

Introduction to the Course - Part 2

David H Smith IV

University of Illinois Urbana-Champaign

Wednesday, Aug 18 2021

What is Computer Science?

- CS is concerned with understanding:
 - Historically concerned with investigating what is computable.
 - How to compute it in one or more of the following in mind:
 - Speed
 - Reliability
 - Security
 - Resource cost

What is Programming?

- Programming \neq Computer Science
 - Rather, programming is a subset of Computer Science
- “Computer Science is no more about computers than astronomy is about telescopes”
 - Edsger W. Dijkstra
- **High Level Definition:** A series of instructions that a computer carries out.



The Journey to a General Purpose Computer

Early “Computers”



Figure: Fire Control System

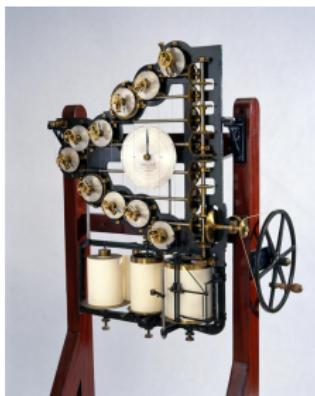


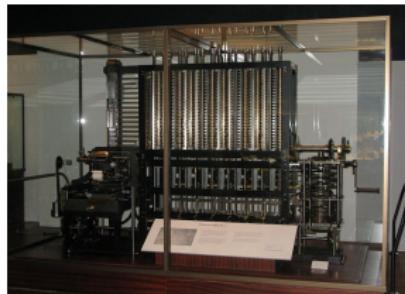
Figure: Tide Predicting Machine



Figure: Antikythera Mechanism

The Difference & Analytical Engines

- ① **Charles Babbage:** Created the Difference Engine and created plans for the Analytical Engine.
- ② **Ada Lovelace:** The first computer programmer who worked with Babbage on the Analytical Engine.
- ③ **Difference Engine:** A large mechanical calculator capable of performing large computations.
- ④ **Analytical Engine:** The first model of a general purpose mechanical computer.



Turing Machines

Turing Machines - An abstract, mathematical model of a machine that moves up and down a strip of paper, one step at a time, and performs operations based on a set of rules.

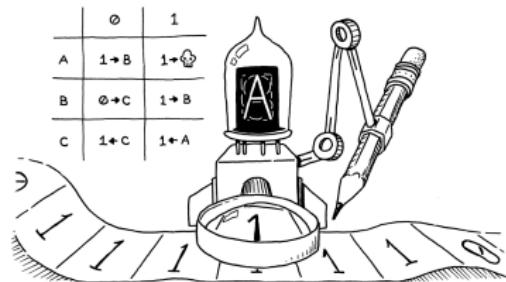


Figure: Artist's Representation of a Turing Machine

Von Neumann Architecture

- ① **Input Devices:** Something with buttons and knobs.
- ② **Central Processing Unit:**
 - ① **Control Unit:** Manages everything.
 - ② **Arithmetic/Logic Unit:** Does math and logical comparisons.
- ③ **Memory Unit:** Supplies info to CPU (i.e., Random Access Memory).
- ④ **External Storage Unit (Not Pictured):** We often need larger storage for data that isn't needed immediately (e.g., Hard Drive).
- ⑤ **Output Devices:** Flashing lights, monitor, etc.

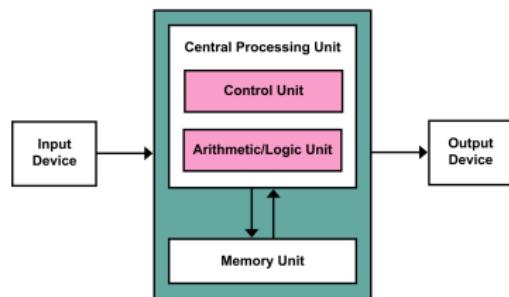


Figure: Von Neumann Architecture

How Programs are Constructed

Computers Run Low-Level Instructions

- Computers don't "Play a video"
- They...
 - ① Move some numbers into memory.
 - ② Do some math, or a comparison, or both
 - ③ Make a decision based on the results
 - ④ Rise a and repeat a few million time before something useful happens
- Programming at this level is:
 - Tedious and error prone.
 - Needs **A LOT** of code to do anything useful.
 - Isn't portable. Each processor is it's own machine and will require a different set of instructions that works with it's parts.

Enter Python (And Other High-Level Languages)

- They are:
 - **Productive** → A few lines do a lot and it's easy to debug.
 - **Safer** → Less likely to write code that is insecure or damaging.
 - **Portable** → Works on all systems that support the Python interpreter.
- Used for everything from machine learning to general scripting.
- We use Python 3.x (not Python 2.x)

How programs are Constructed

- **Algorithms** → A step-by-step process for achieving a result
 - Often Written in pseudo-code
- **Programming** → Express the commands in a form the computer understands.
- **Testing** → Designing inputs that test specific behaviours of the code.
- **Debugging** → Finding errors in the code based on the results of your tests and fixing them.

A Search Algorithm

- **Input:** (1) An array of numbers and (2) number for which we're searching.
- **Output:** Whether or not the number exists in the list of numbers.
- **Algorithm:**
 - ① Look at first item.
 - ② Check if it's the item we're looking for.
 - ③ If it's the item we're looking for stop otherwise go to next item.
 - ④ Repeat steps 2 & 3 if there are still items in the list.

Searching Number: 4

Input List:

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



Is 1 equal to 4? No, so we check the next one.



Is 2 equal to 4? No, so we check the next one.



Is 3 equal to 4? No, so we check the next one.



Is 4 equal to 4? Yes, so we stop searching.

Algorithm:

- ① Look at first item.
- ② Check if it's the item we're looking for.
- ③ If it's the item we're looking for stop otherwise go to next item.
- ④ Repeat steps 2 & 3 if there are still items in the list.

Code:

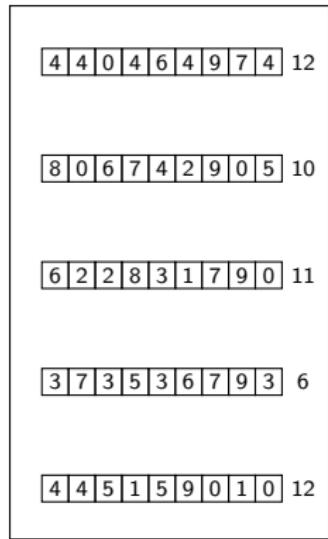
```
def search(list, searchitem):  
    for item in items:  
        if item == searchitem:  
            return True  
    return False
```

How Languages are Constructed

- **Syntax** → The rules specifying the structure and symbols that are allowable within the programming language.
- **Semantics** → Rules defining the behaviour of symbols or combinations of symbols.

Testing and Debugging

Testing



Debugging

- (1) Observe output
- (2) Determine errors
- (3) Propose fixes

