AQA Computer Science

AQA COMPUTER SCIENCE

CHEAT SHEET GUIDES TO SUPPORT LEARNING

BY MRS FRANKS

AQA COMPUTER SCIENCE

CHEAT SHEET GUIDES TO SUPPORT LEARNING

This cheat sheet book has been created to support learning in AQA Computer Science lessons. It is not endorsed by AQA in any way. However, they have commented about my resources on twitter:

AQA Computer Science @AQACompIT 24h

Interesting and informative blog full of helpful teaching resources @skyrocket_2005 @casinclude rfranksblog.wordpress.com

Retweeted by R Franks

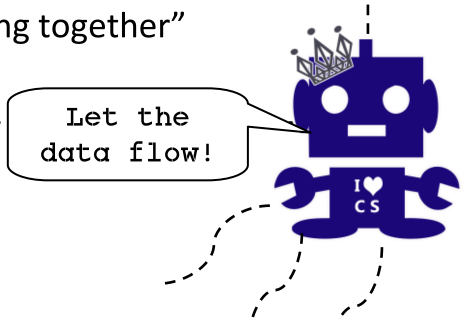
View summary

This guide goes through each topic in the AQA scheme for learning. The guides are designed to jog your memory or explain things in a simple manner so that the concepts can be grasped.

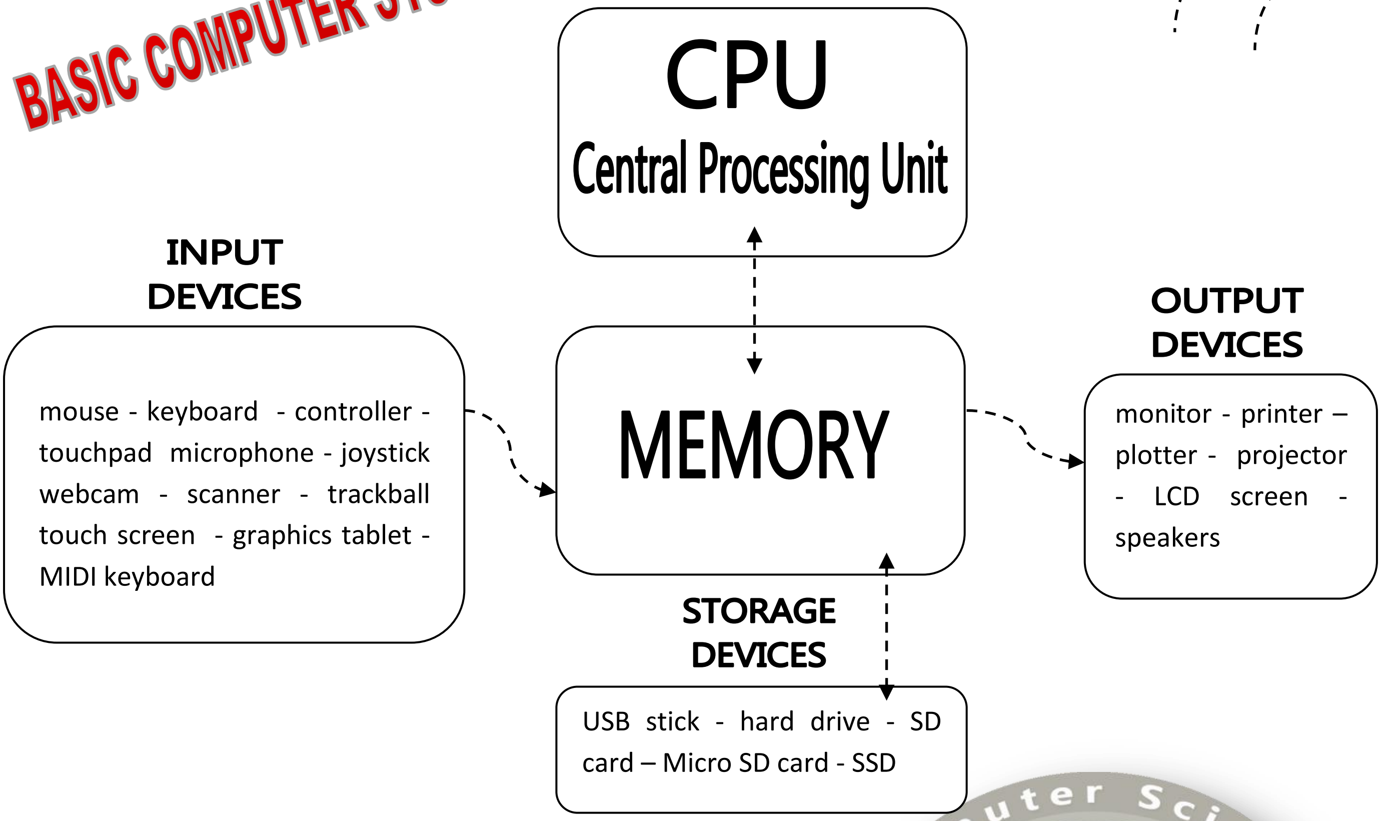
They do not cover ALL the aspects of AQA computer science. Lessons should go into much more depth than in this book. They are just an aid to support learning. There are 35 topics and these are listed below in the order of this book:

Define: Computer System

"A combination of hardware and software working together"



BASIC COMPUTER SYSTEM DIAGRAM!



CPU

Central Processing Unit

INPUT DEVICES

mouse - keyboard - controller - touchpad microphone - joystick webcam - scanner - trackball touch screen - graphics tablet - MIDI keyboard

MEMORY

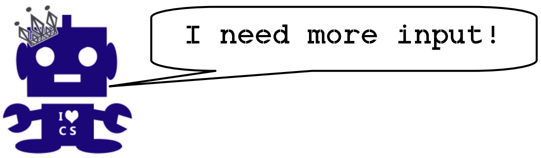
OUTPUT DEVICES

monitor - printer – plotter - projector - LCD screen - speakers

STORAGE DEVICES

USB stick - hard drive - SD card – Micro SD card - SSD

The input device sends data into the CPU via the memory. Input can be in many different forms. The CPU then processes the data and as a result, displays output. As with input, output comes in many forms.



The next time you use a games console/laptop/mobile, think about this diagram!

AQA Computer Science :: Cheat Sheet

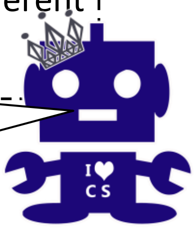
Topic 2 :: Expressions and Types

Define: Expressions

“A combination of precise values, constants, variables, operators, and functions that are interpreted according to the particular rules of priority”

Define: Types

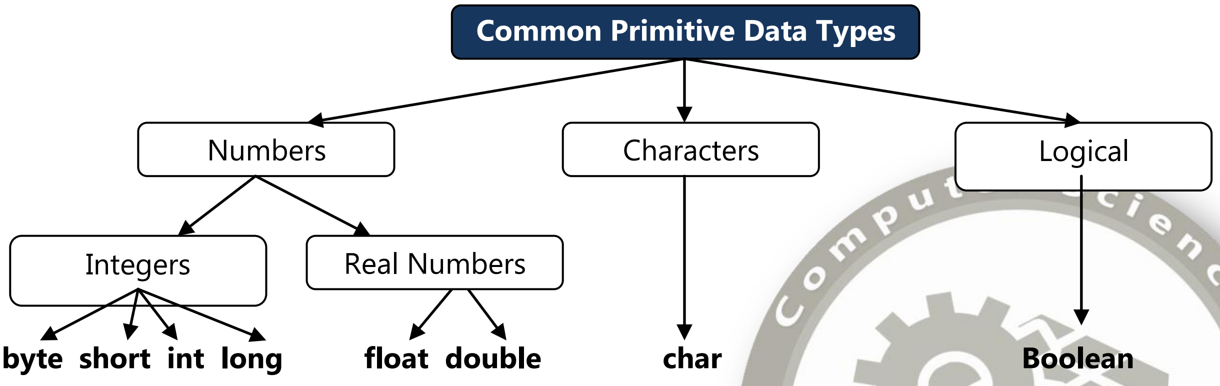
“Also known as a data type, it is the name used to describe the different types of data used in a program”



What are values, variables, operators and functions?

We will learn much more about these later on. But here is a quick explanation.

Value = a value
Constants = something that stays the same
Variables = something that changes depending on a condition
Operators = +, -, *, /, power (these are the main ones)
Function = A portion of code within a larger program



What data types do the following belong to?

1 Yes abc 3.4 No

Don’t know? Find out!

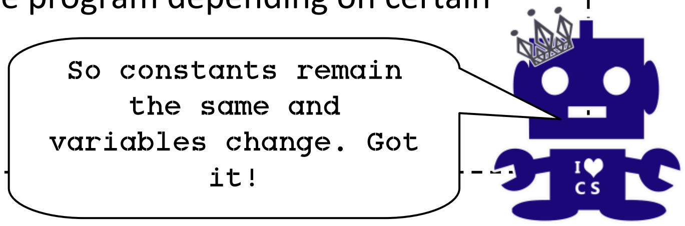
Topic 3 :: Constants and Variables

Define: Constant

"As the word suggests. A constant is something that stays the same throughout the program"

Define: Variable

"A variable changes throughout the program depending on certain conditions"



Constants are useful for making your code simpler. Also, once you have declared the constant, it can be changed by the programmer if required. Declaring the constant at the start means that you don't have to go through the entire program, changing each occurrence in turn.

E.g. A game was designed where the **constant** speed of the character was set to 5. After testing the game, the user thought the speed wasn't challenging enough so the programmer changed the **constant** speed to 10. Once defined, a constant remains the same throughout the execution of the application/game.

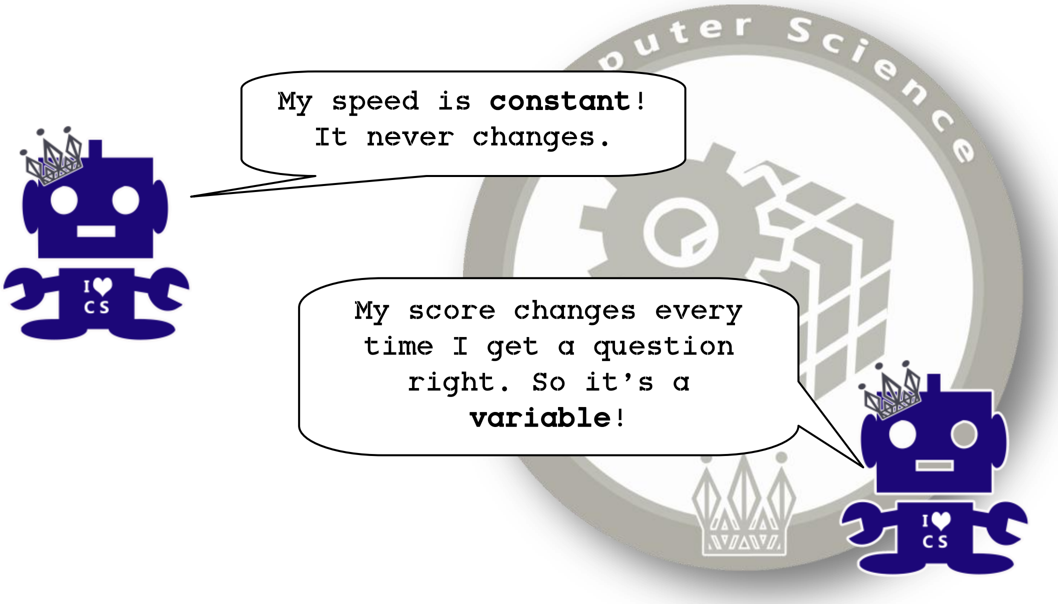
Variables are used when a value needs to change during the execution of the program.

E.g. A game has a scoring system. When the character shoots a target, 1 is added to the score. Score is a **variable**; it changes every time a target is hit. If a value changes within a program then it is called a **variable**.

____

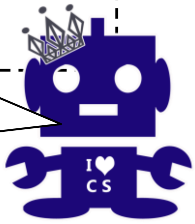
____

____


My speed is **constant**! It never changes.
My score changes every time I get a question right. So it's a **variable**!

Define: Selection

“When a path through the program is selected based on a condition”



```
IF I eat cheese tonight THEN
    I will have nightmares
ELSE
    I will sleep well
ENDIF
```

When using selection in a program we can either use **IF...THEN...ELSE** or we can use a **CASE** statement.

Here is an example of an **IF...THEN...ELSE** statement:

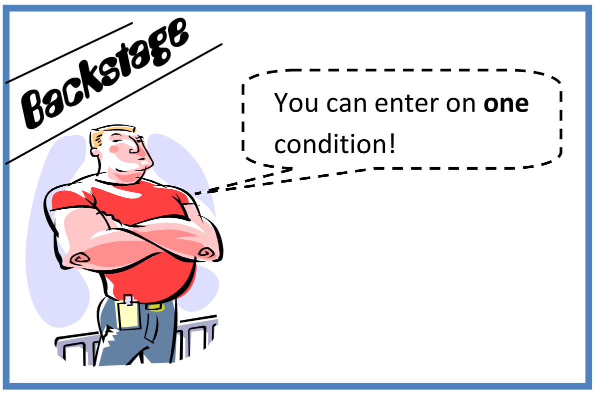
```
IF X > 10 THEN
    OUTPUT “correct”
ELSE
    OUTPUT “incorrect”
ENDIF
```

If whatever is in X is more than 10 then say “Correct”, otherwise say “Incorrect”.

Here is an example of a **CASE** statement:

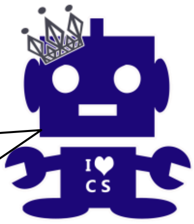
```
CASE num OF
    1: OUTPUT “One”
    2: OUTPUT “Two”
    3: OUTPUT “Three”
    4: OUTPUT “Four”
ELSE
    OUTPUT “Out of range”
ENDCASE
```

The user types in a number and it is stored as *num*. If the number is in the list then it displays the matching output. E.g. the user types in 3 so it outputs *Three*. If it’s not in the list it says “Out of range”.



Define: Repetition

“Repetition is used when the same bit of code is needed several times. Instead of writing it over and over again, you can use the repeat command. Repetition can also be called looping.”



10 green bottles sitting on the wall, 10 green bottles sitting on the wall, and if 1 green bottle should accidently fall. There’ll be 9 green bottles sitting on the wall.

Hold on a minute, why am I repeating myself?

The three most common loops are a **FOR** loop, a **WHILE** loop or a **REPEAT UNTIL** loop.

FOR	FOR i ← 1 TO 5
	    OUTPUT i
	ENDFOR
	
	# this will output: 1, 2, 3, 4 and 5
		For as long as the variable i is 1,2,3,4 or 5. Display the variable i. (each time this loop runs it will add 1 to i)

WHILE	WHILE myVar ≠ 100
	    OUTPUT myVar
	    myVar ← myVar + 1
	ENDWHILE
		While the variable myVar is not equal to 100, output the variable myVar and reduce myVar by 1.

REPEAT UNTIL	REPEAT
	    OUTPUT “Enter a number”
	    num ← INPUT
	    OUTPUT num
	UNTIL num = 5
		Repeat the command, “Enter a Number” until the number 5 has been typed in.

AQA Computer Science :: Cheat Sheet

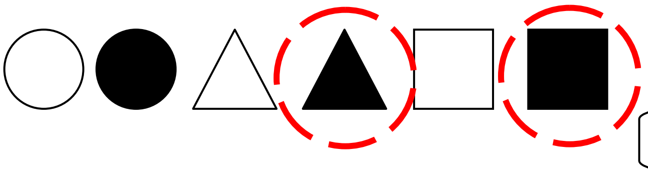
Topic 6 :: Boolean Operators

Define: Boolean Operators

“Boolean operators are AND, OR or NOT. Boolean operators can be used in a Boolean expression. The result of a Boolean expression is always TRUE or FALSE. Arithmetic operators (<,<=, ==,!=, >=,>) can also be used to create Boolean values (TRUE or FALSE). ”

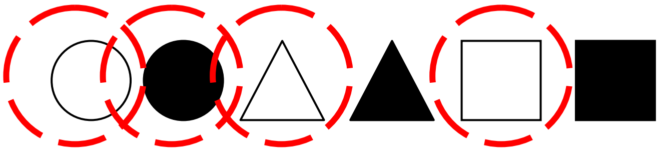
Let’s play a sorting game!

Circle the shapes that are black **AND** have straight edges.

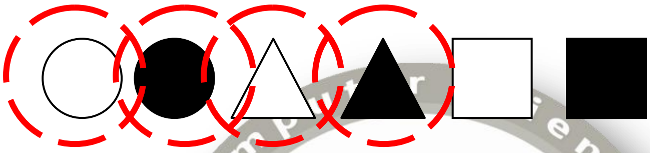


Yippee!

Well done! Now circle the shapes that are white **OR** round.



Super! Now circle the shapes that are **NOT** squares.



Miss, that was too easy!

I’m glad you find it so easy. Now tell me if these arithmetic expressions create the Boolean value TRUE or FALSE.

Arithmetic Expression	TRUE OR FALSE?
12 > 9	TRUE
3 * 6 < 23	TRUE
22 != 2 * 11	FALSE
35 == 7 * 4	FALSE
99 < 98	FALSE

COVER

&

CHECK!

Define: Arrays

“Arrays are useful in programming when you want to use a list of variables with a common link. E.g. if you wanted to use the alphabet, you wouldn’t go:

```
A ← “A”
B ← “B” etc.
```

You would say:

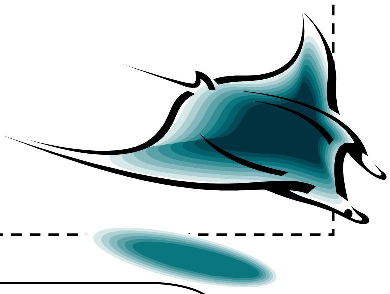
```
alphabet ← [“A”,”B”,”C”,”D”,”E”...]
```

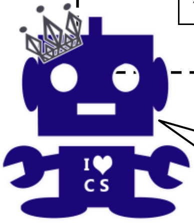
Then if you needed the letter F for example you would call it by saying:

```
alphabet [6]
```

The number 6 refers to the position of F in the array. When finding the position using AQA’s pseudo code we start at 1:

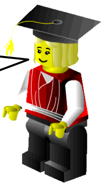
1	2	3	4	5	6	7
A	B	C	D	E	F	G



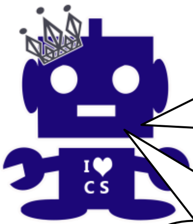


Phew! That’s a tough one. Can I have a little practise?

Certainly! Have a look at the array below. Then call the word fish.



```
Seaside ← [“Bucket”,”Spade”,”Ice Cream”,”Fish”,”Water”]
```

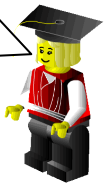


Hmmm...

Seaside [4]

Thanks, you aren’t funny by the way. Having a picture of a “Ray” on here!

Well done! “Fish” is the fourth one in the array. Some programming languages start at 0 when indexing arrays. It is important to find out how they are indexed. You can use a trace command to test it. Read on to find out what a trace is is!



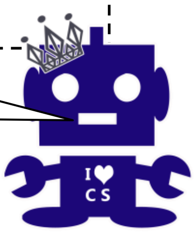
AQA Computer Science :: Cheat Sheet

Topic 8 :: Flowcharts

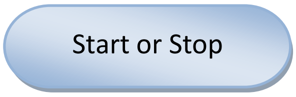
Define: Flowcharts

"Flowcharts are used to plan out a procedure using a diagram"

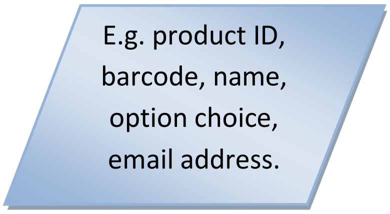
Why so many symbols?



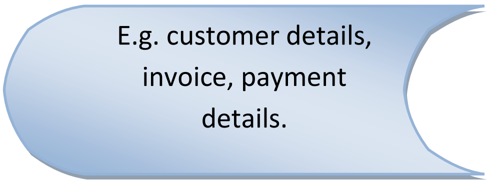
You need to learn the meaning of all the symbols below.



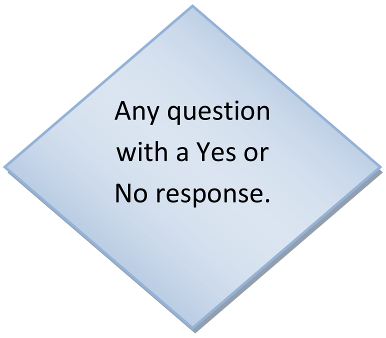
Start / Stop (terminators)


E.g. product ID, barcode, name, option choice, email address.

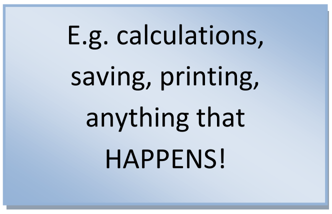
Input box



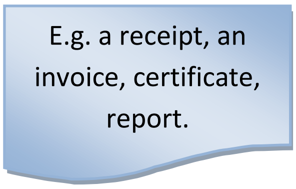
Storage



Decision box



Process



Printed output

Define: Trace Tables

"This is a method that we can use for testing whether our code makes sense logically"

But I am always logical?

This may be the case but when you make a program it is not always easy to spot logic errors. So we use trace tables to spot the things we may not!

Oh!

It is sensible to use a trace table in the design stage of your program.

- Break down your code into steps.
- Each step should be a row in your trace table.
- Each row should show the "state" of your variables for that step.

Balloon Hits	Score
0	0
1	10
2	20
3	30
4	40
5	50
6	60

I have made this one for my "Shoot the Balloons" game. It tracks the score at each stage. When the balloon is hit, the user scores 10 points. Have I done it right?

Yes, this is a good example of a ***simple*** trace table. They will become more complex as you learn more about programming.

Programming languages allow you to trace variables in the console when running your code. It is important to learn how to do this for each programming language that you use.

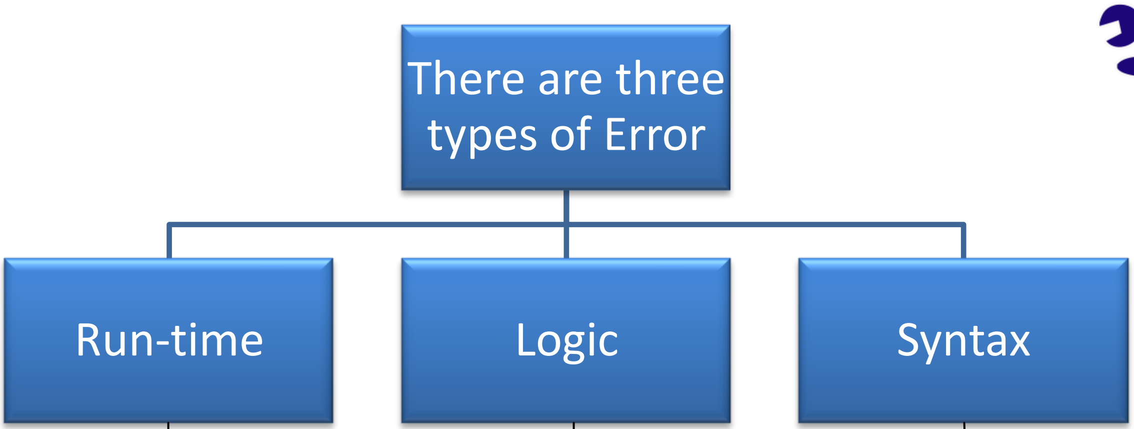
AQA Computer Science :: Cheat Sheet

Topic 10 :: Errors

Define: Errors

“Nobody is perfect! It doesn’t matter how good you are at programming, there will always be errors. Errors can be one of three types; Logical, Run-time or Syntax”

Error, Error, Does not compute! Error!



There are three types of Error

Run-time

Logic

Syntax

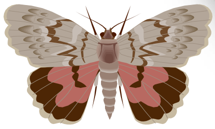
There are many reasons for run-time errors. They generally slow the program down or stop it from working because the computer doesn’t understand what it has to do or the problem is too big to solve quickly. They tend to crash the computer when running the program; this is how you can spot them.

Logic errors are really tricky to spot. You know there is a logic error when the program runs, but it doesn’t work as you would expect. You can fix a logic error by using trace tables or by blocking out code at each point to try and spot where it is going wrong.

Syntax errors are very common and the handler usually picks these up when you run the program. Syntax errors occur when punctuation is missing or in the wrong place.

Where does the word “Debugging” come from?

Well some people believe it came from Admiral Grace Hopper finding a moth stuck in a computer that caused an error. What do you think?



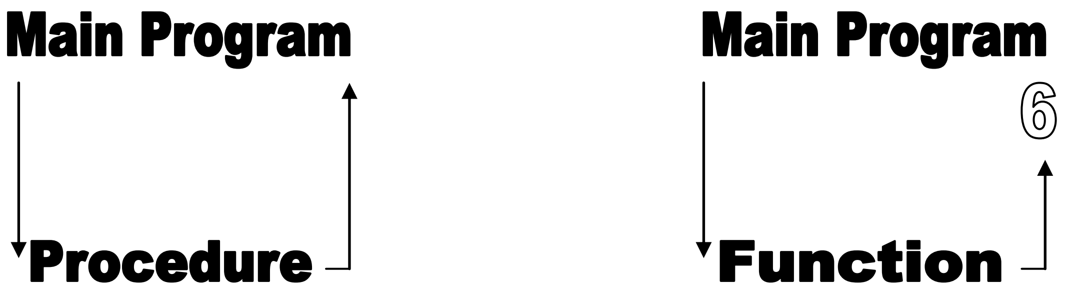
AQA Computer Science :: Cheat Sheet

Topic 11 :: Functions and Procedures

Define: Functions and Procedures

“Functions and procedures are quite similar as they are bits of code that can be recalled at any point in your program and used again and again”

The Difference:



Main Program

Procedure

Main Program

6

Function

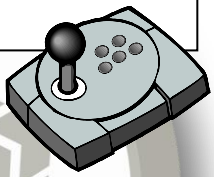
A procedure just “does something”

A function returns a value

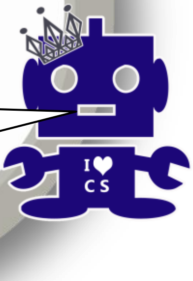
OR in even more simple terms...

Brushing your teeth is a **procedure**. You perform the same routine every day. E.g. put toothpaste on brush, brush teeth for two minutes, spit, and clean brush. These actions can all come under the procedure name “Brushing my Teeth”. Once the routine has finished there is no value to return, it has just finished.

Counting your computer games is a **function**. You look at your pile of games, you count them all and you are left with a value. Every time you count your games you do the same routine, add 1 each time, and at the end you are left with a value.



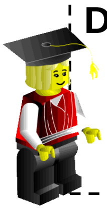
So procedures “do something” and functions “return a value”



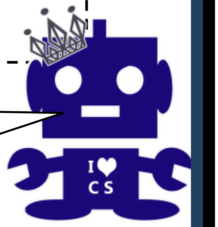
Yes, the value doesn’t have to be a number though. It could be string, characters or anything really!

Topic 12 :: Scope

Define: Scope



“This refers to <u>where</u> a variable is accessible in your code”

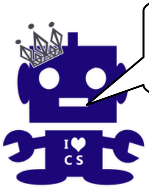


If I use a variable inside a procedure, can I use it somewhere else in my code?

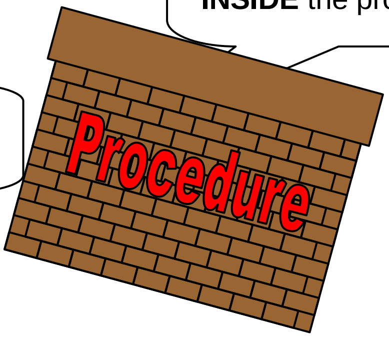
Only if it has been declared as a global variable. For a variable to be accessible anywhere, it needs to be defined as a global variable at the beginning of your code.

Different Types of Scope:

Procedure Scope

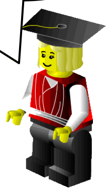


Hello, I am looking for the **Name** variable?



You can’t see me because I was declared **INSIDE** the procedure.

Procedure Scope is where a variable is declared inside a procedure.



Module Scope

Any variable declared **within a module** can be accessed and modified by **any procedure** within that module.

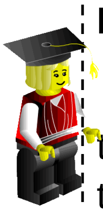
Project Scope

Any variable declared **within a project** can be accessed and modified **by any procedure or module** within that project.

Global Scope

Any variable declared **as global** can be accessed and modified **by any procedure, module or project** within the program.

Topic 13 :: Bespoke Data Structures



Define: Bespoke Data Structures

“When something is bespoke it has been tailored to the needs of the user. When we are talking about ‘Bespoke Data Structures’ we are tailoring a data structure to the user or client needs.”

Please write your personal details below:

RoboChief

23 Bakerland Road

07783 187382

Gemini

Please write your personal details below:

Technokid

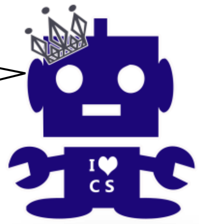
01562 812345

technokid@rockets.com

Please write your personal details below:

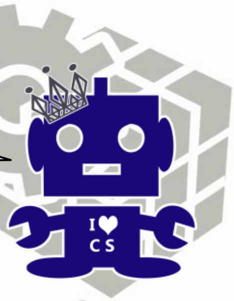
miss robot

25/09/2000



I have given this form to three of my friends. They haven’t given me the information I needed. What should I have done?

Before you ask for data from anyone you need to think carefully about what you needed. You then set up a structure for people to enter their data correctly.



Ahhh. So I should have said “First Name, Surname, Email, DOB etc.”

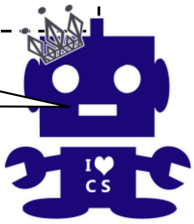
That’s correct! You also need to think of the TYPE of data being entered. Is it text? Is it a number? Then you will have a good, solid data structure.

AQA Computer Science :: Cheat Sheet

Topic 14 :: Software Development Life Cycle

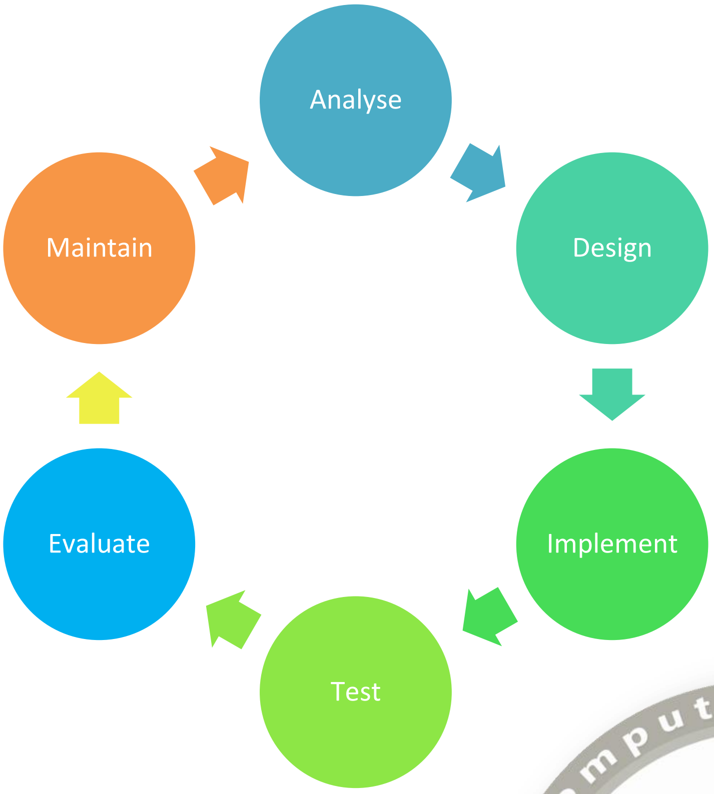
Define: Software Development Life Cycle

“This is the different phases or stages that we need to go through in order to make a successful program”



But I just want to get started and write my code!

It is good that you are keen but you do need to go through all of the steps or else you will make a lot of work for yourself when things go wrong!



Analyse – What does your program need to do?

Design – What will it look like?

Implement – Let’s code it!

Test – Does it work?

Evaluate – Does it work effectively?

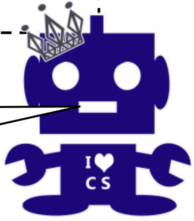
Maintain – Does it still work now it is being used for real?

Define: Prototyping and Testing 1

"Throughout this course you will be set programming tasks. The level of skill required will increase for each one. They will always cover elements that have been covered in your Computer Science lessons"

Yes, and remember to follow the Software Development Life Cycle whilst doing this!

Yay I get to make something!



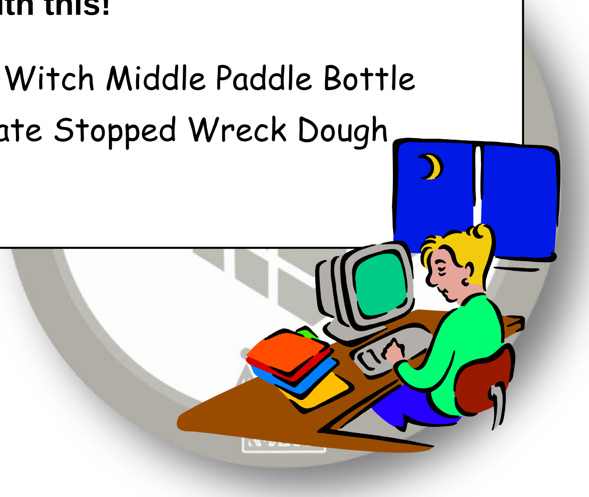
What you should know by now:

Expressions and Types	Variables and Constants	Selection
Repetition	Boolean Operators	Arrays
Flow Charts	Trace Tables	Errors
Functions and Procedures	Scope	Bespoke Data Structures
	Software Development Life Cycle	

Project: You have two weeks to complete this project. You should use lesson time to ask your peers to assess your progress.

Brief: A primary school teacher wants her students to learn how to spell the list of words below. You must develop a program that allows students to test their spelling. Be as imaginative as you like with this!

When What Kneel Knight Knock Honest Which Witch Middle Paddle Bottle Travelled Terrible Frustrating Dancing Evaporate Stopped Wreck Dough Though Through



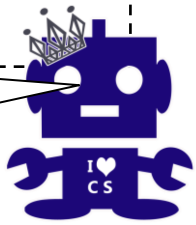
AQA Computer Science :: Cheat Sheet

Topic 16 :: Bits and Bytes

Define: Bits and Bytes

“Computers see everything in 0’s and 1’s, this is called binary. 1 bit is a single digit (0 or 1). 1 byte is 8 bits. We also have something called a nibble which is 4 bits or half a byte. Because bits are so small, we tend to measure things in bytes.”

All this talk of bits and bytes is making me hungry!



Below is a list of different ways to measure storage:

1024 bytes => 1 Kilobyte (KB) or about half a page of text

1024 Kilobyte => 1 Megabyte (MB) or a whole book of text

1024 Megabytes => 1 Gigabyte (GB) or around300 mp3 songs

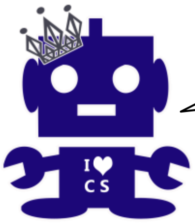
1024 Gigabytes => 1 Terabyte (TB) or around 307,000 mp3 songs

1024 Terabytes => 1 Petabyte (PB) or 350,000,000 images

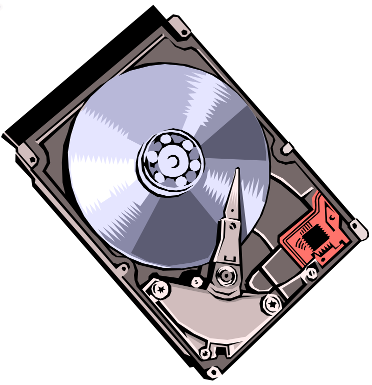
1024 Petabyte => 1 Exabyte (EB) or 360,000,000,000 images

1024 Exabytes => 1 Zettabyte (ZB) or around 375,000,000,000,000 images

1024 Zettabyte => 1 Yottabyte (YB) or around 384,000,000,000,000,000 images



Why is it always 1024?

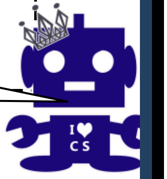


This relates to binary representation. You can find out more about this in T17a Binary Cheat Sheet!

Define: Binary

"Binary is a machine language made of 0's and 1's. The 0 represents OFF and the 1 represents ON"

00000001111010100010
10001010001001010001



We (humans) understand numbers as denary; our numbers have a base of 10. Computers understand binary; binary has a base of 2. In order to communicate with computers, binary numbers are converted to denary.

This is how **denary** works...When we write the number 483,297 we are looking at where those numbers are placed on the number scale.

The symbol **^** means "to the power of". This should be familiar to you!

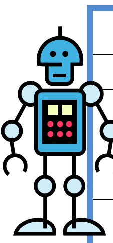
How humans think

10,000,000	1,000,000	100,000	10,000	1000	100	10	1
10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
0	0	4	8	3	2	9	7

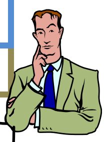
or (4 * 100,000) + (8 * 10,000) + (3 * 1000) + (2 * 100) + (9 * 10) + (7 * 1)

Let's say we have the **binary** number **01001010** and we need to convert it to **denary**. We can do this using a table to help us. Each binary digit **holds a place**. Like denary is 10 to the power of..., Binary is 2 to the power of...

How computers think



128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
○	●	○	○	●	○	●	○
OFF	ON	OFF	OFF	ON	OFF	ON	OFF
0	1	0	0	1	0	1	0
	64			8		2	



When we put our binary number into our table we can begin to convert the number to denary. Wherever there is a 1, we count the value that applies to the **placeholder**. We can only ever multiply the placeholder by 1 or 0. 0 is ignored as you can see!

01001010 = 64 + 8 + 2 = 74

Define: Hexadecimal

"This is a word used to describe a number that has a base of 16. Similar to binary that has a base of 2 and denary that has a base of 10."

Two examples of where hexadecimal is used in terms of computer science are:

- **Colour References** – Colours are assigned a hexadecimal number, for example, white is represented as #FFFFFF
- **Error Messages** – You may have seen an error message on your screen that displays something like "error reading FFFF-1048", this refers to a specific error that can be located using the hexadecimal number

Hexadecimal has 16 values. These compared to decimal can be seen below:

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

When we want to convert from Hexadecimal to Decimal (or denary) we can use the table below, this shows how the Hex number D48 is converted:

268435456	**16777216**	**1048576**	**65536**	**4096**	**256**	**16**	**1**
16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
					D	**4**	**8**

If we place **D48** on our number scale we can work out the denary conversion by looking at where the numbers are placed.

We complete the following calculation:

(8 * 1) + (4 * 16) + (D * 256)

We know that D = 13 from the conversion table above. So this calculation becomes:

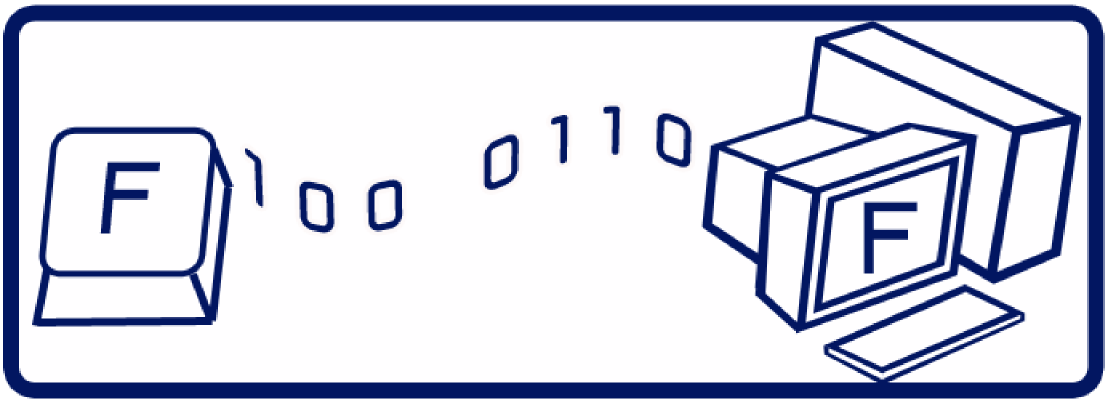
(8 * 1) + (4 * 16) + (13 * 256)

Giving us the conversion: **3400**

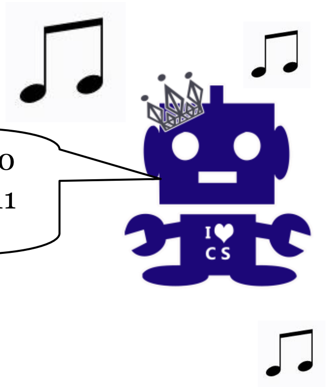
Define: Character Sets

"This refers to the 128 different characters that are on our keyboards. For example, A to Z, @, :, + etc."

Our computers use the ASCII standard for representing characters as a seven-bit binary number. ASCII stands for American Standard Code for Information Interchange. It uses the characters that are in the English language.



Each time a key is pressed on your keyboard, a seven-bit binary number is sent to the computer. It is then outputted on the monitor as the letter that you originally typed. Every character on your keyboard has its own unique code.



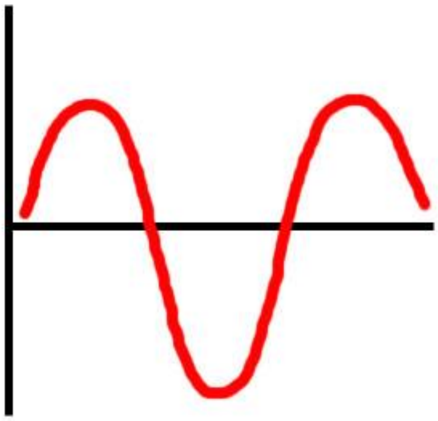
10 00001, 100 0010, 100 0011, 100 0100, 100 0101, 100 0110, 100 0111

A beautiful rendition of the alphabet song!

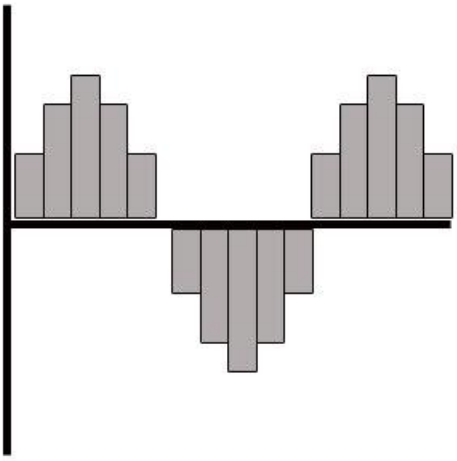
Define: Sound

"Sound travels through a medium, such as air or water, from the source to the ear drum. Sound waves are analogue, they are continuously changing. "

This is an analogue sound wave:



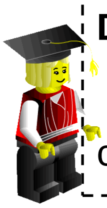
This is a digitised sound wave:



Just like anything else that it is inputted to a computer, it needs to be converted into a language that the computer understands. Analogue sound is stored digitally in order for the computer to understand it.

If we were to convert every single element of sound to digital then the storage space needed would be too much! There is little point in storing data that your ears cannot physically hear so only the sounds that are needed are converted.

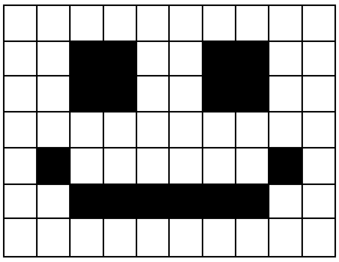
The quality of the sound depends on the **sampling rate**. A higher sampling rate means better quality sound. It also means that it will take up more storage space on a computer.



Define: Bitmap Images

"A bitmap image is made up of lots of tiny little blocks of colour called pixels. Each pixel is represented by a binary number."

A bitmap image file remembers the colour of every single pixel in an image. They take up a lot of storage space because of this. Below is a representation of a 1 bit depth image. It uses two colours, black and white.



Imagine each block above is a pixel. A bitmap image would save this file as:

white, white, white, white, white, white, white, white, white, white, white, white, black, black, white, white, black, black, white, white and so on...

Or in binary this would be:

00000000000011001100

A computer wouldn't be able to output this on the screen with just these binary digits alone. It also needs something called **metadata**.

Metadata is the extra data that is required for the computer to create the image. It will need to know the resolution of the image and how many bits are in each pixel.

The metadata for the image above would be:

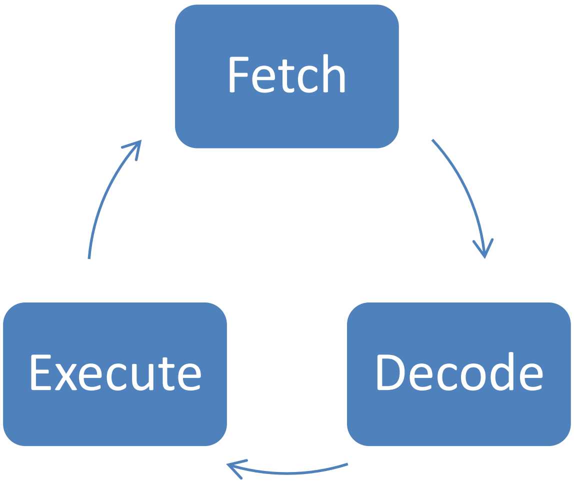
- Resolution - 10 x 7 pixels
- Pixel Depth - 1 bit per pixel

Topic 21 :: Instructions and the CPU

Define: Instructions and the CPU

“The Central Processing Unit (CPU) is the brain of the computer. It does all of the thinking and tells all the other components what to do. The processor runs each instruction in turn, then it decodes it and finally it executes it.”

The processes that a CPU completes can be represented by a diagram called the **Fetch-Execute Cycle**:



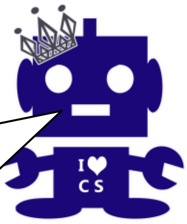
Fetch – Get the next instruction from the main memory

Decode – Make sense of the instruction

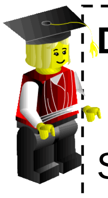
Execute – Perform the instruction

The speed of the processor effects how fast the computer will decipher instructions and execute them.

The speed of my processor is known as the **clock speed**. It is measured in **Hertz**. Think about a real clock, each tick of the clock is a process being performed. The faster my clock ticks, the faster I can process instructions.

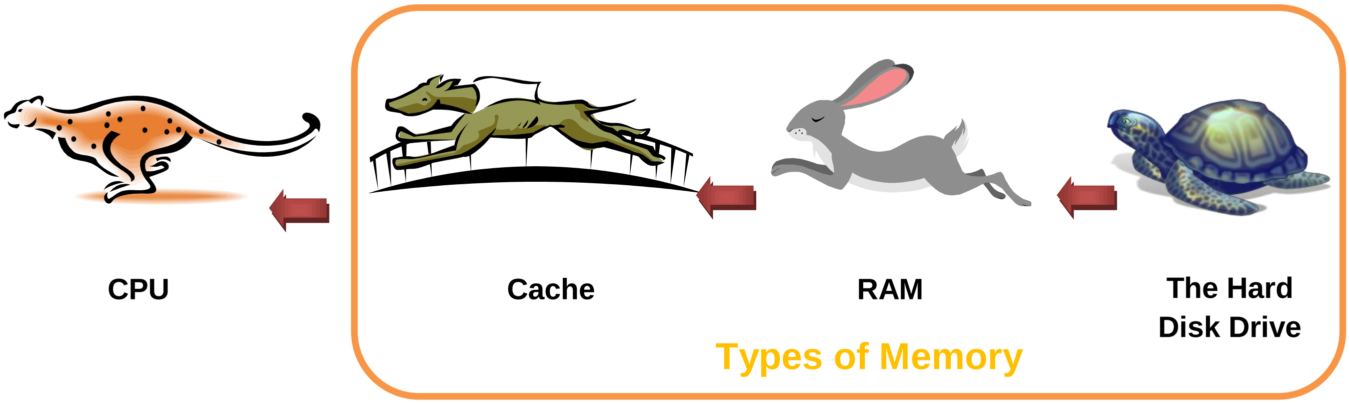


Topic 22 :: Memory



Define: Memory

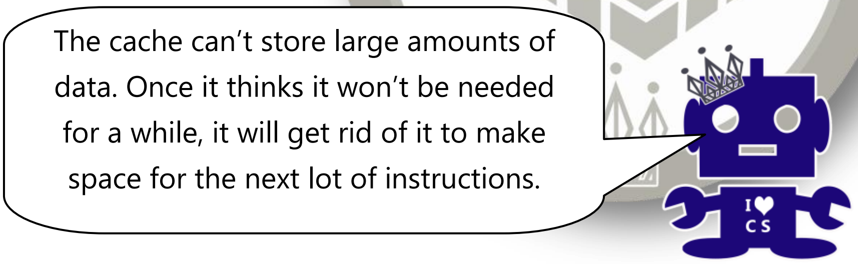
“The memory in our computer comes in lots of different forms. Some are faster than others but they all work together. Their job is to store the instructions needed to perform tasks.”



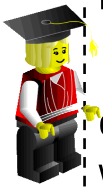
The CPU goes at a tremendous speed. The hard drive operates at a much slower speed. If the CPU tried to get instructions directly from the hard drive then it would not work because they operate at different speeds.

Your applications or programs are saved in the computers hard drive. When you wish to run a program it is loaded into the RAM or the Main Memory of the computer. This is so that it can be accessed faster.

The cache then takes the instructions from the RAM when they are needed. The cache has to be very clever as it predicts what might be needed next in order to speed up the process.



The cache can’t store large amounts of data. Once it thinks it won’t be needed for a while, it will get rid of it to make space for the next lot of instructions.



Define: Secondary Storage

"A primary storage device contains data that we need from day to day, like our applications. A secondary storage device contains data that we may not need day to day. They are often used for backing up files. Examples of a secondary storage device could be; external hard drives, floppy disk, CD RW, tape drive or a memory stick."

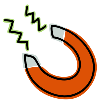
There are three main types of secondary storage device:

Optical

Uses lasers to read and write data to and from the disk. Microscopic lines and dots are "burnt" onto the disk, representing binary digits.

Magnetic

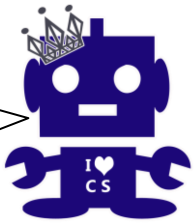
Magnetic dots that represent binary digits are placed on a magnetised material inside the drive.



Solid State

Solid state is a type of drive that has no moving parts. This makes them more robust.

If you have a spare moment, why don't you compare these three types of secondary device on the Internet? Look for things like Capacity, Speed, Portability, Durability and Reliability.



AQA Computer Science :: Cheat Sheet

Topic 24 :: Input and Output Devices

Define: Input and Output Devices

“Input devices send data into a computer. Output devices send data out of the computer.”

Input Devices:

Keyboard

Mouse

Touch Screen

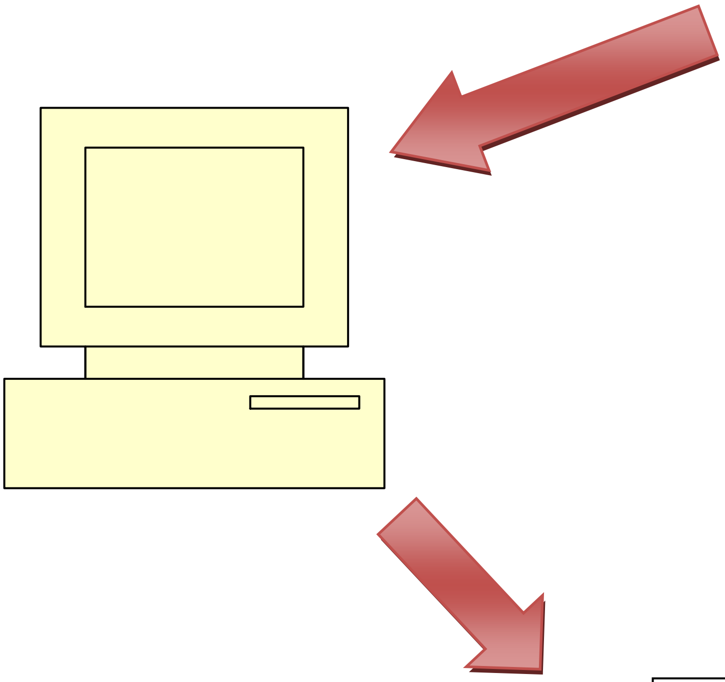
Microphone

Controller/Joystick

Scanner

OMR and OCR

Sensors



Output Devices:

Monitor

Printer

Speakers

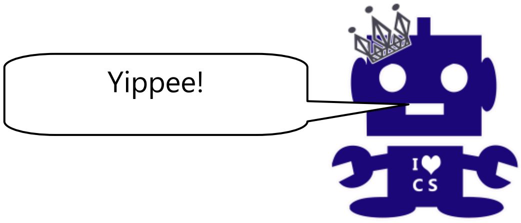
Headphones

Projector

Define: Prototyping and Testing 2

“Throughout this course you will be set programming tasks. The level of skill required will increase for each one. They will always cover elements that have been covered in your Computer Science lessons”

Following the Software Development Life Cycle I would like you to have a go at a project. This project will use sensors.



What you SHOULD know by now!

Expressions and Types	Variables and Constants	Selection	Expressions and Types
Repetition	Boolean Operators	Arrays	Repetition
Flow Charts	Trace Tables	Errors	Flow Charts
Functions and Procedures	Scope	Bespoke Data Structures	Functions and Procedures
Software Development Life Cycle	Bits and Bytes	Number Systems	Character Sets
Sounds	Bitmap Images	Instructions and the CPU	Memory
Secondary Storage	Input and Output Devices		

Project: You have two weeks to complete this project. You should use lesson time to ask your peers to assess your progress.

Brief: Using Scratch and a Sensing device you must create a program that shows off your talents! You should try to use as much of the knowledge in the table above as you possibly can. Maybe your program could teach someone about these things?

Define: Algorithm

"An algorithm is a set of instructions that always finish with a problem solved."

Scenario One

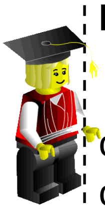
If you wanted to bake a cake, you would gather the ingredients then follow a set of instructions. Once you have finished following the instructions then you will have baked a cake. You will have finished and be left with the results.

Scenario Two

If you wanted to get directions to somewhere new then you may type the post code for a destination into an online mapping service. It would then return a list of instructions. Once you had followed the instructions, you would have reached your destination.

Both of these are examples of algorithms. They are lists of instructions that once followed, a problem is solved.

When you are programming you may create an algorithm that is an overview of your whole program. You may then make further algorithms that solve parts of your program. These could be converted to pseudo code to help you when programming your real product.



Define: Language Libraries

> “All programming languages work on the same fundamental concepts of programming. Each language has its own way of communicating. You can use a language library to find out what syntax is required to perform specific functions.”

Each programming language has its own built in functions that you can use. This is useful because it saves you having to write the code for those, often complex, functions.

Python

Python has a function called `time;` we can import it into our program by typing `import time`.

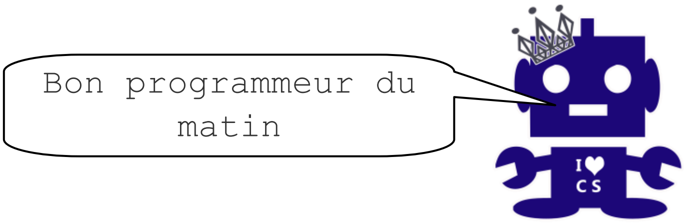
You can then use elements of that time function. If you want to make the program pause for two seconds you can type in:

```
time.sleep(2)
```

Investigating Language Libraries

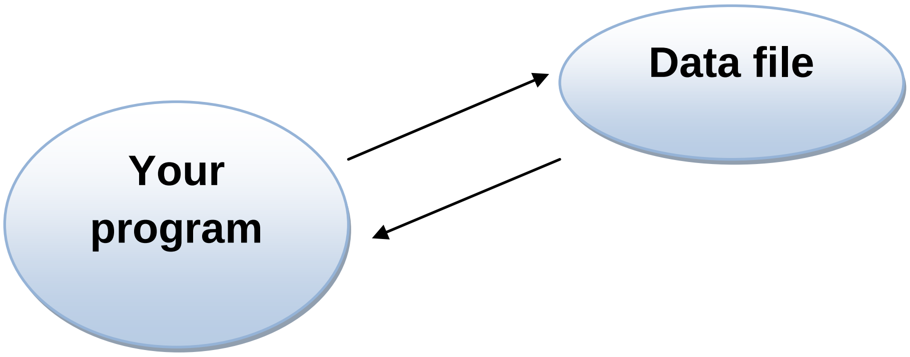
It is important that you read the documentation that is available for your programming language. This will give details of the functions that are available for you to use.

For the controlled assessment, you need to know your chosen language very well. Think about the functions that you might need and investigate how you can use them in your program.



Define: External Databases - TEXT

“Sometimes, we may need to store data externally. This means that it is no longer written into our code, it is linked to externally.”

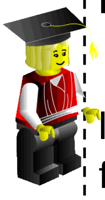


What might we store externally?

- Usernames and passwords
- Scores for a score board
- Products
- Customer details
- Content for a web page

The simplest data file that you can produce is called a text file. These can be created using notepad.

You will need to investigate how your programming language communicates with external data files.



Define: External Databases SQL

“SQL stands for Structured Query Language. It is the most used language for searching relational databases. A simple data file is great for databases that have just one table. Sometimes you need a database that has more than one table and they need to be linked together. This is where SQL can help you.”

You may have a database for a library. One table will have a list of all the members of the library, another will list the books and another will list which member has borrowed which book.

tblMembers

FirstName
Surname
DOB
Address
PostCode

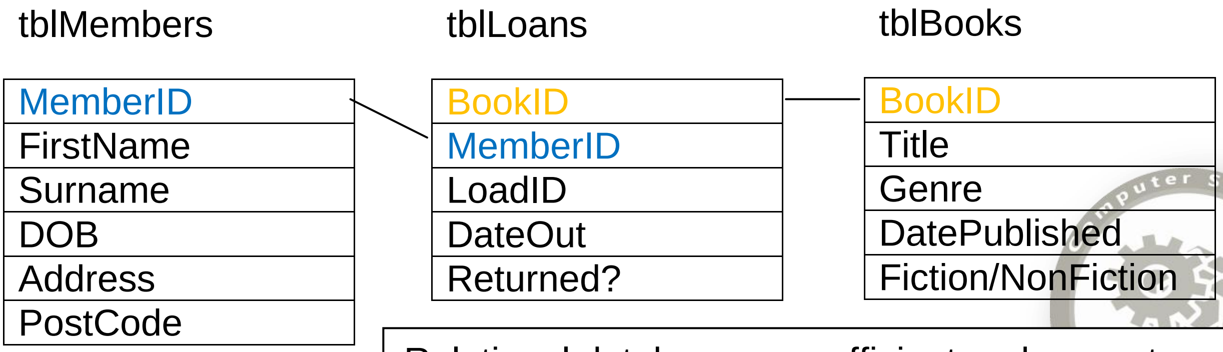
tblLoans

FirstName
Surname
DOB
Address
PostCode
Title
Genre
DatePublished
Fiction/NonFiction
DateOut
Returned?

tblBooks

Title
Genre
DatePublished
Fiction/NonFiction

This database is not efficient! The librarian would need to physically type in the details of each member before allowing them to loan a book. It would be very time consuming and would take up a lot of storage space.



tblMembers

MemberID
FirstName
Surname
DOB
Address
PostCode

tblLoans

BookID
MemberID
LoadID
DateOut
Returned?

tblBooks

BookID
Title
Genre
DatePublished
Fiction/NonFiction

Relational databases are efficient and save storage space. SQL can be used to search a relational database.

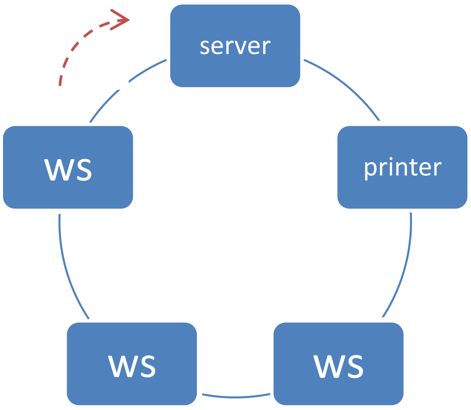
Define: Networks

“Networks link devices together so that they can communicate with each other and share peripherals.”

Networks are arranged in different topologies. It is important that you understand the different topologies. If you understand how they work then you can talk about their advantages and limitations.

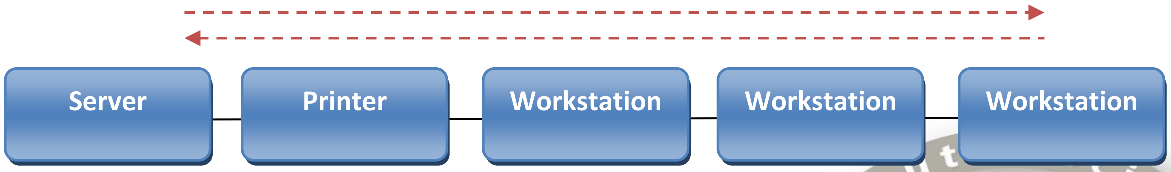
These are the three main topologies:

Ring Topology

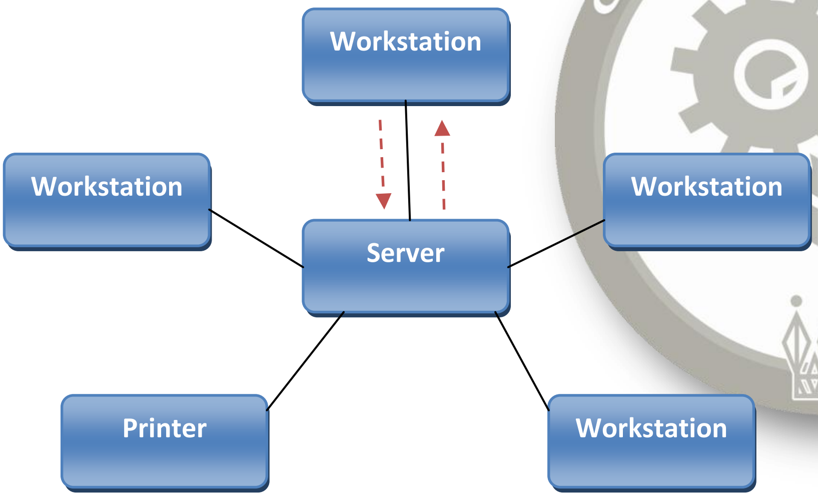


Find out WHY some are better than others!

Bus Topology



Star Topology



Topic 31 :: Client Server Model

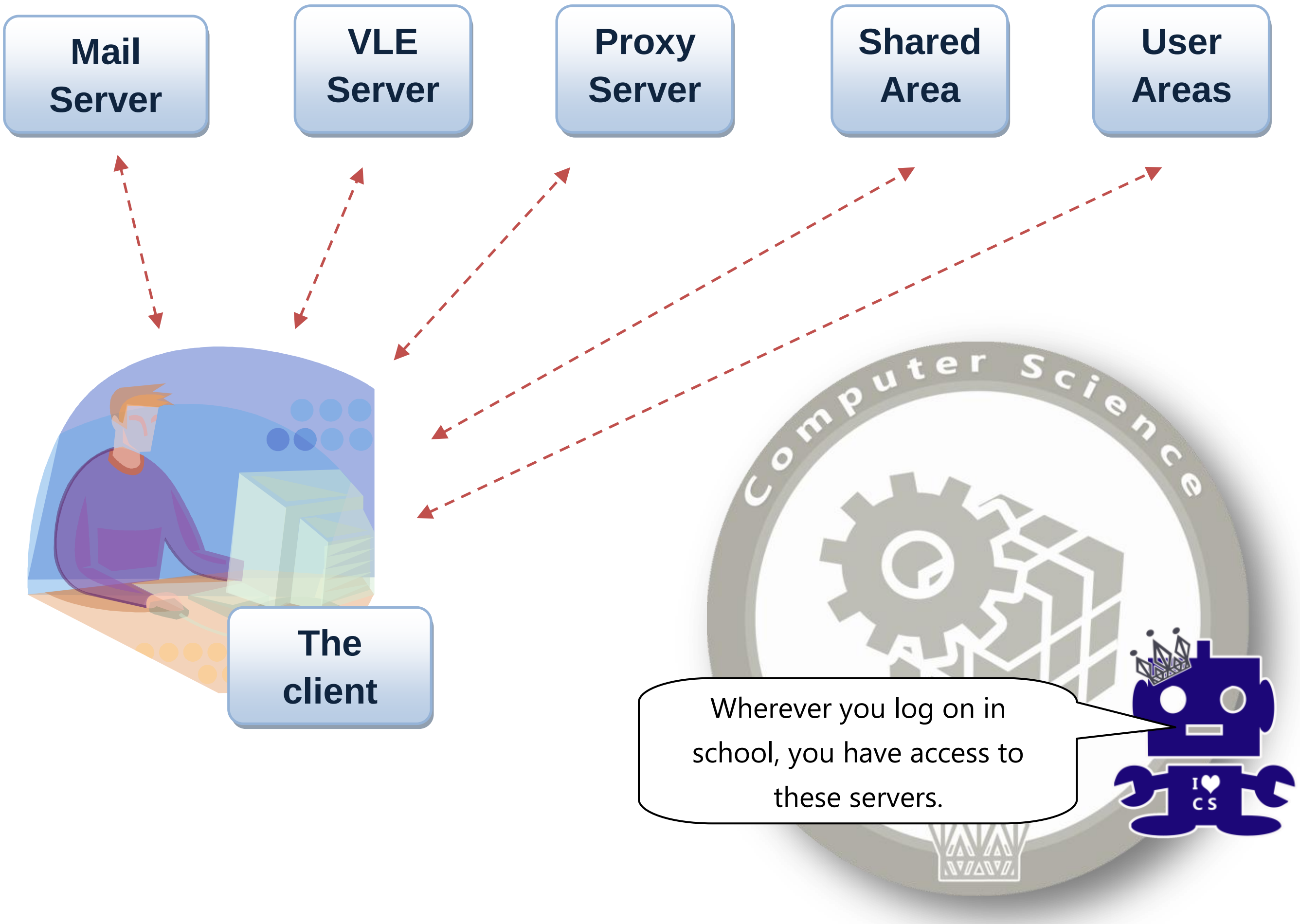
Define: Client Server Model

“There are two ways to configure networks. Peer to peer and Client-Server. Peer to peer is used for small networks in one building like a home or small office. Client-Server networks are used when lots of computers need to be connected over a wide area. This could happen in a school or a large company.”

At school you can access your files from any computer. You also have access to the Internet and your emails. This is because all of these things are stored or processed at a central server.

The computer that you are working at is known as the **client**. The **server** is where you get your files, internet access and emails.

Our school network has lots of servers:



Wherever you log on in school, you have access to these servers.

Topic 32 :: Client Side Programming

Define: Client Side Programming

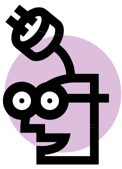
"When you visit a webpage, the code for that page is downloaded to your computer. Client side programming means that no further code needs to be downloaded in order to use that page fully. Everything you need is there."

Here are some examples of things that may work on a web site at the client end:

Cookies – Cookies are text files that contain data that make it easier for you to use a page next time you visit. Have you ever wondered how a website remembers your username? It remembers it because a cookie has stored that data on your PC and tells the webpage your name the next time you visit the page.

Validation – This checks that data entered into a form is VALID or CORRECT. The validation checks will be stored in the code of the webpage.

Plug-ins – Your browser will probably contain plug-ins so that you can use Flash or Java whilst on a web page; these plug-ins will already be saved on your computer.

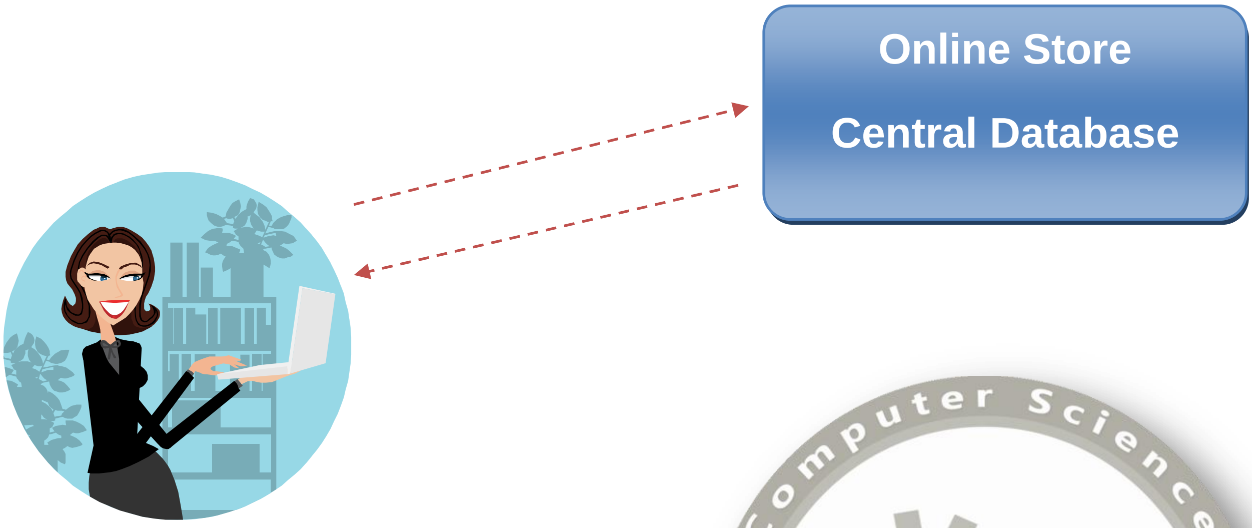


Define: Server Side Programming

"Server side programming is when data needs to be taken from a centralised database for the website to work."

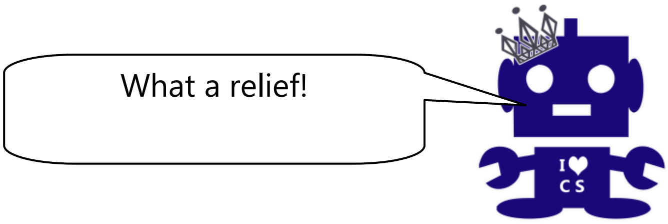
If you have an account with an online company, they will have your details stored on a large, centralised database. Your details won't be saved on your computer; they will be saved on their database.

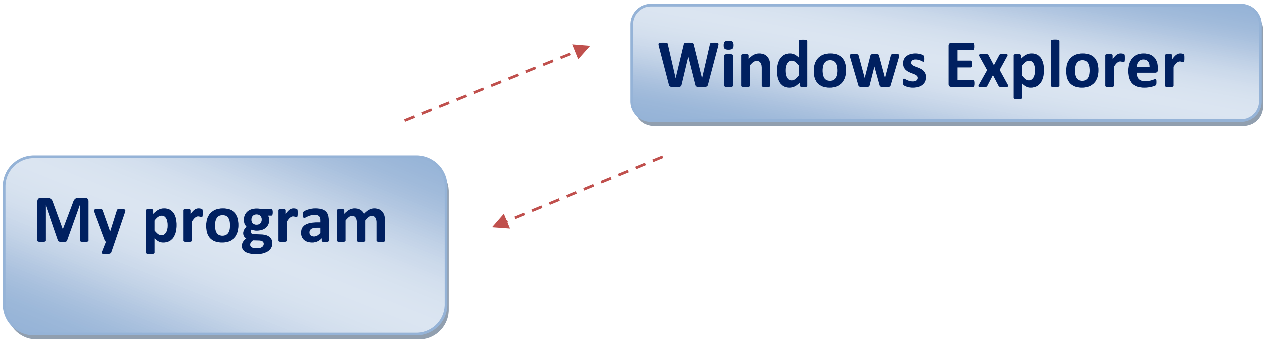
In order for you to purchase something from their store a login is required. This login process retrieves your data from the database.



Define: External Code Sources

“Your program may need to interact with other applications on the computer. It also might want to interact with the files and folders on the computer. Operating systems and applications often provide source code for you to use for this purpose. This saves you having to write the code yourself!”

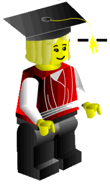


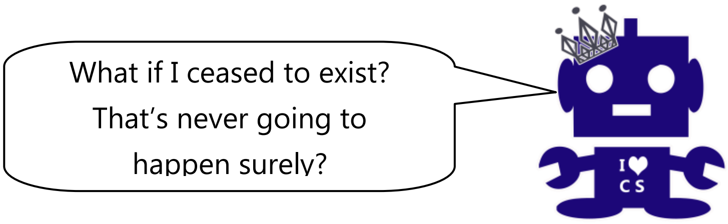


Your program can interact with the external code source for windows explorer so that the user can save their files. It would be silly to attempt to write the code for this yourself when it has already been written and is available to you!

Define: Computer Technology in Society

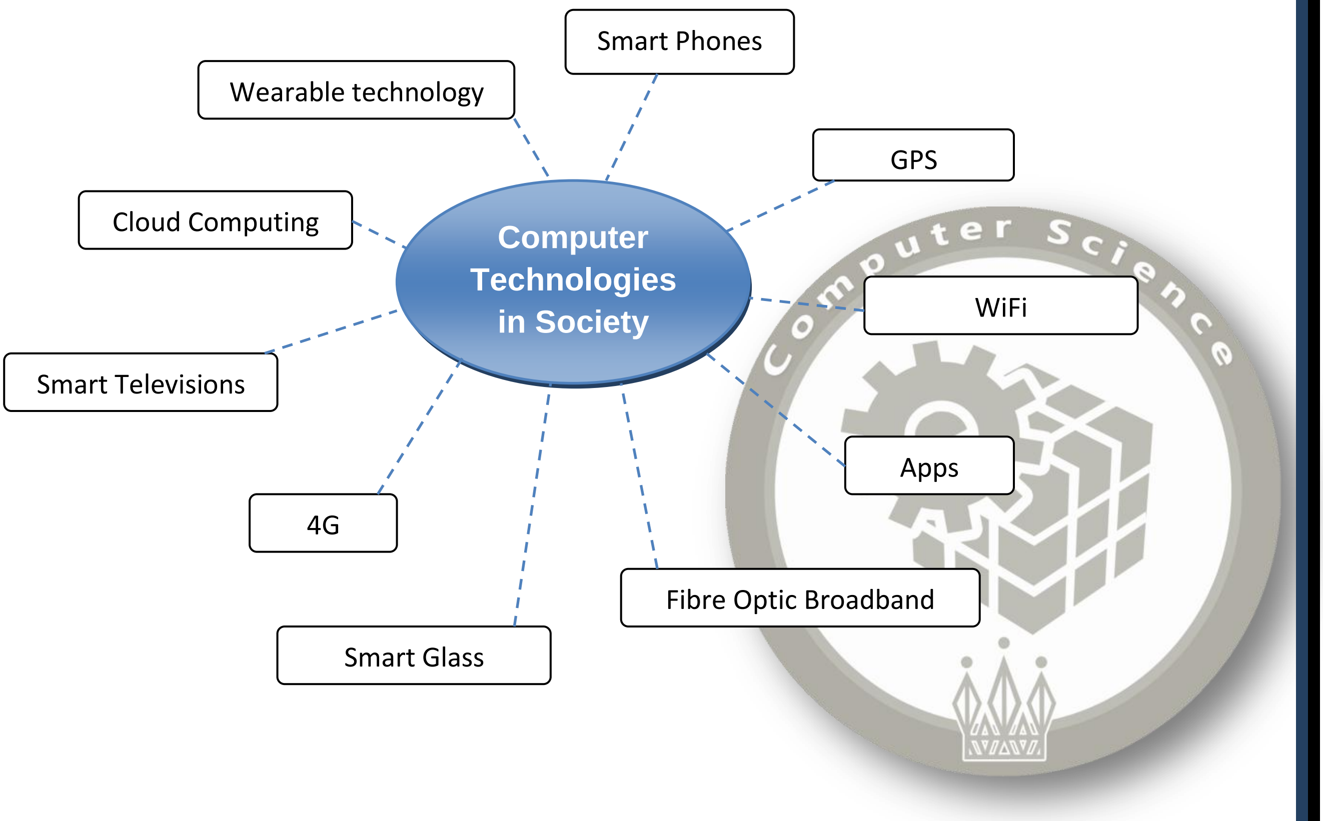
“Computer technology in the real world; how it is used, how it affects us and what would we do without it.”




What if I ceased to exist? That’s never going to happen surely?
I ♥ CS

Your end of year examination will cover some aspect of this topic. It will most likely be connected to a recent technological advance. You will need to answer the question using technical language and make good use of the English language.

Below is a diagram showing some of these technologies:



Computer Technologies in Society

Smart Phones

Wearable technology

GPS

Cloud Computing

WiFi

Smart Televisions

Apps

4G

Fibre Optic Broadband

Smart Glass

My Notes: