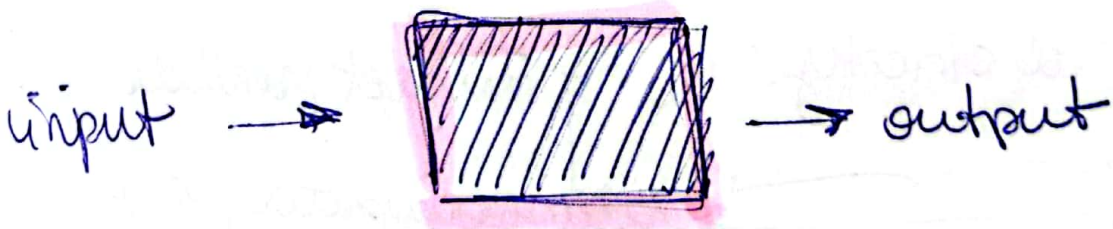
Web Programming with Python and JavaScript

HARVARD

week 0 → Scratch

What is programming?

What is computer science?



What the language so we speak?

computers, samurai, only have these zeros and ones (binary)

They represent one thing or the other, is just really simple shape

decimal system → 0 to 9

10 digits

So why that particular pattern? And why those particular Zeros and ones?

is just really simple for a computer

↓

they're powered by electricity

It's a really simple thing to just either store some electricity OR don't store some electricity.

ON or OFF

1 or 0

In fact, inside of a computer, a phone, anything these days that's electronic, is some number of switches, otherside known as transistors.

these are just tiny little switches that can get turned on and off.

And by turning these things ON and OFF in patterns, a computer can count from 0 on up to 7, and even higher than that.

These switches, really, you can think of being as like switches like this.

But how is it that these patterns come to be?

Well, these patterns actually follow something very similar.

In binary, it's actually the same thing

100 10 1

1 2 3

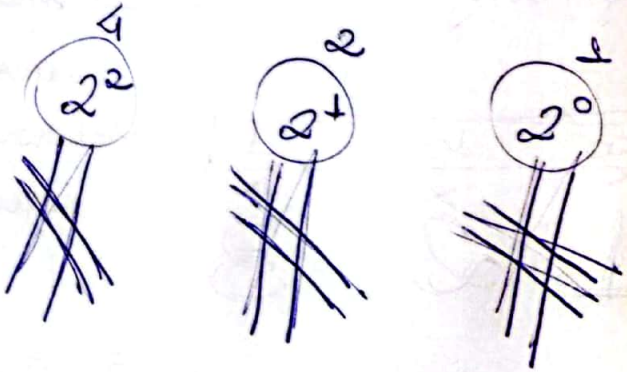
100×1 + 10×2 + 1×3 = 123

they're technically just powers of 10.

→ why 10?

You have 8 and 10 digits, 0 through 9.

In the binary system, if you're going to use three digits, just change the bases if you're using only ZEROS and ONES.



So, why did we get these patterns that we did?

4	2	1	
0	0	0	it's 4 times 0, 2 times 0, 1 times 0, obviously 0.

this is why we got the decimal number 1 in binary; this is why we got the number 2 in binary;

$1^1 = 1$ → 0 0 1

$0\ 1^2\ 0 = 2$ → 0 1 0

$= 2$

$1^4\ 0^2\ 0^1 = 4$ → 4 2 1: 1 0 0 = 4

$1^4\ 0^2\ 1^1 = 5$ → 4 2 1: 1 0 1 = 5

4 2 1: 1 1 0 = 6

4 2 1: 1 1 1 = 7

If you wanted to count as high as 8, what do you have to do?

ADD A BIT

And another light bulb, another switch, and, indeed, computers have standardized just how many zeros and ones, or bits or switches, they throw at these kinds of problems.

In fact, most computers would typically use at least eight at a time.

And even if you're only counting as high as three or seven, you would still use eight and have a whole bunch of zeros.

But that's OK, because the computers these days certainly have so many more, thousands, millions of transistors and switches that that's quite OK.

If we can count as high as seven or, [illegible] as high as we want, that only seems to make computers useful for things like Excel, like number crunching.

But computers, of course, let you send text messages, write documents and so much more.

So how would a computer represent something like a letter, like a letter A of the alphabet, at the end of the day, all they have is switches?

Hmmm...

You can represent letters in numbers

like:

1 A — 2 B — 3 C — 4 D ...

26 Z

We first all have to agree somehow that ONE number is going to represent one letter.

Maybe we can even take into account uppercase and lowercase. We just have to agree and sort of write it down in some global standard.

Humans, indeed, did just that.

They didn't use 1, 2, 3 ...

it turns out they started a little higher up.

Capital A has been standardized as the number 65. And capital B has been standardized as the number 66. And you can kind of imagine how it goes up from there.

And that's because whatever you're representing, ultimately, can only be stored, at the end of the day, as zeros and ones.

01000001

So, if that pattern of zero and ones appear in a computer, it might be interpreted then as indeed a capital letter A, eight of those bits at a time.

→ We might have now created a problem. It might seem, if I play this naively, that ok how do I now actually do math with the number 65?

So how the computer have this mapping from numbers to letter, but still support numbers?

It feels like we've given something up. We could perhaps have some kind of prefix, like some pattern of zeros and ones → 01000001

Here comes another pattern that represents a NUMBER or a LETTER

that indicates to the computer here comes another pattern that represents a letter.

How might a computer distinguish these thos?

// Maybe having a different file format,

odd text or just check the graphic

or indeed, and that's spot-on. "

redmonte "no sense!"

the reason we have all of these different like formats in the world, like JPEG, GIF and PNGs and word documents .docx, and Excel files and is because a bunch of humans got in a room and decided in the context of this type of file, or really more specifically, in the context of this type of program, Excel x Photoshop x Docs or the like, we shall interpret any patterns of 0s and 1s as being maybe numbers for Excel, maybe letters in a text message program or Google Docs, or maybe even colors of the rainbow in something like Photoshop and more.

So its context dependent. And we'll see, when we answer start programming, you the programmer will ultimately provide some hints to the computer that tells the computer, interpret it as follows.

dicas 😉

So, similar in spirit to that, but not quite a standardized with these prefixes.

So this system has actually has a name (ASCII), the American Standard Code for Information Interchange.

And indeed, it began in the US, and that's why it's actually a little biased toward (A's through Z's) and a bit of punctuation as well.

tendencioso

And that quickly became a problem.

But if we start simply raw, in English, the mapping itself is fairly straightforward.

A → 65
B → 66
...

Suppose that you received a text message, or email, and underneath the hood;

embaixo do capô

If you look inside the computer, what you technically received in this text or this email happened to be the numbers 72, 73, 33, or really the underlying pattern of zeros and ones.

What might your friend text you as a message, if it's 72, 73, 33?

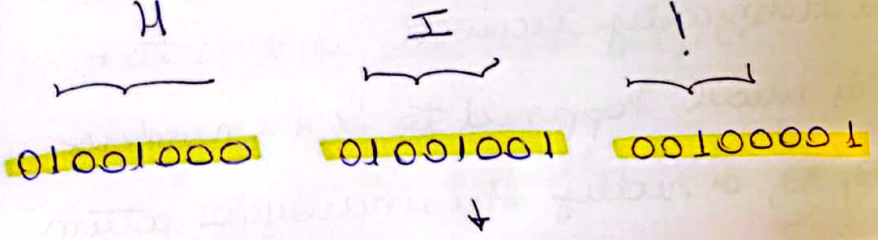
Ascii

Hi !

When I said that we just need to write down this mapping earlier, this is what

people did.

they wrote it down in a book or in a chart. And computers, Android devices, just know this mapping by heart, if you will. they've been designed to understand these letters.



It's import to note that when you get these patterns of zeros and ones in any format, be it email or text or a file, they do tend to come in standard lengths, with a certain number of zeros and ones altogether. And this happens to be 8 plus 8, plus 8.

so just to get the message "hi, exclamation point", you would have received at least, it would seem, some 24 bits.

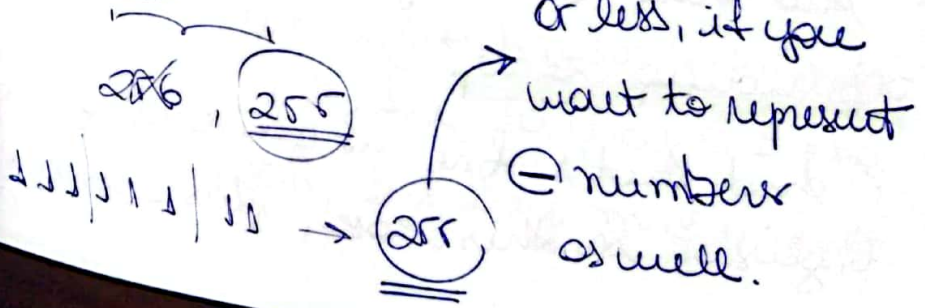
But frankly, bits are so tiny, literally and mathematically, that we don't tend to think or talk, generally, in terms of bits.

You're probably more familiar with bytes.

1 BYTE = 8 BITS

And even those, frankly, aren't that useful if we do out the math.

How high can you count if you have eight bits?



So this is useful because now we can speak, not just in terms of bytes but, if the files are bigger, kilobytes is thousands of bytes, megabytes is millions of bytes, gigabytes is billions of bytes, terabytes are trillions of bytes, and so forth.

(assim por diante)

We have a vocabulary for these increasingly large quantities of data.

The problem is that, if you're using Ascii and, therefore, eight bits or one byte per character, and originally, only seven, you can only represent 255 characters.

And that's actually 256 total characters, including zero.

And that's fine if you're using literally English, in this case, plus a bunch of punctuation. But there's many human languages in the world that need many more symbols and, therefore, many more bits.

↳ including emojis! 😊

There are very much in vogue these days isn't just ↩ characters, like the letters of an alphabet, patterns of 1s and 0s and ones that you're receiving.

What the world has also standardized.

↓

This newer standard is called UNICODE

It's a superset of what we called Ascii.

Unicode is just a MAPPING of many more numbers to many more letters or characters that might use eight bits for backwards compatibility with the old way of doing things with Ascii, but they might also use 16 bits.

And if you have 16 bits, you can actually represent more than 65 000 possible letters. And that's getting up there.

Unicode might even use 32 bits to represent letters and numbers and punctuation symbols and emojis. And that would give you up to 4 billion possibilities.

And one of the reasons we see so many emojis these days is we have so much room. We've got room for billions more, literally.

4,036,991,159

or in binary...

11110000 10011111 10011000 10110111

— emoji of medical mask 😷

And we have different versions of the same. There's bunches of other interpretations by other companies as well.

Unicode, as a consortium, if you will, has standardized the descriptions of what these things are. But the companies themselves, manufacturers out there, have generally interpreted it as you see fit. And this can lead to some human miscommunications.

It's different because we have different interpretations of the data. This has happened too when what was a gun become a water pistol in some manufacturer's eyes. And so it's an interesting dichotomy between what information we all want to represent and how we choose, ultimately, to represent it.

Decimal ↔ Binary → Unary is another ↔ Hexadecimal ↔ too.

00 00 00 → represent the decimal number Zero

Color

How to represent something like color?

What are our options if all we've got are zeros and ones and switches? RGB

- RGB indeed is something that represents some amount of red and some amount of green and blue and indeed computer can represent colors by just doing that.

- If we mix those colors together, you can indeed get a very specific one.

- It we do indeed decide as a group to represent any color of the rainbow with some mixture of some red, some green, and some blue, we have to decide HOW to represent the amount of red and green and blue

72	73	33

It turns out if all we have are 0 and 1, are numbers, let's do just that.

For instance, suppose a computer we're using these three numbers 72, 73, 33. No longer in the context of an email or a text message, but now in the context of something like Photoshop, maybe this first number could be interpreted as representing some amount of red, green and blue, respectively.

- And indeed, you can come up with a numbers between 0 and 255 for each of those colour to mix any other color that you might want.

Images

Pixel's just a dot on the screen and you've got thousands, millions of them these days horizontally and vertically.

Things get pixelated because what you're seeing is each of the individual dots that compose this particular image. And apparently each of these individual dots are probably using 24 bits, eight bits for red, 8 bits for green, 8 bits for blue, in some pattern.

time

And about Videos?

How would a computer, knowing what we know now, represent something like a video? How might you represent a video using only zeros and ones?

What video really adds is just some notion of time. → sequence because time is passing.

With a whole bunch of images, maybe 24 maybe 30 per second, if you flip them by the human's eyes, we can interpret them using our eyes and brain that there is now movement and therefore video.

And about music?

Computers can certainly represent those sounds, too.

Like MIDI and might just represent each note that you saw a moment ago essentially as a sequence of numbers.

But more generally, you might think about music as having notes, for instance, A through G, maybe some flats and some sharps, you might have the durations like

long is the note being hard or played on a piano or some other device, and then just the volume like how hard does a human in the real world press down on that key and therefore how loud is that sound.

It would seem that just remembering little details like that quantitatively we can then represent really all of these otherwise analog human realities.

So that then is really a laundry list of ways that we can just represent information.

Computers or digital have all of these ≠ formats, but at the end of the day and as fancy as these devices in your car, it's just zeros and ones, tiny little switches or light bulbs, represented in some way and it's up to the software that you and I and others write to use those zeros and ones in ways we want to get the computers to do something more powerfully.

How does a file format like .mp4 discriminate between audio and video?

ZIP file → somehow your computer is using fewer zeros and ones to represent the same amount of information, ideally without losing any information.

Multimedia → we'll touch on a little bit in a few weeks, there are both lossy and lossless formats out there.

Lossless → you lose no information whatsoever. But more commonly as you're alluding to one is lossy compression.

You have containers like QUICKTIME and the MPEG container that can combine ≠ formats of video, ≠ formats of audio in 1

life, but there, too, do designers have discretion.

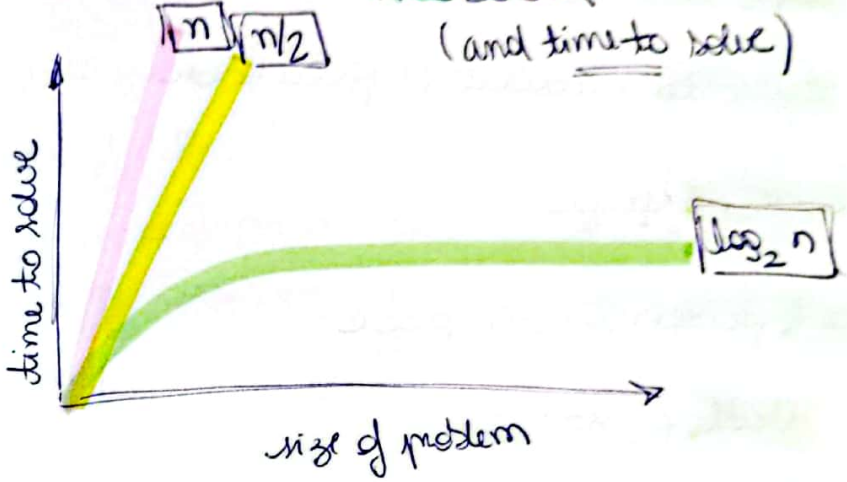
Algorithm

Step-by-step instructions for solving problem incarnated in the world of computers by SOFTWARE.

When you write software / programs, you are implementing one or more algorithms, one or more sets of instructions for solving some problem, and maybe you're using this language or that, but no matter the language you use, the computer is going to represent what you type using just 0 and 1.

STEP-BY-STEP looking for the solution to some problem.

SIZE OF PROBLEM (and time to solve)



Pseudocode

- Is not a specific language
- It's not like something we're about to start coding in, it's just a way of expressing yourself in any human language succintly

How to formalize the code?

1 Pick up phone book

2 Open to middle of phone book

3 Look at page

4 If person is on page

5 Call person

6 Else if person is earlier in book

7 Open to middle of left half of book

8 Go back to line 3

9 Else if person is later in book

10 Open to middle of right half of book

11 Go back to line 3

12 Else

13 Quit

. Highlighted in yellow here are what I'm talking about. We're going to start calling functions

Lots of different programming languages exist, but most of them have what we might call functions, which are actions or verbs that solve some smaller problem.

↓

You might use a whole bunch of functions to solve a big problem because each function tends to do something very specific or precise

> that can be translated in CODE.

→ BOOLEAN EXPRESSIONS

In blue, are what we might call conditions

the things that you do conditionally

based on the answer to some question

It's kind of like FORKS in the road

some DIRECTIONS

bool — YES — TRUE — 1
     — NO — FALSE — 0

We just need to distinguish (1) scenario from another.

The last thing manifests in this pseudocode is in green. →

they CALL LOOPS

Some kind of CYCLE, some kind of directive that tells us to do something.

Our program in C, for example:

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

there's more overhead and more syntax and clutter than there is actual idea.

Today, programming languages have typically a much smaller vocabulary than any actual human language, but at first it might indeed look quite cryptic.

Scratch ↓ (risca)

In Python:

```
print("hello, world")
```

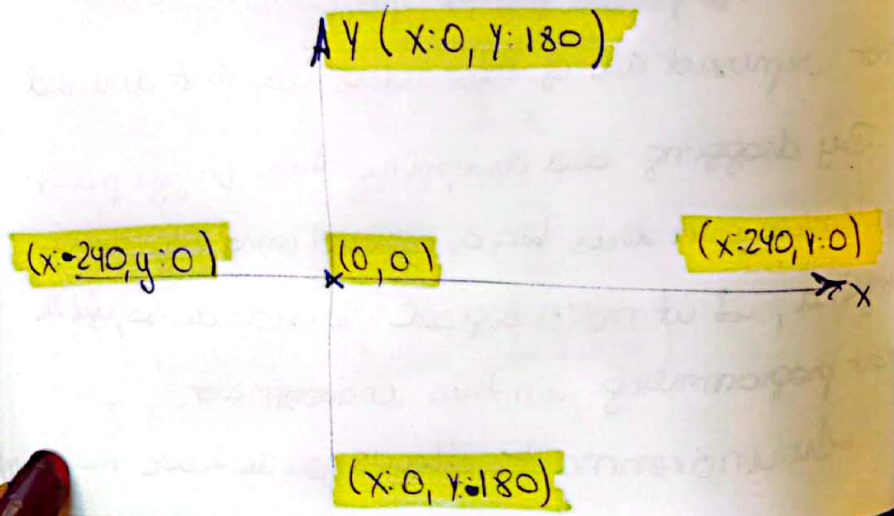
Pallete of puzzle pieces, programming blocks that represent all of those ideas we just discussed

By dragging and dropping these puzzle pieces or blocks over this big area and connecting them together, if it makes logical sense to do so, we'll start programming in this environment.

The environment allows you to have mul

tiple sprites, so to speak multiple characters, things like a cat or anything else, and those sprites exist in this rectangular world up here that you can full screen to make bigger and this hungry default is Scratch, who can move up, down, left, right and do many more things, too.

Within it's Scratch's world you can think of it as perhaps a familial coordinate system with Xs and Ys which is helpful only when it comes time to position things on the screen.



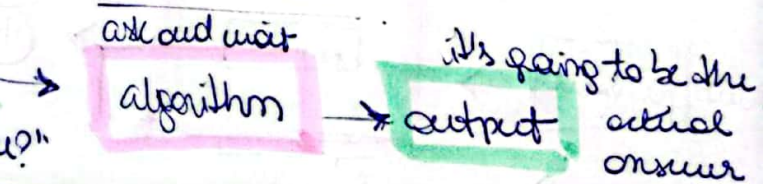
So those numbers generally don't so much matter because you can just max relatively in this world up, down, right, left, but when it comes time to precisely position some of these sprites or other imagery, it'll be helpful just to have that mental model of up, down, left and right.

~~impostor~~ Scratch blocks !

the languages can listen for and respond to. 😊 Indeed, that's why when you tap the phone icon on your phone, the phone application starts up because someone wrote software that's listening for a finger press on that particular icon.

So Scratch has these some things, too.

PUZZLES



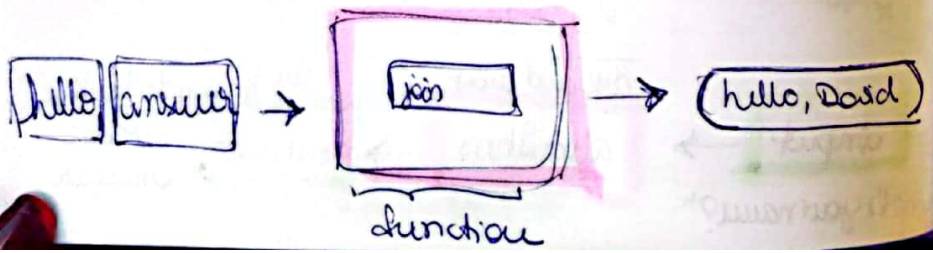
variable called answer

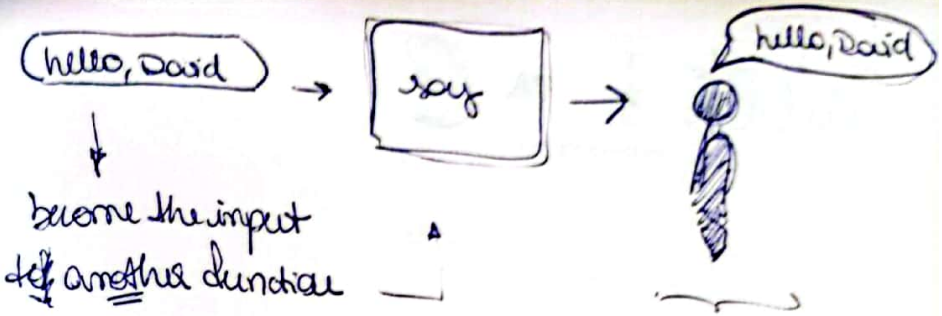
we're joining it with that 1st argument, "Hello,"

So already we see that some functions like join can take not one but 2 arguments or inputs, and that's fine.

The output of join is presumably going to be Hello, David or Hello, Carter or whatever the human typed in.

That output notice is essentially becoming the input to another function, just because we've kind of stacked things or nested them on top of one another.





But you can ultimately really kind of spice these things up.

Identify the SUBPROBLEMS of a bigger problem.

then you can start making progress

Conditionals

(some examples).

Week 0 ✓