COMPUTER SCIENCE KNOWLEDGE ORGANISER

Algorithms

An **algorithm** is a sequence of ordered instructions that are followed step-by-step to solve a problem. This does *not* need to be on a computer.

Decomposition is the breaking down of a complex problem into smaller more manageable problems that are easier to solve.

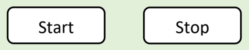
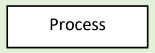
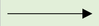
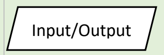
Abstraction allows us to remove unnecessary detail from a problem leaving us with only the relevant parts of a problem thereby making it easier to solve.

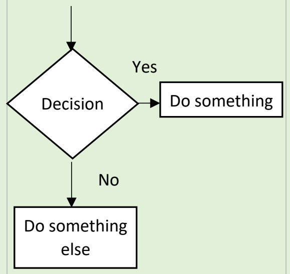
Algorithm Efficiency More than one algorithm can be used to solve the same problem. Normally we use the algorithm that solves the problem in the quickest time with the fewest operations or makes use of the least amount of memory.

Dry run testing is carried out using **trace tables**. The purpose of the trace tables is for the programmer to track the value of the variables and outputs at each step of the program and to track how they change throughout the running of the program.

Flowchart Symbols

We can represent algorithms using flowcharts

Start and Stop 	**Process – An operation that the algorithm performs** 
Connector – Links all the other symbols together 	**Input and Output of data that is read in and written out** 

Decision is the same as a selection (if then … else)

	`IF answer is "yes" THEN`
	`        do something`
	`ELSE IF answer is "no"`
	`        do something else`
	`ENDIF`

Pseudocode

We can represent algorithms using pseudocode

	Example	Python equivalent
Variable assignment	a ← 10	`a = 10`
Constant assignment	constant PI ← 3.142	`PI = 3.142`
Input	a ← USERINPUT	`a = input()`
Output	OUTPUT "Bye"	`print("Bye")`

Arithmetic Operators
 	
Add
Multiply
Divide
Subtract
Integer division
Modulus (remainder)
	
	+
	*
	/
	-
	a ← 7 DIV 2
	a ← 7 MOD 2
		
		+
		*
		/
		-
		`a= 7 // 2`
		`a = 7 % 2`

Relational Operators Less than Greater than Equal to Not equal to Less than or equal to Greater than or equal to	 < > = ≠ or <> ≤ ≥	 < > == != <= >=
Boolean Operators AND OR NOT	 AND OR NOT	 AND OR NOT

Selection
 	
if ..
	
	`IF i > 2 THEN`
	`  j ← 10`
	`ENDIF`
		
		`if i  > 2:`
		`  j=10`

if .. else …	`IF i > 2  THEN`
	`  j ← 10`
	`ELSE`
	`  j ← 3`
	`ENDIF`
		`if i > 2:`
		`    j=10`
		`else:`
		`    j=3`

if … else if … else	`IF i ==2 THEN`
	`  j ← 10`
	`ELSE IF i==3 THEN`
	`  j ← 3`
	`ELSE`
	`  j ← 1`
	`ENDIF`
		`if i ==2:`
		`  j=10`
		`elif i==3:`
		`  j=3`
		`else:`
		`  j=1`

Iteration
 	
While loops
	`a ← 1`
	`WHILE a < 4`
	`  OUTPUT a`
	`  a ← a + 1`
	`ENDWHILE`
		`while a<4:`
		`  print(a)`
		`  a=a+1`

For loops	`FOR a ← 0 TO 3`
	`    OUTPUT a`
	`ENDFOR`
		`for a in range(3):`
		`  print(a)`

Repeat loops	`a ← 1`
	`REPEAT`
	`  OUTPUT a`
	`  a ← a + 1`
	`UNTIL a←4`	

Subroutines
 	
procedure
	`SUB hello()`
	`  OUTPUT "hello"`
	`ENDSUB`
		`def hello():`
		`  print("hello")`

Function (with paramerters and return)	`SUB add(n)`
	`  a ← 0`
	`  FOR a ← 0 TO n`
	`    a ← a + n`
	`  ENDFOR`
	`  RETURN a`
	`ENDSUB`
		`def add(n):`
		`  a=0`
		`  for a in range(n+1):`
		`    a=a+n`
		`  return a`

Built-in functions Length of array	LEN(a)	len(a)

Random integer	RANDOM_INT(0, 9)	import random
		random.randint(0,9)

The Wellington Academy

proud to be part of ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Searching Algorithms

Linear Search Algorithm

- The purpose of the linear search algorithm is to find a target item within a list.
- Compares each list item one-by-one against the target until the match has been found and returns the position of the item in the list.
- If all items have been checked and the search item is not in the list then the program will run through to the end of the list and return a suitable message indicating that the item is not in the list.
- The algorithm runs in linear time. If *n* is the length of the list, then at worst the algorithm will make *n* comparisons. At best it will make 1 comparison and on average it will make (*n+1*)/2 comparisons.
- The performance of the algorithm will be improved if the target item is near the start of the list.

Example
Find the position of letter "Z" within the following list. Assume we do not have visibility of the list

Index position	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Value	V	A	S	Z	X	R	T	G

We compare it with the value in index position 0. We find that the value is "V" so we need to move on to the next index position. At index position 1 and 2 we still have not found Z. However, we get to index position 3 and we compare the target with the value and we find that they match, so the algorithm returns the index position and stops.

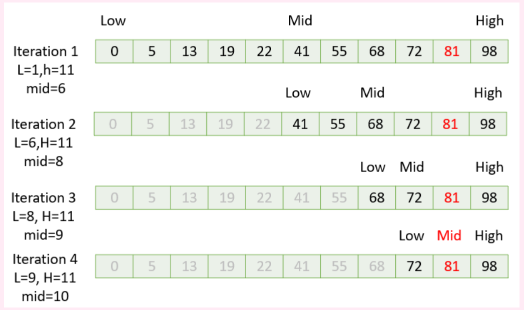
Pseudocode

```
i ← 0
x ← len(listOfItems)
pos ← -1
found ← False
WHILE i < x AND NOT found
  IF listOfItems[i] == itemSearch THEN
    found ← True
    pos ← i + 1
  ENDIF
  i=i+1
ENDWHILE
OUTPUT pos
```

Binary Search Algorithm

- The binary search algorithm works on a sorted list by identifying the middle value in the list and comparing it with the search item.
- If the search item is smaller the mid element becomes the new high value for the search area.
- If the search item is larger the mid element becomes the low value for the search area.
- The keeps repeating until the search item is found.
- When the search item is found the index position of the item is returned.
- At each iteration the search are halved in size consequently this is an efficient algorithm.

Example: Binary search in operation to find 81



Pseudocode

```
low ← 1
high ← LENGTH(arr)
mid ← (low + high) DIV 2
WHILE val ≠ arr[mid]
 IF  arr[mid] < val THEN
  low ← mid
 ELIF arr[mid] > val THEN
  high ← mid
 ENDIF
  mid ← (low + high) DIV 2
  ENDWHILE
OUTPUT mid
```

Linear search versus binary search

	Advantages	**Disadvantages**
Linear Search	• Very simple algorithm and easy to implement • No sorting required • Good for short lists	• slow because it searchers through the whole list • very inefficient for long lists
Binary Search	• much quicker than linear search, because it halves the search zone each step	• The list need to be ordered

The Wellington Academy

proud to be part of ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Sorting Algorithms

Bubble Sort

- The purpose of sorting algorithms is to order an unordered list. Item can be ordered alphabetically or by number.
- Bubble sort steps through a list and compares pairs of adjacent numbers. The numbers are swapped if they are in the wrong order. For an ascending list if the left number is bigger than the right number the items are swapped otherwise the numbers are not swapped.
- The algorithm repeatedly passes through the list until no more swaps are needed.

Example

Sort the following sequence in ascending order using bubble sort: 5,3,4,1,2.

Pass 1	**5**	**3**	**4**	**1**	**2**	
	3	5	4	1	2	Compare 5 and 3 – swap
	3	4	5	1	2	Compare 5 and 4 – swap
	3	4	1	5	2	Compare 5 and 1 – swap
	3	4	1	2	5	Compare 5 and 2 – swap; end of pass 1
Pass 2	3	4	1	2	5	Compare 3 and 4 – no swap
	3	1	4	2	5	Compare 4 and 1 – swap
	3	1	2	4	5	Compare 4 and 2 – swap
	3	1	2	4	5	Compare 4 and 5 – no swap; end of pass 2
Pass 3	1	3	2	4	5	Compare 3 and 1 – swap
	1	2	3	4	5	Compare 3 and 2 – swap
	1	2	3	4	5	Compare 3 and 4 – no swap
	1	2	3	4	5	Compare 4 and 5 – no swap; end of pass 3
	1	**2**	**3**	**4**	**5**	

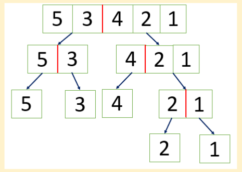
Bubble sort Pseudocode

```
A=[5,3,4,1,2]
sorted ← False
WHILE not sorted
 sorted ← True
 FOR I TO LEN(A)-1:
  IF A[i] > A[i+1]:
   temp ← A[i]
   A[i] ← A[i+1]
   A[i+1] ← temp
   sorted ← False
  ENDIF
 ENDFOR
ENDWHILE
OUTPUT A
```

Merge Sort

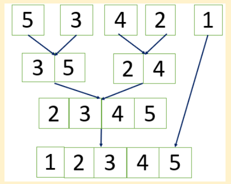
- Merge sort is a type of divide and conquer algorithm.
- There are two steps: divide and combine
- Merge sort works by dividing the unsorted list sublists. It keeps on doing this until there is 1 item in each list.
- Pairs of sublists are combined into an ordered list containing all items in the two sublists. The algorithm keeps going until there is only 1 ordered list remaining.
- Merge sort is a recursive function, that calls itself.

Step 1: Divide



Keep dividing until there is only 1 item in each list

Step2: Combine



1. The first items in the two sublists are compared, and the smallest value is copied to the parent list.
2. The copied item is then removed from the sublist.
3. When there are no items left in one of the sublists the remaining items in the other sublist are them copied in order to the parent list.

Merge sort Versus Bubble sort

	Advantages	Disadvantages
Bubble sort	Very simple and robust algorithm	Can be slow particularly for long lists. As the number of items increases the time taken for the algorithm to run increases dramatically.
Merge sort	Much faster than bubble sort especially when the number of elements is large	More complex to understand Step 1: Divide Step 2: Combine

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Computer Networks

A network is a set of computers that are connected to one another.

Standalone computers are isolated from other devices.

Advantages of a network
- ✓ Share resources, such as software applications, files and hardware (eg printers).
- ✓ Allows communication (eg email) and can transfer files easily.
- ✓ Easier network management (eg can backup data onto a central fileserver; updates can be sent to all computers; users on a network can login to any computer)

Disadvantages of a network
- ✓ Greater security risk as computers can be hacked if they are connected to the internet.
- ✓ Worms can spread from one computer to another
- ✓ A problem with any shared resource, (eg file server goes down) can impact the whole network.

Types of Computer Networks

Personal Area Network (PAN) set up around an individual person. Many people have multiple devices such as tablets, phones and computers that can be interconnected using a PAN. A Bluetooth PAN uses radio waves to communicate wirelessly between devices over a range of a few metres.

Local Area Network (LAN) covers a relatively small geographical area typically extends over the range of a single organisation such as a university campus, school site. LANs are usually managed by a single organisation.

Wide Area Network (WAN) made up of many local area networks and covers a much wider geographical area. The internet the ultimate WAN. It is a network of networks with billions of interconnected devices. No single person or organisation has control over a WAN.

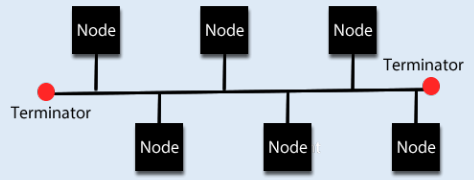
Network Topology

A network topology describes how a set of computers are arranged within a network.

Bus network topology All devices including clients, servers, printers and so on are connected to a cable called a bus. All communication is via the shared bus. At either ends of the bus is a terminator.

Advantages
- ✓ Easy and cheap to install and does not require much cable
- ✓ Easy to add more computers

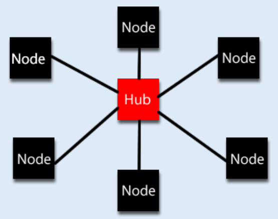
Disadvantages
- ✓ If the main cable fails then then the whole network fails.
- ✓ Less secure as data are broadcast to all devices on the network.
- ✓ Can be slow as there are collisions between data along the shared bus.
- ✓ Will get slower as more computers are added.



Star network topology all devices including clients, servers, printers and so on are connected to a central hub or switch. All communication is via the hub

Advantages
- ✓ Greater security as data are only sent to the intended recipient.
- ✓ If any of the connections fail only a single node will be affected.
- ✓ Fewer collisions between data packets

Disadvantages
- ✓ If the central hub fails then every computer on the network is affected.
- ✓ Expensive as extra cable and hardware (hubs) are needed.



Wired and Wireless

Computers can be connected using wired or wireless methods

Wired transmission methods use cables to communicate

Wireless transmission use radio waves communicate (eg Wi-Fi).

Advantages of wireless
- ✓ Can use computer anywhere and not constrained by cables

Disadvantages of wireless
- ✓ Packets can be intercepted more easily than wired connections
- ✓ Security is a much more difficult challenge, as the network can be accessed from outside the confines of a building.
- ✓ Slower than wired methods
- ✓ Signal can be interfered with by other electronic devices.

Advantages of wired
- ✓ Allows more control, security and reliability. Can restrict who has access to the network.
- ✓ Wired methods have greater speeds than wireless methods.

Disadvantages of wired
- ✓ Cables can be difficult to maintain in big organisations

Wired networks use a variety of cables, including copper and fibre optic.

Copper cables use electrical signals to transmit data. Three main types:

- ✓ **Coaxial cable** – the signal loses strength over long distances
- ✓ **Unshielded twisted pair** – A pair of copper cables are twisted together and allows data to be transmitted over longer distances
- ✓ **Shielded twisted pair** – Shielding around the twisted cables means the signal is less susceptible to interference.

Fibre optic cables are glass or plastic and use use pulses of light to transmit data

Advantages of copper cables
- ✓ Cheaper than fibre optic
- ✓ Reliable because a telephone is powered from the copper cable and does not rely on a separate electrical power supply

Advantages of copper cables
- ✓ Slow
- ✓ Low capacity
- ✓ Can only be used over short distances
- ✓ Interference can occur

Advantages of fibre optic
- ✓ Higher bandwidth than copper so can transmit more data
- ✓ Less attenuation (degrading) of the signal so fibre optic is more suitable over long distances
- ✓ Less "cross talk" interference between fibres compared with copper so the quality of the signal is better

Disadvantages of fibre optic
- ✓ Expensive
- ✓ Difficult to install

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Network Security and Protocols

Why do we need network security?
✓ To prevent unauthorised access to our electronic devices
✓ To protect our data eg to prevent sensitive data being stolen
✓ Prevent cyberattacks

Methods of Network Security

Authentication allows us to confirm the identity an individual. There are lots of ways of confirming the identity of an individual that come under one of three factors:
✓ Knowledge factor: Something the user knows, eg a password
✓ Possession factor: Something the user owns eg a mobile phone
✓ Biometric factor: eg Fingerprint, iris scan

Encryption The message is garbled so if it gets intercepted during transmission it will be almost impossible for anyone without the key to read the original message.

Firewall prevents packets containing malware getting on to the computer

MAC address filtering A MAC (Media Access Control) address is a unique identifier for any device that is connected to a network. Each network interface card has a unique MAC address that is a 12 digit hexadecimal code (e.g. 12-F3-EE-56-44-A1).

✓ *White list filtering* only allows devices on a list to connect to the network.

✓ *Black list filtering* devices in a black list blocked from accessing the network.

Network Protocols

A **network protocol** is a set of rules that allow computers to communicate and exchange information over a network. There are many types of protocols depending on the application.

HTTP (Hypertext transfer protocol) is the protocol used for the World Wide Web. An exchange begins with a request for a web page from a client web browser to a web server. The server then sends the web page to the client.

HTTPS (Secure Hypertext transfer protocol) is a secure way of transferring data between a web browser and a server because the data are encrypted during transfer. Used for e-commerce and online banking.

FTP (File Transfer Protocol) is usually used to download or upload large files from a server to a client.

Ethernet is not a single protocol but a collection of related protocols. LANs most commonly use ethernet. The following is a simplified procedure:

1) Check whether there is any traffic on the ethernet
2) If so wait for traffic to clear
3) Send the packet
4) If collision detected, go to step 1 to resend.

Wi-Fi is a collection of protocol that use radio waves to transmit data between devices. Wi-Fi is a trademark and WLAN (Wireless LAN) is the generic term. Data are transmitted when the medium is clear, and an acknowledgement is received if the transmission was successful. If no acknowledgement is received, then the data are resent as it is assumed that a collision occurred, and the packets did not reach their destination.

Email protocols

SMTP (simple mail transfer protocol) Sends the mail from the user onto the mail server.

IMAP (Internet Message Access Protocol) Retrieves the mail from the mail server to the client (user) and allows access from anywhere on any device because the email remains on the server.

TCP (Transport Control Protocol) When files are sent over the internet they are broken up into small chunks called packets. When they arrive at the destination computer they are reassembled back into the original format. TCP handles and controls all this. TCP waits for acknowledgements to verify whether the packets have reached their destination. TCP will also retransmit packets of they have not arrived at the destination or become corrupted.

IP (Internet Protocol) The internet protocol is a set of rules that govern the transmission of data across the internet.

UDP (User Datagram Protocol) is used as an alternative to TCP. It is used in video conferencing and online gaming when speed is necessary as huge volumes of data are transferred in real time. It improves speed by not checking for lost packets so they do not get re-sent.

TCP/IP

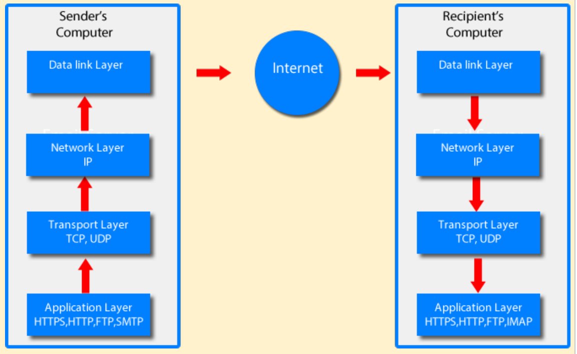
The TCP and IP protocol work closely together and are referred to as TCP/IP. The TCP/IP model consists of four layers that pass data between each layer.

Application layer contains protocols related to the application such as HTTP, HTTPS for web browsers, FTP for file transfer and SMTP and IMAP for email. The application layer interacts with the user via appropriate application software (eg web browser / ftp client).

The **transport layer** establishes the end to end connection. When files are sent over the internet, they are broken up into small chunks called packets. When they arrive at the destination computer they are reassembled back into the original format. It is the role of the transport layer to split the data into packets and pass the data onto the network layer. On the recipient's computer the transport layer reassembles the packets into the original form. The packets are numbered by this layer to allow them to the reassembled. The transport layer chooses the port number for sender and receiver. TCP and UDP are the main protocols used in this layer.

The **network layer** adds the source and destination IP address and route the packets over the network. At the destination the network layer strips out the IP addresses. The IP operates on this layer.

The **data link layer** has a network card and deals with the physical connection and adds the physical addresses (MAC address) of the hardware to the packets that it receives from the network layer. For each step the sender and receiver MAC address is removed then a new sender and receiver MAC address is added. The receiver MAC address becomes the sender MAC address.



COMPUTER SCIENCE KNOWLEDGE ORGANISER

Computer Systems

A computer system has both hardware and software.

Hardware are the physical components that make up a device or computer system. These include both the internal components (eg motherboard, CPU, RAM) and peripheral devices such as printers.

Software is the computer code, programs and algorithms that give instructions to the hardware to make it perform the desired task. Without the software the hardware will not get any instructions and it will not do anything.

Software Classification

Software is split into two types: application software and system software

Application software is a program designed to perform a specific task that the user interacts directly with (eg spreadsheets, web browser and word processor, disk defragmentation).

System software is concerned with the running of the computer. Its purpose is the control the computer hardware and manage the application software. (eg operating system, antivirus, backup tools, firewall)

The **operating system (OS)** is the most important piece of system software. The OS handles management of the processor, memory, input/output devices, applications and security.

- **Application management** - Application software does not need to concern itself with interaction and complexities of managing the hardware because this is dealt with by the operating system. Application software runs on top of operating system which is an intermediary and takes care of interaction with the hardware.
- **Processor resources** – Allows multiple applications to be run simultaneously by manages the processing time between applications and cores and switching processing between applications very quickly. Multiple applications will access the processor resources via a schedule that alternates process between applications. High priority applications will have more CPU time, but it means that lower priority applications will take longer to run.
- **Memory management** – Distributes memory resources between programs and manages transfer of data and instruction code in and out of memory. Ensures that each application does not use excessive memory.
- **Security** – Tools such as anti-virus software and firewalls help protect the computer from attack. In addition requirement for passwords and control of access rights
.
- **Input / Output devices** – OS controls interaction with input (eg keyboard) outputs (eg. Monitor) and storage (eg hard disk) using hardware drivers. Allows users to save files to the hard disk and print documents for instance.

Cloud Computing

- Can store data and files on a server elsewhere that can be accesses via the internet.
- Can use applications over the internet
- Can sync files so that all your devices see the same files
- Can share documents with others
- Can access your files anywhere if you have a good internet connection

Advantages of cloud computing
- Only pay for storage that you use
- Data and files available from anywhere in the world where there is an internet connection
- Data automatically backed up

Disadvantages of cloud computing
- Need a reliable network connection
- Files are hosted elsewhere so a security concern
- the most recent versions of software is often not available
- Transfer of data over the internet will slow down performance.

Advantages of local storage
- Files can be accessed even when there is no internet connection
- More secure as files to not need to be transferred over the network and the user has more control

Disadvantages of local storage
- Users need to organise their backup solutions
- Not so easy to share documents
- Can only access the files locally

Memory

Volatile memory (main memory) When the computer is turned off the contents of volatile memory is lost. When there is no power, volatile memory is erased.

Non-volatile memory (secondary storage) Even when here is no power, the data remain unchanged and can be accessed once again once power has been resumed. This allows you to store files for he long term.

ROM (Read Only Memory) Data can only be read from the device, and cannot the memory cannot be edited or deleted. ROM is only used for situations where you can be sure that updates will not be needed. The computer's BIOS (basic input output system) which controls the boot up sequence is stored on a ROM chip.

RAM (Random Access Memory) - When applications are executed they are loaded into RAM first. RAM is volatile.

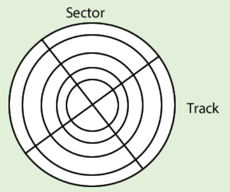
Embedded Systems

An embedded system is a computer system that is designed for a specific function, in contrast to a general-purpose computer that can carry out many tasks. Embedded systems typically have a minimal or no user interface. Thus, they can be optimised for size and power consumption, for instance. Examples of embedded systems include digital watches, MP3 players, washing machines, cars and mobile phones.

Secondary Storage

Secondary storage is necessary for saving files long and software including the operating system. Even when the computer is turned off, the data remain unchanged, and can be accessed again once the power supply has been turned on.

Magnetic Hard Disk
- Tracks on the disk platters contain tiny magnets, each holding 1 bit of data.
- The polarity (negative or positive) of the magnets determines whether the bits are 0 or 1.
- The write head modifies the polarity of the magnet as appropriate.
- The read head identifies whether each magnet is negative or positive.
- The tracks are laid out as a series of concentric rings.



Advantages
- Cheap form of storage

Disadvantages
- Less reliable because it contains moving parts that can break
- Electromagnetic surge can corrupt the data held
- Slow speed of read/write access

The Wellington Academy

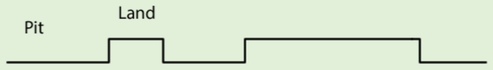
proud to be part of ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Optical Disks
- Tracks on the disk contain pits and lands.
- The track is a spiral.
- A laser is emitted and the laser light is reflected when it hits the lands, but is scattered when it hits the pits.
- Depending on whether the light is scattered light is encoded as a binary value of 0 and reflected light is encoded as a 1.
- The sensor is able to detect light reflected, but not scattered.
- Example: Blue-Ray (25 Gb) DVD (4.7 Gb), CD (700 Mb).

Advantages
- Can transfer easily between computers

Disadvantages
- Can scratch easily
- Not much storage compared with other methods.
- No unlimited writes to the hard disk



Solid state Drive
- Use millions of switches called floating gate transistors on microchips to store data.
- Electrons are stored in gates and this is encoded as 0 when there is an electron present and encoded a 1 when there is no electron present.
- The electros remain trapped even when there is no flow of electricity.
- Contain no moving parts and are therefore more robust that optical and magnetic storage.

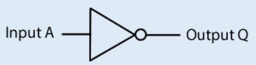
Advantages
- Much faster that other means of storage
- More reliable than other means if you are only reading
- Quiet

Disadvantages
- More expensive per volume of storage
- Reliability might be an issue if you do a lot of writing

Boolean Logic

NOT gate - The output is the opposite of the input

$$Q = \bar{A}$$
$$Q = NOT\ A$$

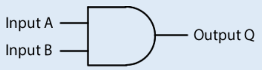


NOT truth table

Input	Output
0	1
1	0

AND gate - has two inputs and will have a true output if the two inputs are true otherwise the output will be false

$$Q = A.B$$
$$Q = A\ AND\ B$$

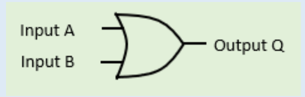


AND truth table

Input - A	Input - B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR gate - has two inputs and will have a true output if either or both the inputs are true

$$Q = A + B$$
$$Q = A\ OR\ B$$

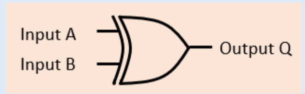


OR truth table

Input - A	Input - B	Output
0	0	0
1	0	1
0	1	1
1	1	1

XOR gate - has two inputs and will have a true output if either the inputs are true but not both

$$Q = A \oplus B$$
$$Q = A\ XOR\ B$$



OR truth table

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	0

Boolean Expression Operators

Operator		Example	
AND	.	Q = A AND B	Q = A . B
OR	+	Q = A OR B	Q = A + B
NOT	$\bar{}$	Q = NOT A	$Q = \bar{A}$
XOR	$\oplus$	Q = A XOR B	$Q = A \oplus B$

Converting a truth table to a logic circuit

There is a general approach to converting a truth table into a logic circuit.

We consider only the lines with an output of 1.
We take in the input of each and then AND.

We then OR between each statement such that
(NOT A AND B) OR (A AND NOT B). We can then draw the logic circuit.

Worked example: What is the logic circuit for the following truth table

Input - A	Input - B	Output
0	0	0
1	0	1
0	1	0
1	1	1

(A AND NOT B) OR (A AND A)

System Architecture

CPU (Computer Processing Unit) or processor Fetches, decodes and executes instructions and performs logical and arithmetic operations.

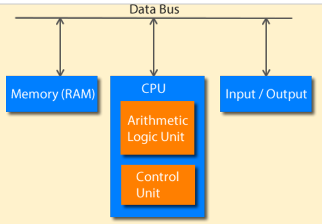
Von Neumann architecture is the stored program concept, where program instructions and the data to be processed can be stored in the same memory.

The Wellington Academy

proud to be part of

ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER



Components of a CPU

Bus Wires through which data and instructions are transferred between computer components

Clock keeps all the CPU components synchronised

Arithmetic Logic Unit (ALU) Every operation takes place here. This is where the arithmetic (eg adding two binary numbers) and logic operations (eg checking to see if one number is bigger than another) take place.

Control Unit Decode the machine code instruction so that the ALU knows what to do with the instruction. Controls and monitors data transfer between different input and output hardware components

Factors affecting CPU performance

Clock speed is the number of cycles that a processor carries out per second. Each cycle of the CPU allows a single action (instruction) to be carried out. The greater the clock speed, the greater the number of operations and the faster the computer will run.

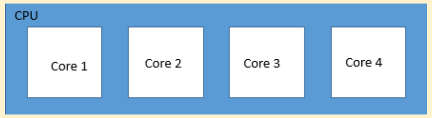
Number of processor cores A core is CPU in its own right. Nowadays most CPUs have multiple cores. Having multiple cores allows instructions to be carried out concurrently (at the same time), whereas a single core will only allow carry out instructions in serial (one at a time).

Latency Delay in transfer of data between components

Cache size Cache is a volatile memory store on the processor. Cache is much faster but smaller that RAM. Frequently used data and instructions within an application can be stored in cache instead of fetching from RAM which is quite slow. The bigger the cache the greater the volume of data and instructions that can be stored thereby reducing latency and improving performance of the CPU.

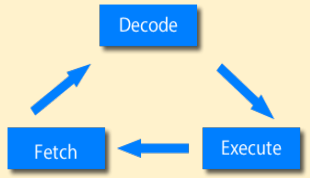
Cache type There are three levels of cache**.** Cache Level is a trade off between size and speed

- *Level 1 Cache* closest to the CPU and is the fastest cache (lowest latency), but does not have much capacity
- *Level 2 Cache* – is slower and further away from the CPU than L1 cache so latency is greater, but has more storage capacity.
- *Level 3 Cache* is the slower than L1 and L2 cache; much faster than RAM; has greater capacity than L1 and L2 cache.



Fetch execute cycle

1. Instructions are loaded into memory
2. Processor fetches the instruction from the main memory
3. Instruction is decoded so the CPU knows what to do with the instruction
4. Processor then executes the instruction
5. Result of the instruction can be stored in memory
6. Next instruction is then fetched from main memory and the cycle repeats itself.



Classification of programming languages

High level programming languages are closer to human language and is therefore easier to understand. A translator is used to convert the instructions into code that the computer understand. High level languages allow programs to be written that is independent of the type of computer. High level programming languages allow code to be written that is independent of the type of computer system. It is up to the compiler to translate the code into the right machine code for a particular code. There is a huge variety of high level programming languages, and the choice depends on the application.

Low level programming languages refer to machine code and assembly language. The Low level refers to low level of abstraction. The low level language is close to the language understood by the computer where operations map to the instruction in the processor instruction set. However it is difficult for humans to understand. Low level languages are appropriate for developing new operating systems, embedded systems and hardware device drivers

Machine code is expressed in binary values 0 and 1. This is the language that computers understand. All codes whether assembler or high level programming languages need to be translated into machine code. Machine code is specific to a processor.

Machine code instructions are made up of two parts the operator and the operand. The processor decodes the operator to identify the task that is to be carried out (eg. Add, load). The operand is the value or memory address that that instruction is to be operated on

Machine code instruction	
Operator	Operand
0011	10010100

Assembly language provides basic computer instructions for programs to run. There is a one to one relationship between machine code and assembly code instructions. One assembly language instruction maps to one machine code instruction, thus the structure of assembly language and machine code is the same, but where machine codes uses 0 and 1 which are very difficult for programmer to understand, assembly language uses mnemonics which is easier for the programmer.

Assembly language sample Instruction set

```
LOAD #23 # Load from RAM to processor
MOV a 23 # Transfer in number 23 into the variable a
ADD 2 3  # Add 2 values
STORE    # store data in RAM
```

Each type of processor has its own instruction set and therefore its own assembly language and machine code. So Assembly code written for one type of processor will not run on another.

Low level languages versus high level languages

	Advantages	**Disadvantages**
Low level	Produce code that is faster and better optimised than high level languages. Appropriate for developing new operating systems, embedded systems and hardware device drivers	Difficult to understand and modify Assembly code is written for a specific processor architecture, and so is not portable to other computer architectures
High level	High level programming languages allow code to be written that is more portable. Thus code can be run on different of the types of computer system with different processor architecture. Easier to understand Easier to modify	Needs a translator run slower because of the layers of abstraction and there is inefficiency in translator.

The Wellington Academy

proud to be part of

ROYAL WOOTTON BASSETT ACADEMY TRUST

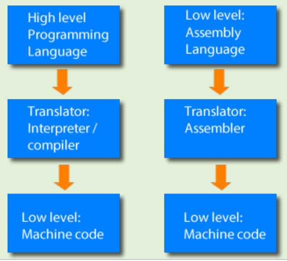
COMPUTER SCIENCE KNOWLEDGE ORGANISER

Program translators allow programs to be translated into machine code so the than programs can be run on a computer.

Interpreter converts high level languages into machine code one instruction at a time on-the-fly while the program is running. Each instruction is converted to machine code once the previous instruction has been executed. Interpreters are good for debugging code because the program stops as soon as the error has been found. However running code this way is much slower running compiled code. The machine code is not saved.

Compiler A program that converts high level languages into machine code before the program is run. A compiler saves the machine code, so the source code is no longer needed A compiler allows a program to be run faster than interpreted code. Software is normally distributed as compiled machine code. For proprietary software this is good because other people cannot copy the code and use it for their own applications.

Assembler Assembler converts assembly language instructions into machine code.


High level Programming Language
Translator: Interpreter / compiler
Low level: Machine code
Low level: Assembly Language
Translator: Assembler
Low level: Machine code

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Cybersecurity

Cybersecurity is concerned with the protection of computer systems, computer networks and data. Its purpose is to:

- to protect computers and networks from cyberattacks
- to prevent unauthorised access to computers
- to protect computers against damage caused by malicious software
- to prevent data from being stolen
- to protect against the disruption of services running on the computer

Cyber Security Threats

Malware is software that has been purposely developed to damage, disrupt or take control of computer systems.

Social engineering techniques manipulate people into giving away confidential and personal information.

Weak passwords are easy to guess. Passwords that use words are easy to crack using an algorithm that systematically goes through all the words in a dictionary until the word matches the password.

Default passwords Upon registration for an online account, users may be given a default password that they do not change. Often these passwords are sent out unencrypted via email so pose a major security vulnerability.

Removable media such as a USB pen drive can be a vector for transmitting malware.

Unpatched/outdated software Software needs regular updates to fix security vulnerabilities in computer systems. Software that remains unpatched is vulnerable to attack.

Misconfigured access rights Users should only have access to files and data that they need, but sometimes they have access that they should not.

Penetration Testing

Penetration testing is legitimate testing of an organisation's computer system to identify whether there are any vulnerabilities that an attacker could exploit. By identifying vulnerabilities, these can be patched before the system gets attacked.

White box testing testers are given some information about the network, such as network architecture, source code, and IP addresses. This is designed to simulate an attack by a malicious insider.

Black box testing testers are given very little information about the network before the test. This is designed to simulate an outside attack or cyber warfare attack.

Cyber Security Threats - Malware

Computer viruses replicate themselves and can transfer from one computer to another. They are activated by a user often as email attachments and attachment to other files and programs.

Trojan gains access to a computer by pretending to be legitimate software. The trojan allows unauthorised backdoor access to a computer without the user being aware.

Spyware records the activity on your computer such as your keystrokes, thereby logging your passwords for instance and then send the data back over the network to a hacker. Spyware can also be used to control your webcam and microphone.

Adware includes banners and popups that are automatically installed onto a computer. Whilst this does not cause any, adware is undesirable and can slow down the performance of a computer.

Worms spread like viruses but do not require human intervention. They attach themselves to network tools to spread automatically around a network very quickly.

Methods to detect and prevent cyber security threats

Biometric measures such as fingerprints, facial recognition and iris scans are increasingly being used to verify a user's identity for mobile devices. These are more secure than passwords that can be guessed and forgotten. Biometric measures require a user to be present when signing into a system.

Automatic software updates to firewalls, operating systems, antivirus and other security software are needed so that software can be kept up-to-date against new malware and to fix recently discovered vulnerabilities.

CAPTCHA is a test that can distinguish between humans and bots. It uses images that machines cannot interpret but humans can.

Password systems Virtually all accounts require passwords to access. Some secure sites such as online banking require 2 passwords. Banks may also contact you by phone to confirm a large transaction. This is called two-factor authentication. Password systems can force users to have strong passwords that regularly need to be changed.

Using email to confirm a person's identity Often when you register for an online service you need to provide your email address. You are then requested to activate a link sent to you in an email. This is to confirm that the email account is actually active. Helps to ensure that the users are human and not bots.

Anti-virus software scans the computer intermittently to identify whether there is any malware on the computer. The software compares each file against a database of known virus codes. If viruses are found (ie contains code that is in the database) the file is quarantined. That is the file cannot be run without explicit authorisation from the user. New malware are regularly being created and so anti-virus software needs to be updated to identify the new viruses. That is why anti-virus software is regularly updated.

Cyber Security Threats – Social Engineering

Blagging (Pretexting) Fraudsters make up a scenario to con victims into revealing something they would not ordinarily do. They may have found out some personal information about you from social media sites, to pretend they already know you.

How to prevent

- Use biometric measures because these cannot be divulged.
- Ensure you have your privacy settings on any social media to maximum so that fraudsters cannot find information about you such as your date of birth, where you live etc.

Phishing Normally an email or text messaging scam where victims are conned into believing that they are being contacted by their bank for instance and can give sensitive personal details such as bank account passwords.

How to prevent

- Awareness and vigilance. Be particularly aware of unsolicited texts, emails and phone calls. Do not give personal confidential information away. Official organisations such as banks will never ask for this information.
- Apply email filtering to prevent dubious emails getting through.

Pharming Users are redirected to a fraudulent website that they believe to be genuine because it looks like the real site. For instance, you could be directed site that pretends to be an online store that asks you for your credit card information.

How to prevent

- Check the URL in the web address. For secure websites such as banking or e-commerce sites the HTTPS protocol should be used.
- Website filter

Shoulder surfing Fraudsters look over the shoulder of users to see what passwords or pin numbers that are being typed into the device. This can easily occur at computer terminals and at ATMs that are out in the street.

How to prevent

- Be aware of who is around you when typing in your pin into an ATM or into a chip and pin device. Make sure you cover your hands and they are shielded from prying eyes.
- Place computers in locations that makes shoulder surfing difficult

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Data Representation

Number bases

Denary (or decimal) is base-10 and is the number system we are most familiar with. We have the columns of units, tens, hundreds, thousands and so on. Base-10 means that we have 10 possible values (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) in each column.

Binary is base-2 and has 2 values, 0 and 1. It requires a greater number of digits in binary to represent a number than denary. This is how data and instructions are stored in a computer.

To calculate the maximum value for a given number of bits we use 2^n-1 where n is the number of bits. For example for 4 bits we have 2^4-1 which is 15.

Bits	Max value binary	Max value denary
1	1_2	1_{10}
2	11_2	3_{10}
3	111_2	7_{10}
4	1111_2	15_{10}
5	11111_2	31_{10}
6	111111_2	63_{10}
7	1111111_2	127_{10}
8	11111111_2	255_{10}

Hexadecimal is base-16. To make up the 16 values we use the ten denary numbers in addition to 6 letters (A, B, C, D, E, F).

Denary	Hex.	Binary
0_{10}	0_{16}	0000_2
1_{10}	1_{16}	0001_2
2_{10}	2_{16}	0010_2
3_{10}	3_{16}	0011_2
4_{10}	4_{16}	0100_2
5_{10}	5_{16}	0101_2
6_{10}	6_{16}	0110_2
7_{10}	7_{16}	0111_2

Denary	Hex.	Binary
8_{10}	8_{16}	1000_2
9_{10}	9_{16}	1001_2
10_{10}	A_{16}	1010_2
11_{10}	B_{16}	1011_2
12_{10}	C_{16}	1100_2
13_{10}	D_{16}	1101_2
14_{10}	E_{16}	1110_2
15_{10}	F_{16}	1111_2

Hexadecimal is used a lot in computing because it much easier to read than binary. There are far fewer characters than binary. So hexadecimal is often used in place of binary as a shorthand to save space. For instance, the hexadecimal number 7BA3D456 (8 digits) is 01111011101000111101010001010110 (32 digits) in binary which is hard to read.

Hexadecimal is better than denary at representing binary because hexadecimal is based on powers of 2.

Converting between number bases

Denary to binary conversion

1. Create a grid:

128	64	32	16	8	4	2	1

2. Add a 1 to the corresponding cell if number contributes to target number and 0 to all the other cells

Worked example: convert 24_{10} to binary.

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

16_{10} + 8_{10}=24_{10}

The binary value is $\underline{11000_2}$ (we can ignore the preceding zeros)

Binary to denary conversion

Worked example: Convert 01011001_2 to denary

1. Create the grid:

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

2. Add up the cells that have a corresponding value of 1:
   64 + 16_{10} + 8_{10} + 1= $\underline{89_{10}}$

Hexadecimal to denary conversion

1) Convert the two hex values separately to denary value
2) Multiply the first value by 16
3) Add the second value

Worked example: Covert $A3_{16}$ to denary

A_{16} = 10_{10}

3_{16} = 3_{10}

(10_{10} x 16_{10}) +3_{10} = $\underline{163_{10}}$

Denary to hexadecimal conversion

1) Integer divide the denary number by 16
2) Take the modulus 16 of the denary number
3) Convert the two numbers to the corresponding hex values.

Worked example: Convert 189_{10} to hex

189_{10} / 16_{10} = 11_{10} remainder 15_{10}

11_{10} = B_{16}

15_{10} = F_{16}

189_{10} = $\underline{BF_{16}}$

Hexadecimal to binary conversion

1. Find the corresponding 4-bit binary number for the two numbers
2. Concatenate the two binary values to give the final binary value

Example: convert $C3_{16}$ to binary

C_{16} = 12_{10} = 1100_2

3_{16}= 3_{10} = 0011_2

$\underline{11000011_2}$

Binary to hexadecimal conversion

1. Split the binary number into groups of 4 bits: 1110_2 1010_2
2. Find the corresponding Hex value for each of the 4-bit groups

Worked example: Convert 11101010_2 to hexadecimal

1110_2 |1010_2

1110_2 = 14_{10} = E_{16}

1010_2 = 10_{10} = A_{16}

$\underline{EA_{16}}$

Units of Information

Unit	Symbol	Number of bytes
Kilobyte	KB	10^3 (1000)
Megabyte	MB	10^6 (1 million)
Gigabyte	GB	10^9 (1 billion)
Terabyte	TB	10^{12} (1 trillion)

A bit is the fundamental unit of binary numbers. A bit is a binary digit that can be either 0 or 1.

1 byte = 8 bits

1 nibble = 4 bits

Character Encoding

Character coding schemes allows text to be represented in the computer. One such coding scheme is **ASCII**. ASCII uses 7 bits to represent each character which means that a total of 128 characters can be represented.

Lower case letters	26
Upper case letters	26
Numbers	10
Symbols (e.g. comma, colon)	33
Control characters	33

ASCII encoded values for some characters

A	1000001_2	65_{10}
B	1000010_2	66_{10}
a	1100001_2	97_{10}
b	1100010_2	98_{10}
“0”	0110000_2	48_{10}
“1”	0110001_2	49_{10}

- ASCII has a limited character set (7 bits, 128 characters), but **Unicode** has 16 bits and allows many more (65K) characters.
- Unicode provides a unique character for different languages and different platforms.
- It allows us to represent different alphabets for instance Greek, Mandarin, Japanese, Emojis etc.
- Unicode and ASCII are the same up to 127.

Binary addition

Binary addition rules

$0_2 + 0_2 = 0_2$

$0_2 + 1_2 = 1_2$

$1_2 + 0_2 = 1_2$

$1_2 + 1_2 = 10_2$ (carry 1)

$1_2 + 1_2 + 1_2 = 11_2$ (carry 1)

Example

$$\begin{array}{rl} & 1\,0\,1\,0\,1\,0\,0\,1_2 \\ & 0\,0\,0\,0\,1\,0\,0\,1_2 \\ + & \underline{0\,0\,0\,1\,0\,1\,0\,1_2} \\ & \underline{1\,1\,0\,0\,0\,1\,1\,1_2} \\ \text{carry} & 1\,1\,1\quad\;\; 1 \end{array}$$

Binary Shift

The binary shift operator is used to perform multiplication and division of numbers by powers of 2

multiply/divide	x 16	x 8	x 4	x 2	/ 2	/ 4	/ 8
shift	<<4	<<3	<<2	<<1	>>1	>>2	>>3

Example: Apply shift operator to 1101_2 (13_{10})

Shift	Result	denary
<<1	11010_2	13_{10} x 2_{10} =26_{10}
<<2	110100_2	13_{10} x 4_{10} =52_{10}
>>1	110	13_{10} // 2_{10} =6_{10}

Note that odd numbers are rounded down to the nearest integer when the right shift operator is applied.

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Sound

Sample - Measure of the analogue signal at a given point in time

Sample rate - number of samples taken per second and is measured in Hertz.

Sample resolution - number of bits used to represent each sample

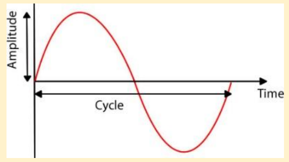
The size of sound files can be calculated using:

size of file = length (seconds) x sample rate x sampling resolution

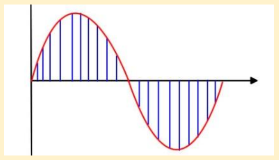
For sound to be stored digitally on a computer it needs to be converted from its continuous analogue form into a discrete binary values. The steps are:

1. Microphone detects the sound wave and converts it into an electrical (analogue) signal
2. The analogue signal is sampled at regular intervals
3. The samples are approximated to the nearest integer (quantised)
4. Each integer is encoded in binary with a fixed number of bits

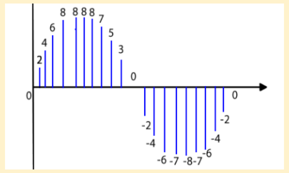
Original analogue signal



Sample signal at regular intervals



Integer values give to each sample



Encode as binary

0 2 4 6 8 8 8 8 7 5 3 0 ->

00000 00010 00100 01000

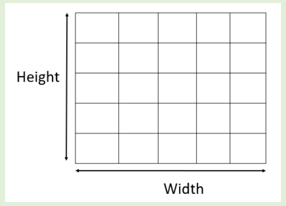
01000 01000 01000 00111

00101 00011 …

Images

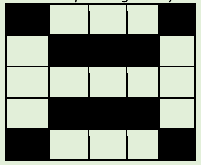
Bitmap images are made up from tiny dots called **pixels**. Each pixel will have a colour associated with it. An image can then be constructed from many of pixels which will have different colours arranged in rows and columns.

Total number of pixels in image = width in pixels x height in pixels



Colour depth is the number of bits used to represent each pixel in an image. If we have a black and white image it has two colours. Each pixel can be represented by a single pixel because a bit value of 0 is black and 1 is white.

Image and corresponding binary encoding



0111010001111111000101110

To represent more colours we can use more bits. For instance if we have 2-bits per pixel we can represent 4 colours because we know have 4 binary code combinations (00, 01, 10 11) where each code represents a different colour

Pixilation occurs when the image is overstretched. In these situations, the image looses quality and has a blocky and blurred appearance. This arises when the image is presented at too large a size and there are not enough pixels to reproduce the details in the image at this larger size.

Calculating the size of a bitmap image

File size in bits = width in pixels x height in pixels x colour depth

File size in bytes = width in pixels x height in pixels x colour depth / 8

Data Compression

The purpose of data compression is to make the files smaller which means that:

- Less time / less bandwidth to transfer data
- Take up less space on the disk

Given that there are 7 bits per ASCII character, the uncompressed size of an ASCII phrase is:

size = number of characters (including spaces) x 7

Run Length Encoding (RLE) is a compression method where sequences of the same values are stored in pairs of the value and the number of those values. For instance, the sequence:

```
0 0 0 1 1 0 1 1 1 1 0 1 1 1 1
```

would be represented as:

```
3 0 2 1 1 0 4 1 1 0 4 1
```

Huffman coding is a form of compression that allows us to use fewer bits for higher frequency data. More common letters are represented using fewer bits than less common letters. For instance, "a" and "e", which occur in many words would be represented with fewer bit than "z" which occurs rarely.
This allows for much more effective compression than RLE.

The steps involved in Huffman encoding as are follows:

1. Do frequency table
2. Order table
3. Create the tree
4. Add 1, 0 to the branches
5. Encode letters
6. Encode message

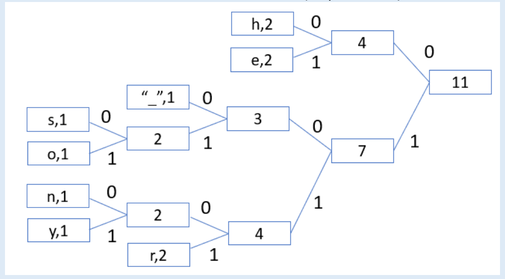
Worked Example: How much smaller is the phrase henry horse encoded using Huffman encoding compared with its uncompressed size.

Calculate the uncompressed size
In the phrase *henry horse* there are 11 characters (including the space). Therefore the uncompressed size is 11 x 7 = 77 bits

Generate ordered frequency table (steps 1 and 2)

letter	frequency
e	2
h	2
r	2
<space>	1
o	1
s	1
y	1
n	1

Create the tree and add 1 and 0 to branches (steps 3 and 4)



Encode letters

Letter	encoding
e	01
h	00
r	111
<space>	100
o	1011
s	1000
n	1100
y	1101

Encode message
00 01 1100 111 1101 100 00 1011 111 1000 01 = 33 bits

Therefore by using compression we have reduced the size from 77 bits to 33 bits a saving of 44 bits.

The Wellington Academy   proud to be part of   ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Databases

A **database** is a collection of **data** stored in an organised and logical way. Data are stored in **tables** and tables are made up of **records** (rows) which can have 1 more **attributes** (columns). An example of a table is given here:

Student ID	First Name	Surname	DateOfBirth	FormTutor
712	Bart	Simpson	1/4/10	Principal Skinner
423	Lisa	Simpson	20/5/12	Mrs Krabapple
917	Ralph	Wiggum	16/6/10	Mrs Krabapple
124	Nelson	Muntz	14/9/09	Principal Skinner

Entity

Each table contains information about an **entity**. A database entity is an object, person, item or thing about which you want the data stored. Examples of database entities are:

Person entity	Object entity	Item entity
✓ Customer	✓ Book	✓ Sale transaction
✓ Employee	✓ Car	✓ Appointment
✓ Student	✓ House	
✓ Teacher		

Data

Data are atomised facts, values and observations that are stored in a database. That is they cannot be broken up further. Data can be stored as any data type.

Field	Student ID	First Name	Height	Date of Birth	Had Flu Vaccination?
Date Type	*Integer/ number*	*Text/ string*	*Real/ float*	*date*	*Boolean – Yes/no or true/false*
Record 1	712	Bart	1.35	1/4/2010	True
Record 1	423	Lisa	1.16	20/5/2012	True
Record 1	917	Ralph	1.05	16/6/2010	False

Database Index

A database index allows for quick speed of retrieval of data from searches of tables. The index is a separate file that has a sorted column of values that link to records in a table.

Record

A record is a single row in a table that can have data stored as 1 or more fields (columns). A record needs to be uniquely identifiable and needs an entity identifier which in this example is Student ID. A table contains multiple records. The following example contains 4 records.

	StudentID	FirstName	Surname	DateOfBirth	FormTutor
Records	712	Bart	Simpson	1/4/10	Principal Skinner
	423	Lisa	Simpson	20/5/12	Mrs Krabapple
	917	Ralph	Wiggum	16/6/10	Mrs Krabapple
	124	Nelson	Muntz	14/9/09	Principal Skinner

The **Student ID** field contains unique values for each record; this means that each value is different. The **Surname** field does not contain unique values. For instance, *Simpson* appears twice.

Field

Fields / attributes form the columns of the database table and refer to the characteristics of a record. For instance, the fields of the table below include:

- ✓ Student ID
- ✓ First name
- ✓ Surname
- ✓ Date of Birth
- ✓ Form tutor

Fields

Student ID	First Name	Surname	Date of Birth	Form Tutor
712	Bart	Simpson	1/4/10	Principal Skinner
423	Lisa	Simpson	20/5/12	Mrs Krabapple
917	Ralph	Wiggum	16/6/10	Mrs Krabapple
124	Nelson	Muntz	14/9/09	Principal Skinner

Data Redundancy

Data redundancy occurs when the same data are stored in multiple places and so we have repeating data. As a result more space is needed to store the same values several times which is not efficient. In the table below notice how the Author Name fields are repeated.

BookID	Title	FirstName	Surname
1	Fantastic Beasts and Where to Find Them	J.K.	Rowling
2	Harry Potter and the Chamber of Secrets	J.K.	Rowling
3	Harry Potter and Order of the Phoenix	J.K.	Rowling
4	The BFG	Roald	Dahl
5	Going Solo	Roald	Dahl
6	Danny Champion of the World	Roald	Dahl
7	War Horse	Michael	Morpurgo
8	Private Peaceful	Michael	Morpurgo

Data Inconsistency

Data inconsistency occurs when data pertaining to the same object are in fact stored in a different format. For instance, JK. Rowling and Joanne Rowling refer to the same person, but the database may record these as two separate authors.

BookID	Title	FirstName	Surname
1	Fantastic Beasts and Where to Find Them	JK	Rowling
2	Harry Potter and the Chamber of Secrets	Joanne	Rowling
3	Harry Potter and Order of the Phoenix	Joanne	Rowling
4	The BFG	Roald	Dahl
5	Going Solo	Roald	Dahl
6	Danny Champion of the World	Roald	Dahl
7	War Horse	Michael	Morpurgo
8	Private Peaceful	Michael	Morpurgo

Relational Databases

Complex databases can be made up of multiple tables linked together by shared values called a key. These relational databases make it easier to search and find information that you want. Relational databases reduce the amount of duplication (redundancy) of data and reduces inconsistencies in the data.

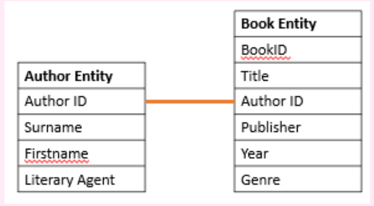
Primary Key

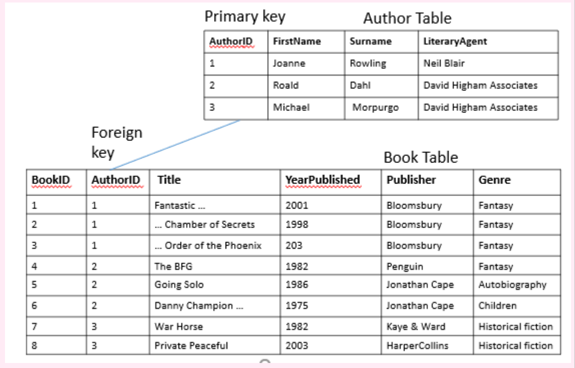
All tables have a field that is the primary key and uniquely identifies each record. This is also known as entity identifier

Foreign Key

These are primary keys that are held as fields in other tables to cross reference tables. They allow tables to be linked together.

For instance, in a book database with two tables including Author table and Book table, AuthorID is primary key in Author table and is used to cross-reference with the AuthorID in the book table which is the foreign key so the two tables can be linked.





Author Table

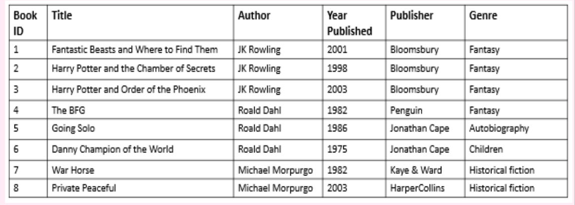
AuthorID	FirstName	Surname	LiteraryAgent
1	Joanne	Rowling	Neil Blair
2	Roald	Dahl	David Higham Associates
3	Michael	Morpurgo	David Higham Associates

Book Table

BookID	AuthorID	Title	YearPublished	Publisher	Genre
1	1	Fantastic …	2001	Bloomsbury	Fantasy
2	1	… Chamber of Secrets	1998	Bloomsbury	Fantasy
3	1	… Order of the Phoenix	203	Bloomsbury	Fantasy
4	2	The BFG	1982	Penguin	Fantasy
5	2	Going Solo	1986	Jonathan Cape	Autobiography
6	2	Danny Champion …	1975	Jonathan Cape	Children
7	3	War Horse	1982	Kaye & Ward	Historical fiction
8	3	Private Peaceful	2003	HarperCollins	Historical fiction

Structured Query Language

We will use this book table in the examples that follow.



Book ID	Title	Author	Year Published	Publisher	Genre
1	Fantastic Beasts and Where to Find Them	JK Rowling	2001	Bloomsbury	Fantasy
2	Harry Potter and the Chamber of Secrets	JK Rowling	1998	Bloomsbury	Fantasy
3	Harry Potter and Order of the Phoenix	JK Rowling	2003	Bloomsbury	Fantasy
4	The BFG	Roald Dahl	1982	Penguin	Fantasy
5	Going Solo	Roald Dahl	1986	Jonathan Cape	Autobiography
6	Danny Champion of the World	Roald Dahl	1975	Jonathan Cape	Children
7	War Horse	Michael Morpurgo	1982	Kaye & Ward	Historical fiction
8	Private Peaceful	Michael Morpurgo	2003	HarperCollins	Historical fiction

Select

To retrieve data from the table

To retrieve all records data from the table we can use the SELECT statement with the wild card operator *.

```
SELECT *
FROM tableName
```

Examlpe

```
SELECT *
FROM book
```

The Wellington Academy

proud to be part of

ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Retrieved data

1	Fantastic Beasts ..	JK Rowling	2001	Bloomsbury	Fantasy
2	..Chamber of Secrets	JK Rowling	1998	Bloomsbury	Fantasy
3	.. Order of the Phoenix	JK Rowling	2003	Bloomsbury	Fantasy
4	The BFG	Roald Dahl	1982	Penguin	Fantasy
5	Going Solo	Roald Dahl	1986	Jonathan Cape	Autobiography
6	Danny Champion ..	Roald Dahl	1975	Jonathan Cape	Children
7	War Horse	Michael Morpurgo	1982	Kaye & Ward	Historical fiction
8	Private Peaceful	Michael Morpurgo	2003	HarperCollins	Historical fiction

We can also choose the fields that we wish to retrieve:

```
SELECT field1, field2, …
FROM tableName
```

Example

```
SELECT Author, Title
FROM book
```

Retrieved data

Fantastic Beasts and Where to Find Them	JK Rowling
Harry Potter and the Chamber of Secrets	JK Rowling
Harry Potter and Order of the Phoenix	JK Rowling
The BFG	Roald Dahl
Going Solo	Roald Dahl
Danny Champion of the World	Roald Dahl
War Horse	Michael Morpurgo
Private Peaceful	Michael Morpurgo

We can sort the output of our SELECT statement by using the ORDER BY clause. ASC and DESC refer to sorting ascending and descending alphabetically or numerically of a specified field.

```
ORDER BY fieldname ASC|DESC
```

Example Sort ascending

```
SELECT Author, Title
FROM book
ORDER BY Title ASC
```

Danny Champion of the World	Roald Dahl
Fantastic Beasts and Where to Find Them	JK Rowling
Going Solo	Roald Dahl
Harry Potter and the Chamber of Secrets	JK Rowling
Harry Potter and Order of the Phoenix	JK Rowling
Private Peaceful	Michael Morpurgo
The BFG	Roald Dahl
War Horse	Michael Morpurgo

Example Sort Descending

```
SELECT Author, Title
FROM book
ORDER BY Title DESC
```

War Horse	Michael Morpurgo
The BFG	Roald Dahl
Private Peaceful	Michael Morpurgo
Harry Potter and Order of the Phoenix	JK Rowling
Harry Potter and the Chamber of Secrets	JK Rowling
Going Solo	Roald Dahl
Fantastic Beasts and Where to Find Them	JK Rowling
Danny Champion of the World	Roald Dahl

WHERE Clause

We can filter our selection using the `WHERE` clause

```
WHERE fieldname operator value
```

Operator	Description
=	Value equal to
!=	Value not equal to
<	Value less than
>	Value greater than
<=	Value less than or equal to
>=	Value greater than or equal to

SELECT using WHERE clause

EXAMPLE 1 – Select books written since 2000

```
SELECT Title, Author, yearPublished
FROM book
WHERE YearPublished > 2000
```

Fantastic Beasts and Where to Find Them	JK Rowling	2001
Harry Potter and Order of the Phoenix	JK Rowling	2003
Private Peaceful	Michael Morpurgo	2003

EXAMPLE 2 – Select books written by Michael Morpurgo

```
SELECT Title, Author
FROM book
WHERE Author = "Michael Morpurgo"
```

Notice how the author name is in speech marks because it is a string datatype.

War Horse	Michael Morpurgo
Private Peaceful	Michael Morpurgo

Example 3 – Select by date

```
WHERE Date < #1/1/2010#
```

For data type date you need to use #. Eg

Boolean Operators

We can use Boolean and relational operators with the WHERE clause if we have multiple conditions that need to be met.

Operator	Description
OR	Allows us to combine multiple conditions. Any of the conditions can be true for the overall expression to return true
AND	Allows us to combine multiple conditions. All conditions need to be true for the overall expression to return true
NOT	Reverses the value of a condition. If it is true it will be false and vice versa

EXAMPLE – Select all books written by Michael Morpurgo since 2016

```
SELECT Title, Author FROM book
WHERE Author="Michael Morpurgo"
AND YearPublished > 2000
```

Private Peaceful	Michael Morpurgo

Update - To update records in a database

To make changes to a record that is already in a table we can use the `UPDATE` statement.

EXAMPLE 1: Update the book table to change the genre of all fields to Children

```
UPDATE book
SET Genre="Children"
```

EXAMPLE 2: Update the book table to change the author name from JK Rowling to Joanne Rowling.

```
UPDATE book
SET Author="Joanne Rowling"
WHERE Author="JK Rowling"
```

Book ID	Title	Author	Year Published	Publisher	Genre
1	Fantastic Beasts .	Joanne Rowling	2001	Bloomsbury	Children
2	Harry Potter ..	Joanne Rowling	1998	Bloomsbury	Children
3	Harry Potter ..	Joanne Rowling	2003	Bloomsbury	Children
4	The BFG	Roald Dahl	1982	Penguin	Children
5	Going Solo	Roald Dahl	1986	Jonathan Cape	Children
6	Danny .	Roald Dahl	1975	Jonathan Cape	Children
7	War Horse	Michael Morpurgo	1982	Kaye & Ward	Children
8	Private Peaceful	Michael Morpurgo	2003	HarperCollins	Children

INSERT INTO - Adding new records

INSERT INTO is a commonly used command in SQL for adding new records to database tables. To insert all attributes for a table we can use:

```
INSERT INTO table
VALUES(value1, value2,…)
```

Example

```
INSERT INTO book
VALUES ('Boy', 'Roald Dahl', 1984, 'Penguin',
'Autobiography')
```

Sometimes we do not enter data into every field. Instead we can explicitly state which fields we would like to add the data to.

```
INSERT INTO table (field1, field2,…)
VALUES(value1, value2,…)
```

The values correspond to the fields in the table i.e.:

- ✓ Field 1: Book ID
- ✓ Field 2: Title
- ✓ Field 3: Author
- ✓ Field 4: YearPublished
- ✓ Field 5: Publisher
- ✓ Field 6: Genre

Example

```
INSERT INTO book (Title, Author, YearPublished,
Publisher, Genre) VALUES ('Boy', 'Roald Dahl', 1984,
'Penguin', 'Autobiography')
```

Deleting records

To delete a record we specify which record(s) from which table we wish to remove.

```
DELETE FROM table WHERE condition
```

COMPUTER SCIENCE KNOWLEDGE ORGANISER

EXAMPLES

Remove all books

```
DELETE FROM book
DELETE * FROM book
```

The `WHERE` clause is used to filter records so that we do not apply a statement to a whole table.

Remove all books written by JK Rowling:

```
DELETE FROM book WHERE Author=‘JK Rowling’
```

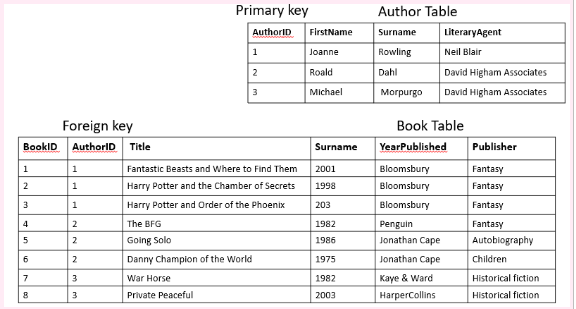
Remove all books written by Michael Morpurgo and written before 2000

```
DELETE FROM book WHERE Author=‘Michael Morpurgo’ AND
YearPublished < 2000
```

SELECT ATTRIBUTES FROM MULTIPLE TABLES

So far we have looked at a database made up of a single table. databases can be made up of multiple tables. We can link tables together using primary keys and foreign keys. We can use SQL statements to select data from multiple tables. When selecting the data from multiple tables we need to specify the name of the table from which each attribute we are wishing to retrieve.

We will use the following database table as an example case study.



Primary key

Author Table

AuthorID	FirstName	Surname	LiteraryAgent
1	Joanne	Rowling	Neil Blair
2	Roald	Dahl	David Higham Associates
3	Michael	Morpurgo	David Higham Associates

Foreign key

Book Table

BookID	AuthorID	Title	Surname	YearPublished	Publisher
1	1	Fantastic Beasts and Where to Find Them	2001	Bloomsbury	Fantasy
2	1	Harry Potter and the Chamber of Secrets	1998	Bloomsbury	Fantasy
3	1	Harry Potter and Order of the Phoenix	203	Bloomsbury	Fantasy
4	2	The BFG	1982	Penguin	Fantasy
5	2	Going Solo	1986	Jonathan Cape	Autobiography
6	2	Danny Champion of the World	1975	Jonathan Cape	Children
7	3	War Horse	1982	Kaye & Ward	Historical fiction
8	3	Private Peaceful	2003	HarperCollins	Historical fiction

We need to specify that we only wish to select the records where the primary key and foreign key match.

EXAMPLES

Retrieve data book title and author surname

```
SELECT book.Title, author.Surname
FROM author, book
WHERE author.AuthorID=book.AuthorID
```

Fantastic Beasts and Where to Find Them	JK Rowling
Harry Potter and the Chamber of Secrets	JK Rowling
Harry Potter and Order of the Phoenix	JK Rowling
The BFG	Roald Dahl
Going Solo	Roald Dahl
Danny Champion of the World	Roald Dahl
War Horse	Michael Morpurgo
Private Peaceful	Michael Morpurgo

Retrieve book title and author surname where genre is *fantasy*

```
SELECT book.title, author.surname
FROM author, book
WHERE author.AuthorID=book.AuthorID
AND book.Genre=“Fantasy”
```

Fantastic Beasts and Where to Find Them	JK Rowling
Harry Potter and the Chamber of Secrets	JK Rowling
Harry Potter and Order of the Phoenix	JK Rowling
The BFG	Roald Dahl

Retrieve book title and author surname where genre is fantasy and sort in descending order Title

```
SELECT book.title, author.surname
FROM author, book
WHERE author.AuthorID=book.AuthorID
AND book.Genre=“Fantasy”
ORDER BY title DESC
```

The BFG	Roald Dahl
Harry Potter and Order of the Phoenix	JK Rowling
Harry Potter and the Chamber of Secrets	JK Rowling
Fantastic Beasts and Where to Find Them	JK Rowling

The Wellington Academy

proud to be part of

ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Ethical, Legal and environmental impacts of digital technology on society

The Ten Commandments of Computer Ethics (From the Computer Ethics Institute)

Thou shalt:

1. not use a computer to harm other people
2. not interfere with other people's computer work
3. not snoop around in other people's computer files
4. not use a computer to steal
5. not use a computer to bear false witness
6. not copy or use proprietary software for which you have not paid (without permission)
7. not use other people's computer resources without authorization or proper compensation
8. not appropriate other people's intellectual output
9. think about the social consequences of the program you are writing or the system you are designing
10. always use a computer in ways that ensure consideration and respect for other humans

Environmental Impacts

- The disposal of computer waste is a big problem because they contain many toxic chemicals. Often old computing equipment is illegally shipped for disposal to developing countries.
- The growth in cloud computing means a greater need for storing data online. For this data centres are used but they require huge amounts of electricity, thereby contributing to climate change.
- Cobalt is a key element requited for Lithium batteries for powering mobile devices. Much of the World's cobalt is mined in the Congo even by very young children in appalling conditions.

Environmental benefits

- Less reliance on paper saving resources
- More opportunity for online global communication and collaboration thereby saving on travel and associated pollution
- Greater insight of environment and climate through using computer to model and analyse and process environmental data

Legislation

Computer Misuse Act (CMA)
The purpose of the CMA is to prevent:

- unauthorised access to computers by hackers
- intentionally impairing the operation of computer systems through denial of service (DOS) attacks on web servers or distributing viruses
- the theft of data

Three levels of offence:

1) Unauthorised access
2) Unauthorised access with intent to commit an offence
3) Unauthorised modification of data

Copyright, Designs and Patents Act (CDPA)

Copyright is a law that protects the creators of original pieces of work. No one else has the right to use or copy it without permission from the owner. This ensures that people can be rewarded for their work.

Plagiarism To pass off some else's work as one's own work.

Patent An inventor has the exclusive right to create, use and sell an invention for fixed period

Piracy Illegally copying and distributing copyrighted material.

Fair use allows copyrighted work to be used legally in certain situations

- personal or educational use (not commercial use)
- use only a small amount of the work (e.g. a short quote)
- acknowledge original source of the work

Copyleft work can be copied, modified used even used for commercial gain as long as the derived works are also distributed under copyleft.

Creative Common Licences (CCL) The creator of the work has explicitly given anyone permission to use the work.

Investigatory Powers Act This is legislation that allows public authorities to carry out mass surveillance on electronic communications.

Justification - By monitoring electronic communications security services can keep us safe from terrorists and other serious criminals

Concerns - Can infringe on our privacy and civil liberties

In a liberal democracy there will always a need to balance security and privacy, but where we draw that line will always be a matter of debate.

Some powers of the security services under the IPA

- can hack into computers, networks, mobile devices, servers
- internet service providers have to store which websites users visit for 12 months and allow access to authorities when requested
- carry out mass surveillance of communications; authorities can collect bulk data including data about people who are not suspected of anything.
- demand that an internet service provider provide access to a customer's communications including keys to encrypted data

General Data Protection Regulation (GDPR)

The purpose of the GDPR is to ensure that personal information collected by businesses and other organisations are protected.

Personal data is defined as anything that allows an individual to be identified (e.g. name, biometric data)

Six principles of the GDPR
Personal information must:

- be used fairly and lawfully
- be used only for specific purposes for which it was collected
- be adequate, relevant and not excessive
- be accurate and kept up to date
- be kept for longer than is necessary and deleted when it is no longer needed
- be kept secure against unauthorised access

Other aspects of the GDPR

- The data subject needs to be notified if their data are shared with other organisations
- Obtain consent from the data subject to their process data
- Obtain consent from parents or guardians to process children's data.
- Allow data subjects to have their data removed
- Allow data subjects to access the data held about them
- Pay big fines for a breach of the GDPR

Other Social Impacts

Artificial Intelligence is replacing people in jobs. More hi-tech jobs but less need for many lower skilled jobs.

The **digital divide** refers to the unequal access to information technology between different groups of people, and the knowledge and skills needed to use the technology.

Online trolling, cyber bulling and fake news on social media sites is undermining freedom of expression

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Programming - Python

Comment – Text within the code that is ignored by the computer. A Python comment is preceeded by a #.

```
# This is an example of a comment
```

Output – Processed information that is sent out from a computer

Python	Pseudocode

`print("Hello World!")`
Hello World!
`print("Hello", "World!")`
Hello World!
`print("Hello"+"World!")`
HelloWorld!
`print("Hello\nWorld!")`
Hello
World!
	`OUTPUT "Hello World"`

Input – Data sent to a computer to be processed

`print("Enter name")`	`OUTPUT "Enter name"`
`name=input()`	`name ← USERINPUT`
`print("Hello", name)`	`OUTPUT "Hello", name`
`print("Enter age")`	`OUTPUT "Enter age"`
`age=int(input())`	`age ← USERINPUT`

Assignment - The allocation of data values to variables, constants, arrays and other data structures so that the values can be stored.

- *Variable* – Value that can change during the running of a program. By convention we use lower case to identify variables (eg `a=12`)
- *Constant* – Value that remains unchanged for the duration of the program. By convention we use upper case letters to identify constants. (e.g. `PI=3.141`)

Data Types

Integer – Whole number	`age = 12`	`age ← 12`
Float (real) number – A number with a decimal point	`height = 1.52`	`height ← 12`
Character – A single letter, symbol or number	`a = 'a'`	`a ← 'a'`
String – multiple characters	`name = "Bart"`	`name ← "Bart"`

Boolean – Has two values: true of false.	`a = True`
	`b = False`
		`a ← True`
		`b ← False`

Arithmetic Operators

Add	`7 + 2    = 9`	`7 + 2`
Subtract	`7 - 2    = 5`	`7 - 2`
Multiply	`7 * 2    = 14`	`7 * 2`
Divide	`4 / 2    = 2`	`4 / 2`
power	`2 ** 3   = 8`	`2 ** 3`
Integer division	`7 // 2   = 3`	`7 DIV 2`
Modulus (remainder)	`7 % 2    = 1`	`7 MOD 2`

Relational Operators – Allows the Comparison of values

Less than	<	<	7<2	-> False
Greater than	>	<	7 > 2	-> True
Equal to	==	==	7==2	-> False
Not equal to	!=	≠ or <>	7!=2	-> True
Less than or equal to	<=	≤	7<=2	-> False
Greater than or equal to	>=	≥	7>=2	-> True

Boolean Operators

AND	and	`7 < 2 and 1 < 2`	-> False
OR	or	`7 < 2 or 1 < 2`	-> False
NOT	not	`not 7 < 2`	-> True

Sequencing represents a set of steps. Each line of code will have some operation and these operations will be carried out in order line-by-line

Using + operator for adding
 	
`a = 1`
`b = 2`
`c = a + b`
`print(c)   -> 3`
	`a ← 1`
	`b ← 2`
	`c ← a + b`
	`OUTPUT c`

Using + operator for concatenation
 	
`a = 'Hello '`
`b = 'World'`
`c = a + b`
`print(c) -> Hello World`
	`a ← 'Hello '`
	`b ← 'World'`
	`c ← a + b`
	`OUTPUT c`

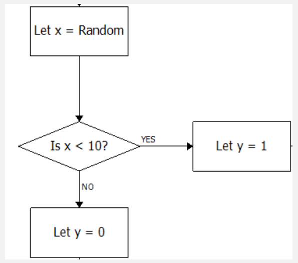
Random number

Random integer	`import random`
	`random.randint(0,9)`
		`RANDOM_INT(0,9)`

Choice	`random.choice('a','b','c')`	
Random value from 0 to 1	`random.random()`	

Selection represents a decision in the code according to some condition. The condition is met then the block of code is executed otherwise it is not. Often alternative blocks of code are executed according to some condition.

```
x=RANDOM_INT()
IF  x < 10 THEN
 y=1
ELSE
 y=0
ENDIF
```



IF …	`IF i > 2 THEN`
	` j ← 10`
	`ENDIF`
		`if i  > 2:`
		` j=10`

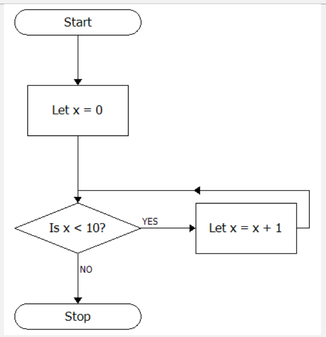
IF … ELSE …	`IF i > 2  THEN`
	` j ← 10`
	`ELSE`
	` j ← 3`
	`ENDIF`
		`if i > 2:`
		`   j=10`
		`else:`
		`   j=3`

IF … ELSE IF … ELSE	`IF i ==2 THEN`
	` j ← 10`
	`ELSE IF i==3`
	` j ← 3`
	`ELSE`
	` j ← 1`
	`ENDIF`
		`if i ==2:`
		` j=10`
		`elif i==3:`
		` j=3`
		`else:`
		` j=1`

Iteration Sometimes we wish the code to repeat a set of instructions

`WHILE` loops are used when the we do not know beforehand the number of iterations needed and this varies according to some condition.

```
x = 0
while (x < 10):
      x = x + 1
```



`while True:`
` print("Hello World")`
	`WHILE TRUE`
	` OUTPUT "Hello World"`
	`ENDWHILE`

`a=0`
`while a<4:`
` print(a)`
` a=a+3`
	`a ← 0`
	`WHILE a < 4`
	` OUTPUT a`
	` a ← a + 3`
	`ENDWHILE`

`FOR` loops are used when we know before hand the number of iterations we wish to make.

`for a in range(3):`
` print(a)`
	`FOR a ← 0 TO 3`
	`    OUTPUT a`
	`ENDFOR`

The Wellington Academy

proud to be part of ROYAL WOOTTON BASSETT ACADEMY TRUST

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Nested structures - Use constructs (e.g. `WHILE, FOR, IF`) inside another.

use a nested FOR loop to print out a grid	`for i in range (10):`
	 `for i in range (10):`
	  `print ("x ",end="")`
	 `print()`

Use a nested while and if to print out only even numbers	`i=0`
	`while i<51:`
	 `if (i%2==0):`
	  `print(i)`
	 `i=i+1`

Lists

Create a list	`shapes=["square","circle"]`
Access element by index pos	`shapes[1] -> circle`
Append item to list	`shapes.append(“triangle”)`
Remove item from list	`shapes.remove(“circle”)`
Remove item from list by index	`shapes.pop(1)`
Insert item into list	`shapes.insert(2,”rectangle”)`
Number of elements in a list	`len(shapes)`
Get index pos of item in list	`shapes.index(“triangle”)`

Concatenating lists	`shapesGroup1[“square”,”circle”]`
	`shapesGroup2=[“triangle”]`
	`shapes=shapesGroup1+shapesGroup2`

Loop through list	`for i in range(len(shapes)):`
	 `print(shapes[i])`

Reverse elements in a list	`shapes.reverse()`
Order elements in a list	`shapes.sort()`

2D lists - A list if lists

Create a 2D list	`d = [ [23, 14, 17], [12, 18, 37], [16, 67, 83]]`

Another way to create a 2D list	`a = [23, 14, 17]`
	`b = [12, 18, 37]`
	`c = [16, 67, 83]`
	`d = [a,b,c]`

Access element by index position	`d[1][2] -> 37`

Strings

Get length of a string	`len(“Hello”)`	`LEN(“Hello”)`
Character to character code	`ord("a") -> 97`	`ORD("a")`
Character code to character	`chr(101) -> ‘e’`	`CHR(101)`
String to integer	`a=int(“12”)`	`a=INT(“12”)`
String to float	`a=float(“12.3”)`	`a=FLOAT(“12.3”)`
integer to string	`a=str(12)`	`a=STR(12)`
real to string	`a=str(12.3)`	`a=STR(12.3)`

Concatenation -merge multiple strings together	`a=“hello ”`
	`b=“world”`
	`c=a+b`
	`print(c) ->`
	`hello world`

Return the position of a character If there is more than 1 of the same character the position of the first character is returned.	`student = “Hermione"`
	`student.index(‘i')`

Find the character at a specified position	`student = “Hermione"`
	`print(student[2]) -> r`

sub strings - select parts of a string

Example	`student=“Harry Potter”`	
Output the first two characters	`print(student[0:2])`	Ha
Output the first three characters	`print(student[:3])`	Har
Output characters 2-4	`print(student[2:5])`	Rry
Output the last 3 characters	`print(student[-3:])`	Ter
Output a middle set of characters	`print(student[4:-3])`	y Pot

*A negative value is taken from the end of the string.

Subroutines are a way of managing and organising programs in a structured way. This allows us to break up programs into smaller chunks.

- Can make the code more modular and more easy to read as each function performs a specific task.
- Functions can be reused within the code without having to write the code multiple times.

- **Procedures** are subroutines that do not return values
- **Functions** are subroutines that have both input and output

Procedure: No input parameters or return	`SUB greeting()`
	 `OUTPUT “hello”`
	`ENDSUB`
		`def greeting():`
		 `print(“hello”)`
		
		`call: greeting()`

Procedure: One input parameter, no return	`SUB greeting(name)`
	 `OUTPUT “Hello”,name`
	`ENDSUB`
		`def greeting(name):`
		 `print(“Hello",name)`
		
		`greeting(“grey”)`

Function: 1 input parameter, and 1 return value	`SUB add(n)`
	 `a ← 0`
	 `FOR a ← 0 TO n`
	  `a ← a + n`
	 `ENDFOR`
	 `RETURN a`
	`ENDSUB`
		`def add(n):`
		 `a=0`
		 `for a in range(n+1):`
		 `a=a+n`
		 `return a`

Function: Two input parameters, and 1 return value	`SUB (num1,num2)`
	 `sum=num1+num2`
	 `return sum`
		`def add(num1,num2):`
		 `sum=num1+num2`
		 `return sum`
		
		`greeting(1,2)`

The **scope** of a variable determines which parts of a program can access and use that variable.

A **global variable** is a variable that can be used anywhere in a program. The issue with global variables is that one part of the code may inadvertently modify the value because global variables are hard to track.

A **local variable** is a variable that can only be accessed within a certain block of code typically within a function. Local variables are not recognized outside a function unless they are returned. There is no way of modifying or changing the behavior of a local variable outside its scope.

Global variables need to defined throughout the running of the whole program. This is an inefficient use of memory resources. Local variables are defined only when they are needed an so have less demand on memory. Local variables only exist within the subroutine.

Reading and writing files

Open file Whatever we are doing to a file whether we are reading, writing or adding to or modifying a file we first need to open it using:

```
open(filename,access_mode)
```

There are a range of access mode depending on what we want to do to the file, the principal ones are given below:

Access Mode	Description
r	Opens a file for reading only
w	Opens a file for writing only. Create a new file if one does not exist. Overwrites file if it already exists.
a	Append to the end of a file. Create a new file if one does not exist.

Reading text files

read – Reads in the whole file into a single string	`f=open("filetxt","r")`
	`print(f.read())`
	`f.close()`

readline – Reads in each line one at a time	`f=open("file.txt","r")`
	`print(f.readline())`
	`print(f.readline())`
	`print(f.readline())`
	`f.close()`

readlines – Reads in the whole file into a list	`f=open("file.txt","r")`
	`print(f.readlines())`
	`f.close()`

Writing text files

Write in single lines at a time	`file=open("days.txt",'w')`
	`file.write("Monday\n")`
	`file.write("Tuesday\n")`
	`file.write("Wednesday\n")`
	`file.close()`

Write in a list	`say=["How\n”,”are\n”,”you\n”]`
	`file=open("say.txt",'w')`
	`file.writelines(say)`
	`file.close()`

COMPUTER SCIENCE KNOWLEDGE ORGANISER

Data Validation Routines

Check if an entered string has a minimum length	`OUTPUT "Enter String"`
	`s ← USERINPUT`
	`IF LEN(S) > 5 THEN`
	` OUTPUT "STRING OK"`
	`ELSE`
	` OUTPUT "TOO SHORT"`
	`ENDIF`

Check is a string is empty	`OUTPUT "Enter String"`
	`s ← USERINPUT`
	`IF LEN(S) == 0 THEN`
	` OUTPUT "EMPTY STRING"`
	`ENDIF`

Check if data entered lies within a given range	`OUTPUT "Enter number" s num ←`
	`USERINPUT`
	`IF num > 1 AND num < 10`
	` OUTPUT "Within range"`
	`ENDIF`

Authentication Routine

```
OUTPUT "Enter Username"
username ← USERINPUT
OUTPUT "Enter Password"
password ← USERINPUT

WHILE username != "bart" OR password !="abc"

 OUTPUT "Login failed"
 OUTPUT "Enter Username"
 username ← USERINPUT
 OUTPUT "Enter Password"
 password ← USERINPUT

ENDWHILE

OUTPUT "Login Successful"
```

Debugging

Syntax errors – Errors in the code that mean the program will not even run at all. Normally this is things like missing brackets, spelling mistakes and other typos.

Runtime errors – Errors during the running of the program. This might be because the program is writing to a memory location that does not exist for instance. eg. An array index value that does not exist.

Logical errors - The program runs to termination, but the output is not what is expected. Often these are arithmetic errors.

Test data

Code needs to be tested with a range of different input data to ensure that it works as expected under all situations. Data entered need to be checked to ensure that the input values are:

- within a certain range
- in correct format
- the correct length
- The correct data type (eg float, integer, string)

The program is tested using normal, erroneous or boundary data.

Normal data - Data that we would normally expect to be entered. For example for the age of secondary school pupils we would expect integer values ranging from 11 to 19.

Erroneous data - Data that are input that are clearly wrong. For instance, if some entered 40 for the age of a school pupil. The program should identify this as invalid data but at the same time should be able to handle this sensibly which returns a sensible message and the program does not crash.

Boundary data - Data that are on the edge of what we might expect. For instance if someone entered their age as 10, 11, 19 or 20.