KNOWLEDGE ORGANISER

COMP 2 REVISION: COMPUTATIONAL THINKING, ALGORITHMS & PROGRAMMING

2.1 – Algorithms - Computational Thinking

What is Computational thinking?

The thought processes involved in formulating a problem and its solution(s), so that a computer, human or machine can effectively carry out

1

How do you think computationally?

To effectively solve problems you need to….

- Decompose
- Abstract
- Algorithmic thinking
- Create algorithms

1

KEY TERMS

Algorithm: Steps to provide a solution to a problem, usually represented in flowcharts or pseudocode

Decompose: Breaking down a large problem into smaller sub-problems

Abstraction: Representing 'real world' problems in a computer using variables and symbols and removing unnecessary elements from the problem e.g

Algorithmic Thinking: Identifying the steps involved in solving a problem.

Sequence: Completing steps in the order which they must happen

Selection: Where a choice is made in a program depending on a condition or outcome

Iteration: Act of repeating or lopping specific sections of code

2

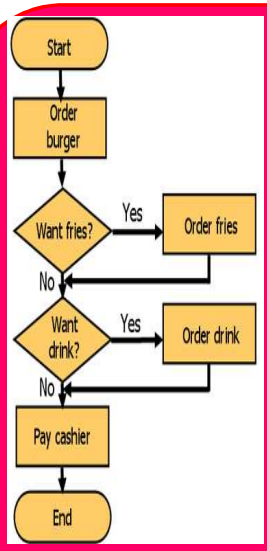
3

Flowcharts

Displays an algorithm in diagram form using symbols and arrows to show to flow of information

Pseudocode

A structured use of English used to define the steps needed to solve a problem.

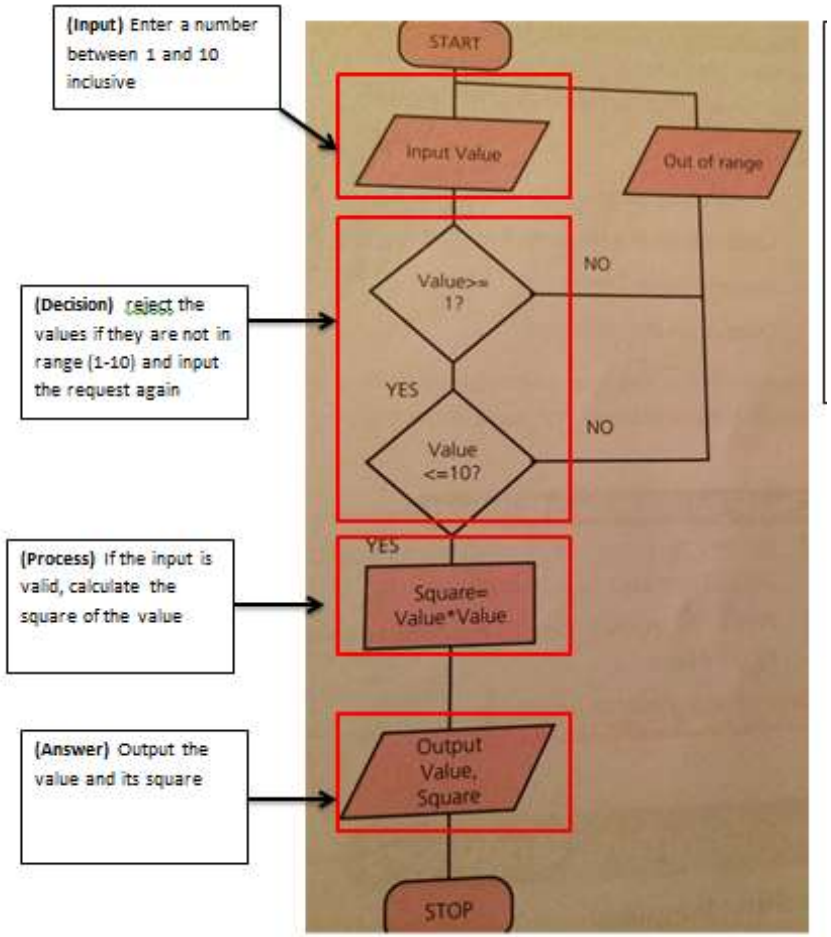


3

```
WHILE TRUE:
temp = “”
IF temp > than 20c THEN
        Open windows AND heaters OFF
ELSE
        Close windows AND heaters ON

TIME = “”
IF TIME = 18.00  THEN
        Sprinklers on
ELSE
        Sprinklers off
        Continue
BREAK
```

2.1 – Algorithms – 4. Flowcharts



START/STOP

Always start and end with this

SUB ROUTINE

Sequence that performs a specific task. You can use this within your flowchart to show more detail in a specific section

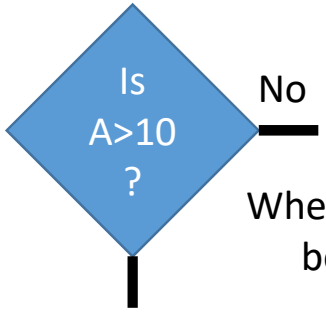
INPUT/OUTPUT

Use when there is an input or output required e.g. user inputs their name, program displays their name

PROCESS

To do something in the program e.g a calculation

Decision



When a **choice** has to be made in the program

Flow lines – show the flow of information in the algorithm

2.1 – Algorithms – 5. Pseudocode

The OCR Pseudocode guide has more information if you wish to do more research.

Pseudocode uses English.
It mimics how your code may look in the programming language BUT it DOESN'T have to be exact

Some useful terms you could use.

START
IF
ELIF
ELSE
FOR
WHILE TRUE
ENDIF
END

Here are some symbols to use

Comparison operators

== Equal to
!= Not equal to
< Less than
<= Less than or equal to
➢ Greater than
>= Greater than or equal to

Arithmetic Operators

+ Add
- Subtract
/ Divide
*** Multiply**
MOD will return the remainder for the division

```
WHILE TRUE:
temp = ""
IF temp > than 20c THEN
        Open windows AND heaters OFF
    ELSE
            Close windows AND heaters ON

    TIME = ""
    IF TIME = 18.00  THEN
            Sprinklers on
    ELSE
            Sprinklers off
            Continue
    BREAK
```

This repeats, it reads the temp of a greenhouse. If greater than 20 degs then open windows and turn heaters off ELSE close windows and heaters on

It checks the time if the time is 6pm then turn sprinklers on otherwise keep sprinklers off

2.1 – *Sorting Algorithms – Bubble, Insertion & merge*

Key vocabulary	
Bubble sort	Works by repeatedly going through the list to be sorted, comparing each pair of adjacent elements. If the elements are in the wrong order they are swapped, else they are left in position.
Insertion sort	Sorts data one element at a time. The algorithm takes one data item from the list and places it in the correct location in the list. This process is repeated until there are no more unsorted items in the list. More efficient than bubble sort.
Merge sort	This is a two-stage sort. Firstly the list is split in half into sub lists repeatedly. The algorithm stops splitting the lists when each list has only 1 element in it. The second stage involves repeatedly merging the lists in order until there is only one sub list remaining.

Figure 1 - Bubble sort example

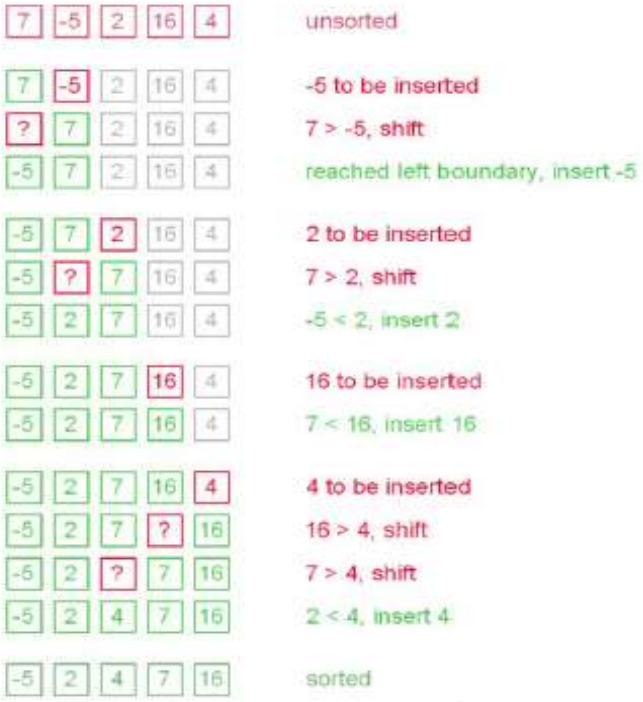


Figure 2 - Insertion sort example

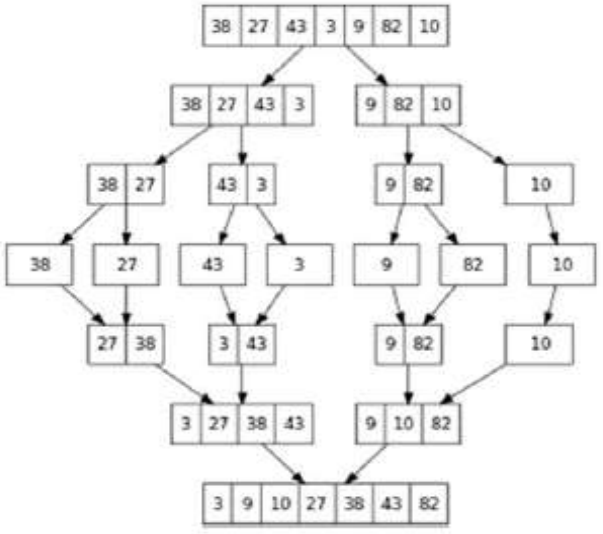


Figure 3 - Merge sort example

2.1 – *Sorting Algorithms - Binary and Linear searches*

Key vocabulary	
Linear Search	Data may be in any order to complete a linear search. Each item is inspected in turn to see whether it is what is being searched for. If an item is found, then True is returned, else the next element is inspected until all items have been searched. If nothing is found by the end of the algorithm then False is returned.
Binary Search	If a list is sorted (numerical or alphabetical order) then a more efficient algorithm can be used. It works by repeatedly dividing the list into half and searching in the appropriate half.

Figure 1 - Linear search example

Figure 2 - Binary search example

2.2 Programming Techniques

Constant
Value STORED IN A **MEMORY LOCATION** that **never changes WITHIN A PROGRAM**

Variable
Value STORED IN **MEMORY LOCATION** that **can change WITHIN IN A PROGRAM**

Syntax Error
An error in the rules/grammar of the language
Logic Error
The program is written to do something other than what the programmer intended
Eg Resetting only the first 9 elements in an array instead of all 10.

Run Time Error:
More difficult to spot as it can run a program without reporting an error. E.g. runs but Doesn't give an output. Or the program hangs or Becomes inactive

Sequence: Completing steps in the order which they must happen

Selection: Where a choice is made in a program depending on a condition or outcome

Iteration: Act of repeating or lopping specific sections of code
Count controlled Iteration: Repeats a set number of times
Condition controlled: Repeats until a condition is met or something in the program changes

Data Types

Real /Float
Number with decimal Point
Integer
Number without a decimal Point
String
A series of characters/TEXT
Character
A single letter or symbol
Date/Time
Date and Time in any format
Boolean
Yes no, true false value

Other Info

Concatenate
To join different data types together

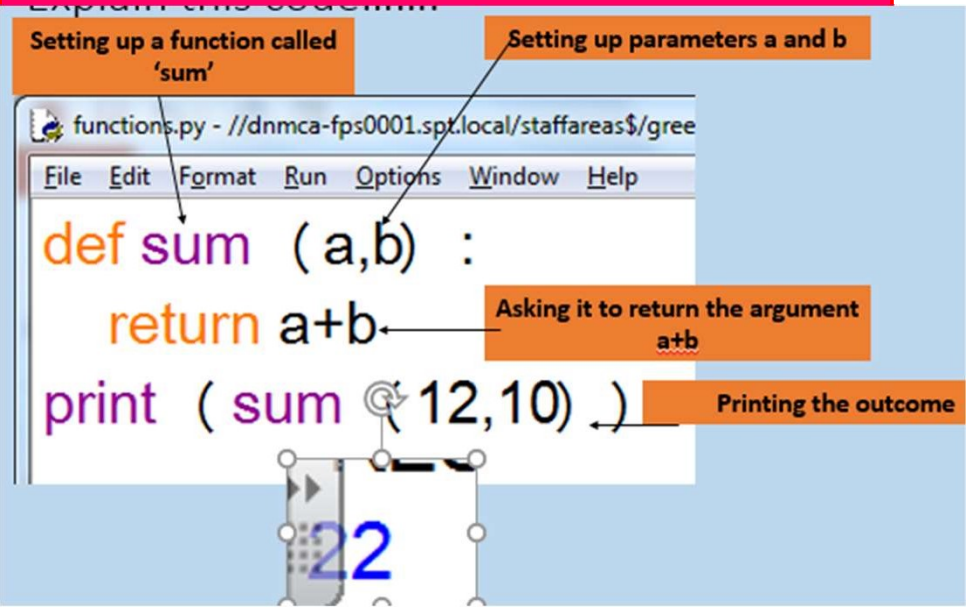
 # Comments
Use these to add comments in to your code to explain what you have done

Validation: An computer check to ensure that the data entered is sensible and reasonable. It does not check the accuracy of data.

2.2 Programming Techniques

A procedure performs a task, whereas a function produces information.

FUNCTIONS –
These are BLOCKS of code that perform a specific task.
Functions will perform calculations and they will RETURN a value



PROCEDURES -
Procedure will perform a specific task and nit necessarily return values.

We can define a PROCEDURE to so something WITHOUT explicitly returning a value

```
def hello__twice (name) :
    print ("Hello" +name)
    print ("Hello"+name)
print (hello__twice ("Bob") )
```

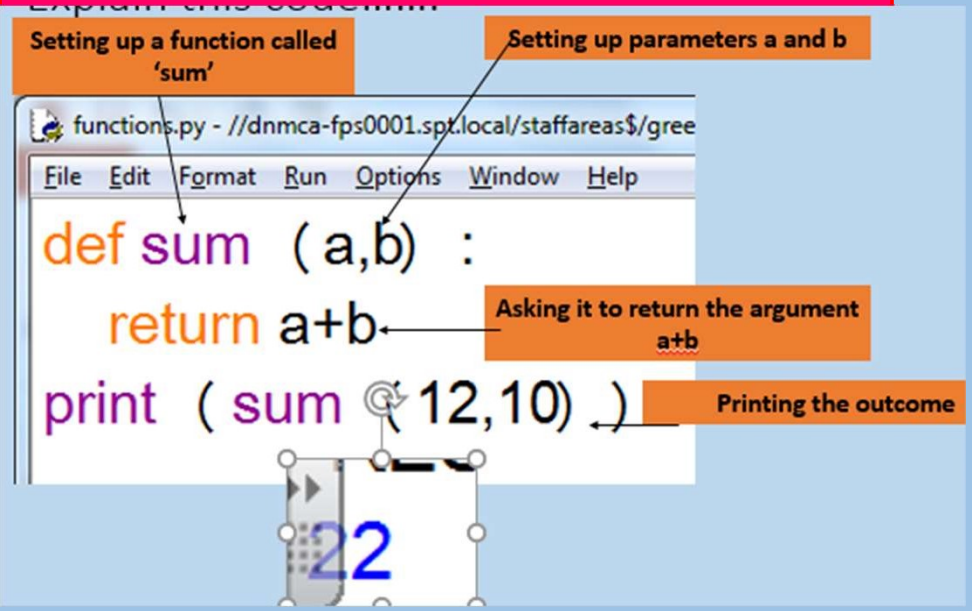
```
HelloBob
HelloBob
None
```

None – as we didn't use the **return** keyword the None is returned automatically

2.2 Programming Techniques

A procedure performs a task, whereas a function produces information.

FUNCTIONS –
These are BLOCKS of code that perform a specific task.
Functions will perform calculations and they will RETURN a value



PROCEDURES -
Procedure will perform a specific task and nit necessarily return values.

We can define a PROCEDURE to so something WITHOUT explicitly returning a value

```
def hello__twice (name) :
    print ("Hello" +name)
    print ("Hello"+name)
print (hello__twice ("Bob") )
```

```
HelloBob
HelloBob
None
```

None – as we didn't use the **return** keyword the None is returned automatically

2.2 Programming Techniques

Comparison Operators

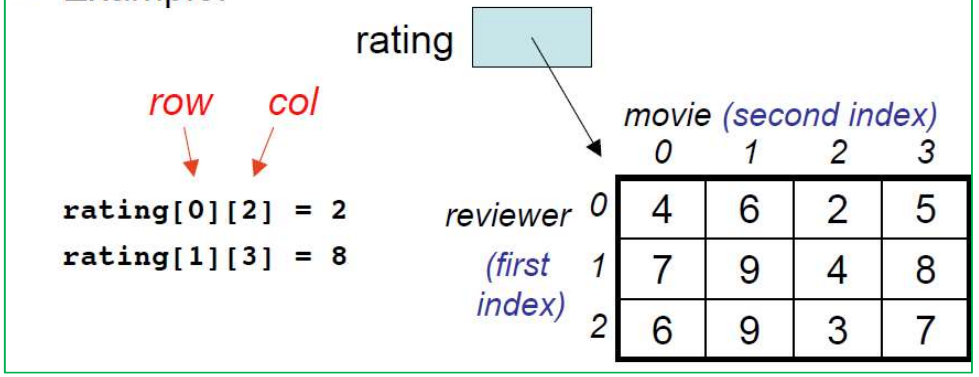
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Aritmetic Operators

+	Addition eg x=6+5 gives 11	
-	Subtraction eg x=6-5 gives 1	
*	Multiplication eg x=12*2 gives 24	
/	Division eg x=12/2 gives 6	
^	**Exponentiation:** When one number increases exponentially (the number of times) to another. The repeated multiplication of a number by itself.	2 ^ 4 = 16 (2 * 2 * 2 * 2 = 16, or 2^4)
MOD	**Modulus:** The remainder that is left over when a number is divided by another. Some programming languages will use the % symbol for MOD.	16 MOD 3 = 1 (16 / 3 = 5, with 1 left over)
DIV	**Integer division:** Used to find the quotient (integer number before the decimal point) after division. Python uses // for this.	100 DIV 3 = 33 (actual value 33.333333333 repeating)

TYPE	INFO	SYNTAX
LIST	MUTABLE DIFFERENT DATA TYPES	[ ] E.G. [1,"HELLO", 3.4]
TUPLE	IMMUTABLE DIFFERENT DATA TYPES	( ) E.G. (1,2, "Hello", 4.3)
ARRAY	IMMUTABLE SAME DATA TYPE	[ ] E.G [1,2,3,4]

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:



2.3 Robust Programs

Producing robust programs so that they are defensive against hacks and attacks.

Data Validation: check to ensure that the data entered is sensible and reasonable. It does not check the accuracy of data.

Data Sanitisation: Trims or strips strings, removing unwanted characters to make sure it contains only permitted characters . E.g. Da%ve the % would be removed.

Authentication: The process of verifying the identify of a user or process

Maintainability : Updating code so that it is compatible with current requirements

Comments: To help describe the code and structures in a program using #

Indentation: used to show the programs structure e.g. where selection or iteration or functions are used .

Data Validation techniques

Check digit	the last one or two digits in a code are used to check the other digits are correct	bar code readers in supermarkets use check digits
Format check	checks the data is in the right format	a National Insurance number is in the form LL 99 99 99 L where L is any letter and 9 is any number
Length check	checks the data isn't too short or too long	a password which needs to be six letters long
Lookup table	looks up acceptable values in a table	there are only seven possible days of the week
Presence check	checks that data has been entered into a field	in most databases a key field cannot be left blank
Range check	checks that a value falls within the specified range	number of hours worked must be less than 50 and more than 0
Spell check	looks up words in a dictionary	MS Word uses red lines to underline misspelt words

Anticipating Misuse: Defensive program design will consider and anticipate misuse. Misuse may be in the form of a brute force attack on the program. E.G Many programs and systems only allow a user to enter a password three or four times before it locks out the system.

Contingency Planning: Once a programmer has anticipated the misuse they can then plan for the these issues. For example: Limiting the number of logon attempts. Ensuring the code is robust in validating the data entered

2.3 Robust Programs

Iterative testing: This happens overtime and is repeated throughout the development of the program

Final/terminal testing: is when the product is released and real end users begin using it. Very often, end users find things wrong with systems that the programmers did not expect.

Syntax Errors: An error in the rules/grammar of the language Eg missing colon / spelling mistake

Logic errors: The program is written to do something other than what the programmer intended Eg Resetting only the first 9 elements in an array instead of all 10.

Test Data: Something which has been specifically identified for use in **tests**, typically of a computer program

Test data – When testing your programs it is important to use of data to test and to create a test plan.

- **Valid** – Data that is correct
- **In Range** – The maximum values of the data that could be entered for example for teenagers 13 - 19
- **Out of Range** – Values higher or lower than the expect range, for teenagers greater than 19
- **Null Value** – when no data is entered or left blank to test what happens.
- **Invalid** - incorrect values such as entering 'Dave' in an age field.

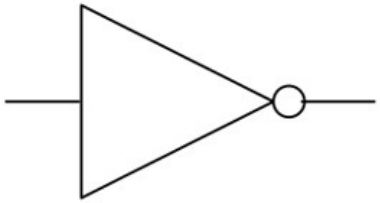
Test Plan

Test No.	Test Type	Target File or Screen	Test Name	Purpose of Test	Test Data or Situation	Expected Result	Actual Result	Outcome and Actions Required
1	Browser	flight_info.php	Rendering of arrivals table	Test that table renders as expected for arrivals	Date set: 2nd July 2007 1. Internet Explorer 7.0.6000 2. Mozilla Firefox 2.0.0.6 3. Safari for Windows 3.0.3	Six rows for arrivals, five coloured blue, one coloured red, displayed in ascending order by time. Column sequence: flight number, from, time expected, status, gate. Row 1 should contain an image arrivals.jpg). Last row should contain an image in right-most cell (corner.jpg)	1. As expected 2. As expected 3. As expected	All screens rendered as expected. No actions required
2								
3								

2.4 Computational logic

Logic Gate: A building block of a digital circuit. They perform logical functions in a circuit and use binary

Truth Tables: Display all possible outcomes for that gate



Not Gate

Inverts the input (0 becomes 1 and 1 becomes 0

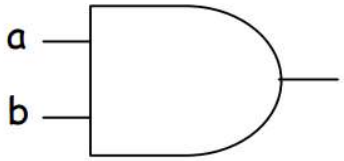
IN	OUT
0	1
1	0

OR Gate

Wait for either inputs to be 1 for output to be 1

A OR B

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1



AND Gate

Both inputs to be 1 for output to be 1

A AND B

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

2.4 Computational logic

TRANSISTOR: A tiny switch that is activated by the electronic signals it receives. The digits 1 and 0 used in binary reflect the on and off states of this.

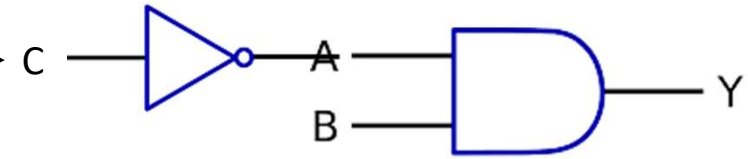
Logic CIRCUIT: A combination of different logic gates used to perform more complex tasks

Notation
The symbols used to describe logic gates

∧ And
V Or
¬ Not

EG. The notation below means – P = A and b and not c. When you draw the circuit the one in brackets goes first

Y = A ∧ B (¬ C)



HINT: when completing the inputs in a truth table – don't forget to count up in binary, to make sure you have all possible inputs
A and B re inputs. There are 4 rows – starting from 0 make the binary numbers 0,1,2,3.

	A	B	Out
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

How many rows do you need in a truth Table?

$Number\ of\ Rows = 2^{number\ of\ inputs}$

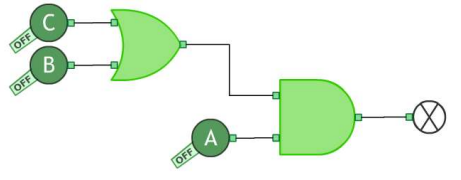
So for 1 input = 1 x 2 = 2
For 2 inputs = 2 x 2 = 4
For 3 inputs = (2x2)x2 = 8
For 4 inputs = 2x2x2x2 = 16

Gate	Input	$2^{number\ of\ inputs}$
NOT	1	$2^1 = 2$
AND	2	$2^2 = 4$
OR	2	$2^2 = 4$

2.4 Computational logic

An example logic circuit, with notations and truth table

A ∧ (B ∨ C)



A	B	C	B V C	A ∧ (B V C)
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	1	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

Why is data represented in Binary in a computer system?:

- Binary is the representation of the 'presence' of electricity
- If present or 'on' we use a: 1
- If absent or 'off' we use a: 0
- We can use this idea to change 1 and 0 states through the use of logic gates
- Logic Gates take inputs and covert them to an output
- The digit 1 or 0 is stored in transistors which are in the processor.

Half Adder: A circuit to add two binary bits together

Full Adder circuit: 2 half adders together. This can add more than two binary bits together and deal with carrying bits over

2.5 Translators and Facilities

High Level Language (eg Python)

- Use Instructions are in Words
- Designed to be read by human programmers
- Portable/translated for different machines

PROS: Easy for humans to understand as written in English
CONS: Needs to be translated in to machine code so not as fast to execute

Machine Code (Binary Code) (low level)

- Instructions are in binary code Designed to be read/executed by the computer
- Specific to a particular machine

PROS: Fast no need to translate already in binary, close to CPU
CONS: Difficult to read and understand for humans

Assembly Language (low level)

- Low Level Language written in mnemonics that closely reflect the operations of the CPU
- Eg LOAD

PROS: A bite easier to understand than binary as uses mnemonics
CONS: Still needs translating to machine code using an Assembler translator

How Instructions are stored in Binary.

- The instruction consists of an operator/op code and an operand
- both stored as bit patterns
- (op code) from a given instruction set
- Each op code has a unique bit pattern

Why Use Binary.

- So that computers can be based on logic circuits.
- (each part of the circuit) can be in one of two states
- 0 and 1/true or false

Translators

Interpreters:

Translates **one line of HL** code at a time…
 … and executes it
 … stops when it finds an error
 … can be resumed

Compilers:

Translates **whole program of HL code** at a time…
 … and executes it
 … stops when it finds an error

Assembler:

Translates **Assembly language** into machine code

Selection

A condition is used to decide executed

- if x = 1 then y = 3 else x whether code should be = 2

Sequence

A list of instructions to be followed one after the other. Step by step

Iteration

code is executed repeatedly

```
For I in range (10);
    Print (i)
```

2.5 Translators and Facilities

TOOLS FOR PROGRAMMING

IDE (intergrated Development Environment): A software application that provides all the facilities to computer programmers for developing programs. It normally consists of a source code editor, build automation tools and a debugger

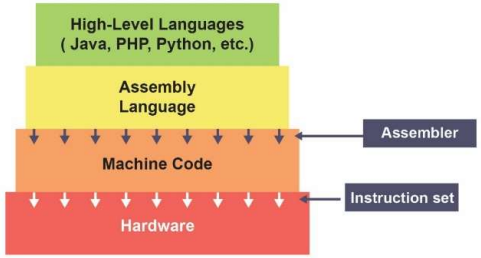
Code Editor – edits program text, you can type our your source code here
Syntax Checks – Highlights syntax errors

Runtime Environment – allows programs to be run one line at a time – helps test programs and locate errors.

Translator – Compiles or interprets the code

Libraries – Provides functions not included in core part of programming languages (eg Random)

Debugger – Helps to detect errors. Suggests what type of error it is and what line it is on.



Source Code: The code the programmer writes

Object Code: code converted by the compiler so that it can be understood by the computer

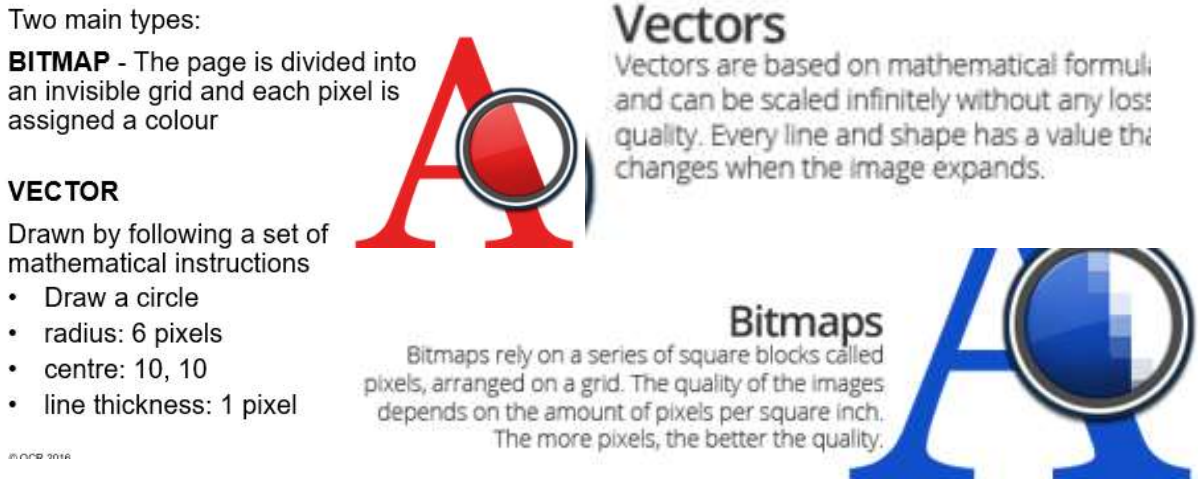
Executable Code: Files which contain **instructions in machine code.** These instructions are **carried out in the computer hardware**

Programming Standards
Code should follow agreed conventions (EG Lowercase for variable names, schemes to be followed).
Language code is written in.
Functions used to tidy up repeated code.
Comments explain the code clearly.
Correct use of indentation.
Useful identifiers (File names & Variable names)
Code should follow agreed conventions

2.6 Data Representation - Images

Images store as binary ….

- Stored as Bitmap file as pixels
- **Each Pixel** of Image is made up of a 1 or 0.
- Following information about image **is stored in file**:
  - Width of the picture in pixels.
  - Number of bits used for each pixel
  - Colour of each pixel.
- **Image Resolution** = The concentration of pixels in an image
- **Higher Resolution** = More Pixels = Larger File Size
- **Lower Resolution** = Less Pixels = Smaller File Size.



Metadata: data about data - Certain information must be defined for the bitmap image. E.g. width, height, pixels, colours,

Colour depth

How many bits will be used to store the colour for each pixel in the grid. E.g. 8 bit (1011001) allows 256 different colours.

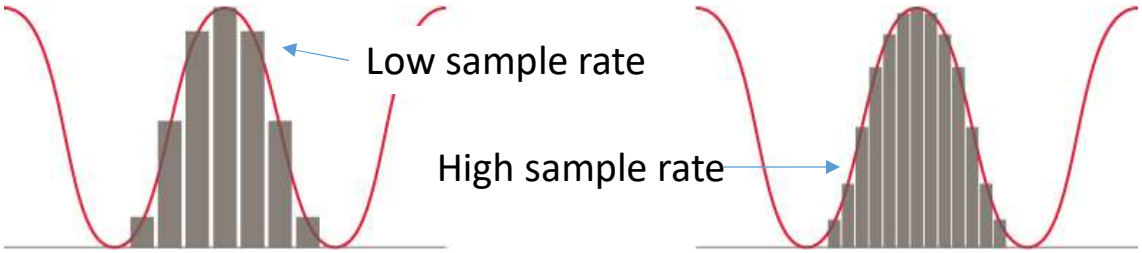
The greater the colour depth: The more realistic colours, The more data needs to be stored and the larger the file size on disk

2.6 Data Representation - Sound

Sound stored as Binary

- The height/amplitude of the sound wave is measured
  - at regular intervals
  - and converted to binary.
- If the interval is smaller
  - More samples taken
  - more data to store
  - larger files
  - the sound reproduced is closer to the original - better quality.

Sound exists as waves – however as computers only understand binary values this **needs to be converted** into such Sounds created on a computer exist as digital information that is **encoded as audio** files



Digital sound is broken down into **thousands of samples per second** – each of these **samples** is then **stored as binary data.**
The **quality** that the samples are stored with depends on different factors:

- ***Sample Frequency*** **-** The number of audio samples captured every second
- ***Sample Size/ Bit Depth*** **-** Number of bits available for each sample
- ***Bit Rate*** **-** The number of bits used per second of audio

Compression: The method computers use to make files smaller by reducing the number of bits (1's and 0's) used to store the information. Can be used for any files including sound and image files

Lossy compression: makes the file smaller by getting rid of bits that aren't really noticeable to the human eye. Once a file has been compressed using a lossy method, any bits that are lost **cannot be recovered**.

Lossless compression: The file is reduced **without losing any quality** so the original file can be restored.

Lossy compression usually has smaller file sizes than lossless

2.6 Data Representation - Sound

Bit Rate

- The bit rate of a file tells us how many **bits** of data are processed every second. Bit rates are usually measured in **kilobits per second (kbps)**.

How do you Calculate bit rate?

- The bit rate is calculated using the formula:

Frequency × bit depth × channels = bit rate

- A typical, uncompressed high-quality audio file has a **sample rate** of 44,100 **samples** per second, a bit depth (the number of bits available for each sample) of 16 bits per sample and 2 channels of stereo audio. The bit rate for this file would be:

44,100 samples per second × 16 bits per sample × 2 channels = 1,411,200 bits per second (or 1,411.2 kbps)

A four-minute (240 second) song at this bit rate would create a file size of:

14,411,200 × 240 = 338,688,000 bits (or 40.37 megabytes)

Compression: The method computers use to make files smaller by reducing the number of bits (1's and 0's) used to store the information. Can be used for any files including sound and image files

How is sound stored in Binary?

Audio file is **inputted through a microphone** - broken down into thousands of samples per second,

More samples recorded **per second** the **higher the quality** of the audio file, but the **more memory** it will consume.
Each sound sample is stored as **Binary Data.**

The **more bits per sample** also increases **the higher the quality** of the audio, on a CD the bit depth is usually 16 bits

Lossy compression: makes the file smaller by getting rid of bits that aren't really noticeable to the human eye. Once a file has been compressed using a lossy method, any bits that are lost **cannot be recovered**. **+ can produce smaller file sizes. - No good if 100% accuracy required e.g. text files**

Lossless compression: The file is reduced **without losing any quality** so the original file can be restored.
+ Original file can be restored without losing any data. - Not all files can be compressed using lossless compression

2.6 Data Representation - Characters

Character Set
is used to describe the possible characters that can be represented in a computer system. E.g A a, 123, @!”£, emoji’s

Ascii (American Standard Code for Information Interchange)
- Each character is given a binary code
- Uses **7 Bits** this gives **128 possible characters**
- **Extended Ascii** used **8 bits** – **256 characters** – enough for the English language
- Some codes are reserved for control characters (eg TAB, Carriage Return)

Unicode
- Unicode has a much larger character set
- can represent many more characters/characters from all alphabets
- uses 16 bits
- It **uses 2 bytes** that give us 2^{16} possibilities (65,536).
- This is used universally to represent many more languages than our own

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

2.6 Data Representation

Binary – Known as **BASE 2** – **Uses 2 digits (1 or 0)** – This is why we DOUBLE the headings in our tables when we convert binary to denary and vice versa

Denary is known as **BASE 10** – **Uses 10 digits (0-9)**

Hexadecimal is knows as **BASE 16** - uses **16 digits (0-9 and A-F)**

Denary to Binary Conversion

- Write out the column headings (128,64,32,16,8,4,2,1)
- Work out **how** to make the denary number **from those columns.**
- **If you don't use all the table headings DON'T just fill them with 0's**

20 denary is 10100 in Binary…..

16	8	4	2	1
1	0	1	0	0

Does 16 go in to 20? – yes – put a 1.
There is 4 left
Does 8 go in to 4? – no – put a 0
Does 4 go in to 4? – yes put a 1.
At this point we had no numbers left so we add 0's

Binary to Denary Conversion

- Write out the column headings (128,64,32,16,8,4,2,1) over the binary number.
- Add all the column headings in which there is a 1

10000111 = 71 in Denary….. How? I added up the numbers

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

HINT: A typical exam question will not give you the table headings so make sure you remember to use these

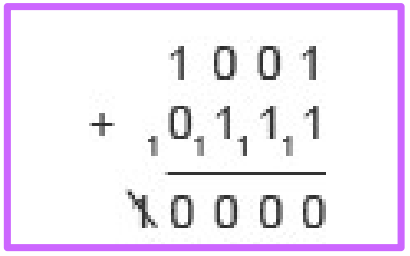
Exam questions usually give you 2 marks
1 for correct answer
1 for showing your working out

Binary Addition

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 Carry 1
1 + 1 + 1 = 1 Carry 1

If the result of the addition has a 1 in the 9th bit then Result cannot be held in 1 byte (8 bits) so will need to have a 2nd Byte.

This is an **OVERFLOW ERROR**.



2.6 Data Representation

HEXDECIMAL
BASE 16. Uses 0-9 and then A-F
Notice that we use the values A-F to represent 10-15

Denary to Hex Conversion

e.g. Convert 167 into Hexadecimal
1. Divide the number by 16:
167/16= 10
2. Record the remainder:
Remainder = 7
So 10 in Hex is A and 7 in Hex is 7
Therefore the answer = A7

Convert Hex to Denary

- Take the hex number and split it
- Times the left by 16 and the right by 1
- Add these together to get Denary
- E.g. See table to convert 4C

16	1
4	C (12 in denary)
4*16=64	12*1=12
64+12=76	

Binary to Hex Conversion

To convert 10101011 into Hex, we can use the following steps:

1. **Split into 2 nibbles**: **1010 1011**
2. Convert **each nibble** into decimal: 1010 = 10 1011 =11
3. 10 in hex is A and 11 in Hex is B
4. **Therefore 10101011 in Hex is AB**

2.6 Data Representation

Hex to Binary Conversion

To convert from Hexadecimal to Binary we can do the opposite, where we convert each digit into a nibble.

To convert **3B** into Binary, we can use the following steps:

1. Split into 2 digits: **3 B**
2. Convert each digit into 4 bits (a nibble!):

3= 0011 B=1011

3. Therefore 3B in Binary is 00111011