one folder (directory) to another, or from one storage medium to another. You may wish to carry out either of these tasks in order to organise your files better, using subfolders or to back-up your work onto a secondary storage device, such as a flash memory stick.

### Formatting

Formatting is the process of preparing a disk for use. During this process, a new file system is set out on disk and all data may be erased in readiness for new data to be stored.

#### INTERESTING FACT

Certain specialist software can be used to "unformat" a formatted disk and recover all the data originally stored on it.



## Compressing

Compression is the process of making a file size smaller. This may be advantageous as it allows more data to be stored on the disk and files may also be transferred more quickly. There are two methods of achieving disk compression; one is software based and the other hardware based.

Software based disk compression is often included as a facility of an operating system and so it is readily available on most computer systems. The disadvantage of this is that it slows down the process of reading and writing to disk.

Hardware disk compression requires specialist hardware, which can be expensive. However, it does not affect the speed of access as much as software based disk compression.

Disk based compression is always lossless. For further discussion see page 54.

## System restore (roll back), disk defragmentation, control panel, system maintenance tools.

Many different system maintenance tools are included with operating systems that allow users to maintain the upkeep of their computer systems. Here are some of the tools below.

### System restore (roll back)

System restore is the process of replacing lost or corrupt data by replacing it with an earlier backup.

#### INTERESTING FACT

Some modern viruses exploit the system restore facility by deliberately seeking out back-ups and placing copies of themselves there.

### Disk defragmentation

Files are stored on computer systems that can, over time, become fragmented. This means that they are split and stored on different parts of the disk. If a file is fragmented, it takes longer for the disk heads to move between parts of the file, which slows the process of loading it.

Defragmentation is the process where files are physically re-arranged on disk so that they are no longer fragmented and the parts of each file are stored together. This improves the speed of accessing data from disk.

## **Control panel**

Many operating systems use a control panel to give the user control of software and hardware features. It enables the user to change settings, such as sound, device and display settings all from one convenient location.

## 6. Principles of programming

### High level languages

A high level language is a programming language that allows code to be written. It is similar to a natural human language, such as English. Some programmers prefer to use high level programming languages, as they are easier to understand, learn and program. Their commands are similar to natural languages like English and identifiers can be long and meaningful. High level programming languages also allow the use of powerful commands that perform quite complex tasks such as MsgBox in Visual Basic or the SORT clause in COBOL. Examples of common high level programming languages include:

- Basic
- Java
- Pascal
- COBOL
- C#
- C++

### Machine code

Machine code is the opposite of a high level language in that it does not resemble any natural language and is made up entirely of bit patterns (instructions or data) that can be executed directly by the CPU. Examples of machine code instructions are opcodes and operands. High level languages must be converted into machine code before they can be executed by the CPU.

### Low level languages

Programming in a low level language, such as assembly code requires knowledge of the internal structure of the CPU and is therefore very specialised. The program statements are written for a particular type of CPU and make direct reference to specific internal registers. Assembly code uses mnemonics and is converted to machine code for execution using an assembler. Source code produced in a low level language, is not portable, but it can be very efficient and the programs can be made to run faster than programs produced using a high level language.

## Uses of high and low level languages

High level languages are used when the execution speed is not the most critical factor, e.g. in common productivity applications, such as a word processor, or spreadsheet. Most modern applications such as commercial database packages, operating systems, e-commerce software and social media apps are developed using a high level programming language.

Although uncommon, some programmers may wish to program directly in machine code or use assembly code. This is primarily done when programming device drivers or embedded systems, where fast execution speeds are critical. Professional game developers may need to use console specific development software, which is likely to include low level features for optimum performance.

## 7. Software engineering

### Software development environment

Software development environments, also known as Integrated Development Environments (IDE), provide programmers with various tools that are needed to create computer programs. Here are some of the tools and facilities offered by a typical software development environment:

Facility	Use
<b>Editor</b>	Allows a programmer to enter, format and edit source code
<b>Compiler</b>	Converts source code into executable machine code. Once compiled, a program can be run at any time
<b>Interpreter</b>	Converts each line of source code into machine code, and executes it as each line of code is run. The conversion process is performed each time the program needs to be run
<b>Linker</b>	A program which allows previously compiled code, from software libraries, to be linked together
<b>Loader</b>	A program which loads previously compiled code into memory
<b>Debugger</b>	A program which helps locate, identify and rectify errors in a program
<b>Trace</b>	A facility which displays the order in which the lines of a program are executed, and possibly the values of variables as the program is being run
<b>Break point</b>	A facility which interrupts a program on a specific line of code, allowing the programmer to compare the values of variables against expected values. The program code can then usually be executed one line at a time. This is called <i>single-stepping</i>
<b>Variable watch</b>	A facility that displays the current value of any variable. The value can be 'watched' as the program code is single-stepped to see the effects of the code on the variable. Alternatively a variable watch may be set, which will interrupt the program flow if the watched variable reaches a specified value
<b>Memory inspector</b>	A facility which will display the contents of a section of memory
<b>Error diagnostics</b>	Used when a program fails to compile or to run. Error messages are displayed to help the programmer diagnose what has gone wrong

## Libraries

A library is a collection of commonly used private functions and subprograms. Examples of private functions include standard mathematical operations such as square root, random number generators. Examples of subprograms include standard input/output routines, such as saving data to disk. These functions and subprograms can be called from within your program at any stage, but only when the appropriate library has been linked.

The advantages of using standard libraries are that:

- related private functions and subprograms are stored in the same location
- time is saved as the programmer can simply use the private functions and programs stored in a library
- subroutines contained in a library have already been tested, so they should work reliably and not need further testing
- programs will contain less code and will therefore be easier to maintain

Most computer languages use standard libraries, although it is also possible to create your own custom libraries.

## 8. Program construction

### Assemblers

An assembler is a program which converts the low level assembly programming language into machine code. The assembler does this by converting the one-word assembly instructions into an opcode, e.g. converting AND to 0010. It also allocates memory to variables, often resulting in an operand.

Assembly code	Assembler conversion	Opcode	Operand
AND A	→	0010	0001
LOD B	→	0110	0010

### Interpreters

Before high level programming languages can be run, code is converted by an interpreter, one line at a time, into machine code, which is then executed by the CPU.

### Compilers

A compiler is used when high-level programming languages are converted into machine code, ready to be executed by the CPU. There are four main stages of compilation:

#### Lexical analysis

- Comments and unneeded spaces are removed.
- Keywords, constants and identifiers are replaced by 'tokens'.
- A symbol table is created which holds the addresses of variables, labels and subroutines.

#### Syntax analysis

- Tokens are checked to see if they match the spelling and grammar expected, using standard language definitions. This is done by parsing each token to determine if it uses the correct syntax for the programming language.
- If syntax errors are found, error messages are produced.

#### Semantic analysis

- Variables are checked to ensure that they have been properly declared and used.
- Variables are checked to ensure that they are of the correct data type, e.g. real values are not being assigned to integers.
- Operations are checked to ensure that they are legal for the type of variable being used, e.g. you would not try to store the result of a division operation as an integer.

#### **Code generation**

- Machine code is generated.
- Code optimisation may be employed to make it more efficient/faster/less resource intense.

#### **Translators**

A translator changes (translates) a program written in one language into an equivalent program written in a different language. For example, a program written using the PASCAL programming language may be translated into a program written in one of the C programming languages using a translator.



## Types of error that may occur in programming code

Error	Description	Example
Syntax	An error that occurs when a command does not follow the expected syntax of the language, e.g. when a keyword is incorrectly spelt	<ul style="list-style-type: none"> <li>• <b>Incorrect:</b> IF A <b>ADN</b> B Then</li> <li>• <b>Correct:</b> IF A <b>AND</b> B Then</li> </ul>
Runtime/execution	An error that only occurs when the program is running and is difficult to foresee before a program is compiled and run	<ul style="list-style-type: none"> <li>• Program requests more memory when none is available, so the program <i>crashes</i></li> </ul>
Logical	An error that causes a program to output an incorrect answer (that does not necessarily crash the program)	<ul style="list-style-type: none"> <li>• An algorithm that calculates a person's age from their date of birth, but ends up giving negative numbers</li> </ul>
Linking	An error that occurs when a programmer calls a function within a program and the correct library has not been linked to that program	<ul style="list-style-type: none"> <li>• When the square root function is used and the library that calculates the square root has not been linked to the program</li> </ul>
Rounding	Rounding is when a number is approximated to nearest whole number/tenth/hundredth, etc.	<ul style="list-style-type: none"> <li>• 34.5 rounded to nearest whole number is 35, an error of +0.5</li> </ul>
Truncation	Truncating is when a number is approximated to a whole number/tenth/hundredth, etc. nearer zero	<ul style="list-style-type: none"> <li>• 34.9 truncated to whole number is 34, an error of -0.9</li> </ul>

### INTERESTING FACT

In 1947, Grace Murray Hopper, an admiral in the U. S. Navy and a computer programming pioneer, documented the first actual computer bug when a moth got trapped in a computer.

## 8. Security and data management

Information stored on computers can be sensitive and needs to be looked after very carefully. This means restricting access to the information. Inappropriate and unauthorised access to this information is likely to have serious consequences and could result in legal penalties, identity theft, financial loss, fraud and invasion of privacy.

Other risks to information stored on computers include loss due to accidental deletion, or overwriting parts of files in error; mechanical damage to hard discs, which are the most fragile parts of a computer; power failure whilst work is in progress; accidental damage to hardware, such as fire or damage caused by spilling a drink.

Most of these risks can be managed by adopting efficient procedures for saving work and making regular backups.

### Network security

Security is of paramount importance to any network as the loss of data, in particular, personal or confidential data can have many serious consequences. Risks to data become greater as it is shared across a network.

### User access levels

It is not desirable that every user should be able to access all the data on a computer system. User access levels are one method used to allow certain users read and/or write access to data on a computer system. For example, in a program used within a company, an administrator, possibly the owner, will have read and write access to all data on the system. An assistant, however, will not have access to confidential data such as employees' salaries. User access levels will define which users can change and view, view but not change, or not view stored data.

### Suitable passwords

Passwords are commonly used to prove a person's identity to a computer system, thus allowing them access to relevant data.

Different programs may require a user to use different complexities of password, as well as different character lengths. An example of a simple password may be the user's town of birth, or the word 'password'. A more complex

#### INTERESTING FACT

The average Internet user has 25 online accounts, 6.5 passwords and waits an average of 3.1 months before changing passwords.

password may require the user to use a combination of upper and lower case alphanumeric characters, for example 'Pa55word1234'. Increasingly, computer programs require users to use a combination of upper and lower case alphanumeric characters as well as other non-alphanumeric characters such as @ ! ~ - / \ %, for example 'P@55word/1234!'.

Another user can guess short simple passwords, or a hacker may have access to programs that have the ability to try multiple guesses in quick succession. This is known as a *brute force attack*. Passwords that use a combination of upper and lower case alphanumeric characters as well as other non-alphanumeric character, will be much harder to guess and will take longer to 'brute force'.

As a rule of thumb, the following formula can be used to determine the number of attempts it would take to brute force a password.

$$\text{Attempts} = \text{Number of characters}^{\text{Password length}}$$

So a password, such as 'computer' (8 characters), which only contains lower case characters from the 26 letter English alphabet will take:

$$\text{Attempts} = 26^8 = 208,827,064,576$$

(on a typical 3.5GHz computer, this would take less than 6 seconds to brute force)\*

Whereas a password that contains upper and lower case alphanumeric characters, such as 'Computer1' (9 characters), has 26 + 26 + 10 = 62 possible characters. This will take:

$$\text{Attempts} = 62^9 = 13,537,086,546,263,552$$

(on a typical 3.5GHz computer, this would take just over 1 hour to brute force)\*

\*assuming one attempt per clock-tick

## Encryption techniques

Encryption is the conversion of data, using an algorithm, into a form, called cyphertext that cannot be easily understood by people without the decryption key.

When data is encrypted, a logical operator is sometimes used, called the XOR logical operator.

## XOR

The XOR logical operator has two inputs and one output. The output is 1 only if A and B are different.

Input (A)	Input (B)	Output (A XOR B)
0	0	<b>0</b>
0	1	<b>1</b>
1	0	<b>1</b>
1	1	<b>0</b>

When encrypting data, the XOR logical operator is performed on the original data and a *key*. The key is a secure binary number, known only to the sender and recipient.

In this example, we will encrypt the data 10101010, using the key 11110000.

<b>Original Data</b>	<b>10101010</b>	
<b>Key</b>	<b>11110000</b>	XOR
<hr/>		
<b>Cyphertext</b>	<b>01011010</b>	

The original data, 10101010, is now encrypted and can be transmitted as 01011010.

To recover the original data, the cyphertext is *XOR'ed* with the key.

<b>Cyphertext</b>	<b>01011010</b>	
<b>Key</b>	<b>11110000</b>	XOR
<hr/>		
<b>Original Data</b>	<b>10101010</b>	

Other, more complex techniques are also used to encrypt data e.g. SHA256 and Blowfish.

## Compression and compression types

Compression is the process of making a file size smaller. This may be advantageous as it allows more data to be stored on the disk and files may also be transferred more quickly. There are two primary methods that are used to compress files stored on a computer system; these are *lossy* and *lossless*.

## Lossless compression

Lossless compression uses an algorithm that compresses data into a form that may be decompressed at a later time without any loss of data, returning the file to its exact original form. It is preferred to lossy compression when the loss of any detail, for example in a computer program or a word-processed document, could have a detrimental effect.

A simplified version of lossless compression on a word-processed document may be to replace a common string, such as 'the', with a token such as the symbol @. One character takes 1 byte of memory; therefore, the string 'the' would take 3 bytes.

Original uncompressed text	The word <b>the</b> , is <b>the</b> most frequently used word in <b>the</b> English language.	71 characters (bytes)
Compressed text	@ word @, is @ most frequently used word in @ English language.	63 characters (bytes)




This is an 11% reduction in the file size!

## Lossy compression

Lossy compression is a technique that compresses the file size by discarding some of the data. The technique aims to reduce the amount of data that needs to be stored.

The following versions of the Eduqas logo show how much of the data can be discarded, and how the quality of the images deteriorate as the data that made up the original is discarded. Typically, a substantial amount of data can be discarded before the result is noticeable to the user. The compression ratio is calculated using the simple formula:

$$\text{Compression ratio} = \frac{\text{Original file size}}{\text{Compressed file size}}$$

		
Original image 100 kB	Compressed image 10 kB (compression ratio = $100/10 = 10$ or 10:1)	Compressed image 5 kB (compression ratio = $100/5 = 20$ or 20:1)
<b>File size</b>	<b>File size</b>	<b>File size</b>

Lossy compression is also used to compress multimedia data, such as sound and video, especially in applications that stream media over the Internet.

## Network policies

### Acceptable use

Network policies are documents written to outline the rules that users are required to follow while using a computer network. Each document is often several pages long, written and agreed by a committee. Following its publication, network users will be expected to adhere to the rules.

Typical rules set out in these policies include a list of unacceptable types of website that should not be visited and activities that are not allowed on the network, such as gambling and installation of unauthorised software.

## Disaster recovery

Given the amount of important data often stored on a computer network, it is essential that an effective disaster recovery policy be in place. Disasters include:

- fire, flood, lightning, terrorist attacks etc.
- hardware failure, e.g. power supply unit failing
- software failure, e.g. virus damage
- accidental and malicious damage, e.g. hacking

There are usually three parts to a disaster recovery policy:

- **before the disaster:** risk analysis, preventative measures and staff training
- **during the disaster:** staff response – implement contingency plans
- **after the disaster:** recovery measures, purchasing replacement hardware, reinstalling software, restoring data from backups

## Backup

A **backup** is a copy of data that can be used if the original data is lost.

Backups of all data should be made regularly as the older the backed up data becomes, the less likely it is to match any current data stored on a computer system.

### INTERESTING FACT

Key causes of data loss are:

- 78% hardware failure
- 11% human error
- 7% software failure
- 2% computer viruses
- 1% other

A backup policy sets out how often and to what medium backups are made. The backup medium is generally different to the active storage medium. Historically, the medium used was magnetic tape backup.

A typical backup policy would require that three different backups be kept at any given time, with one of these being stored off-site. The oldest backup copy would be named the *grandfather*, the second oldest backup being named the *father* and the most recent backup being called the *son*. When a new backup is made, the oldest backup, the *grandfather* is overwritten and becomes the *son* backup, with the original son becoming the *father* and the father becoming the *grandfather*. This backup policy is called the *grandfather-father-son* method.

## Archiving

Data held on computer systems is often **archived**. Archiving is the process of storing data that is no longer in current or frequent use. It is held for security, legal or historical reasons. The process of archiving data frees up resources on the main computer system and allows faster access to data that is in use.

eBay – Customer’s usernames, encrypted passwords, email addresses of 145 million users stolen. No payment information was taken

use. It is reasons. resources allows

## Cybersecurity

Online networks are vital to many business operations, but they are liable to attacks targeted to access confidential data, such as customers’ details or technical information about products etc. This level of data is very expensive to gather and its loss could result in loss of reputation and even business failure.

### INTERESTING FACT

October 2013. Email addresses and passwords for 150 million users and credit card data for 2.9million users stolen from Adobe’s network.

### INTERESTING FACT

eBay – Customer’s usernames, encrypted passwords, email addresses of 145 million users stolen. No payment information was taken.

Cybersecurity refers to the range of measures that can be taken to protect computer systems, networks and data from unauthorised access or cyberattack. Cyberattacks are carried out using various types of **malware** (malicious software), including;

**Viruses.** Viruses are programs that can replicate themselves and be spread from one system to another by attaching themselves to host files. They are used to modify or corrupt information on a targeted computer system.

**Worms.** Worms are self-replicating programs that identify vulnerabilities in operating systems and enable remote control of the infected computer.

**Spyware.** Installed by opening attachments or downloading infected software. Spyware can be used to collect stored data without the user’s knowledge.

Keyloggers are a type of spyware that can be used to track keystrokes and capture passwords, account numbers for fraudulent use.

**Trojans.** A Trojan is a program that appears to perform a useful function, but also provides a ‘backdoor’ that enables data to be stolen.

Parents can use keylogger software to monitor their children’s online activity.



## Protection against malware

**Install Virus protection software**, also called anti-virus software, is a program that can be loaded into memory when the computer is running. It monitors activity on a computer system for the signs of virus infection. Each virus has its own unique 'signature' that is known

to virus protection software and stored in a database. Data stored on a computer system is scanned to see if any of the virus signatures within the database exist on the system.

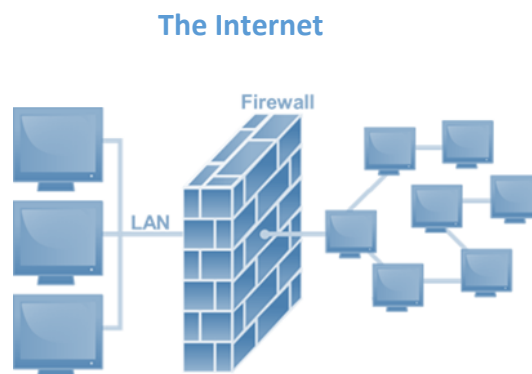
### INTERESTING FACT

Some advanced viruses attempt to evade the virus protection software by changing their own code so that they no longer match the "signature" in the virus signature database.

These are known as *polymorphic viruses*.

There are many thousands of known viruses, and new viruses are created daily. Virus protection software therefore needs to be updated regularly to combat these.

**Use a firewall.** A firewall can be a software or hardware security system that controls the incoming and outgoing network traffic. Packets of data are analysed to determine whether they should be allowed through or not.



The basic function of a firewall is to monitor where data has come from and where it is going and to determine if this communication is allowed. It does this by checking a list of pre-defined rules.

**Keep your operating system up to date.** New ways to bypass the operating system's built-in security are often discovered and can be covered by installing the security patches issued by the operating system manufacturer.

**Use the latest versions of web browsers.** As for operating systems the manufacturers of web browsers seek to continually improve their products and remove possible security vulnerabilities. Most browsers will download updates automatically, but will need a restart for the update to be installed.

**Look out for phishing emails.** Emails that ask you to confirm personal details are usually fakes. They should be caught by the spam filter, but be suspicious and do not provide any sensitive information.

If you suspect you have malware on your computer you will need to download and run a **malicious software removal tool** that should detect and remove malware not blocked by the anti-virus software.

## Forms of Cyberattack

Internet protocols, operating systems and network equipment all present inherent technical weaknesses that must be recognised and protected against. User behaviour can also compromise security e.g. sending sensitive documents to unintended recipients, opening malicious attachments to scam emails, or using the same passwords for multiple systems. Specific forms of attack include;

### Shoulder surfing

Shoulder surfing is using direct observation to get information. It is relatively simple to stand next to someone and watch as they fill out a form, or enter a PIN number, but shoulder surfing can also be carried out long distance with the aid of binoculars or even CCTV.

### SQL injection

SQL injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input. Injected SQL commands can alter SQL statements and compromise the security of information held in a database.

### DoS attack.

Denial of service (DoS) attacks do not attempt to break system security, they attempt to make your website and servers unavailable to legitimate users, by swamping a system with fake requests—usually in an attempt to exhaust server resources.

A DoS attack will involve a single Internet connection. Distributed denial of service (DDoS) attacks are launched from multiple connected devices that are distributed across the Internet. These multi-person, multi-device attacks target the network infrastructure in an attempt to saturate it with huge volumes of traffic.

#### **Carphone Warehouse data breach DDoS – 2015**

In July 2015, major UK smartphone retailer Carphone Warehouse suffered a serious data breach, which it later transpired, might have been aided using a DDoS 'distraction' attack. Up to one in five DDoS incidents are later found to be part of a data theft snatch in which IT staff are occupied fending off the DDoS, giving attackers more opportunity to sneak in and out.

### Password-based attacks.

Passwords are a good starting point, but are they secure? Cyber criminals have ways of finding out your password.

<b>Dictionary attack</b>	This uses a simple file containing words found in a dictionary. This attack uses exactly the kind of words that many people use as their password.
<b>Brute force attack</b>	Similar to the dictionary attack but able to detect non-dictionary words by working through all possible alphanumeric combinations from aaa1 to zzz10. It's not quick, but it will uncover your password eventually.
<b>Guess ???</b>	A user-generated password is unlikely to be random. Passwords are likely to be based upon our interests, hobbies, pets, family etc. Educated guesses often work.

### IP address spoofing.

A spoof is a hoax, or a trick. IP address spoofing involves an attacker changing the IP address of a legitimate host so that a visitor who types in the URL of a legitimate site is taken to a fraudulent or spoofed web page. The attacker can then use the hoax page to steal sensitive data, such as a credit card number, or install malware.

### Social engineering

Internet users frequently receive messages that request password or credit card information to “set up their account”. Social engineering involves tricking a user into giving out sensitive information such as a password, by posing as a legitimate system administrator.

Examples of social engineering attacks carried out by deception include **phishing**, which is an attempt to acquire users’ details using fake emails and websites, and **pharming**, where users are unknowingly re-directed to a fake website, again with the intention of identity theft.

## Identifying vulnerabilities

### Footprinting.

Footprinting is the first step in the evaluation of the security of any computer system. It involves gathering all available information about the computer system or network and the devices that are attached to it. Footprinting should enable a penetration tester to discover how much detail a potential attacker could find out about a system and allow an organisation to limit the technical information about its systems that is publicly available.

### Ethical hacking

Ethical hacking is carried out with the permission of the system owner to cover all computer attack techniques. An ethical hacker attempts to bypass system security and search for any weak points that could be exploited by malicious hackers. This information is then used by the system owner to improve system security.

### Penetration testing

Penetration testing is a sub set of ethical hacking that deals with the process of testing a computer system, or network to find vulnerabilities that an attacker could exploit. The tests can be automated with software applications or they can be performed manually.

Penetration test strategies include;

- Targeted testing, testing carried out by the organization's IT team and the penetration testing team working together.
- External testing, to find out if an outside attacker can get in and how far they can get in once they have gained access.
- Internal testing, to estimate how much damage a dissatisfied employee could cause.
- Blind testing, to simulate the actions and procedures of a real attacker by severely limiting the information given to the team performing the test.

## Protecting software systems

### Secure by design

Secure by design is an approach that seeks to make software systems as free of vulnerabilities as possible through such measures as continuous testing and adherence to best programming practices. At the design stage malicious practices are taken for granted and it is assumed that the new system will have invalid data entered or will be the subject of hacking attempts. These issues are taken into account and corresponding security measures

are considered to ensure security is not an afterthought and so reduce the need for addressing vulnerabilities and patching security holes as they are discovered in use.

Some examples of attacks that should be prevented during design and testing include;

### **Buffer overflow attacks**

A buffer overflow occurs when a program tries to store more data in a buffer (temporary data storage area) than it was intended to hold. This may occur accidentally through programming error, or it may be caused intentionally in a buffer overflow attack, where the overflow data may contain codes designed to change data, or disclose confidential information. Thorough testing, particularly of any library routines used, will help to prevent this type of attack.

### **Permissions**

Every time you want to install an app you are asked to give permission for the software to access certain settings and features of your device e.g. Facebook's Messenger App, which boasts over 1,000,000,000 downloads, requires permission to access a large amount of personal data and requires direct control over your mobile device. It is unlikely that many of those who downloaded this app read the full 'Terms of Service' before accepting them. It is not always easy to understand what you are permitting an app to do. Should you uninstall an app because its permissions are suspicious?

App developers are keen to develop interactive products that are useful, but they need to consider the scope of access and limit the number of permissions required at the design stage. There are malicious apps, but you can avoid them by using common sense.

### **Scripting restrictions**

Same Origin Policy (SOP) is a security measure that prevents a web site's scripts from accessing and interacting with scripts used on other sites. Running scripts from other sites would be dangerous because a malicious script from a compromised site could interact with a script from a legitimate site without restriction, potentially leading to malware infections or sensitive data being compromised.

A programmer can control the range and type of scripts allowed by setting the restrictions in, e.g. an HTML page header, or by using standard script execution settings such as unrestricted, trusted, restricted etc.

### **Accepting parameter without validation**

Dynamically generated HTML pages can introduce security risks if inputs are not validated on the way in. Malicious script can be embedded within input that is submitted to web pages and this could then appear to browsers as originating from a trusted source.

Approaches to prevent this type of cross-site scripting attack rely on the design of validation rules that will check and filter input parameters.

## **The role of cookies**

**Cookies** are data stored on a computer system. They allow websites to store a small amount of uniquely identifying data on your computer system while you are visiting. This may be useful as the website can then identify you in future without requesting that you identify yourself each time, i.e. by entering a username and password. Another use of a cookie would be when adding items to a shopping basket over a period of time. The cookie allows you to store this information between separate browsing sessions.

# 10. Ethical, legal and environmental impacts of digital technology.

**Ethics** - the branch of philosophy that is concerned with right and wrong. What is good for an individual and what is good for society as a whole? Developments arising from computer science and the resulting digital technologies produce ethical implications for individuals and society. These implications are not always obvious and sometimes lead to unanticipated problems.

## The Digital Divide

The digital divide is a term that refers to the gap between populations that have full access to modern information and communications technology, and those who have restricted access. It is used mainly to describe the split between those with and those without broadband Internet access.

The divide traditionally exists between those in cities and those in rural areas; between the educated and the uneducated; between socioeconomic groups; and, globally, between the more and less industrially developed nations. Even among populations with some access to technology, the digital divide can be evident in the form of lower-performance computers and lower-speed connections.

The next billion people coming online will do so from cheap, mobile phones. While the cost of phone services is falling globally, fixed broadband, typically more reliable and faster than cellular connections, is becoming more expensive in the poorest countries.

Some UN statistics from 2015

- For the least developed countries, the average broadband cost grew by more than 30%, “a sharp increase that will certainly not improve the already very low uptake of fixed-broadband in the world’s poorest countries.”
- 43.4% of the world’s population will use the internet in 2015, but that figure falls to 9.5% for the least developed countries
- women in low and middle income countries are 21% less likely to own a mobile phone, helping perpetuate inequality between men and women.
- 8% of females will have used the internet in 2015 compared with 11.3% of males

The UN has set goals for broadband services to cost no more than 5% of average monthly incomes in developing countries by 2020 and to achieve gender equality among Internet users by 2020.

Do you think achieving these the goals will be enough to close the digital divide? Who should provide the technology and meet the cost?

The World Bank's 'World Development Report 2016: Digital Dividends' summarises (<http://www.worldbank.org/en/publication/wdr2016>)

“Digital technologies have spread rapidly in much of the world. Digital dividends, that is, the broader development benefits from using these technologies, have lagged behind. In many instances, digital technologies have boosted growth, expanded opportunities, and improved service delivery. Yet their aggregate impact has fallen short and is unevenly distributed. For digital technologies to benefit everyone everywhere requires closing the remaining digital divide, especially in Internet access. But greater digital adoption alone will not be enough.”

What other measures will be required to close the digital divide?

### **Some example discussion topics**

#### **Drones.**

The idea of remotely controlled unmanned vehicles flying through the air either raises concerns over personal privacy, or leads us to consider citizens who live in fear of drones used for military purposes.

NASA have successfully tested a prototype system that allows unpiloted drones to detect and avoid other aircraft in their midst. The agency's drones are able to sense when something was in their flight path and make adjustments on their own.

In the UK the Civil Aviation Authority is warning that drones being flown as high up as 2,000ft are putting passenger aircraft in danger. Drones can be used to monitor pollution levels and e.g. to track wildlife poachers. Should their use be controlled?



### **Self-driving cars?**

Standard features on many ordinary cars include intelligent cruise control, parallel parking programs, and even automatic overtaking—features that allow you to sit back and let a computer do the driving.

Many car manufacturers are beginning to design cars that take the driving out of your hands altogether. It is claimed that these cars will be safer, cleaner, and more fuel-efficient than their manual counterparts, but can they ever be perfectly safe?

The idea of the computer controlling the car raises some ethical questions. How should the computer be programmed to act in the event of an unavoidable accident? Should it minimize the loss of life, even if it means sacrificing the occupants, or should it protect the occupants at all costs?

### **Artificial Intelligence.**

Google's AlphaGo computer learned to play Go at an expert level by watching people compete and then simulating millions of its own games against itself. It eventually became good enough to defeat even the best software that had been pre-programmed to play Go. In October, Google pitted AlphaGo against Fan Hui, the best player in Europe. They played five games. The computer won all of them. In 2016 AlphaGo defeated multiple world champion Lee Sedol, 4 games to 1.

Neuromorphic chips configured more like brains than traditional chips make computers far more astute about what is going on around them - computers that learn by experience. New generation robotics can react without pre-programming. The possibility of creating thinking machines raises a host of ethical issues relating both to ensuring that such machines do not harm humans and to the moral status of the machines themselves. If something goes wrong who is responsible - should it be the robot's programmer, designer, manufacturer, human overseer or his superiors?

## Issues of privacy and cybersecurity

Causes of loss of privacy;

- The monitoring of online activity, including browsing histories and use of social media.

**Consider.** In 2014, the world discovered that US security agency NSA had been spying on the communications of millions of its own citizens. In 2014 the UK government amended the Computer Misuse Act to provide a new exception for law enforcement and GCHQ to hack without criminal liability. This was done without public consultation or any debates over mass surveillance.

- The interception and reading of email messages.

One ethical problem that relates to the private communications of an individual involves the interception and reading of email messages which is often justified in terms of security.

- Distribution of databases storing personal information.

The individual may not be aware of the extent of the personal information being distributed, or who has access to the database, or whether the information is accurate.

**Consider.** Genome-based treatment, based on wider and cheaper availability of genome data, will provide new and personalised ways to fight life-threatening diseases, although privacy risks associated with data storage of genome data will invariably arise, particularly as such databases are often shared for security reasons (for example, between international police forces), increasing the possibility of hacking or abuse by authorities.

- Theft of private information by hackers.

In the handling and processing of private and personal information organisations are confronted with several ethical issues:

November 2014 – hackers leak a release of confidential data from Sony Entertainment, including personal information about employees and their families, salaries, internal emails and copies of unreleased films.

- Deciding the scope of personal and private information they can gather.
- The confidential treatment of such information.
- The accuracy of information – who checks that the information is correct.
- The purposes for which various categories of information may be used.
- The rights of a person - question of consent?

February 2016 – there was confrontation between Apple and the FBI over a dead terrorist's iPhone. The FBI wants Apple to write new code that would unlock an iPhone belonging to a dead terrorist. Apple is refusing, arguing that they should not be forced to weaken the iPhone's encryption in the name of national security, as this would compromise the privacy of its customers and the strength of its product security?

## Codes of conduct

A code of ethics, or code of conduct, defines acceptable behaviour within an organisation. Higher standards are generally promoted when a code of ethics is accepted and followed by members of an organisation. It is useful as individuals working for the organisation have a benchmark upon which they can judge their own behaviour and that of others.

### Informal and formal codes

Most small organisations do not have a formal written code of ethics and instead rely on senior members of staff to lead by example, showing what acceptable behaviour is. Members understand the informal code by observing how senior members conduct themselves, e.g. the type of language used in emails and behaviour towards clients.

Formal codes are written documents that outline expected behaviours within an organisation. Formal codes of ethics are usually enforced by the threat of disciplinary action should the code not be adhered to. Each code of ethics is different and usually reflects an

organisation's ethos, values and business style. Some codes are short and set out general guidelines, whereas other codes are large documents that include a variety of aspects relating to an organisation's values, ethics, objectives and responsibilities.

### **An individual's own personal code**

An individual's own personal code often supersedes the bare minimum requirements of an organisation's ethics code. An individual's own personal code will vary from person to person as they choose to act upon their own ethical standards in their everyday actions.

### **Legislation relevant to computing**

Many pieces of legislation govern the use of computer systems. Relevant examples of legislation include:

- Data Protection Act 1998
- Computer Misuse Act 1990
- Freedom of Information Act 2000

#### **Data Protection Act 1998**

The Data Protection Act 1998 (DPA) was put in place by the Government in response to growing concerns about the amount of personal data being stored on and processed by computer systems. Organisations that store and process personal data are required to register with the Information Commissioner, who is the person responsible for the DPA. Organisations must register information on the type of data they wish to store and why it is being collected.

Organisations are required to adhere to the eight principles of the DPA. These specify that personal data must be:

- processed against loss, theft or corruption
- accurate and where relevant kept up to date
- adequate, relevant, not excessive
- prevented from being transferred outside EU to countries without adequate provision
- fairly and lawfully processed
- processed within the rights of subjects
- deleted when no longer needed
- used only for the purpose collected

There are a number of exemptions from the DPA. These include:

- the prevention or detection of crime
- the capture or prosecution of offenders
- the assessment or collection of tax or duty
- personal data by an individual for the purposes of their personal, family or household affairs
- national security and the armed forces
- personal data that is processed only for journalistic, literary or artistic purposes
- personal data that is processed only for research, statistical or historical purposes
- personal data relating to an individual's physical or mental health
- personal data that consists of educational records or relates to social work
- personal data relating to human fertilisation and embryology
- adoption records
- statements of a child's special educational needs
- personal data processed for, or in connection with, a corporate finance service
- examination marks and personal data contained in examination scripts

### **Computer Misuse Act 1990**

When the use of computer systems became widespread, the Computer Misuse Act 1990 (CMA) was put in place to help combat issues arising from their misuse.

The CMA makes it an offence to:

- access data without permission, e.g. looking at someone else's files
- access computer systems without permission, e.g. hacking
- alter data stored on a computer system without permission, e.g. writing a virus that deliberately deletes data.

## **Freedom of Information Act 2000**

The main principle behind freedom of information legislation is that people have a right to know about the activities of public authorities, unless there is a good reason for them not to have this information.

Most countries operate some form of freedom of information law. In the UK it is the Freedom of information act 2000. The Act provides public access to information held by public authorities in two ways;

1. public authorities are obliged to publish certain information about their activities;
2. members of the public are entitled to request information from public authorities.

The Act covers any recorded information that is held by a public authority in England, Wales and Northern Ireland, and by UK-wide public authorities based in Scotland.

There are some exemptions, including information held for criminal investigations or relating to correspondence with the royal family and where disclosure may cause a specific type of harm, such as , endangering health and safety, prejudicing law enforcement, or prejudicing someone's commercial interests.

## **Environmental impacts**

The technologies we use every day consume a lot of resources and power and can present health hazards such as obesity and RSI arising from technology addiction.

Building the hardware can cause harm to the environment, including air, water, heat and noise pollution arising from manufacturing processes and the use of non-renewable resources, including precious metals such as gold used in circuitry.

Carbon emissions are released into atmosphere when electricity created from burning fossil fuels is used. Creating electricity takes a lot of resources, and it can be expensive to use it. It is sensible to reduce how much you use, by taking measures, such as:

- Turning off computers and peripherals when not in use.
- Adjusting the settings of your power options to help minimise power consumption.
- Choosing more energy efficient and environmentally friendly options;
  - Laptop computers use 75% less power than desktop machines.
  - Monitors account for up to half of the energy used by a computer, and the larger the monitor, the more power it uses.

- Ink jet printers use about 90% less energy than laser jets.
- Any product that earns the Energy Star label uses 30 to 75% less electricity than a standard product.

## **Landfill**

Old computers get thrown out when they become out dated. They contain all sorts of hazardous materials that need to be disposed of using special methods; otherwise the waste would become landfill.

Most electronics contain non-biodegradable materials, and heavy metals and toxic materials like cadmium, lead and mercury. Over time, these toxic materials can leak into the ground, where they can contaminate water, plants and the animals that live around the area. Many countries have banned technology products from landfills.

## **Increased populations**

The negative impacts of technology on society include increased pollution and the depletion of scarce resources. A further negative impact arises from improving health research helping people to live longer, resulting in increases in population.

This is good news for people in developed countries, but causes problems in developing countries that may not be in a position to access the health care benefits brought about by technology. In these countries mortality rates remain high, food is scarce and health care is poor.

## **To repair or re-cycling?**

Before throwing away old computers, or mobile devices, consider repair or re-cycling. There are many charities that will try to refurbish and repair old computer equipment and then donate the equipment to worthwhile causes, either at home, or abroad.

Repair pro-longs the life of the equipment, delaying the need to manufacture a replacement. Re-cycling is also an environmentally friendly solution that allows components, such as precious metals, to be retrieved and re-used.

Before donating a machine for repair or re-cycling it is important to remove all of your files and data from it. The recycle bin only partially removes the information - you need to run a special program that erases your hard drive.

## **Paper and packaging waste**

You can reduce waste paper by thinking twice about printing documents, email messages, pictures, and things you find on the Web. Buy paper that is made from recycled products and recycle the paper that you do use. Software companies are reducing their waste by offering their products as an online download instead of selling it in a box.

## **Positive impacts of technology on the Environment**

Advances in computer technology have produced many positive impacts on society, including;

- In the development of new materials and processes that are sustainable and do not harm the environment.
- Enabling the study of our environment to better understand how it works and the impact of our actions on it
- Smarter technologies that respond to how we use them and adjust themselves to reduce their environmental impact
- Helping experts from all fields share their research, experience and ideas to come up with better solutions.
- Communications that reduce the environmental impact people would normally cause from traveling
- Improved education, including distance learning and visual learning using integrated technologies.



# Component 2

## 1. Problem solving

### Problem solving

Programmers can only write programs for computers to solve problems if they understand how to solve the problem themselves. This means that a programmer has to have a toolkit of methods that they can use to consider the ways in which the problem can be solved.

To write successful problem solutions programmers must develop good computational thinking skills that will help them break the problem down into manageable chunks and simplify the situation.

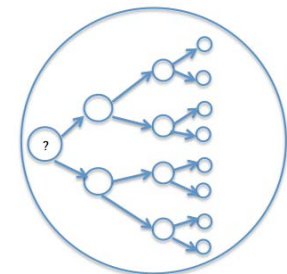
### Computational thinking

To be able to represent a problem as a set of steps that can be carried out by a computer requires good computational thinking skills. These steps could be presented as algorithms.

### Decomposition

Programmers use a technique called decomposition to break a large program down into a series of sub problems.

The starting point is to decompose the large problem so that each sub problem is described in the same level of detail and can be solved independently from the other sub problems. The solutions to the sub problems can then be brought together to provide a solution to the whole problem.



One of the advantages of decomposition is that different people can work on different sub problems. However, a disadvantage would be that the solutions to the sub problems might not come together to provide a solution to the whole problem.

### Example problem

A school has decided to introduce a smart card system for its pupils to pay for food and drinks in the school's dining hall. It has asked a programmer to design the system for the school.

The whole problem is the introduction of the smart card system but the programmer must break the problem down into manageable chunks. The system must be able to:

- Allow parents to add money to their child's smart card
- Allow pupils to pay for their food and drink

- Update the amount of money remaining on the card

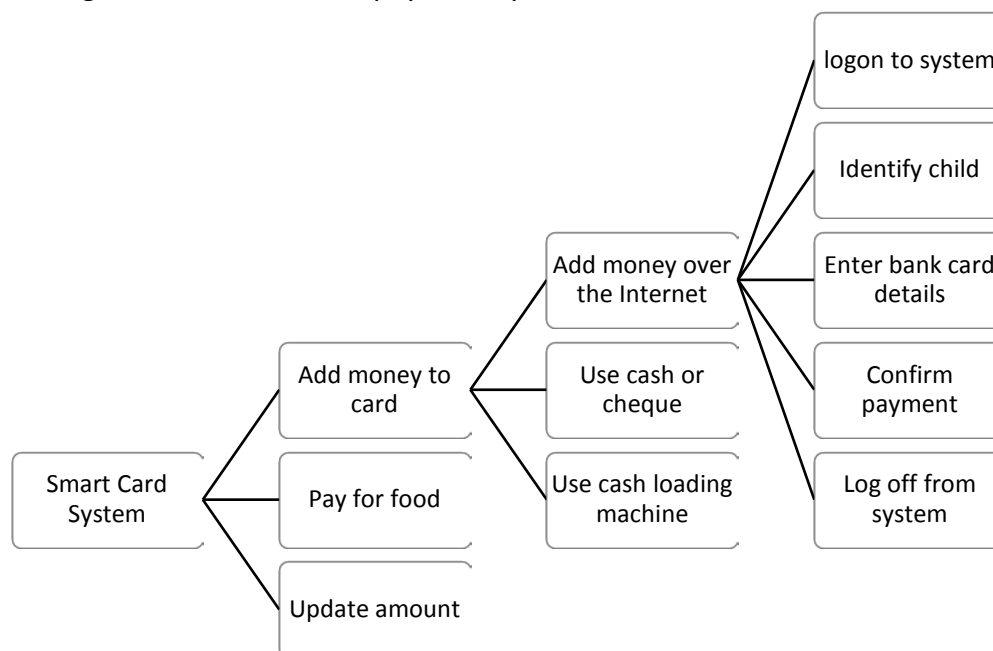
So now the programmer has three sub problems but each of the three is quite complex and can be broken down further.

Allow parents to add money to their child's card can be broken down into:

- Add money over the Internet
- Add money by the pupil taking cash or a cheque to the school office
- Add money by the pupil using a cash-loading machine in the dining hall.

Add money over the Internet can be broken down into:

- Logon to the school's payment system
- Identify the child who uses the smart card
- Enter the details of the bank card to be used for the payment
- Confirm the payment
- Logoff from the school's payment system

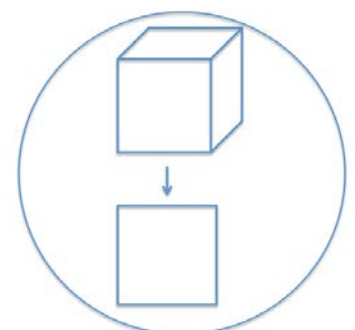


This process has broken part of the problem into distinct sub problems that can be solved and combined to form part of the solution to the original problem.

### Abstraction

Abstraction is a technique to reduce something to the simplest set of characteristics that are most relevant to solving the problem.

The programmer has to concentrate on the most important aspects of the problem without worrying about fine details.

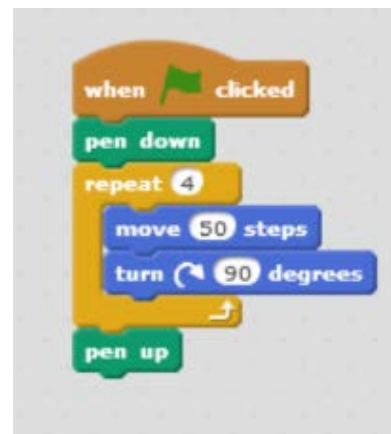


### A simple example of abstraction

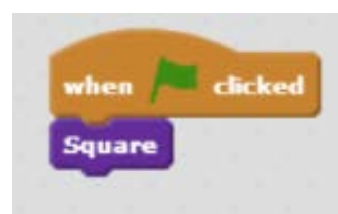
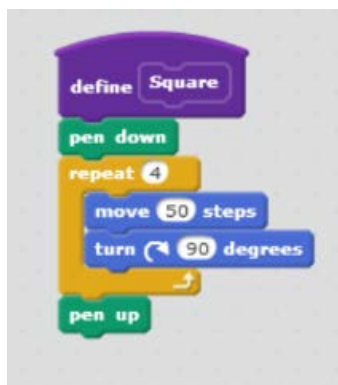
I want to create a program to draw a square in Scratch. The square must have four sides of 50 units with an angle of  $90^\circ$  between each side. The code in Scratch looks like this:



There are a lot of repeated commands in the code. It can be simplified by using a loop for the repeating code.



This code can be defined as a procedure that can be called whenever it is needed.



So instead of considering a series of issues or commands the problem has been simplified down to one procedure without all the unnecessary details that have been hidden.

## Conclusion

Decomposition and abstraction are two ways of computational thinking that are used to analyse complex problems. Computational thinking allows us to understand a complex problem and to present our analysis in a way that a human being can use to create a computerised solution.

## Variables

In computer programming, a variable may be required to store data that can change. Variables are given identifiers (names) that should reflect the data being stored in them. An example of a variable is `txt_FirstName`. This variable has a **self-documenting identifier**, which implies the type of data being stored in it is text containing a person's first name.

## Constants

Constants are used in computer programming to store data that does not change. Constants are also given self-documenting identifiers that should reflect the data being stored in them. An example of a constant would be `Pi = 3.14`, as this is a self-documenting identifier that does not change.

## Variables such as: local, global, static, dynamic,

### Local and global variables

A **local** variable is a variable that is defined within a sub procedure and as such is only accessible from within that same sub procedure. This is known as its *scope*. A **global** variable is a variable that has a larger scope as it is defined globally and is therefore accessible from anywhere within a program. The advantage of defining a local variable over a global variable is that it is easier to track the changes to a variable and the reason for the changes when it is only used within a sub procedure. An advantage of defining a global variable over a local variable is that sometimes it can be the most efficient way of ensuring that an important piece of data is accessible to all sub procedures, e.g. the details of the user currently logged into a program.

Below is an example of an algorithm that calculates the area of a circle:

```

1 Pi = 3.142
2 Radius is real
3
4 declare subprocedure FindArea {procedure to calculate the area fo a circle}
5 Area is real
6
7 start
8     Area = PI * Radius * Radius
9 end
10
11 startmainprog
12     output "Type in the radius"
13     input Radius
14
15     call FindArea
16
17     output "The area is ", Area
18 endmainprog
19

```

An example of a global variable here is *Radius* as it can be accessed throughout the entire program. An example of a local variable is *Area*. It is a local variable as it has been defined within the sub procedure called *FindArea*.

Can you spot the error with the following line from the algorithm above?

```
17     output "The area is ", Area
```

How would you amend the algorithm to correct this problem?

### Static and dynamic variables

Each time a program is run, **static** variables are stored in a location in memory and have a lifespan that lasts the entire time that program is running. For example, if you assign the number 2.14 to a static variable (and do not change it), the static variable will still contain 2.14 the next time you run the sub procedure and until the whole program stops running.

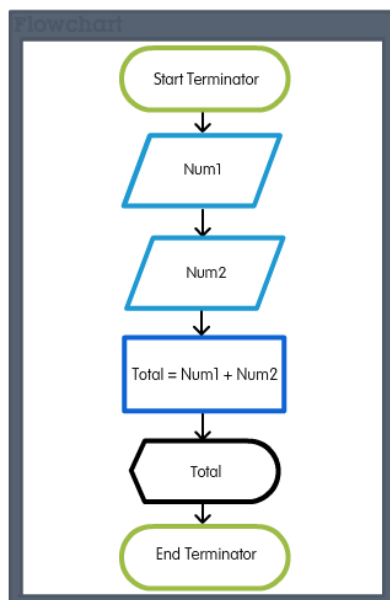
Each time a **dynamic** variable is defined, it is assigned a new location in memory and has a lifespan that ends when the sub procedure in which it was defined ends. For example, if within a sub procedure you assign the number 2.14 to a dynamic variable (and do not change it) and the sub procedure ends, the next time you run that sub procedure the dynamic variable will no longer contain any value.

## 2. Algorithms and programming constructs

In programming an algorithm is a set of instructions that can be used to solve a given problem. The instructions must be clear and in the correct order to produce the solution. These instructions will be the starting point for writing a computer program to solve the given problem.

An algorithm should not be in computer code. It should be presented in a way that it can be used in different programming languages. We are going to present algorithms as pseudo code or flowcharts. Pseudo code is a way of writing instructions in plain English. A flowchart is a diagram showing the instructions to be carried out in the order they should be carried out.

Here is an algorithm to add together two numbers and output their total shown as a flowchart and in pseudo code.



```

Pseudocode
Num1 as integer
Num2 as integer
Total as integer
Input "Enter 1st number", Num1
Input "Enter 2nd number", Num2
Total = Num1 + Num2
Output "Total is", Total
  
```

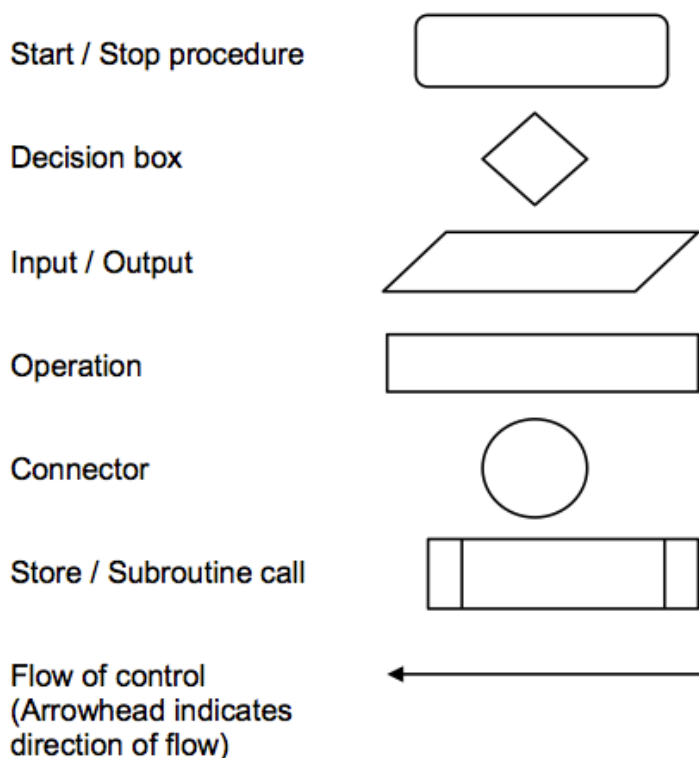
A good algorithm should:

- Be finite – if an algorithm never ends trying to solve the problem then it is not going to be of use.
- Have well defined instructions – each step of the algorithm should be clear so that it can be carried out exactly as intended.
- Be effective – the algorithm should give the correct result/solution to the problem.

We are going to use the following rules for pseudo code and flow charts.

Construct	Example usage
Declare subroutines	Declare CapitalLetterOfName End Subroutine
Call a subroutine	call SubroutineNeeded
Declare and use arrays	myarray[99]
Literal outputs	output "Please enter a number"
Variable names	myvariable
Define variable data type	myvariable is integer
Data types	integer, character, string, boolean
Assignment	set counter = 0
Selection	if . . . else . . . end if
Indent at least single space after if or do or repeat etc.	if counter = 1 output counter end if
Annotation	{Some annotation goes here}
Comments (for Java only)	/** Comments for Java */
Repetition	for i . . . next i repeat . . . until do . . . loop do . . . while while . . . repeat

Algorithms represented using a flowchart will use the following convention:



## Sequence, selection and iteration

These are the three basic constructs that we use to design algorithms.

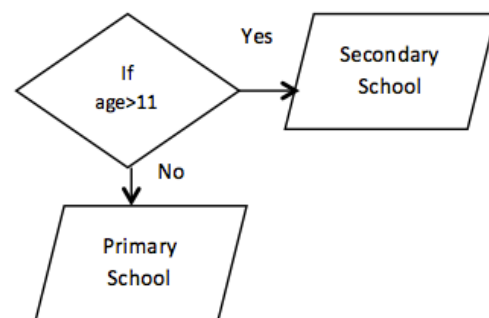
Algorithms consist of a series of instructions in a specific order. This is the order or **sequence** in which the instructions must be carried out for the algorithm to work. A computer can only follow instructions in the order they are given. If the sequence is not right the computer will still follow the order in which the instructions are given.

A **selection** instruction is one where a decision must be made. There are times when an instruction in an algorithm may give different options.

```

1
2 if age > 11 then
3     print "Secondary school"
4 else
5     print "Primary School"
6 end if
7
8

```



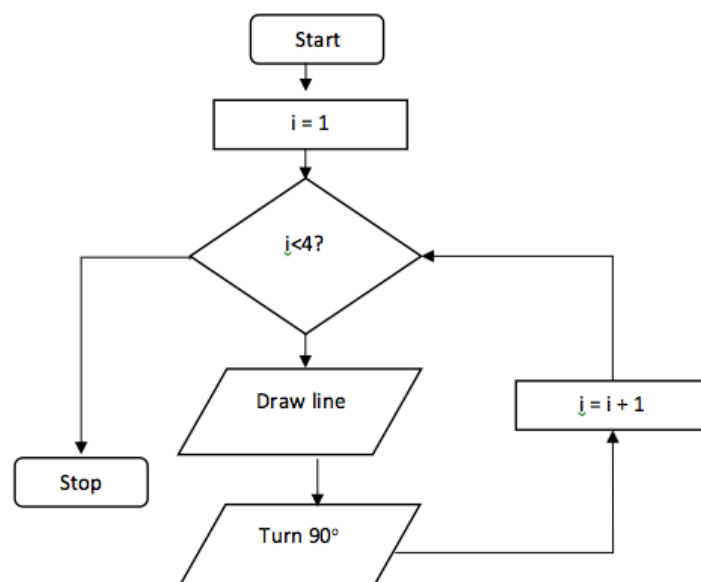
Sometimes an algorithm will require a set of steps to be carried out more than once or many times. This is called **iteration** and often referred to as a loop in the program.

If we want a program to repeat a set of two instructions four times we can use a 'for...next' loop.

```

1
2 for i = 1 to 4
3     draw line
4     turn 90
5 next i
6

```





## Using count and rouge values with loops

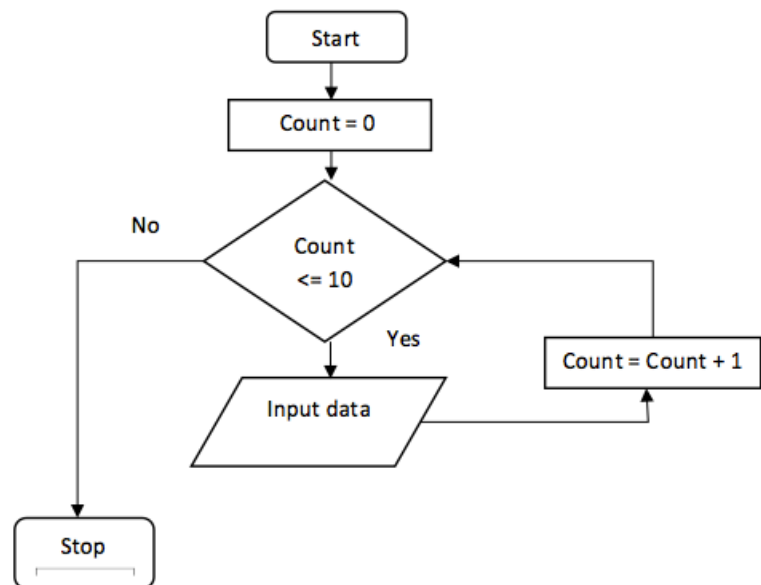
Eventually all loops must be terminated. Sometimes we will not know how many times we will need the instructions in the loop to be used. If this is the case we can control the program by using a count or a rouge value.

Count will record the number of times a process is carried out. When the count reaches the required number the loop will terminate.

```

2 count = 0
3
4 repeat
5     input data
6     count = count + 1
7 until count = 10
8
9

```

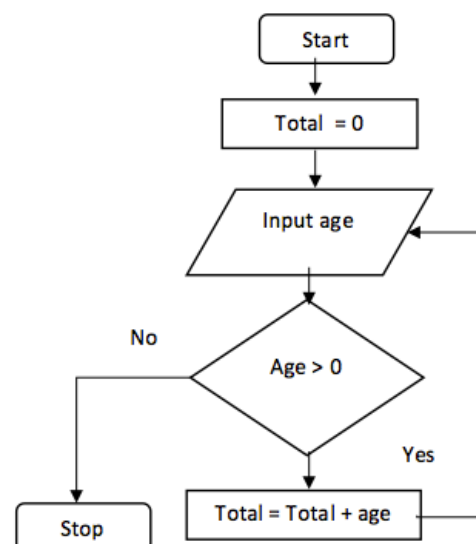


A rogue value is a value that falls outside the range of possible values for the data that is being processed. If we were calculating the average age of a class of children, we could set a rogue value of -1 to stop the loop as no child can have a negative age.

```

1
2 total = 0
3
4 repeat
5     input age
6     if age > 0 then
7         total = total + age
8     end if
9 until age = -1
10
11

```



Why is the 'if ...then' statement included in the algorithm?

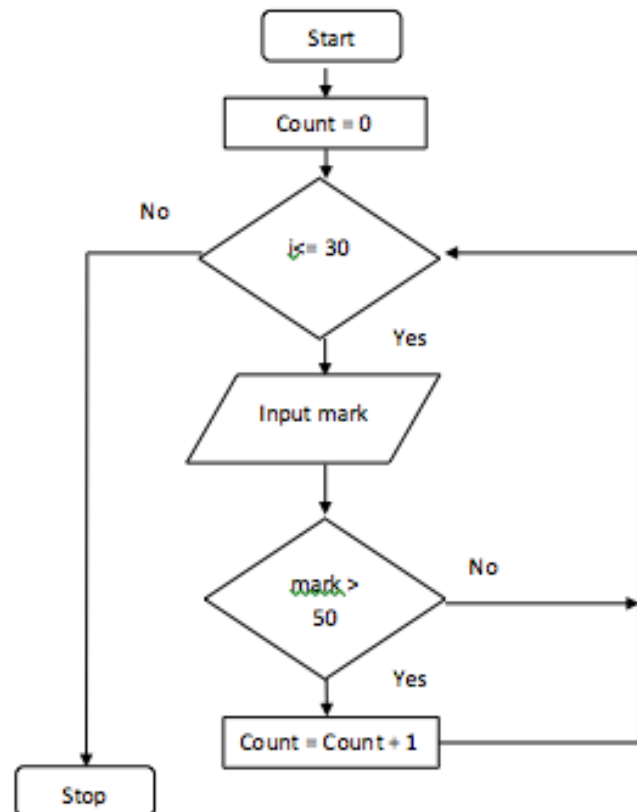
A count can also be used to check whether a condition has been met.

A set of 30 test results is to be entered into a program. Each time the program is run the teacher wants to know how many pupils had marks over 50.

```

1
2 count = 0
3
4 for i = 1 to 30
5   input mark
6   if mark > 50 then
7     count = count + 1
8   end if
9 next i
10

```



## String handling

We are going to use the following rules for pseudo code for string handling.

Construct	Example usage
Variable names	myvariable
Define variable as string	myvariable is String
Length of string	len(string)
Mid String where x is the offset and y the length	mid(string, start, length)
Replace part of a string	Replace(string, find, replacewith)
Compare two strings	strComp(string1, string2)

### Creating strings

To create a string we need to define a variable as a string and assign a value to the variable.

```
greeting is String
greeting = "Hello from me!"
print greeting
```

```
Hello from me!
```

### Measuring the length of a string

To find out how many characters are in a string we used the 'len' command. Any keyboard stroke is counted as a character so spaces and special characters like an exclamation mark are counted as well as letters and numbers.

```
greeting is String
greeting = "Hello from me!"
length = len(greeting)
print length
```

```
13
```

### To return all or part of a string

To discover the contents of all or part of a string we have to use the 'mid' command. Normally you would use the 'len' command to find out how long the string is before you would use the 'mid' command.

```
txt is String
partMessage is String
txt = "Have a happy birthday"
partMessage = mid(txt, 8, 5)
print partMessage
happy
```

### Replace part of a string

To replace part of a string you need to use the 'replace' command.

```
txt is String
message is String
txt = "Have a happy birthday"
message = replace(txt, "happy", "fantastic")
print message
```

```
Have a fantastic birthday
```

```
txt is String
message is String
txt = "Have a happy birthday"
message = replace(txt, "a", "xx")
print message
```

```
Hxxve xx fxxntxxstic birthdxxxy
```

### To compare two strings

To compare the contents of two strings we use the 'strCompare' command. This command will turn a value of 0 (true) if the two strings are identical and a value of -1 (false) if they are not.

```
txt1 is String
txt2 is String
response = integer
txt1 = "Hello"
txt2 = "Hello"
response = str(Compare, txt1, txt2)
print response
```

0

Sometimes we just want to compare if the content is identical and not the use of upper and lower case letters. We can add another parameter (1) to the 'strCompare' command to carry out a textual comparison.

```
txt1 is String
txt2 is String
response = integer
txt1 = "Hello"
txt2 = "hello"
response = str(Compare, txt1, txt2,1)
print response
```

1

### To join strings together and add text

The process of combining a string with text or combining two strings with or without additional text is called concatenation. The example below combines someone's first name and surname with a welcome message.

```
txt1 is String
txt2 is String
concatString is String
txt1 = "Sarah"
txt2 = "Smith"
print "Welcome " & txt1 & txt2
```

Welcome Sarah Smith



### INTERESTING FACT

Data takes longest to sort when it is in the reverse order of the order that is required. For example if data are sorted into ascending data when it needs to be sorted into descending order.

## Sorting

A sorting algorithm will put items in a list into a particular order, which may be alphabetic or numeric. As sorting a large list of items can be timely computer algorithms have been developed to do the work for us.

### Merge Sort

The merge sort is a sorting technique based on the idea of 'divide and conquer'. A merge sort first divides a list into two equal halves and then combines them in a sorted list.

To understand how the merge sort works we will use a merge sort algorithm to sort the following list:

37	14	10	27	24	19	44	35
----	----	----	----	----	----	----	----

We need to divide the list into two equal halves.

37	14	10	27
----	----	----	----

24	19	44	35
----	----	----	----

We then split the halves into half

37	14
----	----

10	27
----	----

24	19
----	----

44	35
----	----

We keep dividing the list until each list is a single item.

37
----

14
----

10
----

27
----

24
----

19
----

44
----

35
----

We then combine them in the same way they were divided but in order.

14	37
----	----

10	27
----	----

19	24
----	----

35	44
----	----

In the next move (iteration) we combine the lists to make lists of 4 sorted items.

10	14	27	37
----	----	----	----

19	24	35	44
----	----	----	----

After the final iteration the list should be fully sorted.

10	14	19	24	27	35	37	44
----	----	----	----	----	----	----	----

## The merge sort algorithm

```

1  Declare MergeSort
2  mergelist [99]
3  if len(mergelist)>1 then
4      mid = len(mergelist)/2
5      lefthalf = mergelist(0:mid)
6      righthalf = mergelist(mid-1:end)
7
8      MergeSort(lefthalf)
9      MergeSort(righthalf)
10
11     i = 0
12     j = 0
13     k = 0
14     while i<len(lefthalf) and j< len(righthalf)
15         if lefthalf[i]< righthalf[j]
16             mergelist[k] = lefthalf[i]
17             i = i+1
18         else
19             mergelist[k] = righthalf[j]
20             j = j+1
21         end if
22         k = k+1
23     end while
24
25     while i < len(lefthalf)
26         mergelist[k] = lefthalf[i]
27         i = i+1
28         k = k+1
29     end while
30
31     while j < len(righthalf)
32         mergelist[k] = righthalf[j]
33         j = j+1
34         k = k+1
35     end while
36 end Subroutine
37
38 mergelist = [37,14,10,27,19,24,35,44]
39 MergeSort(mergelist)
40 Print mergelist
41

```

## The bubble sort

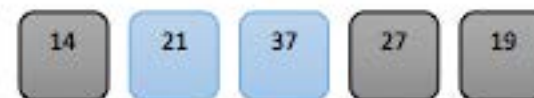
The bubble sort is a simple sorting algorithm. The algorithm is based on the comparison of adjacent data items, swapping them if they are not in the correct order. The bubble sort is not suitable for large sets of data.

### How the bubble sort works

The bubble sort starts with comparing the first two items, in this case 37 and 14. They are out of order so they need to be swapped over.



Next we compare the 37 with 21. These two items need to be swapped.



Now we compare 37 with 27. Again these two items need to be swapped.



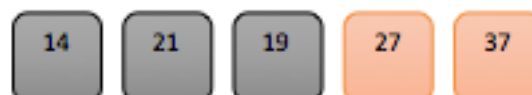
Comparing 37 with 19 means that the two items need to be swapped again.



After the first set of comparisons, known as an iteration, the list looks like this:

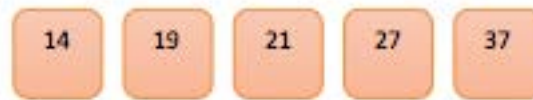


After the second iteration it will look like this:





Notice that after each iteration at least one item has moved to the correct position. After the third iteration the list will look like this.



When an iteration results in no swaps being needed then the bubble sort is stopped.

### The bubble sort algorithm

```

1 Declare BubbleSort (bubbleList) :
2     exchanges = True
3     passnum = len(bubbleList)-1
4     while passnum > 0 and exchanges = True
5         exchanges = False
6         for i = 1 to n
7             if bubbleList[i]>bubbleList[i+1]:
8                 exchanges = True
9                 temp = bubbleList[i]
10                bubbleList[i] = bubbleList[i+1]
11                bubbleList[i+1] = temp
12            end if
13        next i
14        passnum = passnum-1
15    end while
16 end Subroutine
17
18 bubbleList=[20,30,40,90,50,60,70,80,100,110]
19
20 BubbleSort (bubbleList)
21 print (bubbleList)
22

```

## Searching

Searching for data is one of the basic operations of computing.

### Linear search

The linear search is a very simple search algorithm. Each item in the data set is compared with the search condition in sequence until the item is found or the end of the data set is reached. As the items are compared in sequence the linear search is sometimes known as the sequential search. The items in the list are not in any particular order within the data set.

```

1  Declare linearSearch(dataList, searchItem)
2
3  position = 0
4  found = false
5
6  while position < len(dataList) and found = false
7      if dataList[position] = searchItem then
8          found = true
9      else
10         position = position + 1
11     end if
12 end while
13
14 testList = [1,3,21,45,57,17,34,65]
15 linearSearch(testList, 45)
16 linearSearch(testList, 20)
17

```

The test data included at line 15 would produce a successful search for the search item, 45. The test data at line 16 would be unsuccessful as the search item, 20, is not included in the test data.

The linear search is not an efficient search. We can measure the efficiency of a search by considering the number of comparisons that are made before the required item is found. The items in the list are not ordered so the probability that the item we are looking for is in any particular position is exactly the same for each position in the list. This means that we may have to compare every item before we find the required one in the last position. We would also have to search the entire list before we discovered that the item was not included in the list.

### Binary search

A more efficient method of searching a list of data is an algorithm known as the binary search.

For a search to be more efficient it is necessary to sort the data into order. Once the data is in order you can adopt methods that would not work on unordered data. Rather than

starting the search at the beginning of the list, the search starts at the mid point of the list. The data item is compared with the search item with three possible outcomes:

- The item at the mid point will match the search item
- The search item will be less than the item at the mid point
- The search item will be greater than the item at the mid point.

If the search item is less than the item at mid point then all the items after the mid point can be ignored. Similarly if the search item is greater than the mid point then the first half of the list can be discarded.

This process is then repeated on the remaining data time after time until the required item is found.

The list has been sorted. We are going to search this list for the number 31.

11	17	19	23	29	31	37	41	43
0	1	2	3	4	5	6	7	8

To calculate the mid point we use this formula:

$$\text{mid point} = \text{low} + (\text{high} - \text{low})/2$$

$$\text{mid point} = 0 + (8 - 0)/2$$

$$\text{mid point} = 4$$

With the mid point being 4 we need to compare our search item (31) with the data at position 4 (29).

The search item is greater than the data at position 4 we can discard the first part of the list.

11	17	19	23	29	31	37	41	43
0	1	2	3	4	5	6	7	8

We need to recalculate the mid point:

$$\text{mid point} = 5 + (8 - 5)/2$$

$$\text{mid point} = 6.5$$

In this situation the mid point is taken as the integer part of the result so the mid point will be 6.

The search item is less than the data item at position 6 so we can discard the from the mid point upwards.

11	17	19	23	29	31	37	41	43
0	1	2	3	4	5	6	7	8

When we recalculate the mid point:

$$\text{mid point} = 5 + (5 - 5)/2$$

$$\text{mid point} = 5$$

The search item is now equal to the item at position 5. We have had to make 3 comparisons to find the required data. This is far more efficient than a linear search. It is important to remember that our examples have limited data. If you were searching thousands of data items the difference between the two methods would be significant.

## The binary search algorithm

```
1 Declare binarySearch(dataList, searchItem):
2   if len(dataList) = 0
3     return False
4   else
5     midpoint = low + (high - low)/2
6   end if
7   if dataList[midpoint]= searchItem
8     return True
9   else:
10    if item < dataList[midpoint]:
11      return binarySearch(dataList[low:midpoint],searchItemtem)
12    else:
13      return binarySearch(dataList[midpoint+1:high],searchItem)
14    end if
15  end if
16
17  testlist = [11,17,19,23,29,31,37,41,43]
18  print(binarySearch(testlist, 23))
19  print(binarySearch(testlist, 13))
20
21
```

# 3. Programming languages

## HyperText Markup Language

### Why HTML is important as a standard for web page creation

HTML, which is an abbreviation of HyperText Markup Language, is a standard used when creating web pages. This is discussed in further detail on page 60.

Web standards, such as HTML, are important as the development of web pages is simplified, as web programmers will be able to understand another developer’s code. It is also important for the end users of web pages as following standards ensures that different web browsers are able to display web pages in the way in which they were intended.

### Use HTML tags and their closures

HTML tags commonly come in pairs, such as <html> and </html> or <b> and </b>. The first tag in a pair is called the opening tag and the second tag is called the closing tag. Between these tags, programmers can add text, more tags, comments and other types of text-based content.

The purpose of a web browser, such as Internet Explorer, Google Chrome and Safari is to read HTML code and render it on screen. The browser does not display the HTML tags, but instead uses the tags to interpret the content of the page.

**INTERESTING FACT**

Browser use statistics (March 2016)

- Chrome 69.9%
- Internet Explorer 6.1%
- Firefox 17.8%
- Safari 3.6%
- Opera 1.3%

- The text between the <html> and </html> tags describes the web page.
- The <head> element is a container for all the head elements. The head elements can include a title for the document, scripts, styles and meta information.
- Meta elements are used to describe the page, include key words, author of the document and when last altered.
- The text between <body> and </body> tags includes the contents of the web page.
- The <p> tag define a paragraph. Web browsers will automatically add some space before and after each <p> element.
- When an element within a web page is hyperlinked, it is placed within the <a href="url"> and </a> tags. For example, <a href="http://www.eduqas.co.uk">Eduqas</a> will be displayed as [Eduqas](http://www.eduqas.co.uk).
- The <img> tag is slightly different, as it does not contain a closing tag. For example,  will display the image file logo.gif.

The table below shows how unformatted text will look when placed within the commonly used formatting tags.

Command	Opening tag	Closing tag	Unformatted text	Formatted text
Headings	<h1> to <h6>	</h1> to </h6>	Computer science	<b>Computer science</b>
Bold	<b>	</b>	Computer science	<b>Computer science</b>
Italic	<i>	</i>	Computer science	<i>Computer science</i>
Centre align	<center>	</center>	Computer science	Computer science
Underline	<u>	</u>	Computer science	<u>Computer science</u>

### Bulleted list

To create a bullet pointed list we need to use the tag for an unordered list, <ul>. For each line in the list we use the <li> tag.

For example:

```

<ul>
  <li>Apples</li>
  <li>Oranges</li>
  <li>Pears</li>
</ul>

```

will display as:

- Apples
- Oranges
- Pears

### **Blockquote**

The Blockquote tag is used to specify a section that is quoted from another source. Web browsers indent `<blockquote>` elements.

The horizontal rule tag, `<hr>`, is used to separate content in an HTML page.

For example:

```
<h2>HTML</h2>
<p> We use HTML to control the way a web page looks</p>
<hr>
<h3>XHTML</h3>
<p>XHTML stands for eXtensible HyperText Markup Language</p>
```

will display as:

## **HTML**

We use HTML to control the way a web page looks

### **XHTML**

XHTML stands for eXtensible HyperText Markup Language

Here is an example of how original text is formatted using HTML tags.

Original text
<p>For Sale</p> <p><b>Bluetooth Hands Free Car Kit</b></p> <p>Make calls without wearing a headset with this Bluetooth v1.2 EDF Multipoint Hands-free Speakerphone! Visit <a href="http://www.edfweb.com">www.edfweb.com</a> to see. Simply pair this device to any Bluetooth enabled phone and talk hands-free today!</p>

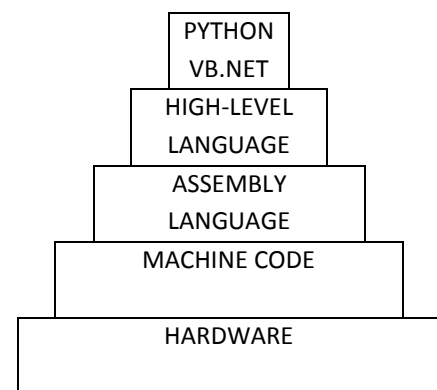
HTML
<pre>&lt;html&gt; &lt;body&gt; &lt;h1&gt;&lt;center&gt;For Sale&lt;/center&gt;&lt;/h1&gt; &lt;p&gt; &lt;b&gt;Bluetooth Hands Free Car Kit&lt;/b&gt;&lt;/p&gt; &lt;p&gt;Make calls without wearing a headset with this Bluetooth v1.2 EDF Multipoint Hands-free Speakerphone!&lt;/p&gt; &lt;p&gt;Visit &lt;a href="http://www.edfweb.com/"&gt;www.edfweb.com&lt;/a&gt; to see.&lt;/p&gt; &lt;p&gt;&lt;i&gt;Simply pair this device to any Bluetooth enabled phone and talk hands-free today! &lt;/i&gt;&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>

Formatted web page
<h2>For Sale</h2> <p><b>Bluetooth Hands Free Car Kit</b></p> <p>Make calls without wearing a headset with this Bluetooth v1.2 EDF Multipoint Hands-free Speakerphone!</p> <p>Visit <a href="http://www.edfweb.com">www.edfweb.com</a> to see.</p> <p><i>Simply pair this device to any Bluetooth enabled phone and talk hands-free today!</i></p>



## Assembly Language

Assembly language is a programming language that is once removed from machine code. Machine code is made up of 0s and 1s and is extremely difficult for a programmer to use. Assembly language has the same structure and instruction set as the commands in machine code but they use mnemonics (names) rather than binary code.



Each type of computer has its own machine code and assembly language. This means that a program written in machine code for one CPU will not run on another type of CPU.

Initially computer programs were all written in low level languages but over time high level languages have been developed that are closer to written English and easier to use.

However, assembly language programs:

- Require less memory and execution time
- Allow code to interact directly with hardware, such as device drivers
- Suitable for time-critical processes.

Instruction	mnemonic	What does it do?
Input	INP	Inputs a value and stores in the accumulator
Output	OUT	Displays the contents of the accumulator
Store	STA	Transfers a number from the accumulator to RAM
Load	LDA	Transfers a number from RAM to the accumulator
Add	ADD	Adds the contents of the accumulator to the contents of a RAM address
Subtract	SUB	Subtracts the contents of the accumulator from the contents of a RAM address
Branch	BRA	Jump to the RAM location specified – used for loops
End/Stop/Halt	HLT	Stops the processor
Data definition	DAT	Defines variables

## Some example programs

This is an assembly language program to add together two numbers.

INP	'input a number
STA 99	'store the number in memory slot 99
INP	'input a number
ADD 99	'add this number to the number stored in memory slot 99
OUT	'output the result of the addition
HLT	'halt/stop/end

This is an assembly language program for subtraction.

INP	'input a number
STA first	'store the number a variable called first
INP	'input a number
STA second	'store the number in a variable called second
LDA first	'load the number in variable first into the accumulator
SUB second	'subtract the contents of second from the accumulator
OUT	'output the number in the accumulator
first DAT	'declare first as variable
second DAT	'declare second as variable

## Object oriented programming

An object-oriented programming (OOP) can be viewed as a collection of objects that communicate with each other.

To understand OOP we need to know about objects, classes, methods and instance variables.

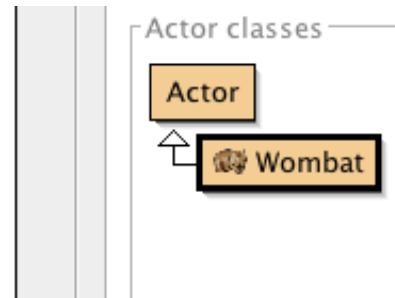


**Object** – an object has states and behaviours. Let's think about the little wombat from Greenfoot.

The wombat has states – name and breed.

It also has behaviours – grunting, burrowing and moving.

**Class** – a class is used to describe one or more objects. The class is a plan or template for creating objects within a program. Each object is created from a single class but a class can be used to create many objects. Each wombat in a scenario will be class instance (object).



Class names must always begin with an uppercase letter.

**Methods** – a method is behaviour. One class can contain many methods such as the wombat grunting, burrowing and moving. Method names must always begin with a lowercase letter.

**Instance variables** – instance variables are variables that are bound to class instances. Imagine that you have created a Greenfoot games in which wombats eat leaves. Each wombat will have an instance variable called leaves. This variable will record how many leaves each wombat has eaten. Wombat1 could have eaten 4 leaves and Wombat2 6 leaves. These are both values of the instance variable but independent of each other.

### Properties of OOP languages

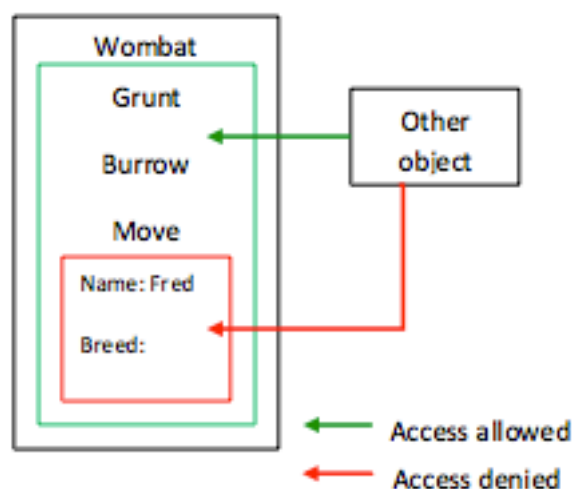
OOP languages have four main properties that reduce the amount of code required in a program. They are:

**Encapsulation** – this is the process of wrapping data and the code that operates on it into a single entity. The variables and methods are wrapped up inside the class.



**Abstraction** – this is the process of hiding non-essential features and showing the essential features.

The wombat's states are hidden from other objects but its behaviours are not.

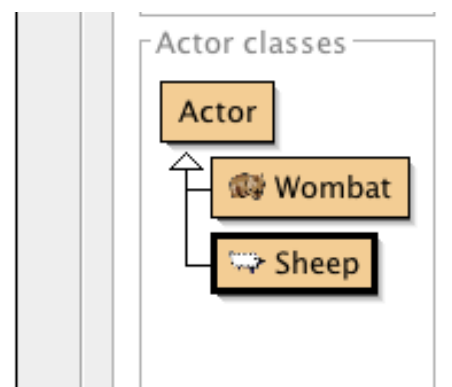


## Inheritance

Inheritance is a feature of OOP. Inheritance allows a new class to use the properties and methods of another existing class. The new (derived) class inherits the states and behaviours of the existing (base) class. The derived class is also called the subclass and the base class is called the superclass. In our example Actor is the superclass and Wombat is the subclass.

The subclass can have its own variables and methods in addition to those inherited from the superclass. A superclass can have many subclasses but a subclass can inherit from only one superclass.

**Polymorphism** – this concept allows actions to act differently based on the object performing the action.



One of the wombat's behaviours or actions is to grunt when it makes a sound. We will

call this action `makeSound()`. The sheep will bleat (Baaaa!!) when it makes a sound. We can also call this action `makeSound()`. The `makeSound()` action makes the wombat object grunt and the sheep object bleat. The action acts differently depending on the object.

## WHAT DO YOU NEED TO BE ABLE TO CODE IN GREENFOOT?

You should be able to:

- Create new and extend existing classes
- Create new and edit existing objects
- Create new and edit existing worlds
- Write and invoke methods
- Change existing methods
- Create new and edit existing properties (including public, private, static, etc.)
- Add and remove objects from worlds
- Use actors
- Move objects around a world
- Keyboard input
- Add and play sounds
- Implement and use parameter passing (by value and by reference)
- Access one object from another
- Implement object collision detection
- Implement random number generation
- Use the concept of inheritance and encapsulation.

## 4. Data structures and data types

### USING DATA STRUCTURES AND DATA TYPES

You should be able to:

- Create new data structures including:
  - One-dimensional arrays
  - Two-dimensional arrays
  - Files and records
- Implement data types including:
  - Integer
  - Boolean
  - Real
  - Character
  - String
- Assign, identify and explain the use of constants and variable in algorithms and programs
- Describe the scope and lifetime of variables in algorithms and programs.

## 5. Security and authentication

### USING SECURITY TECHNIQUES

You should be able to:

- Use appropriate security techniques such as usernames and passwords in algorithms and programs
- Use appropriate validation techniques.