CSE 101: Computer Science Principles

Unit 1:

What is Computational Thinking?
Overview of Computational Thinking
What is Computer Science?

• **Computer science** is all about using computers and computing technologies to solve challenging, real-world problems in science, medicine, business and society

• Although computer programming is an important aspect of computer science, it would be wrong to say that “computer science equals computer programming”

• Rather, computer programs often provide (parts of) the solutions to challenging technological problems

• Computer science is also not:
  • computer literacy
  • computer maintenance/repair
  • a fast track to becoming a nerd (but it might help)
Are You a Good Fit for CS?

• You are a good fit for computer science if:
  • You are naturally curious and inquisitive.
  • You feel compelled to solve problems and puzzles.
  • You have a creative spirit and like making things.
  • You think in a logical, step-by-step manner.
  • You approach issues from unconventional angles.
  • You are willing to evolve and learn new things every day.
  • You are self-driven and have enough grit to endure long periods of frustration.
  • You know how to search the web for answers.
• List courtesy http://www.makeuseof.com/tag/what-is-computer-science
A Modern Computing Problem

- Electronic health records are becoming increasingly important as time goes on.
- Consider some of the issues (technical and otherwise) that would arise in solving the problem of providing a hardware/software system to medical professionals and other people who need access to digital medical records:
  - What data will be stored? How? In what format?
  - How will the data be accessed and displayed?
  - Who will have access? How will the data be secured?
  - How will the data be backed up and preserved?
- Answering these questions requires computational thinking.
What is Computational Thinking?

- Computational thinking refers to how computer scientists think – how they reason and work through problems.
- Computer science encompasses many sub-disciplines that support the general goal of solving problems:
  - Computer theory areas: algorithms, data structures – these are the heart and soul of computer science.
  - Computer systems areas: hardware design, operating systems, networks.
  - Computer software and applications: software engineering, programming languages, computer graphics, databases, simulation, artificial intelligence.
- A major goal of this course is to help you develop your computational thinking and problem-solving skills.
A Classical Problem: Sorting Data

- Suppose we have a deck of cards we want to put in order
- This is the important problem of **sorting** that arises very frequently in computer science
- To keep things simple, let’s just use the Ace through 8 of Hearts
- We are given:

![Card Deck](image1)

- But we want:

![Card Deck](image2)
A Classical Problem: Sorting Data

• Imagine you wanted to explain to a young child how to put the cards in order. What steps would you give?
A Classical Problem: Sorting Data

- One sorting technique is called **selection sort**
- It repeatedly searches for and swaps cards in the list

![Playing cards](image)
A Classical Problem: Sorting Data

- First, find the smallest item and exchange it with the card in the first position
A Classical Problem: Sorting Data

• First, find the smallest item and exchange it with the card in the first position
A Classical Problem: Sorting Data

- First, find the smallest item and exchange it with the card in the first position
A Classical Problem: Sorting Data

• Now, find the second-smallest item and exchange it with the card in the second position
A Classical Problem: Sorting Data

• Now, find the second-smallest item and exchange it with the card in the second position
A Classical Problem: Sorting Data

- Now, find the second-smallest item and exchange it with the card in the second position
A Classical Problem: Sorting Data

• Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

• Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

• Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

- Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

• Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

• Continue in this fashion, *selecting* the third-smallest, fourth-smallest, etc., until the list is sorted.
A Classical Problem: Sorting Data

• Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

• Continue in this fashion, *selecting* the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

- Continue in this fashion, *selecting* the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

- Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted.
A Classical Problem: Sorting Data

- Continue in this fashion, selecting the third-smallest, fourth-smallest, etc., until the list is sorted
A Classical Problem: Sorting Data

- Continue in this fashion, *selecting* the third-smallest, fourth-smallest, etc., until the list is sorted

Finished!
A Classical Problem: Sorting Data

• Another sorting technique is **insertion sort**
• It repeatedly inserts the “next” card into its correct spot
A Classical Problem: Sorting Data

• We begin by leaving the first card (#5) where it is
A Classical Problem: Sorting Data

• The second card (#3) is smaller than the first card
• *Insert* it in front of the first card
A Classical Problem: Sorting Data

• The second card (#3) is smaller than the first card
• Insert it in front of the first card
A Classical Problem: Sorting Data

- The third card (#6) is larger than the first two cards
- So, we don’t need to move it
A Classical Problem: Sorting Data

• The fourth card (#1) is smaller than the first three cards
• Insert it in front of the first card, shifting the others
A Classical Problem: Sorting Data

- The fourth card (#1) is smaller than the first three cards
- Insert it in front of the first card, shifting the others
A Classical Problem: Sorting Data

- The fifth card (#7) is larger than the first four cards
- So, we don’t need to move it
A Classical Problem: Sorting Data

- The sixth card (#4) should be inserted in between the second (#3) and third (#5) cards.
A Classical Problem: Sorting Data

• The sixth card (#4) should be inserted in between the second (#3) and third (#5) cards
A Classical Problem: Sorting Data

• The seventh card (#8) is larger than the first six cards
• So, we don’t need to move it
A Classical Problem: Sorting Data

- The eighth card (#2) should be inserted in between the first (#1) and second (#3) cards
A Classical Problem: Sorting Data

• The eighth card (#2) should be inserted in between the first (#1) and second (#3) cards
A Classical Problem: Sorting Data

- The eighth card (#2) should be inserted in between the first (#1) and second (#3) cards

Finished!
Algorithms for Sorting

• We have just confirmed that there can be different ways to solve the same computational problem
• That is, we could derive many different algorithms for solving the sorting problem
• An algorithm is a set of concrete steps that solve a problem or accomplish some task in a finite amount of time
• For example, the Selection Sort and Insertion Sort algorithms are just two ways of sorting a list of values
• Suppose we wanted to sort a list of student records by the students’ GPAs. Would both of these algorithms work?
• Yes! A hallmark of a good algorithm is that it is general and can work to solve a wide variety of similar problems
Basics of Computer Systems
Computing Systems

• Let's take a tour of modern computing systems
• A computing system consists of two major parts: the **hardware** and the **software**
• What are some of the hardware elements of a computer?
  • Screen, keyboard, mouse
  • Central processing unit, main memory
  • Hard drives and other storage units
• What kinds of software exist?
  • Applications software, like office productivity programs, video games, web browsers
  • Systems software, like operating systems, database systems
Computing Systems

• Can hardware exist without software?
  • Sure, but it’s not very useful, is it!
• Can software exist without hardware?
  • In a literal sense, no – hardware is needed to execute software
  • But the underlying problem-solving techniques employed by the programmer to create the software do exist separately from the hardware and software
• We can throw in one more part to a computer system: data
  • The software needs some kind of data to process: numbers, text, images, sound, video
A Quick History of Computing

• We think of computers as modern inventions, but computing devices go back thousands of years and have many of the same basic features of digital computers

• Abacus – an early device to record numeric values and do basic arithmetic (16th century B.C.)

• What does an abacus have to do with laptops, smartphones and tablet computers???
A Quick History of Computing

• Modern computers borrow four important concepts from the abacus:
  1. Storage
  2. Data Representation
  3. Calculation
  4. User Interface

  1. Storage: an abacus can store only numbers, but numbers are the most fundamental kinds of data we deal with in modern computing.

• In a modern computer, all data – text, images, audio, video – is represented using binary numbers (ones and zeros)
A Quick History of Computing

2. Data Representation: the abacus represents numbers using beads on spindles.
   • Modern computers employ a variety of techniques – magnetic, optical, electrical – for representing data on a variety of storage media

3. Calculation: by moving beads on the abacus’s spindles, the user can perform addition, subtraction, multiplication and division
   • Modern computers contain powerful **central processing units** that perform calculations at astonishing speeds

4. User Interface: the beads and spindles on the abacus
   • Modern computers provide a wide variety of input and output devices for the user
A Quick History of Computing

• In the 17th century people began tinkering with physical devices that could do computations and calculations
• Blaise Pascal – the French mathematician and philosopher – was one of a few people to design and build a physical calculator
• His calculator could do only addition and subtraction
• Input is given using dials, and output is read on small windows above each dial
Programmable Devices

• Pascal’s calculator and other similar devices of that time were not programmable
• One of the first programmable devices in history was a loom
• Joseph Marie Jacquard’s loom (1804) could be programmed to weave different patterns in cloth by feeding in a set of punched cards
• This is not all that different from closing a program that’s running on your computer and starting another one!
Programmable Devices

- Another leap forward came in the 19th century with Charles Babbage’s design of the Analytical Engine, a mechanical, programmable computer
- It was never built in Babbage’s time due to a lack of manufacturing capabilities (ahead of his time!)
- The design called for punched cards to be fed into the machine to program it to perform mathematical calculations
- Output would go to a printer or punched cards
Programmable Computers

• We could go on and on with the history of computing, so let’s move forward to the 20th and 21st centuries

• A modern computer has three basic requirements:
  1. It must be electronic and not exclusively mechanical.
  2. It must be digital, not analog. This means that it uses discrete values (digits) and not a continuous range of values to represent data. (Contrast a digital thermometer with an alcohol-based or mercury-based one.)
  3. It must employ the **stored-program concept**: the device can be reprogrammed by changing the instructions stored in the memory of the computer.
Programmable Computers

- The ENIAC (Electronic Numerical Integrator and Computer) of the 1940s was among the first computers to employ the stored-program concept.

- Note that a modern computer has four major kinds of components:
  - Input device(s) – examples?
  - Output device(s) – examples?
  - Memory – for data storage, both temporary and permanent
  - Processor – for doing computations
Programmable Computers

- Again, the **stored-program concept** is the idea that programs (software) along
  with their data are *stored* (saved) in the memory of a computer
- We’re not talking about storing data on hard drives, flash drives or optical discs – we’re talking about the **main memory** of the computer, sometimes called the **RAM** (random access memory)
- A modern processor reads the **machine instructions** *stored* as ones and zeroes in the main memory and then executes those instructions in sequence
- The key point here is that these instructions can be changed to easily reprogram the computer to do new tasks
Transistors

• Over time, a variety of devices was used to represent the digits and to control the operation of computing machines
• In the 1940s, Bardeen, Brattain and Shockley invented the transistor, which is an electronic switch with no moving parts
• In the 1950s and 1960s, Kilby, Noyce and others used transistors to develop integrated circuits
• They devised a way to manufacture thousands of transistors on a single wafer of silicon
• A single chip contains an integrated circuit, a ceramic or plastic case, and external pins to attach it to a circuit board
Transistors

- Noyce and businessman Gordon Moore commercialized this technology by co-founding Intel Corporation in 1968.
- As manufacturing technologies improved in the 1950s and 1960s, engineers were able to pack many more transistors per unit area on silicon wafers.
- Moore’s law: Moore observed that the number of components within an integrated circuit was doubling every 18 months, a trend which has continued steadily since then. But transistors can be only so small!
- To combat miniaturization challenges, manufacturers like Intel and AMD (Advanced Micro Devices) now make processors that feature multiple processing cores that perform calculations in parallel with each other.
Modern Computer Architecture

- The stored program approach we use today is implemented using von Neumann architecture, named after U.S. mathematician John von Neumann.
- The von Neumann architecture consists of input devices and output devices, a processor and a memory unit.

- We’ll see now how they work together to form a functioning computer.
Modern Computer Architecture

- In a modern computer, the major components in a von Neumann machine reside physically in a circuit board called the **motherboard**
- The CPU (central processing unit), memory, expansion cards and other components are plugged into slots so that they can be replaced
- Hard drives, optical disc drives and other storage devices are connected to the motherboard through cables
- The central processing unit is the “brain” of the machine: its **arithmetic/logic unit** (ALU) performs millions or billions of calculations per second
- The CPU’s **control unit** is the main organizing force of the computer and directs the operation of the ALU
Modern Computer Architecture

• The memory unit in this diagram refers to the main memory, not hard drives and other forms of external storage

• The CPU, main memory and I/O devices communicate over a shared set of wires known as the system bus
The Fetch-Decode-Execute Cycle

- The system bus carries electrical signals that encode machine instructions and data
- The CPU **fetches** the instructions and data from memory as needed
- The control unit **decodes** each instruction to figure out what it is (an addition, subtraction, etc.)
- Data values (e.g., numbers to be added and their resultant sum) are stored temporarily in memory cells called **registers** within the CPU
- The ALU **executes** the instruction, saving the result in the registers and main memory
- This whole process is known as the **fetch-decode-execute cycle**
The Fetch-Decode-Execute Cycle

1. Fetch Instruction
2. Decode Instruction

FETCH CYCLE

3. Get Data
4. Execute Instruction

EXECUTION CYCLE

Main Memory

Control Unit

Registers

ALU
What About the Software?

• Software consists of instructions for the CPU to execute
• The problem is that CPUs “understand” something called machine language, which consists of zeroes and ones
• A single instruction for a modern computer might consist of some combination of 32 or 64 zeroes and ones!
• Most programming now is done using high-level programming languages, which consist of English and English-like words with some mathematical notation thrown in
• In this course you’ll be learning the fundamentals of Python, which is a popular and easy-to-learn, high-level programming language
To Sum Up...

- Computer science is the discipline of how to solve problems using computers
  - We strive for efficient, general solutions that will work on a wide variety of problem types
- Although modern computer science has existed as a field for about 80 years, its roots in mathematics and computation go back thousands of years!
- CS is a very peculiar field in that it relies partly on old mathematical ideas, yet it advances in development at an extraordinary pace
- This course will expose you to some of the modern topics in CS and also to some of the older mathematical content that is still very relevant today