# software

# digital-culture-2019

# overview

This is going to talk about software, what it is, what makes it strange, how it became what it is today, looked at it through the lens of artificial intelligence.

What are the socio-cultural dynamics which made software what it is today? What are some of the myths and beliefs that pushed research forward? What are the actions we are taking to sustain that myth?

In the second part of this session, we will review what we've learned last time about programming, and apply it towards building a simple Twitter surveillance program.

# summary

## intro

---

## digital culture - software

welcome!

## plan for today

-> history of programming

-> practice of programming

## housekeeping

final submissions should be made on **glitch**

- introduction
- methodology
- results
- analysis
- conclusion

you have the option to have additional media for your submission (youtube channel, instagram account, tiktok account, twitter account, etc.)

What you should be focusing at this point is to finish up all of your data collection, and start analyzing it:

- What are the most important findings? What are the secondary findings?
- Does it correlate with what you've read? Which aspects of your bibliography does it reinforce?
- Does it contradict what you've read? Why do you think that is?

As for problems, if there is anything that you're not certain about, you should be getting to it right away, in order to set aside any possible obstacle. Never recorded a podcast? Record a test interview. Never shot a video? Shoot a quick video. Make sure you know how to use all the software needed to make it sound/look good (respectively Audacity/OpenShot/iMovie/Windows Movie Maker, etc.)

## software

Software didn't just "appear", it has been constructed through a complex interplay of science, engineering, anthropology and sociology. It is what we call a **sociotechnical object**, an object which reaches from, and back into, both society and technics.

Because software can be so elusive, we will look at it from multiple perspectives:

- its history,
- its language,
- its specificities
- and its metaphors.

---

https://mitpress.mit.edu/books/codespace
Code/Space, a book about how software reconfigures physical spaces.
https://cidadeinseguranca.files.wordpress.com/2012/02/deleuze_control.pdf
One of the best pieces on the change of political paradigms brought about with the appearance of computers.
https://kathmandupost.com/world/2019/09/05/india-s-citizenship-check-exercise-excludes-nearly-100-000-nepali-speaking-people-in-assam
A modern example of how counting has very real political causes and consequences.
https://thereader.mitpress.mit.edu/algorithms-are-redrawing-the-space-for-cultural-imagination/
A summary of the book which introduced the concept of "effective computability"
https://en.wikipedia.org/wiki/G%C3%B6del's_incompleteness_theorems
Gödel's Incompleteness Theorem
https://www.ascii-code.com/
ASCII table, or how to turn letters into numbers
https://en.wikipedia.org/wiki/George_Boole
George Boole, father of boolean logic and founding component of computing
https://patentsgazette.uspto.gov/week02/OG/html/1482-2/US10891948-20210112.html
Patent filing from Spotify to detect user's mood based on microphone access
https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf
The original Turing paper

## the question

are computers very smart or very stupid?

- they are not conscious
- they start stupid but they can become smart
- they break down very fast
- they can program things fast
- they are relatively smart

computers are very *fast*

Software is a set of instructions that can be executed at any scale, and that can calculate anything that can be calculated. Software is both everything and nothing. It is like speech: powerful, but ephemeral, ordering the doing, both subduing and subdued by action.

Software is abstraction. It is a series of operations that can operate on any numerical formula.

Software is the moment when the human interfaces with the (computing) machine, before the machine interfaces back with the world. Software is the language of the digital and, in order to understand the digital, we must

understand software.

However, software is also human-made, with its messiness appearing whenever it is confronted with the analog nature of real life.

## timeline

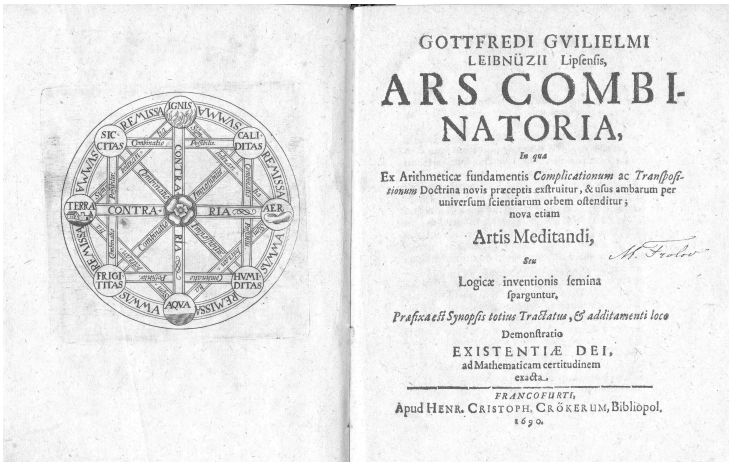
3000 BC -> invention of writing

300 BC -> invention of numbers

1950 AD -> invention of software

## software history

it starts with a desire for a universal, non-ambiguous language characteristica universalis



GOTTFRED GVILIELMI
LEIBNÜZII Lipſenſis,

ARS COMBI-
NATORIA,

*In qua*

Ex Arithmeticæ fundamentis *Complicationum* ac *Tranſpoſitionum* Doctrina novis præceptis exſtruitur, & uſus ambarum per univerſum ſcientiarum orbem oſtenditur; nova etiam

Artis Meditandi,

*Seu*

Logicæ inventionis ſemina ſparguntur,

*Præfixæeſt Synopſis totius Tractatus, & additamenti loco*

Demonſtratio

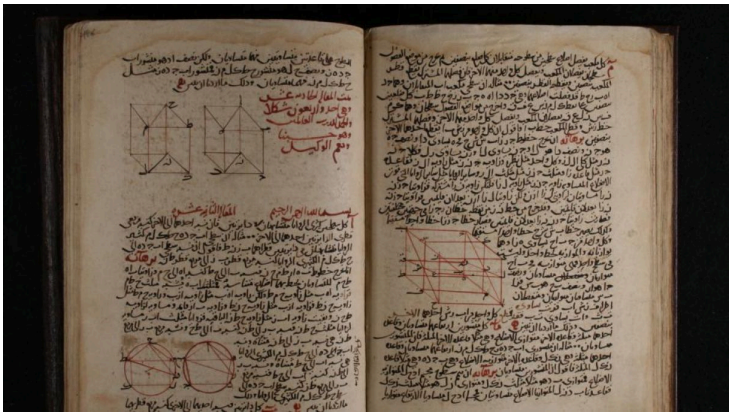
EXISTENTIÆ DEI,

ad Mathematicam certitudinem exacta.

FRANCOFURTI,
Apud HENR. CRISTOPH. CRÖKERUM, Bibliopol.
1690.

de arte combinatoria (1666)

and reusable numerical solutions

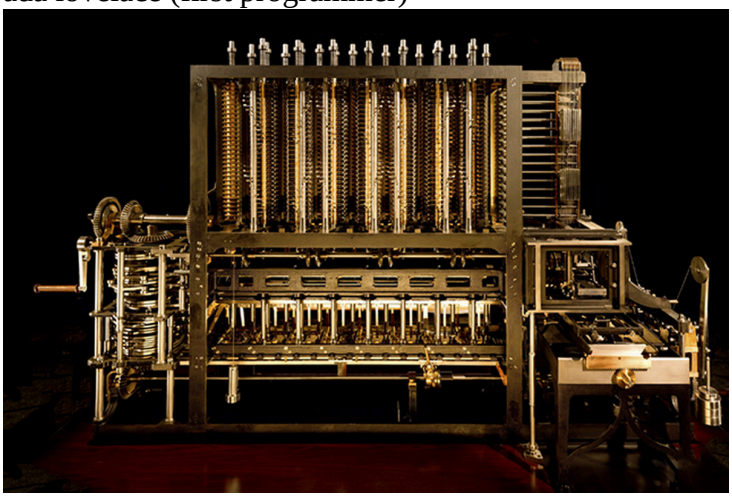


the compendious book on calculation by completion and balancing (820 AD)

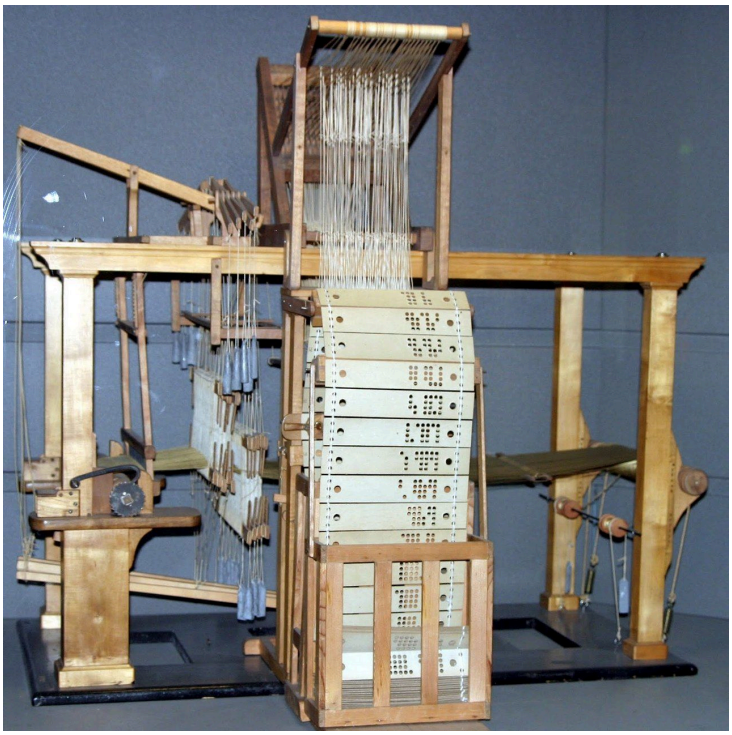
*an algorithm is a process which solves a problem, without having to reinvent a solution each time one is confronted to the problem*

charles babbage (proto-computer)

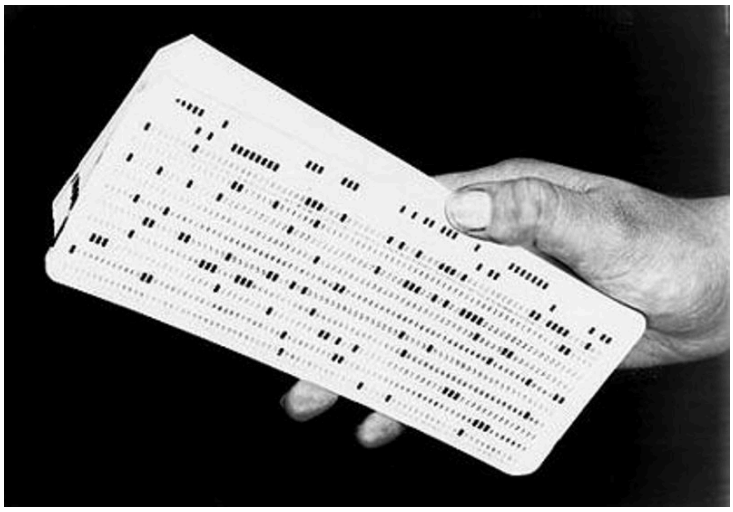
ada lovelace (first programmer)



*differential engine*, 1857

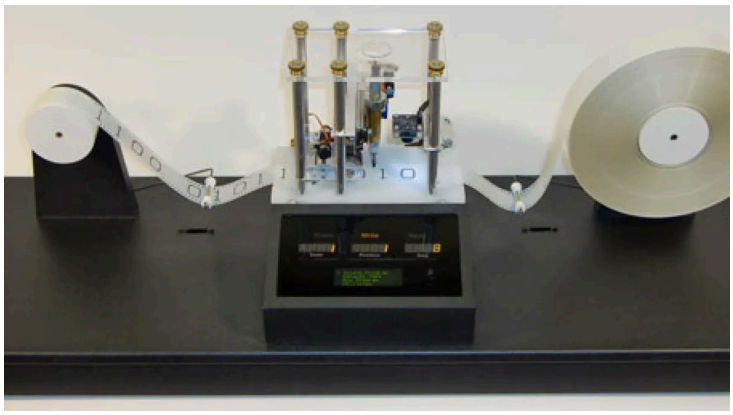


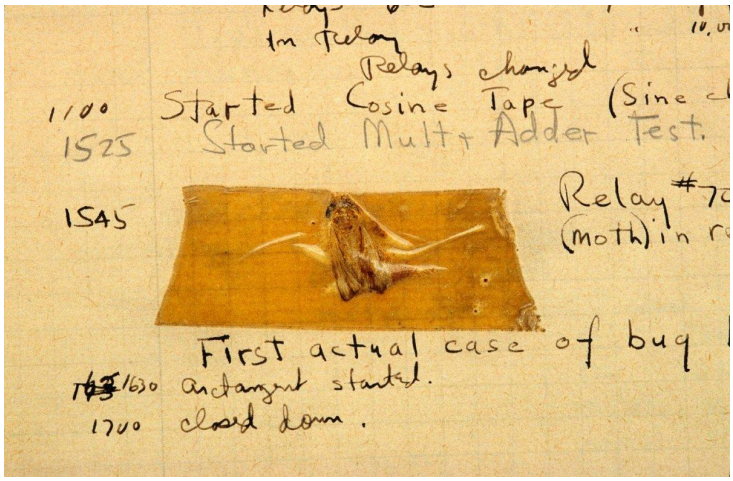
jacquard loom



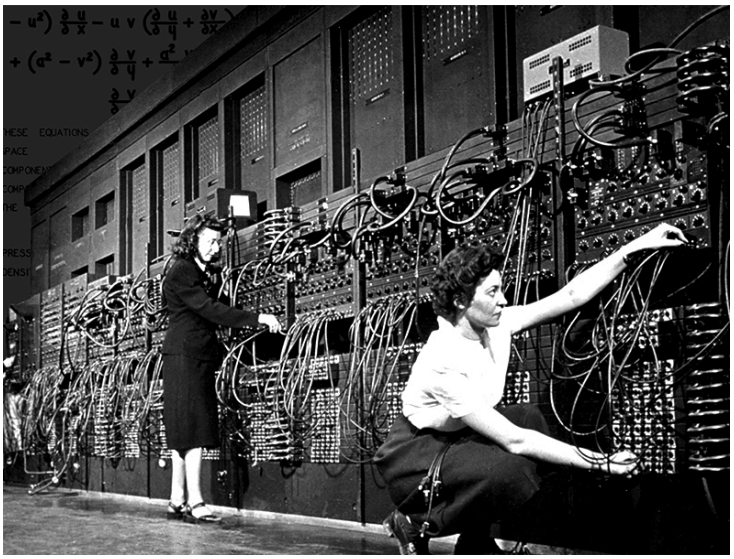
IBM punch cards, 1920s

alan turing (theoretical machine)






the first bug



computers working on computers

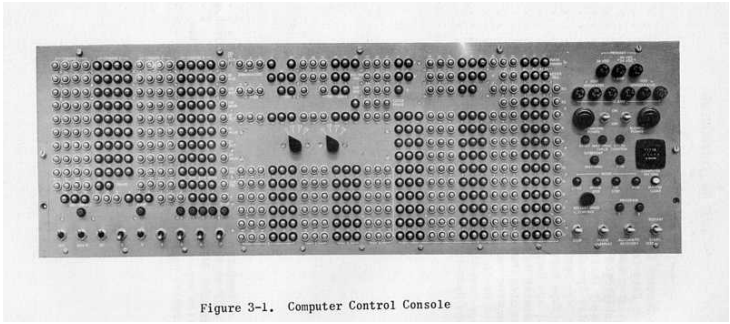


Figure 3-1. Computer Control Console

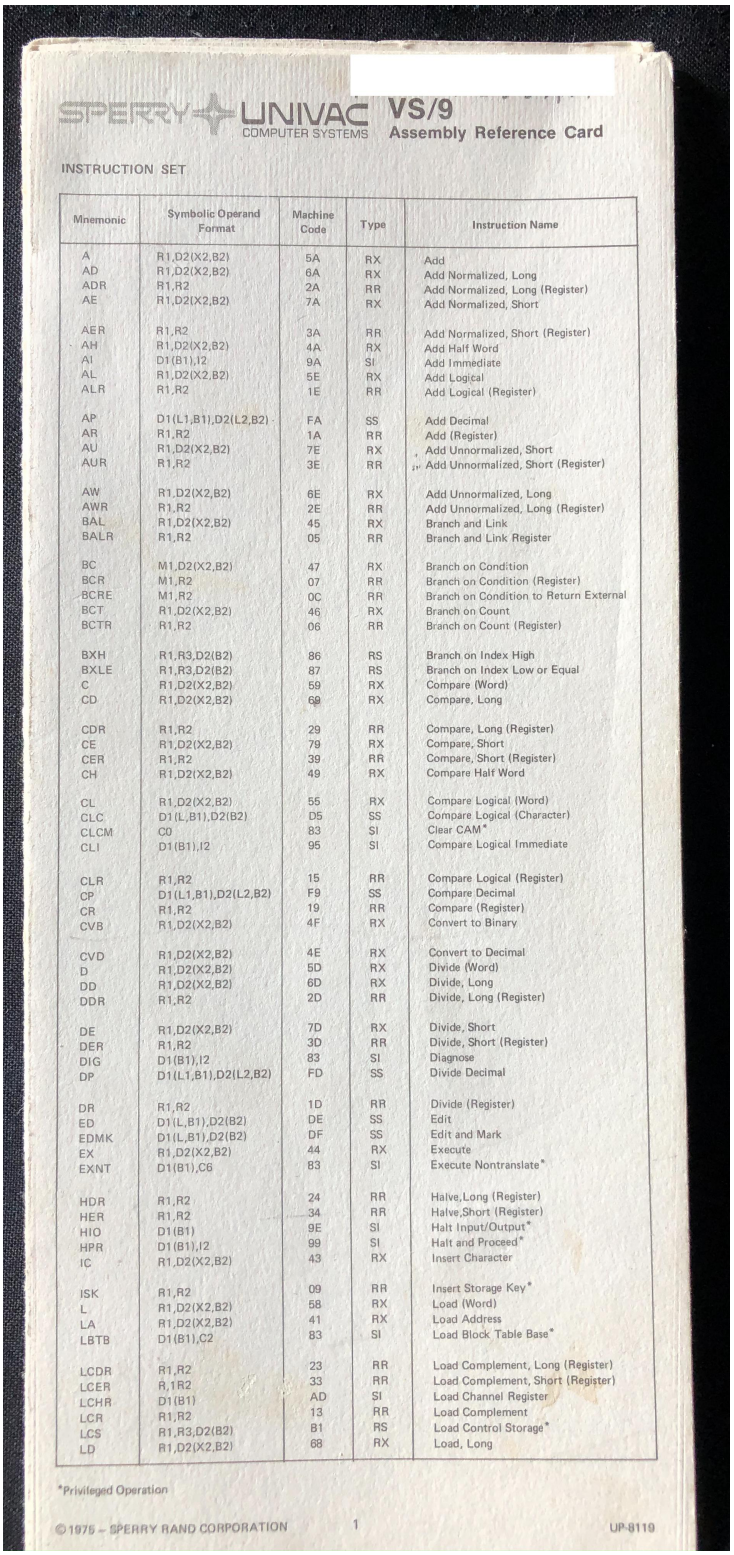
0001 1001

a bit is just a 0 or a 1

a byte is 8 bits grouped together

(basically george boole turned into electricity)

computer love letters ref



SPERRY UNIVAC COMPUTER SYSTEMS **VS/9 Assembly Reference Card**

INSTRUCTION SET

| Mnemonic | Symbolic Operand Format | Machine Code | Type | Instruction Name |
|---|---|---|---|---|
| A | R1,D2(X2,B2) | 5A | RX | Add |
| AD | R1,D2(X2,B2) | 6A | RX | Add Normalized, Long |
| ADR | R1,R2 | 2A | RR | Add Normalized, Long (Register) |
| AE | R1,D2(X2,B2) | 7A | RX | Add Normalized, Short |
| AER | R1,R2 | 3A | RR | Add Normalized, Short (Register) |
| AH | R1,D2(X2,B2) | 4A | RX | Add Half Word |
| AI | D1(B1),I2 | 9A | SI | Add Immediate |
| AL | R1,D2(X2,B2) | 5E | RX | Add Logical |
| ALR | R1,R2 | 1E | RR | Add Logical (Register) |
| AP | D1(L1,B1),D2(L2,B2) | FA | SS | Add Decimal |
| AR | R1,R2 | 1A | RR | Add (Register) |
| AU | R1,D2(X2,B2) | 7E | RX | Add Unnormalized, Short |
| AUR | R1,R2 | 3E | RR | Add Unnormalized, Short (Register) |
| AW | R1,D2(X2,B2) | 6E | RX | Add Unnormalized, Long |
| AWR | R1,R2 | 2E | RR | Add Unnormalized, Long (Register) |
| BAL | R1,D2(X2,B2) | 45 | RX | Branch and Link |
| BALR | R1,R2 | 05 | RR | Branch and Link Register |
| BC | M1,D2(X2,B2) | 47 | RX | Branch on Condition |
| BCR | M1,R2 | 07 | RR | Branch on Condition (Register) |
| BCRE | M1,R2 | 0C | RR | Branch on Condition to Return External |
| BCT | R1,D2(X2,B2) | 46 | RX | Branch on Count |
| BCTR | R1,R2 | 06 | RR | Branch on Count (Register) |
| BXH | R1,R3,D2(B2) | 86 | RS | Branch on Index High |
| BXLE | R1,R3,D2(B2) | 87 | RS | Branch on Index Low or Equal |
| C | R1,D2(X2,B2) | 59 | RX | Compare (Word) |
| CD | R1,D2(X2,B2) | 69 | RX | Compare, Long |
| CDR | R1,R2 | 29 | RR | Compare, Long (Register) |
| CE | R1,D2(X2,B2) | 79 | RX | Compare, Short |
| CER | R1,R2 | 39 | RR | Compare, Short (Register) |
| CH | R1,D2(X2,B2) | 49 | RX | Compare Half Word |
| CL | R1,D2(X2,B2) | 55 | RX | Compare Logical (Word) |
| CLC | D1(L,B1),D2(B2) | D5 | SS | Compare Logical (Character) |
| CLCM | C0 | 83 | SI | Clear CAM* |
| CLI | D1(B1),I2 | 95 | SI | Compare Logical Immediate |
| CLR | R1,R2 | 15 | RR | Compare Logical (Register) |
| CP | D1(L1,B1),D2(L2,B2) | F9 | SS | Compare Decimal |
| CR | R1,R2 | 19 | RR | Compare (Register) |
| CVB | R1,D2(X2,B2) | 4F | RX | Convert to Binary |
| CVD | R1,D2(X2,B2) | 4E | RX | Convert to Decimal |
| D | R1,D2(X2,B2) | 5D | RX | Divide (Word) |
| DD | R1,D2(X2,B2) | 6D | RX | Divide, Long |
| DDR | R1,R2 | 2D | RR | Divide, Long (Register) |
| DE | R1,D2(X2,B2) | 7D | RX | Divide, Short |
| DER | R1,R2 | 3D | RR | Divide, Short (Register) |
| DIG | D1(B1),I2 | 83 | SI | Diagnose |
| DP | D1(L1,B1),D2(L2,B2) | FD | SS | Divide Decimal |
| DR | R1,R2 | 1D | RR | Divide (Register) |
| ED | D1(L,B1),D2(B2) | DE | SS | Edit |
| EDMK | D1(L,B1),D2(B2) | DF | SS | Edit and Mark |
| EX | R1,D2(X2,B2) | 44 | RX | Execute |
| EXNT | D1(B1),C6 | 83 | SI | Execute Nontranslate* |
| HDR | R1,R2 | 24 | RR | Halve,Long (Register) |
| HER | R1,R2 | 34 | RR | Halve,Short (Register) |
| HIO | D1(B1) | 9E | SI | Halt Input/Output* |
| HPR | D1(B1),I2 | 99 | SI | Halt and Proceed* |
| IC | R1,D2(X2,B2) | 43 | RX | Insert Character |
| ISK | R1,R2 | 09 | RR | Insert Storage Key* |
| L | R1,D2(X2,B2) | 58 | RX | Load (Word) |
| LA | R1,D2(X2,B2) | 41 | RX | Load Address |
| LBTB | D1(B1),C2 | 83 | SI | Load Block Table Base* |
| LCDR | R1,R2 | 23 | RR | Load Complement, Long (Register) |
| LCER | R,1R2 | 33 | RR | Load Complement, Short (Register) |
| LCHR | D1(B1) | AD | SI | Load Channel Register |
| LCR | R1,R2 | 13 | RR | Load Complement |
| LCS | R1,R3,D2(B2) | B1 | RS | Load Control Storage* |
| LD | R1,D2(X2,B2) | 68 | RX | Load, Long |

*Privileged Operation



UP-8119

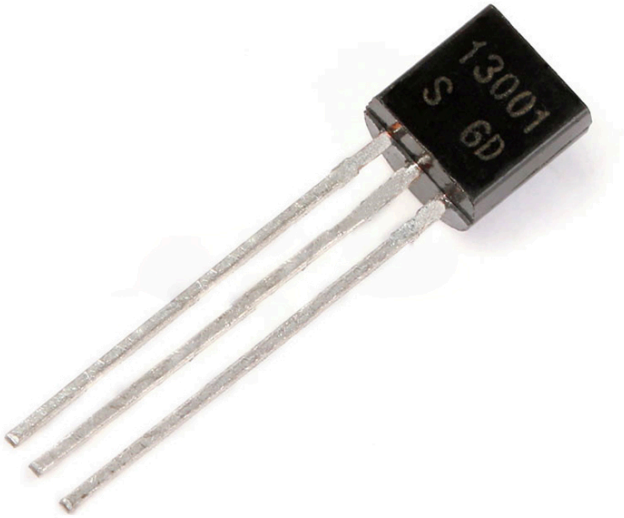
john von neumann (practical computer)

grace hopper (professional programmer and navy rear admiral)

The first computers did not have any software with them. Software is what tells the hardware how it should behave, and the first computers were not *programmable* computers, they couldn't be reprogrammed without being actually *physically rewired*. The software, then, the process of rewiring a computer, was the responsibility of the computer, the female secretaries and clerks which followed the orders of the male mathematicians and scientists to implement the algorithms they wanted to calculate. It was considered at the time that "proper" computer science was to be done as far away from the computer as possible, since it was "a girl's job".

This early form of **command and control** architecture, in which we tell the computer to execute some tasks, comes from another existing form of control, that of power relationships between gender. It is only when software became more interesting than hardware (i.e., when software could be run on more than a handful of computers) that programming gained in popularity and that the involvement of women dropped drastically.

## the decoupling



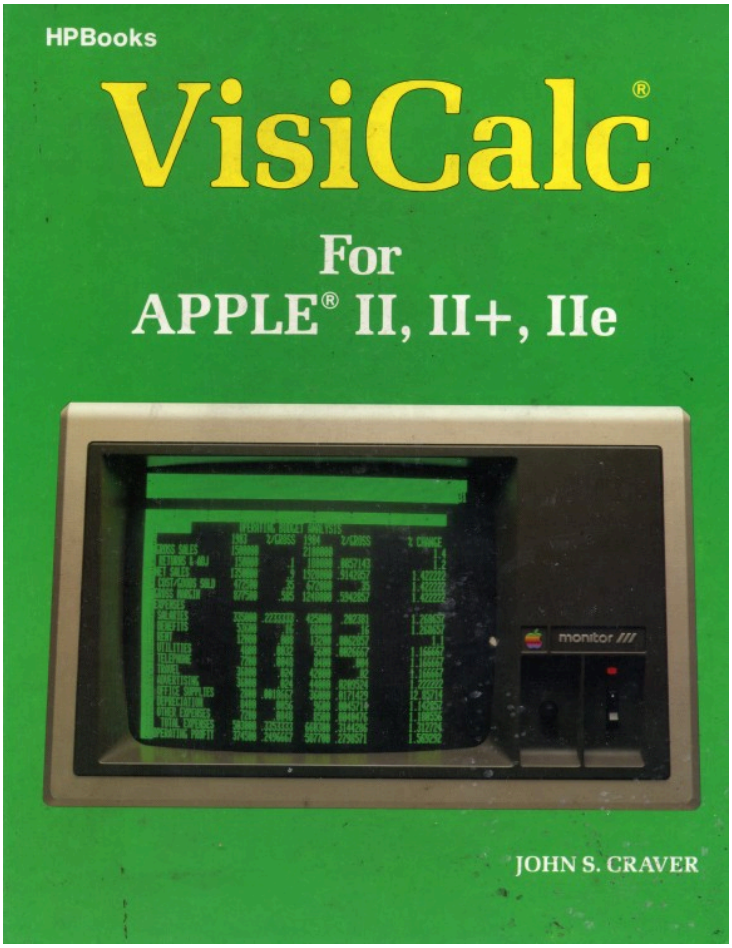

1947, new jersey, AT&T, transistor

always more

(tsmc - taiwan semiconductor manufacturing co, makes about 40% of the world's transistors)

1969, new jersey, AT&T, UNIX

1979, the Apple II




visicalc, 1979

(aka spreadsheets run the world)

1981, MS-DOS

1982, commodore 64

## programming languages

programming languages are the glue between **the absolutely concrete** and **the absolutely abstract**.

between a low-level and a high-level

from:

```
1001 1110 0000 1101 1010 0000
```

to:

```
if(guilt.likeliness() > 0.76) {
     alert(court.get())
}
```

1970s, **smalltalk** and objects

- 0 -> integers
- 0.1 -> floating point numbers
- 'a' -> character
- "word" -> strings (of characters)
- true / false -> boolean

```
claimant -> {
 name: string,
 age: integer,
 country_of_origin: string,
 last_known_location: [number, number],
 isValidPassport: boolean
}
```

The development of programming languages follows the attempts of humans to communicate clearly to the machine what they have in mind. Alan Turing, when he theorized his Universal Computer, stated that it could do anything that was computable. The total sum of what was computable wasn't just math (i.e. missile trajectories, radar detection, etc.), but also people.

In parallel to the development of mathematics for military purposes (e.g. cryptography), a new political paradigm appeared: that of counting populations to control them. An example of that is the role that IBM played in the US Census of 1890 (or rather, the founder of the company went on creating IBM once he invented a punch card system for the said Census), but also the role that IBM played in Nazi Germany, when the company provided the infrastructure to count Germans, including Jews, Gipsies and other undesired ethnic groups. Such developments still continue in 2019, in India and Myanmar, where marginalized populations are excluded through the sole process of counting.

These two dynamics were finalized during **the shift between functional programming towards object-oriented programming**. Essentially, this meant that one no longer needed to deal exclusively with building blocks such as `integers`, `floating point numbers`, `booleans` or `characters`, but could instead create their own building blocks. For an instance, a building block could be a `User` or `Client` or `Terrorist`. Object-Oriented Programming allowed us to start describing the world in computer terms, and treat it like a computable object.

## the language of new media

code is what makes computer-based media different from existing media.

- numerical
- modular
- automated
- variable
- transcoding

The two key works here are *Understanding Media*, by Marshall McLuhan, and *The Language of New Media*, by Lev Manovich. McLuhan who founded the field of media studies, and offered an understanding of **media as extensions of ourselves** and of **media as more important than messages**. Now that the attention of some scholars had been directed to the specificty of media as relevant objects of study, Manovich theorized the unique specificities of new media, which are:

- numerical representation *everything can be a number*
- modularity *a system can be broken into smaller parts and replaced in different places, physical integrity is no longer a problem*
- automation *a set of rules can act on their own*
- variability *new media is customizable*
- transcoding *through numerical representation, data can be coded in different ways: e.g. data visualization*

## computer expressivity

from the **problem domain** to the **computing domain**

aka *how do you turn something into a number*

are computers political? do they hold inherent values?

Our interactions with software are both mental, and physical. For now, we will just focus on the physical part.

Software, to us, is a something that reacts to, and triggers, buttons, switches, toggles, sliders, screens, LEDs, infra-red lights, etc. These are called *sensors* (when it's an input, like your keyboard) and *actuators* (when it's an output, like your screen). These are interfaces, and interfaces can be tricky things, because they both make visible and make invisible, because they do not want to overwhelm you with choices. By showing what can be done, they can create a potential "stage", an environment in which an individual becomes a user. Indeed, while PCs and Macs are essentially the same machine (both based on the same Von Neumann, and sometimes even the same Intel processor), their interfaces make us feel like they couldn't be more different from one another, and even go as far as to differentiate socially their users.

More importantly, interfaces (or at least good interfaces) disappear, become obvious and *natural*; if we don't think about the action we are doing because if feels *right*, how can we critically examine the social construction of that action?

## software and mind

*cybernetics* as means of control and communications in the animal and the machine (norbert wiener)

adaptation based on feedback

*postscript on the societies of control*, gilles deleuze

And so this is the question. If the physical world can be computed, to what extent can the psychological world be computed?

Variability reinforces customization and individuality. Transcoding condenses further the media landscape (more text, more images, more sounds, etc.). Modularity reinforces universality.

Cybernetics are based on **(1)** a universal intellectual system and **(2)** a way to quantify change and evolution within that system. The point is to measure and modulate communications in order to steer the system into a particular state. The systems we're talking about here are physics, biology, societies, etc.

This connection between biology and computer science prefigured neural networks that we see today, and makes the human mind exist within the realm of effective computability.

And yet, there is still the problem of embodiment: all of this needs to happen *within a medium* and that medium is always imperfect.

## politics

software code = data + algorithms

and data is political

technology and politics are never separated, as they change ourselves and our perception of the world

the computers themselves hold as much value as concrete does for the built environment. it's not so much concrete as urban and land planning that is political, just as it is the companies and states which decide which code to run.

## conclusion

the main question is whether code can represent, process and decide upon *everything* in our lives, or if it is limited to certain domains (and which domains?)

## recap

---

## software

the ability to turn everything (?) into numbers

the ability to make complex mechanical actions

the dynamic execution of meaning

## artificial intelligence

Artificial intelligence has a long history of rule-based systems, but has switched today to focus on machine learning, a black-box approach to inputs and outputs.

---

https://en.wikipedia.org/wiki/Perceptron
The Perceptron was the first "computational brain", which foreshadowed our use of AI today.
https://www.youtube.com/watch?v=-P28LKWTzrI
The Mythbusters demonstrating the difference between a CPU and a GPU. AI today switched from using CPUs to using GPUs.
https://en.wikipedia.org/wiki/ELIZA
Wikipedia page about ELIZA, the "first AI"
http://www.ilmarefilm.org/archive/weizenbaum_archiv_E.html
A documentary about Joseph Weizenbaum, the MIT professor that stood against the first AI hype
https://www.youtube.com/watch?v=_XdQugsDz8E
Can we live with sentient AI agents? Unsurprisingly, the japanese are very cool with robots.
https://www.instagram.com/p/ByKg-uKlP4C/
The manipulation of human faces (art project)
https://www.moralmachine.net/
moral machines being trained to make choices
https://plato.stanford.edu/entries/chinese-room/
The Chinese Room experiment: does following rules equate understanding?

## types of smartness

what kind of impact is chatGPT going to have on the world?

## the singularity

when AI can out-perform humans at any tasks

op-ed by GPT-3

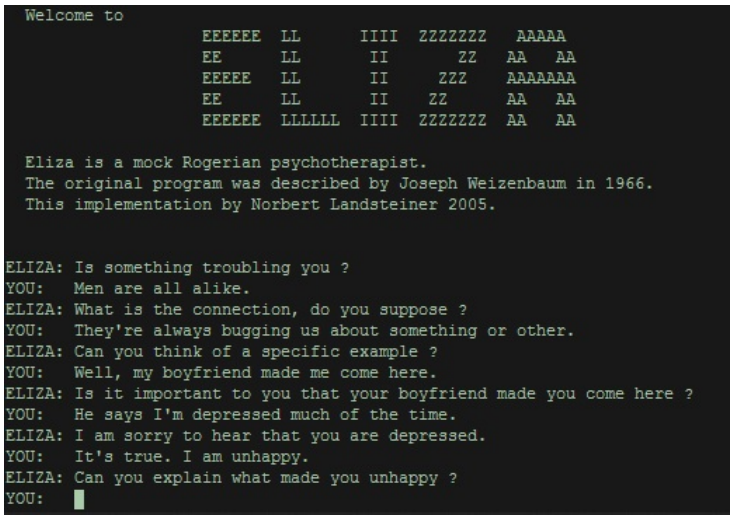
## the dream

artificial intelligence is coined in 1956 at the dartmouth workshop

Artificial intelligence is the attempt to recreate biological intelligence (let alone human intelligence) through computation. While this obviously asks the question of *what* is intelligence, most efforts have been directed towards developing responses to never-before-seen situations, as well as ambiguous situations.

Research in artificial intelligence started with Alan Turing and his Turing Test, according to which one shouldn't be able to determine whether or not one was conversing with a machine. In the late 1950s, Marvin Minsky and a team of other computer scientists founded the field of AI, focusing again on language, and general procedures to solve problems (i.e. rules of thumbs). The following years until 2011 is a series of ups and downs, based on the ebb-and-flow of grant money. Funding would come following hopes and hypes, and wade out after findings proved limited.

## the history of AI




developed by joseph weizenbaum in the 1960s.

- I feel X.
- Why do you feel X?
- Because I think that Z.
- Is it really Z?
- Yes.
- Tell me more.

---

software as psychotherapy?

AI summers and winters

- 1956 -> 1970 (ELIZA, SHRDLU)
- 1980 -> 2000 (Deep Blue, Aladdin)
- 2010 -> now (Machine learning, deep learning, neural networks)

the fear of the singularity

ELIZA was one of those early programs which led us to believe that machines could become intelligent. The first chatbot, ELIZA was based on the personality of a Rogerian therapist (the kind of therapist which spits questions back at you... "i'm unhappy" "why are you unhappy?" "because i don't have any money" "why don't you have any money", etc.).

This led psychologists across the country to contact its creator, Joseph Weizenbaum, in order to use ELIZA in their hospitals. However, Weizenbaum refused, stating that this was but a parlor trick, a very simple system, a verbal mirror of sorts. The lesson here is that, no matter how rough or rudimentary the underlying algorithm might, our desire to believe in some sort of intelligence that we can interact with is a large component of how we construct artificial intelligence. None of the "smart" things that we have are actually smart, but for psychological and marketing reasons, we want to believe that these exist.

## machine learning


if you give a machine enough examples of how to do one thing, it will figure out how to do it

[you can make mathematical operations on human faces]

The recent shift in machine learning has happened due to two changes: first, a change in how we approach the problem and, second, a change in how we can calculate the problem.

The approach of the problem was no longer to pre-establish a system of rules which could determine any outcome, but rather to present the outcome to the system and let it determine its own rules. Such an approach, like the Perceptron, invented in 1958, mimicked the human brain by connecting individual neurons to each other, and using the distributed power of these neurons for a single, binary output (i.e. determine whether a statement is true or not).

The second change is based both on hardware and on people. Hardware has been following Moore's law, which states that computing power doubles every other year for the same price, so we keep getting increasingly fast hardware to do the needful. People, on the other side, have been increasingly connected to computers of all sorts and, by doing so, have created more and more data to work with. Gigantic datasets, such as MNIST (handwritten numbers for check detection) or IMAGENET (for labelled images) have been handpieced by humans and can now be used to train **neural nets**.

Neural nets are a version of AI which only needs to see examples of what it needs to figure out, in order to figure it out. If you show it enough pictures of a cat, it will know what a cat is. If you show it enough criminal records, it will know when someone will be likely to commit a crime again.

## issues in AI

what are some problems in the deployment of AI?

- opacity
- biais

To what extent do we want our rationality, judgment and morals to be handled by automated systems? What is it that's important that we might want to preserve? What is it that we might *not* want to preserve?

## big data

---

https://www.tensorflow.org/datasets/catalog/celeb_a_hq
The dataset on which human faces are trained. As celebrities, they are high-quality but also overwhelmingly white.
http://www.oracle.com/us/solutions/cloud/data-directory-2810741.pdf
The Oracle data marketplace
https://firstmonday.org/ojs/index.php/fm/article/view/4901/4097
The use of big data by the 2012 Obama campaign
https://weaponsofmathdestructionbook.com/
Weapons of Math Destruction is an inquiry as to how mathematical models have very real, unjust economic an social consequences.

## what is data

data vs. information

information is cultural - the moral machine

Data is any pattern of signal which can be meaningfully parsed (i.e. understood by a computer). Data only becomes information through interpretation.

While data means given, a more appropriate term would be capta, taken. In order for data to be useful, it needs to be accessed, and therefore catalogued. The structure of data, i.e. in relational databases becomes a political act on its own.

## big data

big data is just a lot of data, with the assumption that you can get a qualitative step from quantitative mass.

*when you talk to clients, it's artificial intelligence, when you talk to engineers, it's linear regression*

Having incredibly large amounts of data can feed machine learning algorithms, and because more data results in more accurate results and could potentially prove that the whole of our existence is subject to the rules of computation.

Big data is quantification of all of human interactions on a computer system. With those data points, it becomes possible to build up a pool of raw material upon which to develop predictions.

## the biases of big data

machine learning recreates the state of society as it is given to it.

man is to computer programmer as woman is to homemaker?

the way you connect the dots is also a decision

-> structured databases

-> relational databases

## the fallacy of big data

you can always have more

*computers exist to solve problems we wouldn't have without computers*

## responses

---

## responses

EU AI Act proposes a risk-based approach for CE labelling

- **unacceptable risk** is banned (e.g. social scoring)
- **high-risk** is legally framed (e.g. job filtering, grade assessment)
- **limited risk** (e.g. self-driving cars)
- **legal sandboxes** to help innovation

high risk classification

## refraining

on the dangers of stochastic parrots: can large language models be too big?

## group checkin

---

## requirements

- data collection
- web write-up
- conclusion addressing your initial hypothesis

## programming basics

---

## concepts in programming

- variables -> data
- functions -> algorithms
- if statements -> intelligence

```
let x = 4
let name = "max"
let isVaccinated = true

checkAsylumClaim()

if(claimant.age < 18) { ..... } else { ''''''''' }
```

## organizing data

data structures:

```
person = {}
person['first'] = "justyna"
person['last'] = "poplawska"
```

---

algorithms:

```
if person['first'] in french_names:
    print("welcome!")
else:
    print("an error occurred, please retry later.")
```

## practice

we are going to write a short program which automates border control.

1. input data
2. process data
3. output result

- asylum_claim.html
- asylum_claim.css
- asylum_claim.js

## web design

---

## extending our show/hide feature

we can give arguments to functions:

```
function toggleElement(toggleID){
  var el = document.getElementById(toggleID);
}

toggleElement("intro-text");
```

## intra-page links

use the `id="my-anchor"` property to jump across points within a page

## adding a sidebar menu

make two elements side-by-side with `display: flex`

make a function to scroll smoothly

## conclusion

---

## software today

algorithms are only as dangerous as the social context they support

## homework

15min presentations - 10 + 5q&a