Chapter 0
The Study of Computer Science

What is Computer Science?

Definition
- Computer science is a discipline that involves the understanding and design of computers and computational processes.

A well-educated computer scientist should be able to...
- apply the fundamental concepts and techniques of
  - computation,
  - algorithms, and
  - computer design
  - to a specific problem

...including
- detailing of specifications,
- analysis of the problem,
- provide a design that
  - functions as desired,
  - has satisfactory performance,
  - is reliable and maintainable,
  - and meets desired cost criteria.

Aspects of Computer Science
- Theory of computation - limits of computer
- Computational Efficiency - how hard is the problem to solve
- Algorithms and Data Structures - org. of data
- Parallel Processing - multiple computers
- Software Engineering - creating programs
- Others
Our goals

- Our goals are not to just write copious amounts of code, our goals are to:
- increase our problem solving skills
- design good solutions to problems
- test somehow how well they are indeed solutions to the problem
- provide the solution as a readable document

Why is this hard?

- I cannot precisely explain why it is hard, only that it is indeed hard.
- Typically quote is "Never have I worked so hard and gotten so low a grade"

An analogy

Let us say that you have signed up to study French poetry (how about Marot) in the original
You have two problems:
- you don't speak French
- you don't know much about poetry

Clement Marot

1496-1544

Crappy-literal translation

- My aunt, dad,
  - You watch,
  - Where you stay,
  - In a world
  - Though to park,
  - Ever your bed,
  - Render not
  - The poor, the will,
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The awe, the awe
  - The aw...
How does this apply
You have two related problems:
- the syntax of French is something you have to learn
- the semantics of poetry is something you have to learn
You have two problems you have to solve at the same time.

Programming, Syntax and Semantics
- You have to learn the syntax of a particular programming language
- many details about the language, how to debug and use it
- You have to learn about problem solving and how to put it down on computer.
- There probably is no better way. It's hard!

Computers & problem solving?
This is both the problem and difficulty of computers.
- The promise (perhaps the hope) of computers is that, somehow, we can embed our own thoughts in them. To some extent we can!
- The problem is the difficulty of doing so, and the stringent requirements, the real rigor, required to put simple thoughts into a working program.

Focus of Computer Science
There are two foci for computer science
- Learning the difficult task of truly "laying out" a problem-solving task
- Providing tools to make this process as easy (though it will never be "easy") as possible.
Your focus should be on problem-solving, and adding rigor/focus to your ability to do problem-solving.

Good Program (1)
What makes a good program?
- a program is a reflection of the writer and their thoughts
- First, you must have some thoughts! The difficulty for most people is to figure out what has to be done, the problem solving, before writing a program

Rule 1
Think before you program!
How do I solve this problem???
Good Program (2)

- It will be said repeatedly that the goal of a program is not to run, but to be read.
- A program communicates with other people as well. It stands as a document to be read, repaired and, yes, run.

Rule 2

A program is a human-readable essay on problem solving that also happens to execute on a computer.

Why Python?

This book utilizes the programming language known as Python.

Why?

Why Python (1): Simpler

- Python is a simpler language
- Simpler means:
  - Fewer alternatives (one way to do it)
  - Better alternatives (easier to accomplish common tasks)
- This allows us to focus less on the language and more on problem solving.

Why Python (2): Best Practices

- Many of the best parts of other languages are included in Python
  - data structures (lists, dictionaries)
  - control (iteration, exceptions)
  - many packages for common tasks
- Python is often described as "batteries included"

Why Python (3): User base

- While we want to (and will) teach the fundamentals of computer science, we want what you learn to be useful
- Python is Open Source:
  - freely available
  - large user base constantly contributing
  - new packages available to meet changing needs
Why Python (4): Useful

- As a result, Python is more generally useful for getting work done.
- One course in Python makes you a capable (though not expert) programmer
- Can use available packages and your new skill to solve problems.

Computational Thinking

Having finished this course, we want you to have the following thought in your subsequent college career.

"Hey, I'll just write a program for that". For us, that is "computational thinking"

Python allows this to happen more readily.

Is Python the best language?

- The answer is no. This is because there is no best language.
- Computer languages, like tools, are suited for different tasks (what's the best shovel? Depends on what you are doing).
- For introductory students, we think Python is a very good language.

What is a computer?


Computer as a toaster

50 years ago when computers came into common existence, they were expensive, rare items used for research.

Today they are as common as a toaster (likely more common, do you own a toaster?)

Good to know a little bit about your toaster

Computers do computation

- Kind of obvious, but a computer is something that does computation.
- What's interesting is what counts as computation
**Human Brain**

- In the 1960's, when computers were becoming more prevalent, they were commonly called "electronic brains".
- However, the brain as a computer has very little in common with the modern computer (except that they both do computation).

**Neuron**

- The human brain consists of on the order of 100 billion \(10^{11}\) small cells called neurons.
- Neurons have a simple task, to act as a kind of switch.

**Transmission of signal**

- A neuron "fires", sends a signal, when enough chemical transmitters accumulate at its dendrites.
- When it fires, the signal, as a chemical/electrical impulse moves down the axon.
- When the signal reaches the terminals, more chemical transmitters are released to other neurons.

**Interesting facts**

- Signal transmission is very slow – millions times slower than a computer.
- Neuron's require recovery time before they can fire again.
- What causes a firing depends on the number of signals it receives at its dendrites and how fast they arrive.
  - that is, it depends on the number of connections.

**Evolution as Computation**

- There is an entire field devoted to the use of the principles of Darwin and Mendel to solve a problem: evolutionary computation.
- Evolution focuses on solving a problem by generating solutions (i.e. organisms) that adapt to the present environment.
### Facts

- maintains a population of solutions
  - solutions are an encoding of the problem
- Each solution is evaluated
- Good solutions tend to be used to create new solutions, variations on themselves
- Over time, the population contains better and better solutions

### People!

Computers used to refer to people

In WWII, computers were people who did difficult calculations by hand, for things like ballistic tables.

### The Modern Computer

What is Computation?

It is the manipulation of data by either humans or machines.

A computer is something that does computation.

### How does it work?

It's all about the switch

- The basic component of most digital circuitry is nothing more complicated than a simple switch.
- A switch's function is pretty obvious, said in a number of different ways:
  - On or Off
  - True or False
  - 1 or 0

### Electronic switch

- Early computers used vacuum tubes as switches
- Later, transistors were used as substitutes
- Using switches we can create logic circuits
Switches for Boolean Circuits

- Switches can be used to construct more complicated functions, such as Boolean circuits (and on left, or on right).

<Diagram showing AND and OR logic gates with flow of electricity>

Transistor is the key

- Transistor is an electronic switch:
  - If gate has current, then signal flows from source to sink.
  - If gate has no current, no flow.

<Diagram of transistor and its equivalent "latch" site.>

But a very interesting switch

- Transistors have three interesting "features" that have made them the fundamental element of the computer revolution.
  - Size
  - Quantity
  - Speed

<Images of Shockley transistor and Kilby integrated circuit.>

Size

- Originally very large

<Images of early computer hardware.>

Intel's first CPU

- By 1971, Intel had created a "computer on a chip", the 4004 microprocessor, the size of a fingernail with 2300 transistors.

<Image of the 4004 microprocessor.>

But now, really small!

- Chips now have gates measured in billionths of a meter (nanometers, nm)

<Images of microscopic view of transistors.>
Moore's Law

- Gordon Moore is one of the founders of the chip maker Intel
- In 1965, he observed (over that last 15 years or so) the growth rate of the number of transistors in a circuit
- Made a famous prediction

The "law"

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer."

Electronics Magazine 19 April 1965

What it means

- Roughly, since 1965, the number of transistors on a chip doubles every 18 months for approximately the same cost
- Often quoted as the speed of a cpu doubling every 18 months for the same cost
- Speed and density were related
How fast
Typical processor this day runs at GHz speeds. That is 1 billion operations a second.

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU</th>
<th>Clock Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>Intel 8080</td>
<td>4 MHz</td>
</tr>
<tr>
<td>1973</td>
<td>AMD 640</td>
<td>8 MHz</td>
</tr>
<tr>
<td>1977</td>
<td>Intel 8085</td>
<td>5 MHz</td>
</tr>
<tr>
<td>1985</td>
<td>Pentium II</td>
<td>333 MHz</td>
</tr>
<tr>
<td>1996</td>
<td>Pentium III</td>
<td>666 MHz</td>
</tr>
<tr>
<td>1999</td>
<td>Pentium 4</td>
<td>2.8 GHz</td>
</tr>
<tr>
<td>2000</td>
<td>Duron</td>
<td>2.2 GHz</td>
</tr>
<tr>
<td>2002</td>
<td>Pentium 4</td>
<td>3 GHz</td>
</tr>
<tr>
<td>2003</td>
<td>Xeon</td>
<td>3 GHz</td>
</tr>
<tr>
<td>2004</td>
<td>P4-M</td>
<td>1.8 GHz</td>
</tr>
<tr>
<td>2005</td>
<td>Core 2 Duo</td>
<td>1.6 GHz</td>
</tr>
<tr>
<td>2007</td>
<td>Core 2 Duo</td>
<td>3 GHz</td>
</tr>
</tbody>
</table>

How fast is a nanosecond?

Nanoseconds

- Speed of light = 186,000 miles/sec, 3x10^8 meters/sec (actually 299,792,458)
- nanosecond = 10^-9 seconds
- thus 30 centimeters or 11.8 inches

Speed pushing up against physics

- the "clock" is like a drummer in the band. The faster it beats, the faster the operations.
- in 1 clock tick (at 1 GHz), electricity can travel 1 ft. At 2 GHz, 6 inches. At 4 GHz, 3 inches
- It becomes very difficult to get electricity to travel any distance in that short of time!

How to get around physics

- So if physics is getting in the way (and it is), you find a way to get around it.
- If you can't make processors faster, what do you do?

Not faster, but more CPUs
Now more transistors mean more CPUs on a single chip. Below a quad-core CPU

Computer Parts
Main Components

- People!
- Hardware
  - Physical Devices: processor, memory, keyboard, monitor, mouse, etc.
- Software
  - Executable Programs: word processor, spreadsheet, internet browser, etc.

Processor

- The processor is the "brain" of a computer.
- The processor controls the other devices as well as performing calculations.

Hardware

- Memory Storage
  - secondary storage
  - input device
  - output device
  - network

Primary Storage

- stores instructions and data for current program(s)
- other names: primary or main memory, RAM (Random Access Memory)
- memory is dynamic so it requires power to retain information
- often hundreds of Megabytes (million-bytes) or Gigabytes (billion-bytes)

Peripheral Devices

- Secondary storage devices
  - disk (hard & floppy), tape, USB drives, flash drives, etc.
- Input devices
  - keyboard, mouse, camera, mic, etc.
- Output devices
  - monitor, printer, speaker, etc.
- Network
  - wireless, Bluetooth, Ethernet, etc.

Secondary Storage

- nonvolatile -- information is recorded magnetically so power is not needed
- disks hold Gigabytes (billions of bytes)
- cheap, but slow
  - RAM access is a hundred CPU clock ticks
  - disk access is a million CPU clock ticks
- not directly accessed by CPU
Software

- the programs available for execution
- simple classification
  - system software
  - application software

Fetch-Decide-Execute-Store

CPU operates on a simple (but fast) cycle:
- **Fetch**: fetch instruction from memory
- **Decode**: Decode requirements (args, etc.)
- **Execute**: Perform operation
- **Store**: move needed results to memory
- **Repeat**

What is a program?

- A sequence of instructions written in machine language that tells the CPU to take action (i.e. add two numbers) in a specific order
- In this course we will learn to create programs

Program Storage

- machine language instructions are encoded as bit patterns (binary, remember our transistors)
- memory can only hold binary info.
- a bit is represented by two-states, e.g. L-R magnetism, high-low voltage
- it takes many bits to represent reasonable amounts of information

Binary numbers

- The switch nature of transistors make storing numbers in binary a natural fit.
- Binary is a change of base for our number system, base 2
- In a number, its position represents powers of 2

Example

- Decimal uses digits 0-9 and positions in a number as powers of 10
  - $735_{10} = 7 	imes 10^2 + 3 	imes 10^1 + 5 	imes 10^0$
- Binary uses digits 0,1 and positions in a number as powers of 2
  - $1012_{2} = 1 	imes 2^3 + 0 	imes 2^2 + 1 	imes 2^1 + 2^0$
- We can convert back and forth
  - $1012_{2} = 5_{10}$