

# PROGRAMMING LANGUAGES

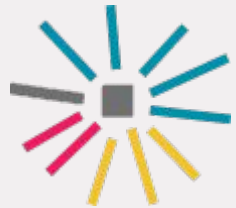
PALIMPSEST

# PROGRAMMING LANGUAGES

---

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

# PROGRAMMING LANGUAGES



What are they?

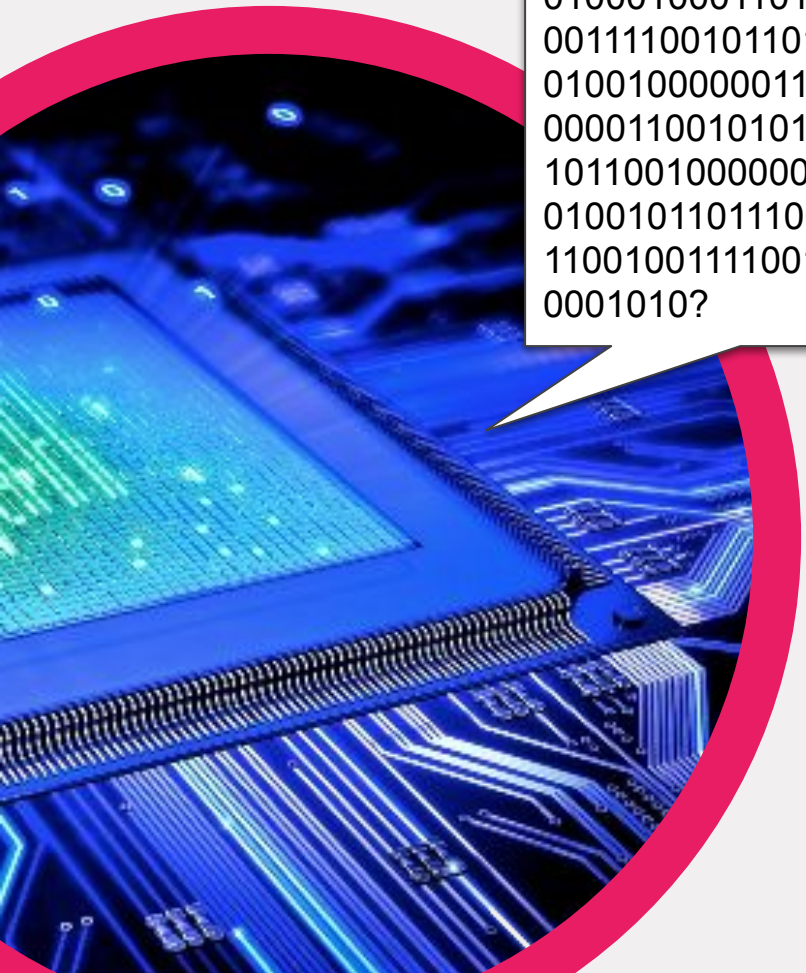


“The Analytical Engine has no pretensions whatever to originate anything. **It can do whatever we know how to order it to perform...**

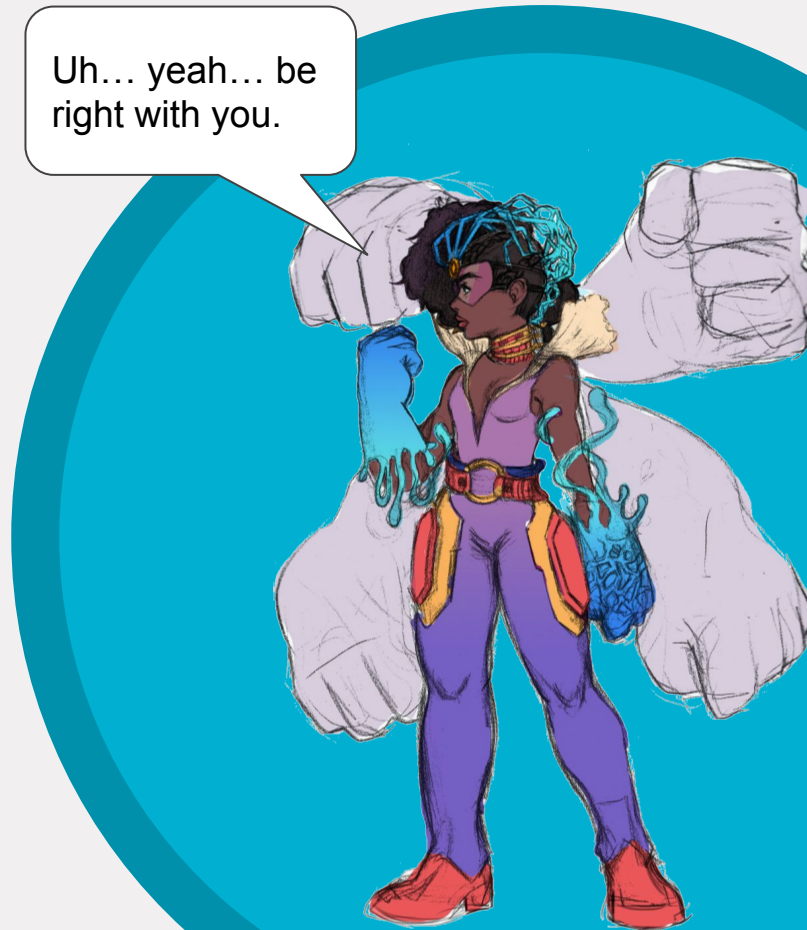
But it is likely to exert an indirect and reciprocal influence on science itself.”



**Remember...**



```
01000100011011110010000
00111100101101111011101
01001000000111001101110
00001100101011000010110
10110010000001100010011
01001011011100110000101
11001001111001000011010
0001010?
```



Uh... yeah... be  
right with you.

[illegible]

A close-up shot of Keanu Reeves as Neo in the movie The Matrix. He is wearing a dark suit and has a serious, slightly shocked expression on his face. The background is dark and out of focus, with some light coming from a window on the left.

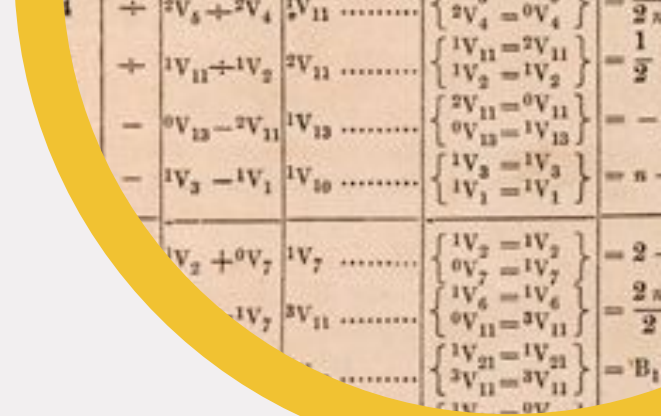
**Don't look so shocked, Neo.**

A close-up shot of Keanu Reeves, looking directly at the camera with a serious, intense expression. He is wearing a dark, high-collared coat or jacket. The background is dark and out of focus, with some light-colored architectural elements visible on the left and right sides.

**It's true.**



**BUT WE  
*CAN* GET  
THERE**



$2V_5 + 2V_4$	$1V_{11}$	$\begin{cases} 2V_4 = 0V_4 \\ 1V_{11} = 2V_{11} \\ 1V_2 = 1V_2 \end{cases}$	$= \frac{1}{2}$
$1V_{11} + 1V_2$	$2V_{11}$	$\begin{cases} 2V_{11} = 0V_{11} \\ 0V_{13} = 1V_{13} \end{cases}$	$= -$
$0V_{13} - 2V_{11}$	$1V_{13}$	$\begin{cases} 1V_3 = 1V_3 \\ 1V_1 = 1V_1 \end{cases}$	$= n$
$1V_3 - 1V_1$	$1V_{10}$	$\begin{cases} 1V_2 = 1V_2 \\ 0V_7 = 1V_7 \\ 1V_6 = 1V_6 \end{cases}$	$= 2$
$1V_2 + 0V_7$	$1V_7$	$\begin{cases} 0V_{11} = 3V_{11} \\ 1V_{21} = 1V_{21} \\ 3V_{11} = 3V_{11} \end{cases}$	$= \frac{2}{2}$
$1V_7$	$3V_{11}$	$\begin{cases} 1V_{21} = 1V_{21} \\ 3V_{11} = 3V_{11} \\ 1V_{11} = 0V_{11} \end{cases}$	$= B_i$



**...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 Language

Low Level Language

Binary

Microarchitecture

Logic-Design at Device

Semiconductors /  
Transistors

Atoms / Electrons,  
Quantum Dynamics





# 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...**

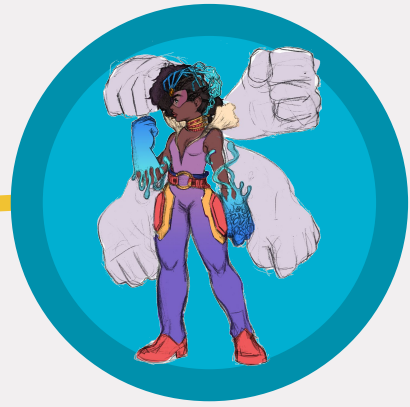


WHAT IS MY GOAL?

# THEN

What information do I have?

What information do I need?



Starting Point

Goal

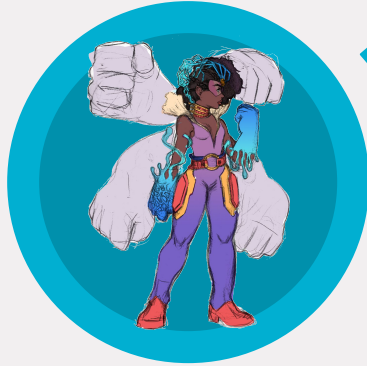


# Pseudocode

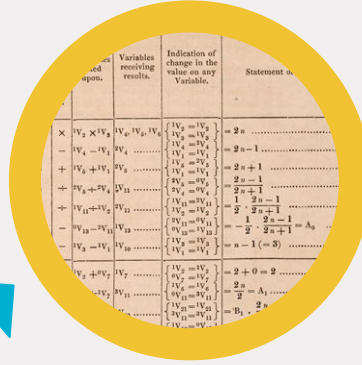
Write out the steps

**EXERCISE**

**Programmer**  
Writes Code

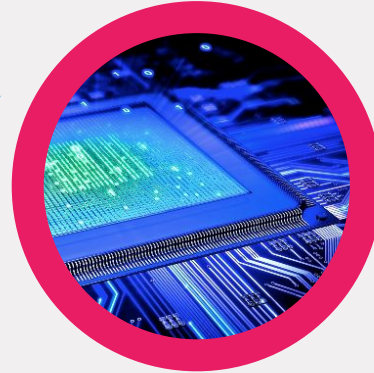


WRITES



**Program**  
Compiles Code  
into Binary

TRANSLATES



**Processor**  
Processes and  
displays results

DISPLAYS

DISPLAYS



Here! Enjoy.



```
010001000110010101101
100011010010110001101
101001011011110111010
101110011001000010010
000001010100011010000
110000101101110011010
110010000001111001011
011110111010100100001
```

(Delicious! Thank you!)



NEXT UP

Lab

