Harvard CS50

Lecture 0: Computational Thinking & Scratch

representation (tally marks —> hand —> order of fingers)
expressing patterns (32) on a single hand
binary language - 1's place, 2's place, 4's place (powers of 2)
123 = 100*1 + 10*2 + 1*3 - 100's place, 10's, 1's (powers of 10)
2 = 0 1 0
3 = 0 1 1 = 2 + 1
1 1 1 = 7 = 4 + 2 + 1
decide on pattern of bits to represent the letter A (65), B(66),...
transistors (on/off)
ACII (asky) ^^
leading 0's don't matter
abstraction = low level implementation details (not useful) -> simplifying them to have more useful conversation
Unicode (32 bits to represent characters - thousands or millions of characters)
UTF-8
emojis represented in pattern of bits
RGB with ranges 0-255
^^ all this to represent inputs to outputs

Algorithms translate inputs to outputs
Phone book - divide and conquer
Pseudocode - english-like syntax so computer/robot understands
functions & conditions/branches, Boolean expressions
threads - do more than one thing at a time
events - listen for things happening

Scratch (from MIT's media lab)
libraries are abstractions

Byte = 8 bits

Lecture 1: C Programming Language

PB&J instructions
Scratch —> C
Source code —> Compiler —> Machine code
clang hello.c

—> “C language”

#s stored using 32 bits = 4 billion
Integer overflow: If you double numbers big enough, ran out of bits when carrying the 1 to a 33 bit value
use the left most bit to determine positive or negative
Boeing 787 - programming error loses electrical power. Software overflows after 248 days of continuous power = tracking hundredths of a second
Can also underflow - Civilization game