1. Binary and Data

Computers work on ones (1) and zeros (0), aka binary.

Binary is responsible of all digital information in the world from letters, numbers, images, and soundwaves. Binary is also the backbone of how all computers **INPUT**, **OUTPUT**, **STORE**, **PROCESS**, and **OUTPUT** information.

Say we have a single wire, an electricity flowing through that wire represents the wire is turned ON. When the electricity is no longer flowing through the wire, it means the wire is turned OFF.

In binary, **1** is *ON* and **0** is *OFF*

> ***1*** *= TRUE which is also ON*
> ***0*** *= FALSE which is also OFF*

These 1's and 0's as individual pieces of information can also called a BIT.

POSITION	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7	BIT N
VALUE	1	0	0	0	1	1	1	...

The table above shows that each position in a binary sequence representing a specific value based on its position. For example, BIT 3 has the value 0 and BIT 5 has the value 1.

Basically, 1 is a single bit value as well as 0. You *cannot* have both 1 and *0* as an individual bit value at the same time. That's only for quantum computers lmao!!

	BIT 1	BIT 2
Correct	1	0
Wrong	10	11

Essentially everything is represented with 1's and 0's, and with more bits you can represent complex information. But to understand that, you must learn #binary .

Binary Number System

In the decimal system, we have 0,1,2,3,4,5,6,7,8,9 which is where we learnt how to count.
In the binary number system, there is only 1 and 0 which you can still count bigger numbers using just those two digits.

> *With binary, you can represent ANY number*

To put the difference aside, the decimal number system has the following positions:

...	2	0	2	3
and so on...	1000's	100's	10's	1's

In the binary number system we have the following positions:

...	1	0	1	0
and so on...	8s	4s	2s	1s
	2^3	2^2	2^1	2^0

> *The "and so on..." means that the position goes on forever, in binary there is 16s, 32s, etc.*

Notice how the positions for the binary system jump by 2^n times?

Current Position	*MATH*
1s	2^0
2s	2^1
4s	2^2
8s	2^3
...	...

We can use these binary positions to calculate decimal numbers.

To count in binary we add every position that is ON or 1 and ignore the ones that are OFF or 0. So say we position 1s and 2s turned on:

1s	2s
1	1

The decimal value will be 3 because 1 + 2 = 3.

If we turned off the 1s position, it would become 0 + 2 = 2

1s	2s
0	1

Now let's turn on the 4s position: 0 + 2 + 4 = 6

1s	2s	4s
0	1	1

If we want to create the decimal number 9, we would do this:

1s	2s	4s	8s
1	0	0	1

This would equate to 1 + 0 + 0 + 8 = 9.

If we turned on the 4s position as well we would get:

1s	2s	4s	8s	
1	0	1	1	

Which will be 1 + 0 + 4 + 8 = 13. The way I think this is if the 4s lightbulb is turned on, it will give the number 4 and if the 4s lightbulb is turned off then the lightbulb will gives us nothing thus `0` because it is turned off.

Representing Letters with Binary

So this is an ascii table...

```
jush@ranedeer:~/Mr-Ranedeer-v3 $ ascii -d
    0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
    1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
    2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
    3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
    4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
    5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
    6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
    7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
    8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
    9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
   10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
   11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
   12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
   13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
   14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
   15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
```

Looks complicated so let's simplify it down:

Number	Letter
65	A
66	B
67	C
68	D
69	E
70	F
	...
90	Z

Each of these letters are represented by an assigned number that is represented as binary under the hood. This is important because these ASCII values are what is inside **Binary Files** which your computer reads and executes programs.

To create the letter A in binary we do:

1s	2s	4s	8s	16s	32s	64s	RESULT	LETTER
1	0	0	0	0	0	1	65	A

$1 + 0 + 0 + 0 + 0 + 0 + 64 = 65$
And which the number 65 is equal to the letter A. Great we made the letter A!

Say we want to create the sequence letters "D O G". The ASCII reference table says that
D --> 68
O --> 79
G --> 71

This would result in the following binary sequence:

1s	2s	4s	8s	16s	32s	64s	RESULT	LETTER
0	0	*1*	0	0	0	*1*	68	D
1	*1*	*1*	*1*	0	0	*1*	79	O
1	*1*	*1*	0	0	0	*1*	71	G

$0 + 0 + 4 + 0 + 0 + 0 + 64 = 68$ ---> D
$1 + 2 + 4 + 8 + 0 + 0 + 64 = 79$ ---> O
$1 + 2 + 4 + 0 + 0 + 0 + 64 = 71$ ---> G

In full binary it would look like this:
0 0 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1
Source: How Computers Work: Binary & Data - YouTube

See Also:
8. Hexadecimal
9. Binary Negative Numbers
10. Binary Fractions
11. Bytes