What is this course about?

• Programming:
  • Writing the instructions to control a computer, causing it to perform some useful task

• What tasks do we want our programs to perform:
  • Word processing, communicating, gaming, watching videos, listening to music, scientific experiments, creating videos/music, managing finances...

• Who needs to know how to program?
  • Just about everyone uses computer programs
  • Many people write computer programs as a small part of their jobs (especially scientists and engineers)
  • Some people write programs as the main part of their job
• To understand the activity of programming
• To learn about the architecture of computers
• To learn how numbers are represented by computers
• To learn about machine code and high level programming languages
• To recognize syntax and logic errors
• To learn about the compilation process
What Is Programming?

• Computers are programmed to perform tasks
• Different tasks = different programs
• Program
  • *Sequence of basic operations executed in succession*
  • *Contains instruction sequences for all tasks it can execute*
• Sophisticated programs require teams of highly skilled programmers and other professionals
Hardware / Software

• **Hardware** is the physical aspect of the computer system
  • It is composed of printed circuit boards, integrated circuits (i.e. chips), wiring, power supplies, cases, and mounting hardware
  • It is rarely modified
  • "The parts of a computer system that can be kicked." Henri Karrenbeld

• The programs that run on a computer are also known as **software**
  • A program is a pattern of information, not a physical device
  • Unlike hardware, software is frequently modified
Schematic Diagram of a Computer

Figure 5  Schematic Diagram of a Computer
Central Processing Unit

• Executes the instructions given in software
  • Usually contained within a single **integrated circuit**
  • Integrated circuit (a.k.a. a chip): Electronic circuitry housed within a small package, with metal pins sticking out

• Consists of millions of **transistors**
  • Transistor: Tiny electronic components that can control whether current flows through a circuit or not

• Executes very simple instructions, very rapidly:
  • e.g. copy this number from location A to B
  • e.g. multiply these numbers

• General purpose device (used for all types of programs)
• Programs and data are all represented as binary numbers (see later) and stored on various media
  • **Random-access memory (RAM)**
    • Electronic storage which is lost when the power is turned off
    • Called "random-access" because it is just as fast to access any (i.e. a random) position in memory
    • The cost per byte (defined below) is high
    • Access time is fast
    • a.k.a. (Also Known As) primary storage

  • **Secondary storage**: e.g. hard disk
    • Information stored on a rotating magnetic disk (actually, a multi layered platter of disks)
    • The cost per byte is low
    • Access time is much slower than RAM

• Programs are usually loaded from secondary storage into RAM when they are executed

• Removable storage devices: e.g. CDs, USB keys, memory cards
**Figure 2**
A Memory Module with Memory Chips
A Hard Disk

Figure 3  A Hard Disk
The CPU is connected to RAM, hard disk, and other devices through the **bus**
- **Bus**: A set of wires used for communication between various devices

The CPU and bus are located on the **motherboard**
- **Motherboard**: A large printed circuit board that all of your computer's devices connect to
- Contains slots for RAM
- Slots for other devices
  - Sound card
  - Graphics card
  - Network card
A Motherboard
The ENIAC: The World's First Electronic Computer
Where is a program stored when it is not currently running?

**Answer:** In secondary storage, typically a hard disk.
Self Check 1.5

Which part of the computer carries out arithmetic operations, such as addition and multiplication?

**Answer:** The central processing unit.
Bistable Devices

• Numbers are represented in a computer using **bistable devices**, a device which can be in two stable states.
  • Electronic circuits can be designed such that the voltage in part of the circuit is either 0 or 5 volts and can be controlled by an external input signal.
  
  • A tiny segment of a magnetic disk or tape can be magnetically polarized in up or down configuration by a "head". The head can also read the configuration.

• On a CD there are "pits" and "grounds". A laser hitting a pit will not get reflected. If the laser hits a ground it will get reflected.
Bistable devices can be combined to represent **binary numbers**

First, let's revisit what we mean by decimal numbers

\[ 346 = 3 \times 10^2 + 4 \times 10^1 + 6 \times 10^0 \]

We can write \( 346 \) \(_{\text{decimal}}\) to make it clear we mean decimal

Decimal numbers use base 10 but this is arbitrary. Some cultures have used number systems with bases of 12, 20, or 60.

Binary numbers use base 2

\[ 1101 \text{\(_{\text{binary}}\)} = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]

\[ = 8 + 4 + 1 = 13 \text{\(_{\text{decimal}}\)} \]
• To convert from binary to decimal, sum up all of the powers of 2 corresponding to digits with value 1

\[ 2^0 = 1, \ 2^1 = 2, \ 2^2 = 4, \ 2^3 = 8, \ 2^4 = 16, \ 2^5 = 32, \ldots \]

• To convert from decimal to binary, apply the following procedure to the number X:
  – Divide X by 2
  – The remainder is the next right-most digit in binary
  – Let X be the quotient
  – If X = 0 we are finished
  – Otherwise go to step 1

e.g. \( X = 6 \)

6 / 2 = 3 remainder 0
3 / 2 = 1 remainder 1
1 / 2 = 0 remainder 1

\( 6_{\text{decimal}} = 110_{\text{binary}} \)
• **e.g. X = 100**

  100 / 2 = 50 remainder 0
  50 / 2 = 25 remainder 0
  25 / 2 = 12 remainder 1
  12 / 2 = 6 remainder 0
  6 / 2 = 3 remainder 0
  3 / 2 = 1 remainder 1
  1 / 2 = 0 remainder 1

  $100_{\text{decimal}} = 1100100_{\text{binary}}$

• The binary number system applies also to non-integer numbers

  $$1.101_{\text{binary}} = 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

  \[= 1 + \frac{1}{2} + \frac{1}{8} = 1.625_{\text{decimal}}\]
Bits and Bytes

• A single binary digit is called a **bit**
• 8 bits makes a **byte**
  • A byte can represent one of the standard characters (a-z, A-Z, 0-9, plus special characters and punctuation)
• $10^3$ bytes make up a **kilobyte** (kB)
• $10^6$ bytes make up a **megabyte** (MB)
• $10^9$ bytes make up a **gigabyte** (GB)
• Typical file sizes:
  • Small document: 100 kB
  • PDF file: 1 MB
  • Powerpoint presentation: 2.5 MB
  • Image: 0.5 - 10 MB
  • MP3 Audio: 3 - 8 MB
• Typical PC specification:
  • 1 GB RAM
  • 160 GB Hard disk
Machine Code and Assembly Language

• The instructions that a CPU executes are encoded as binary numbers
• A computer can be programmed simply by specifying instructions as numbers
  • Such instructions are known as **machine code**
• It is slightly more intuitive to write a program using short commands that translate directly into machine code
  • Such instructions are known as **assembly language**
  • e.g.
    
    ```
    add edx, eax
    mov eax, 1
    jmp L2
    ```

edx and eax are places within the internal memory of the CPU. L2 is a label of a place in memory that the CPU is about to jump to (hence "jmp").
Machine code and assembly language are difficult to use.
Most programmers utilize a **high-level language**, which uses concepts like variables to refer to locations in memory and has a structure that looks more like English:

```java
if ( balance <= 0 )
{
    System.out.println("You have no money left!");
}
```

This is an example of **Java** code. There are many other high-level languages: C, C++, Python, Perl, etc...

For some of these languages the **source code** (i.e. instructions written by the programmer) is converted to machine code by a special program called the **compiler**
The Java compiler translates Java source code into an intermediate form known as \textit{bytecode}.

A special program called the Java Virtual Machine (JVM) executes the bytecode.

Actually, there are two ways in which bytecode can be executed:

- Interpreted line-by-line
  - This is slower but works the same on any computer running a JVM
- Compiled into machine code just before execution
  - It takes a little bit of time for compilation, but the program will then run faster. However, the compilation has to be done differently for every type of computer.
We refer to the individual instructions of bytecode as machine instructions – a typical sequence of machine instructions is:

- Load the contents of memory location 40.
- Load the value 100.
- If the first value is greater than the second value, continue with the instruction that is stored in memory location 240.

Machine instructions are encoded as numbers:

```
21  40
16  100
163 240
```

These instructions are shown in decimal, but inside the computer they are represented in binary.
Self Check 1.6

What is the code for the Java virtual machine instruction "Load the contents of memory location 100"?

**Answer:** 21 100
Does a person who uses a computer for office work ever run a compiler?

**Answer:** No – a compiler is intended for programmers, to translate high-level programming instructions into machine code.
The Java Programming Language

• Java has a number of excellent qualities:

  • Simplicity: Java bears a lot of similarity to the C++ language (its closest rival), but is simpler and therefore much easier to program in.

  • Portability: Java code written on one computer can be compiled into bytecodes and executed on any computer that has a JVM ("write once, run anywhere")

  • It has a rich library of packages for graphics, networking, input/output devices, etc...

  • It was designed for the internet. A special type of JVM can run within a web browser and execute Java applets.
Figure 6  An Applet for Visualizing Molecules ([1])
## Java Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Year</th>
<th>Important New Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1996</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>1997</td>
<td>Inner classes</td>
</tr>
<tr>
<td>1.2</td>
<td>1998</td>
<td>Swing, Collections</td>
</tr>
<tr>
<td>1.3</td>
<td>2000</td>
<td>Performance enhancements</td>
</tr>
<tr>
<td>1.4</td>
<td>2002</td>
<td>Assertions, XML</td>
</tr>
<tr>
<td>5</td>
<td>2004</td>
<td>Generics, enhanced for loop, auto-boxing, enumerations</td>
</tr>
<tr>
<td>6</td>
<td>2006</td>
<td>Library improvements</td>
</tr>
</tbody>
</table>
Our First Java Program!

```java
public class HelloPrinter {
    public static void main(String[] args) {
        // Display a greeting in the console window
        System.out.println("Hello, World!");
    }
}
```

Output:
Hello, World!
HelloPrinter in a Console Window

```
~$ cd BigJava/ch01/hello
~/BigJava/ch01/hello$ javac HelloPrinter.java
~/BigJava/ch01/hello$ java HelloPrinter
Hello, World!
~/BigJava/ch01/hello$
```

Figure 10  Running the HelloPrinter Program in a Console Window
HelloPrinter in an IDE

Figure 11
Running the HelloPrinter Program in an Integrated Development Environment
A Simple Program

• The program HelloPrinter has four main parts:
  • public class ClassName
  • public static void main(String[] args)
  • // comment
  • Method call

Figure 12
Calling a Method

System class
System.out object
println method
Syntax 1.1 Method Call

object.methodName(parameters)

Example:
System.out.println("Hello, Dave!");

Purpose:
To invoke a method of an object and supply any additional parameters.
Self Check 1.12

How would you modify the HelloPrinter program to print the words "Hello," and "World!" on two lines?

**Answer:**

```java
System.out.println("Hello," );
System.out.println("World!" );
```
Would the program continue to work if you omitted the line starting with `//`?

**Answer:** Yes – the line starting with `//` is a comment, intended for human readers. The compiler ignores comments.
What does the following set of statements print?
System.out.print("My lucky number is");
System.out.println(3 + 4 + 5);

**Answer:** The printout is
My lucky number is12
It would be a good idea to add a space after the is.
Syntax Errors (a.k.a. Compile-time Errors)

• **Syntax errors** are violations of the rules of a language. For example, there is an error in the syntax of the following English-language sentence:

   "She to went the store."

• Here are two syntax errors in Java:

   ```java
   System.ouch.print(".
   System.out.print("Hello);
   ```

• The compiler will print out a list of the syntax errors in your program. Always try and fix the first one in the list first.
Logic Errors (a.k.a. Run-time Errors)

- **Logic errors** are mistakes made in the logical flow of your program, with respect to what you want your program to do. For example, if it is raining outside and I want to tell someone, I don't say "It is Wednesday."

  The statement by itself is logical, but it does not achieve what I set out to do.

- Here is a logic error in a Java program:

  ```java
  System.out.print("Hell");
  ```

- The compiler will **not** tell you about logic errors because it does not know what you are trying to do.

  **You must test your code to make sure it works!**
Suppose you omit the // characters from the HelloPrinter.java program but not the remainder of the comment. Will you get a compile-time error or a run-time error?

**Answer:** A compile-time error. The compiler will not know what to do with the word Display.
Self Check 1.16

How can you find logic errors in a program?

**Answer:** You need to run the program and observe its behavior.
The Compilation Process

• Create your program by typing it into an **editor**.
  • An editor is a program that is used to view and modify source code files, as well as other simple types of files (word processors such as Microsoft Word are not considered editors)
    • e.g. notepad in Windows
    • e.g. gedit in Linux
  • A Java program should go in a file whose name ends with ".*java*" (without the quotes).

• Compile your program by running the Java compiler program **javac**
  • **javac** will generate a **class file** from a ".*java*" file.
  • e.g. from "HelloPrinter.java" we get "HelloPrinter.class"
  • A class file contains the bytecode that will be executed by the JVM
The Compilation Process

Figure 13  From Source Code to Running Program
The Edit-Compile-Test Loop

Figure 14
The Edit–Compile–Test Loop
What do you expect to see when you load a class file into your text editor?

**Answer:** A sequence of seemingly random characters, some funny-looking. Class files contain virtual machine instructions that are encoded as binary numbers.
Self Check 1.18

Why can't you test a program for run-time errors when it has compiler errors?

**Answer:** When a program has compiler errors, no class file is produced, and there is nothing to run.