Computer Science
Introduction to ~~Game Development~~ (by Harvard University) → accidently chose the wrong course

CS50

- by David Malin and ~~Colton Ogden~~
- computer science ≠ programming per se → basic concepts of programming
  - ↳ Problem Solving
- ! Where you end up !

Start: knowing nothing → Super Mario Game
End: Own Online Site, Databanks, Login, etc.
building pyramid in Mario → building Web Application +

→ Final Project

Computer Science

What is Computer Science? | course/week 1
- → solving problems
- ▫ what does it mean to solve problems?

input (problem you wanna solve) → [blackbox] → output (solution for that problem)
(code)

↳ how to present?
- → input + output → we have to speak a common language

↳ how to represent?

representation methods of information
- ☒ people in class (count) → hand (finger = number) → unary notation (1 finger = 1 Digit)
- ☐ decimal digits (10 → 0-9) → computers don't understand
- ☐ binary system (bi = 2)
  - ▫ 0 and 1, instead of 0-9 ← computers don't understand → they are simple
  - ↳ how to represent more? (3)

bits
- → 0 and 1 = bits
- → why invent new digits? → plug pc in

input of pc
- → only physical input = electricity (→ electrons flowing)
- ▫ represent information?
  - ☐ light on / light off → store electricity and say 1
    - 1 / 0

the binary system
- → how high can you count with 3 bulbs?

how to display numbers

0 0 0 → start counting at 0
1 1 1 → not how many, the pattern
0 0 1
1 0 1
...
→ eight total patterns
→ from 0 - 7

D	2^2 (4)	2^1 (2)	2^0 (1)
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
...			

2 digits (0 + 1)

→ computer's "lightbulbs" = transistors
- ▫ millions that can be in different patterns

1 | 2 | 3 ≠ 123 → 1 ($1 \cdot 100$, $2 \cdot 10$, $3 \cdot 1$ → 100 · 1, 100, 10)

10^2 10^1 10^0 → 100, 10, 1 — 10 digits (0-9) decimal

2^2 2^1 2^0 → 4, 2, 1 — 7 = 1 | 1 | 1

32 16 8 4 2 1 (2^5 2^4 2^3 2^2 2^1 2^0)

binary: 1 1 0 0 1 0 → 32 + 16 + 0 + 0 + 2 + 0 → 50

how to display letters
- ▫ computers original task was to be a calculator
- → how to display letters? (we only have on/off switches)
- → assign letters to numbers (A = 0, B = 1) → years ago they decided A = 65 → A = 01000001
- → computer "looks" inside of hardware and sees how many switches are on ↳ needs context

ASCII	▫ system that maps 65 = A, 66 = B → ASCII ▫ American Standard Code for Information Interchange ▫ mapping for everything: punctuation, lowercase letters, ...
display letters	▫ text message: 73, 72, 33 = HI! H, I, ! → we need map, table; computer = knows, programmed 01001000 (8 bits), 01001001 (8 bits), 00100001 (8 bits) → 1 bit = 0 or 1 ↓ 1 byte, 1 byte, 1 byte → 1 byte = 8 bits = 8 × 0 or 1
bits and bytes	↗ letters of the alphabet ▫ how many symbols can you represent with 8 bits? → 256 ▫ 2×2×2×2×2 (possibilities) ↳ what is missing from ASCII? ↗ accented characters ↗ asian / arabic languages ▫ why are 256 possibilities not sufficient? ▫ humans use way more to communicate ▫ emojis are characters, in a different alphabet ↳ pictoral
Unicode	→ ASCII → Unicode ▫ supports all human languages ▫ emoji with tears of joy = 128514 → Does not use 8 bits, but 16 or 24 or 32
display colors	electricity → representing numbers → represent letters + emojis → ? ▫ how to display colors? → we only have bits, assign those to colors (Photoshop) ↑ RGB → can create every color in the rainbow 73 72 33 → HI! ↓ amount of red, amount of green, amount of blue (yellow dot) → R: 8 bits ↓ 256; G: 8 bits ↓ 256; B: 8 bits ↓ 256 → range from 0 to 255 each → 24 bits in total ↓ color ▫ displaying e.g. an emoji: ▫ control colors of dots on the screen (→ pixels) ↳ tiny little squares that represent color
pixels	resolution = how many pixels/dots there are horizontally (–) and vertically (|) image — one pixel ▫ 24 bits / 3 bytes image = grid of pixels
display music/videos	video: changing pixels really fast, many pictures displayed after another → flipbook
formats	music: number to represent each possible note, another number for duration formats (png, gif, word) tell the computer how to store patterns of zeros and ones in a file
input/output	represent with 0s and 1s ↘ INPUT ↳ [how to take inputs and convert them into actual solutions?] → OUTPUT ↙ represent with 0s and 1s

Algorithms

Algorithms

- step-by-step instructions for solving problems → we can follow someone elses instructions
- not always happening inside of computers
  - following a recipe
- no room for ambiguity in computers

computer and human

Computers Algorithms:
- have to be correct (get the right outputs)
- need to be precise → humans can read between the lines → "I get what you mean" → computers take you literally

→ programming = translating an algorithm into a language the computer understands

bugs and good/bad algorithms

Phonebook → Input → Algorithm → Output → Phone number
David → Input

a) Turning every page till you find the name → correct, but very slow → 1000 → 999 → 998 → 997 ...

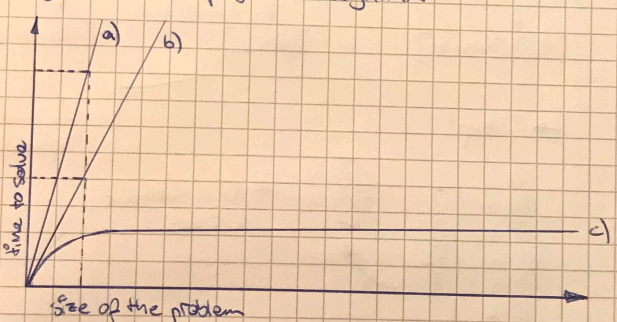
Take more pages at once → you could skip the name → 1000 → 998 → 996 → ... still slow, but works

look at the pages you see and decide if the name lays right or left → fix that "Bug" b)

c) open in the middle, decide if left or right, throw other part away, go to the middle, decide if left or right → 1000 → 500 → 250 → ... → way faster than 2 or 1 page at once

bug = mistake in a program/an algorithm

efficiency/"it works"



pseudocode = algorithm implemented in English
- important: correct and precis (!)

basic terms in form of pseudocode

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 ← important, often forgotten

Go back to line 3 → loop/cycle = programming construct or principle of an algorithm that gets you to do smt again and again

Pick up → Functions = Statement that gets the computer to do something

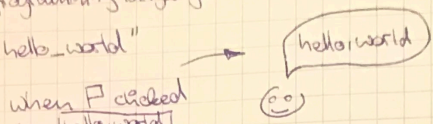
Else if → conditions/branches = which fork in the road to take → "Scratch"

If … → Boolean expression = question whose answer is y/n, true/false, 0/1

programming with scratch

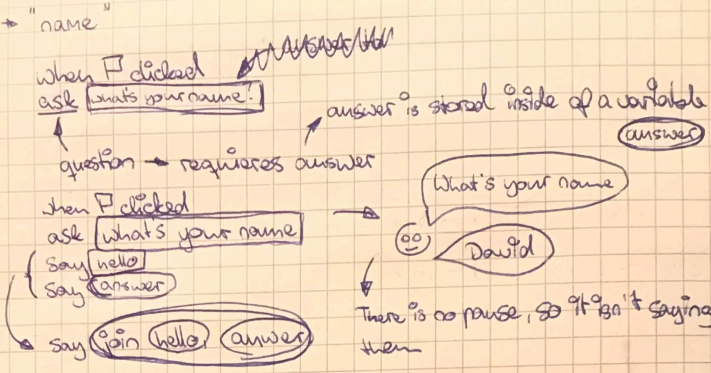
Scratch
a programming language with blocks

→ "hello_world"

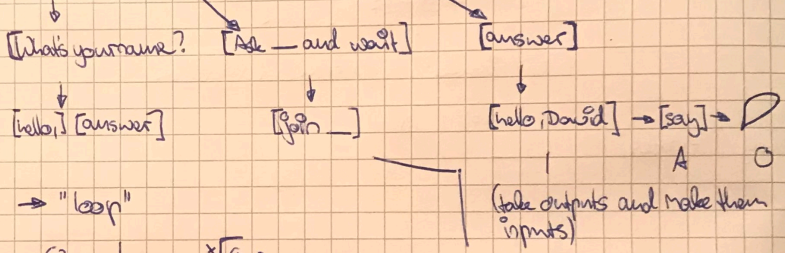


output and variables

→ "name"

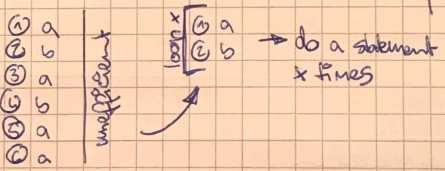


Scratch as an Algorithm



loop

→ "loop"



conditions

→ "condition"

boolean expression

```
if [mouse down] then
    pen down
else
    pen up
```

condition

how to program

- Start programs simple, then add more complex things
- abstraction = way of problem solving
  - take a complicated idea → defining a function / puzzlepiece for it
    - → get a cat to meow (not worrying about complementation details)