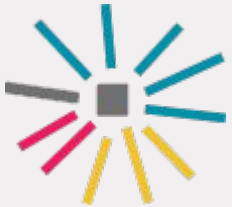
PROGRAMMING LANGUAGES

PALIMPSEST

PROGRAMMING LANGUAGES

- Definition
- Purpose
- High Level Languages
- Low Level Languages
- EXERCISE

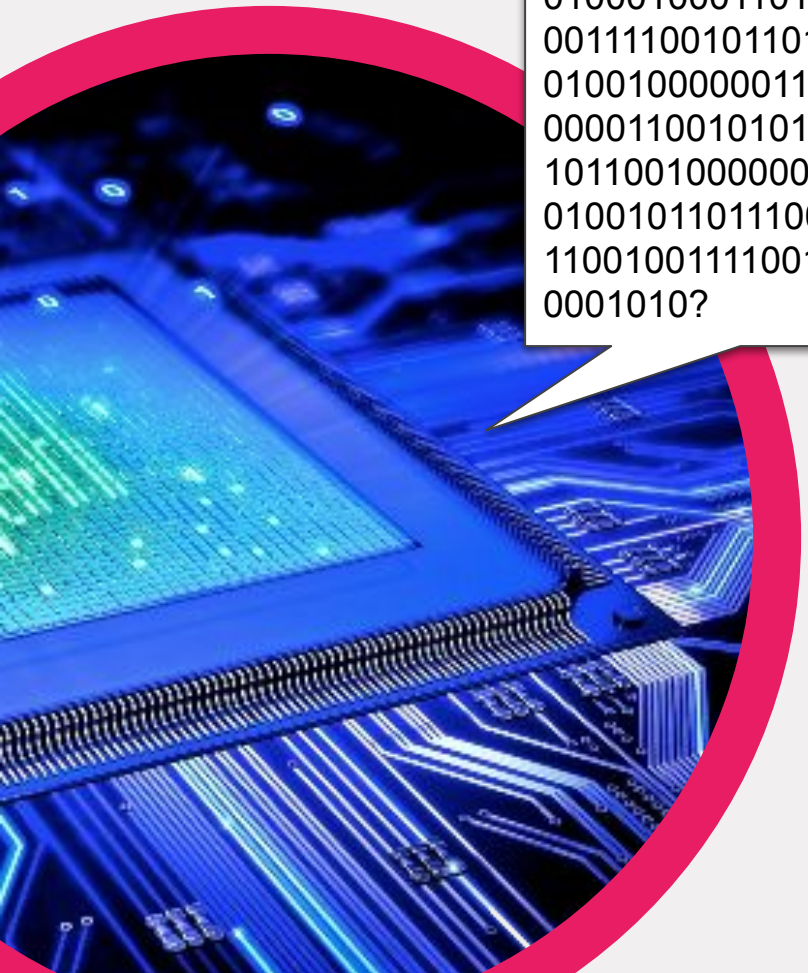
PROGRAMMING LANGUAGES



What are they?

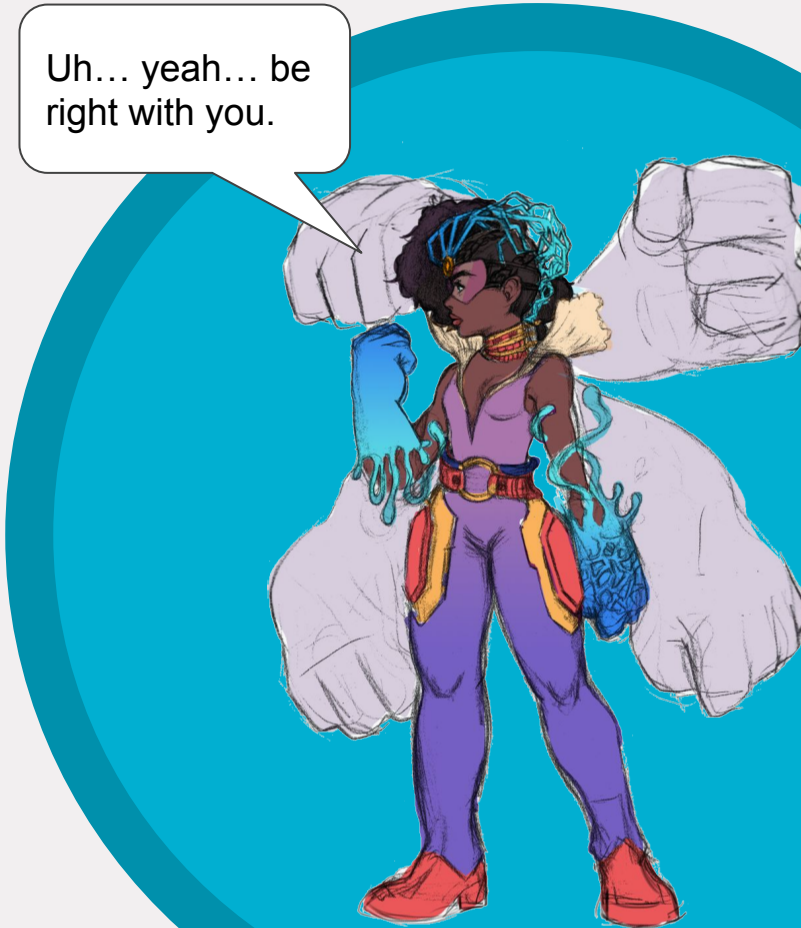
"The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform...

But it is likely to exert an indirect and reciprocal influence on science itself."



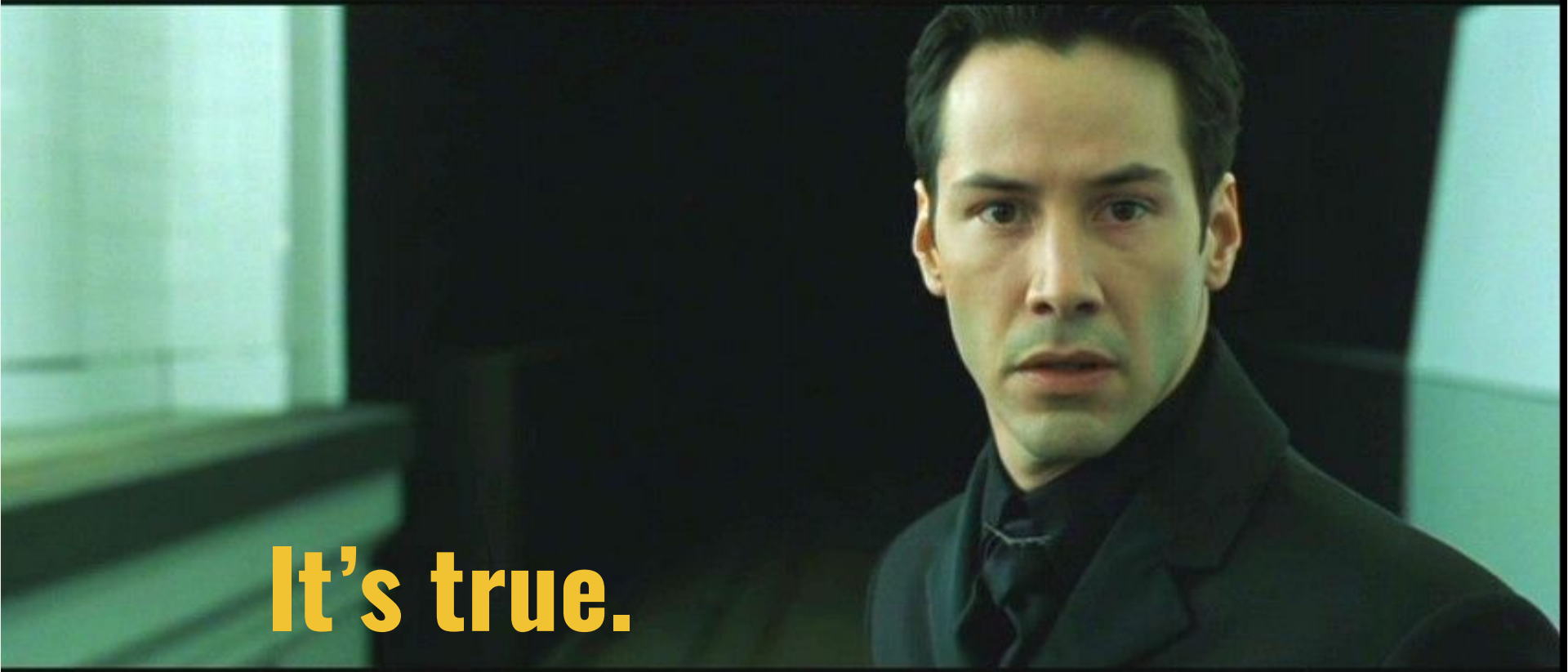
01000100011011110010000
00111100101101111011101
01001000000111001101110
00001100101011000010110
10110010000001100010011
01001011011100110000101
11001001111001000011010
0001010?

Uh… yeah… be right with you.

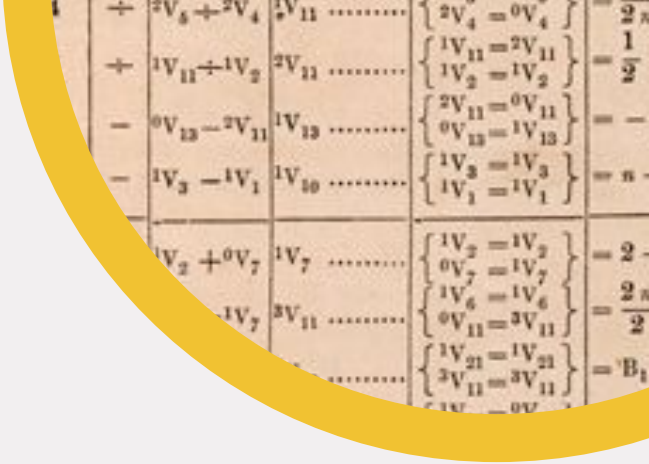


MOST PEOPLE DON'T SPEAK


Don’t look so shocked, Neo.


It’s true.

BUT WE *CAN* GET THERE



...it's a process...

...with many constraints...

Constraints... Let's Break It Down

- You write programs in a High Level Language
- the Compiler translates that
- into the Low Level, Assembly Language
- the Assembler translates that
- into Machine Language (binary!)
- the Control Unit interprets that
- for the Microarchitecture
- where the Microsequencer interprets the binary
- for the Logic-Design at the Device Level
- made up of Semiconductors / Silicon Transistors

...and CONSTRAINED by the properties of Atoms, Electrons, and Quantum Dynamics!

Compiler

Assembler

Control Unit

Microsequencer

Frosting

More frosting

High Level Languages

- C/C++/C#, Java, Fortran, Lisp, etc.
- Used by application programmers and systems programmers
- Can we build machines executing HLL right away?
- Compiler’s job is not only translating

Low Level Languages

Assembly

- More primitive instructions than HLL
- English version of the machine language + some more
- User mode and kernel mode
- Can we go from this level to HLL?

ISA

Instruction Set Architecture

A very important abstraction

- interface between low-level software and hardware
- **advantage:** different implementations of the same architecture
- **disadvantage:** sometimes prevents using new innovations

ISA

Instruction Set Architecture

Modern instruction set architectures:

- X86_64
- IA-32
- PowerPC
- MIPS
- SPARC
- ARM
- and more...

Instructions

- Language of the Machine
- Platform-specific
- A limited set of machine language commands "understood" by hardware
    - ADD, LOAD, STORE, RET, etc.

- We'll talk MIPS instruction set architecture and x86 instruction set architecture

Microarchitecture

- Resources and techniques used to implement the ISA
  - Pentium IV implements the x86 ISA
  - Motorola G4 implements the Power PC ISA

- Register files, ALU, Fetch unit, etc.
- Realize intended cost/performance goals
- Interpretation done by the control unit

Logic-Design

- Gates
- Multiplexers, decoders, PLA, etc.
- Synchronous (i.e. clocked) : the most widely used
- Asynchronous

Device

- Transistors and wires
- Implement the digital logic gates
- Lower level:
  - Solid state physics
  - Machine looks more analog than digital at that level!

Device

- Transistors and wires
- Implement the digital logic gates
- Lower level:
  - Solid state physics
  - Machine looks more analog than digital at that level!

We will use C#

HOW TO BEGIN?

Before you write one line, type one character...


WHAT IS MY GOAL?

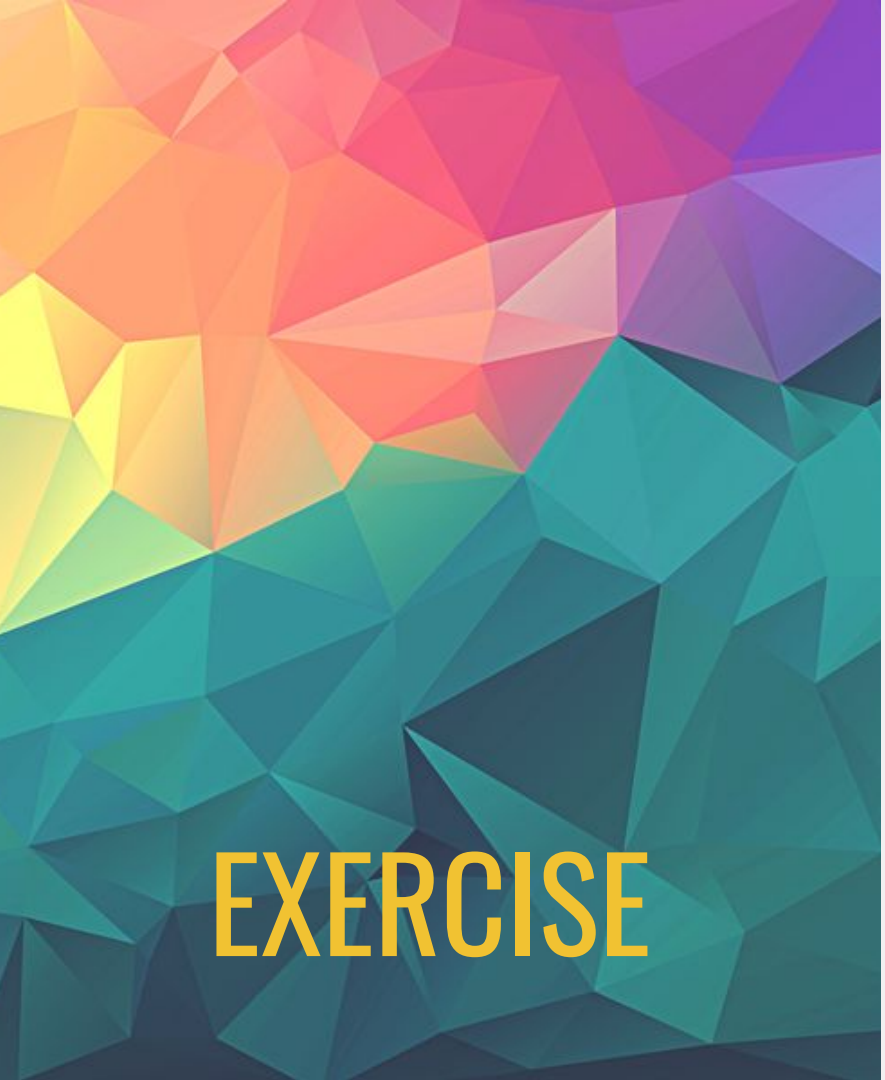
THEN

What information do I have?

What information do I need?

Starting Point

Goal

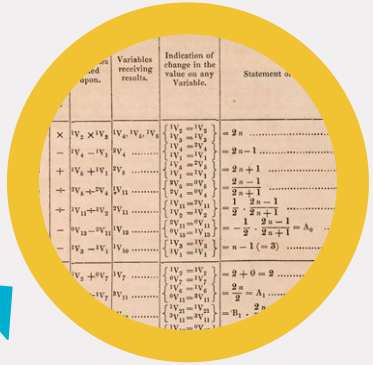


Pseudocode

Write out the steps

Program

Compiles Code into Binary



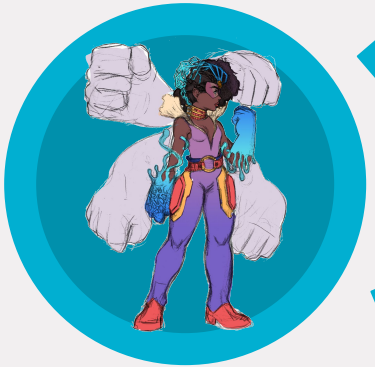
WRITES

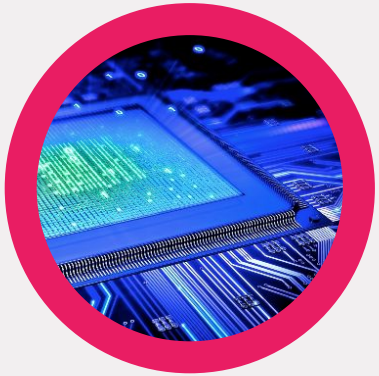
TRANSLATES

DISPLAYS

Programmer

Writes Code





Processor

Processes and displays results

DISPLAYS


Here! Enjoy.
0100010001100101011011000110100100101100011011010010111011110111010100100110011001000010010000001010100001101000011000010110111001101011001000001111001011011110101010010000101
(Delicious! Thank you!)

NEXT UP

Lab