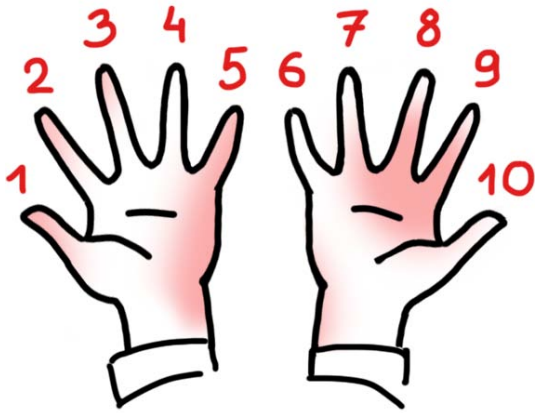
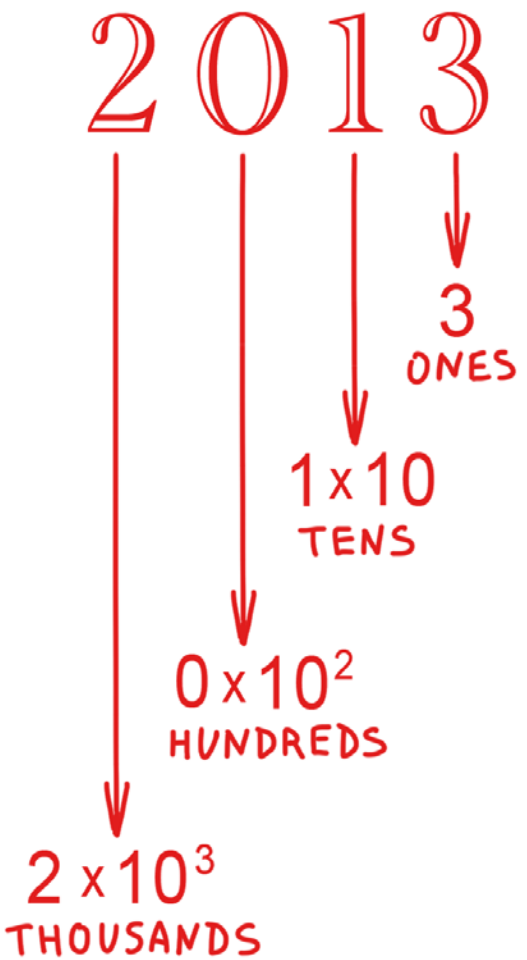
Numeral Systems



Karina Kudymova, 2013

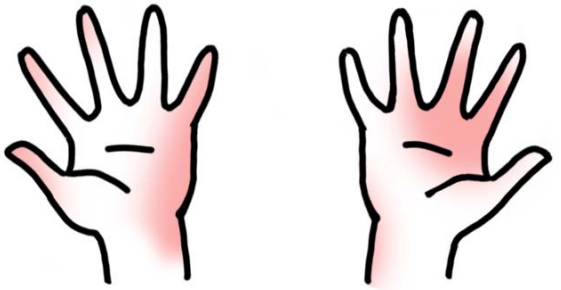
Numbers we use consist of digits. Let's see what each digit means... For example, how many is 2013?



The digit in the right-most position tells us how many ones are in the number, the next one shows how many 10s are there, next - 100s, and then 1000s

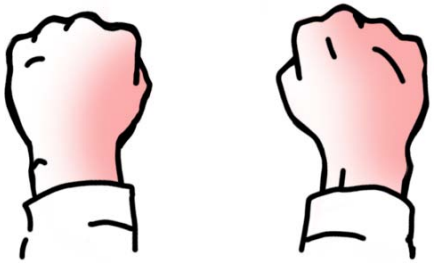
Do you know that 2013 is the first year with all different digits in a long while? When was the last year with four different digits?

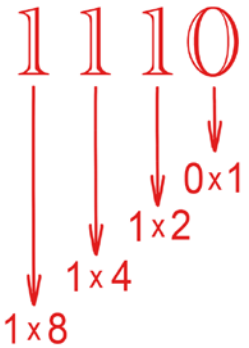
A way to represent numbers is called a *numeral system*. We use the representation based on the powers of 10, called *decimal* and 10 is its *base*. 10 is a very convenient base for counting because we have 10 fingers.



Decimal form is a very compact representation of numbers. It is much shorter to write 14 than to repeat something 14 times.

Instead of 10 we could use any other number as a base of the numeral system. Let's start with base 2 and consider the *binary* system. Instead of counting with fingers that would be counting with one's fists.

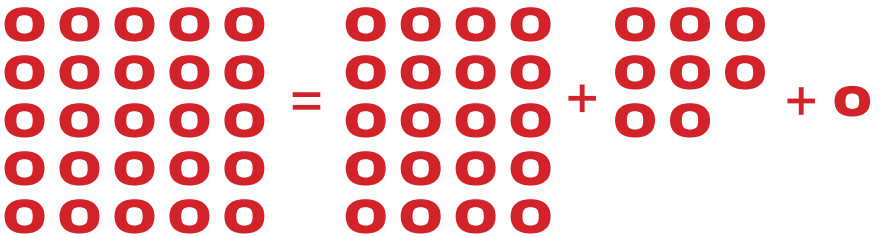




In binary, instead of powers of ten, digits correspond to powers of two: 1, 2, 4, 8, 16, 32... We place units in the right-most position and then increase powers of two as we move to the left. Let's see what binary number 1110 is.

8+4+2=14 bin(1110)=dec(14)

Let's try to represent decimal number 25 in binary. What's the highest power of two that is less than 25. It's 16, so we'll put 1 in the corresponding position. Once we covered 16, we are left with 9. What's the largest power of two that we can subtract from it? 8 — we put 1 in corresponding position. And then we are left with just 1 — final 1 in the right-most place.



$$25=1\cdot16+1\cdot8+0\cdot4+0\cdot2+1$$
$$1 \quad 1 \quad 0 \quad 0 \quad 1$$

dec(25)=bin(11001)

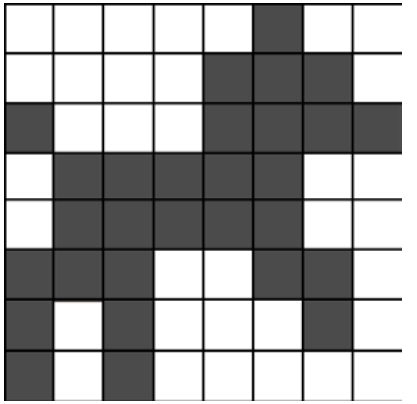
Can your write your age in binary?

Note that in binary we are just using two digits, 0 and 1.

A digit in a binary number is called *bit* (*b*inary dig*it*). Eight bits form a byte.

Binary numbers find their use in computers. For example, an image can be represented by a simple *bitmap*.

⇨



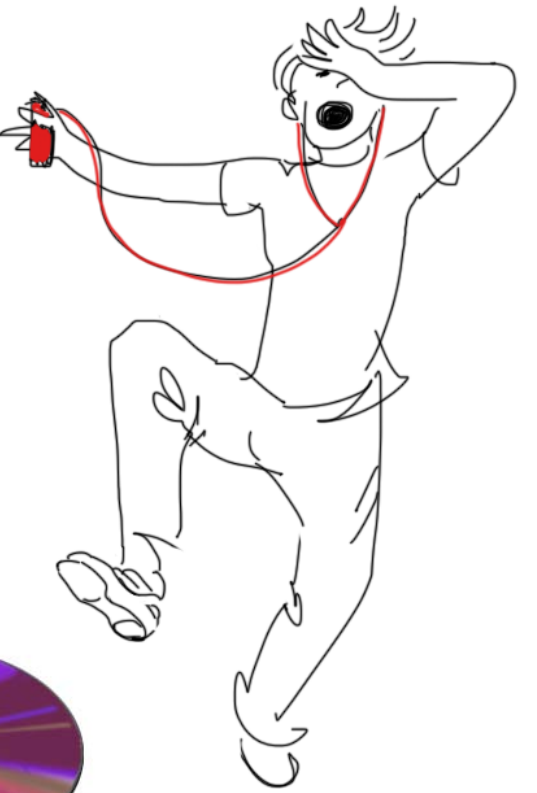
⇨

0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
1	0	0	0	1	1	1	1
0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	0
1	1	1	0	0	1	1	0
1	0	1	0	0	0	1	0
1	0	1	0	0	0	0	0

⇨ 00000100,00001110,10001111,01111100,01111100,11100110,10100010,10100000

For computers, every object is a binary number, but these numbers can get very long.

Thousand bytes = *Kilobyte*
Million bytes = *Megabyte*
Trillion bytes = *Gigabyte*

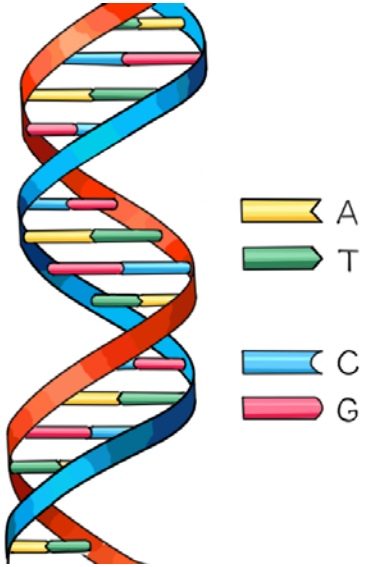


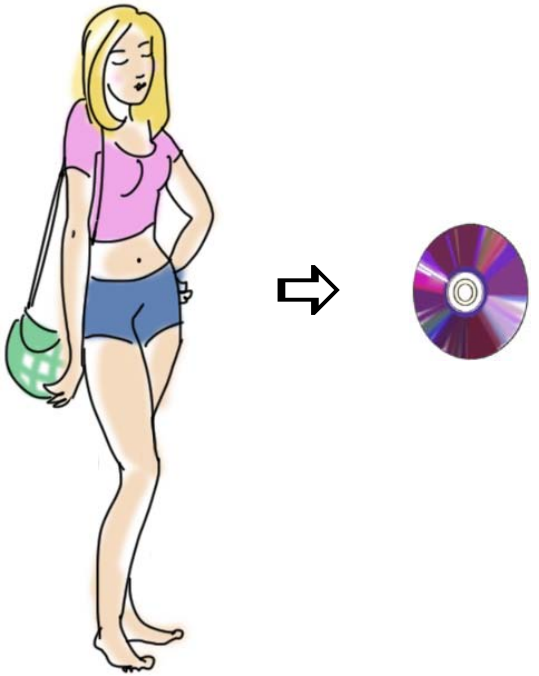
An e-mail can be in kilobytes, an MP3 song is typically a few megabytes, and a movie is typically a few gigabytes.

Next time you watch a move, keep in ming that what's on the DVD or in a digital download is just a very long binary number.

Are there large numbers like that in nature? Yes, in biology DNA is formed from four types of nucleobases: A, C, T, G

That's why DNA is just a number in base 4. Human genome has approximately 3.2 billion base pairs (digits in base 4).





Each base 4 digit can be represented by 2 bits, so human genome has approximately 6.4 billion bits, or 800 Megabytes. That's almost exactly the capacity of a CD.

In addition to binary and decimal, computer programmers often use numbers in base 8 (*octal*) and 16 (*hexadecimal*). Octal numbers use digits from 0 to 7 and are based on powers of 8. Can you tell what is octal number 31, oct(31), is in decimal.

So oct(31)=dec(25). There is a standing joke that this why computer programmers mistake Halloween for Christmas.

We talked about numeral systems with bases 2, 4, 8, and 10. All these system are *positional*: the meaning of each digit depends on its position in the number. An example of non-positional numeral system is Roman Numerals, but that's a topic for another section.