What is Programming?

Computer science is not the study of programming. Programming, however, is an important part of what a computer scientist does. **Computer programming** is:

"*the process of developing and implementing various sets of instructions to enable a computer to do a certain task*".

These instructions are considered computer programs and help the computer to operate smoothly or perform in a certain way.

Hardware vs Software

Computer **hardware** refers to the physical components of a computer system. These include the monitor, mouse, keyboard, speakers, hard drive, motherboard, graphics/sound card, screen, battery, computer chips, discs etc. All of these are physical, tangible objects which you can see and touch.

Computer **software** (or simply software) refers to the series of instructions (programs) that directs a computer or device to perform specific operations or tasks. Software allows a user to interact with a computer and its hardware and instructs hardware on how to respond to inputs. These are not physical elements and instead are composed of code. Examples of software include programs or applications that you may use on your computer/phone such as Operating Systems (*Windows 10, Mac OS X*), Applications (*Whatsapp, Photoshop*), Games (*Call of Duty*), Antivirus (*McAfee*), Browsers (*Chrome, Safari*) etc.

Of course **Hardware** and **Software** go hand in hand. You can't do anything with a device unless you have the software to interact with it and what the software is able to achieve is limited by the hardware that is available.

Programming Languages

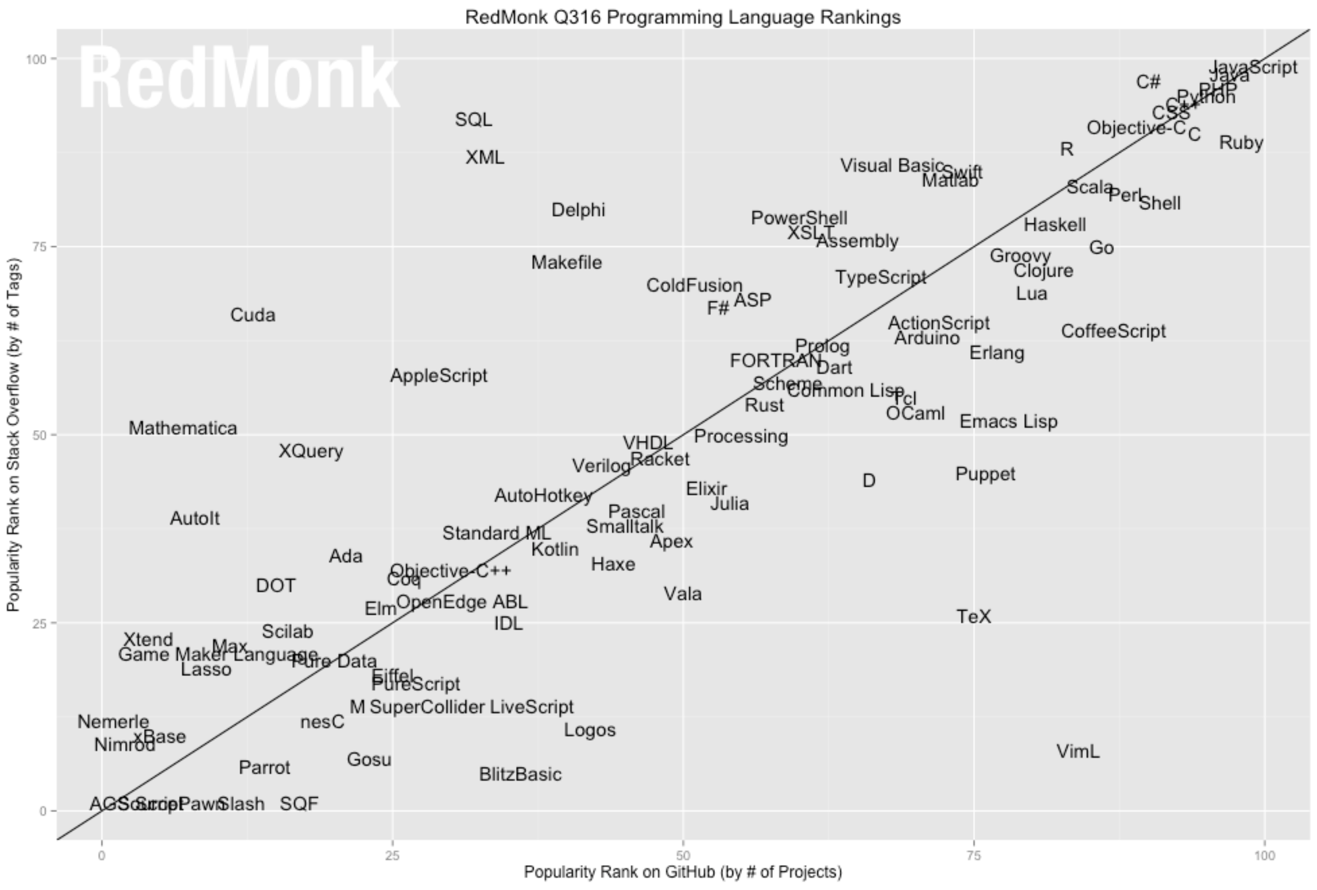
A **programming language** is a special language programmers use to develop software programs, scripts, or other sets of instructions for computers to execute. Typically higher level programming languages have strong **abstraction** from the complex details of the computer. That is to say they are designed to be easier to understand and use when a human wants to write instructions to a machine.

To put the term "abstraction" in another real world example let's look at how a car engineer makes it easier for a driver to accelerate a car. All the driver has to do is press the accelerator to speed up. In this way the engineer abstracts the complex process of injecting extra fuel into the engine under high pressure to move the car forward by making a relatively simple process to carry out all of this.

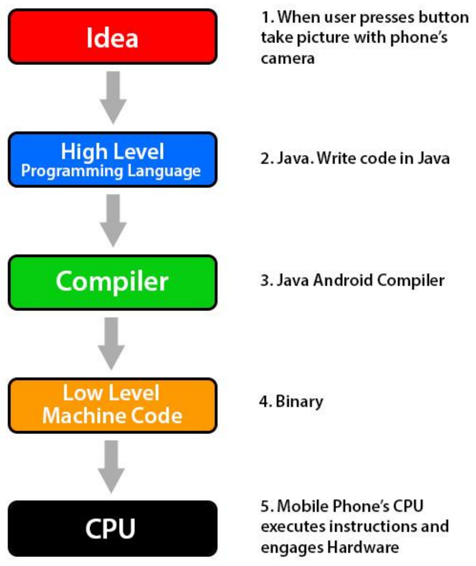
As such these programming languages may use real English words and terms. While coding may seem very complicated at first, these high-level programming languages are much easier to use than resorting to machine code or binary data e.g. 11010010010101100101010101, which is what a machine understands.

The terms high-level and low-level are entirely relative however. Generally speaking lower-level code is more efficient because it has a faster execution speed, lower memory consumption and smaller binary program size.

The most popular programming languages for 2017:
https://blog.appdynamics.com/engineering/the-most-popular-programming-languages-for-2017/



Binary/Machine Code


Idea
1. When user presses button take picture with phone's camera
High Level Programming Language
2. Java. Write code in Java
Compiler
3. Java Android Compiler
Low Level Machine Code
4. Binary
CPU
5. Mobile Phone's CPU executes instructions and engages Hardware

Software is written with Programming Languages. Programming languages translate human instructions into binary code which is interpreted (understood) by computers. This binary code may often be referred to as Machine Code and is regarded as the lowest-level representation of a compiled or assembled computer program. This is executed directly by the computer's CPU (Central Processing Unit).

While it is possible for humans to write this low level code themselves, in practice it is very tedious and highly prone to error. Except for highly optimized situations and extreme debugging (the process of finding and resolving bugs or defects that prevent correct operation of computer software or a system), all programs are created with higher-level programming languages.

High-level programming languages are translated into executable low-level machine code by utilities such as compilers, assemblers and linkers.

High-Level	Translation	**Low-Level**
Example Languages: JavaScript, PHP, C++, Java	*Interpreter* Compilers Assemblers Linkers	Example: 1010111001110101010101011
Code may use natural language elements		Code is made up of numerical binary notation
Easier to understand and create by humans		Difficult to understand and write by humans
Translated by compilers / interpreters		Executed by CPU

What differentiates programming languages from each other?

Some languages are better designed to handle particular things. For instance JavaScript is great for web development while Objective-C is used to create apps for iOS. There are a number of different ways in which programming languages differ. Some of the ways are very specific while others are relative and open to interpretation. The main topics of differentiation are listed below:

- **Syntax**
- **Low-Level** vs **High-Level** - degree of abstraction.
- **Specialized** vs **General purpose**
- **Compiled** vs **Interpreted**
- **Strongly-Typed** vs **Loosely/Weak-Typed**
- **Enforced Programming Style** vs **Non-Enforced Programming Style** - this is usually described as Object Orientated vs Non-Objected Oriented Language

Syntax

A computer language's syntax is the set of rules and symbols that are used throughout it. It essentially boils down to what the code looks like and how it is to be used. Much like French, English and Spanish, while they use the Latin alphabet and try to convey the same information, they look different, have different rules and use different symbols.

Low-Level vs High-Level

As described above, this refers to how close to the complicated machine code the programming language gets. Generally all programming languages are considered High-Level however in recent times some people are of the opinion that the C based languages should be considered Low-Level.

Specialized vs General purpose

This refers to what the purpose of the language is. Almost all popular programming languages will be classified as general purpose because they can be used for multiple endeavours. For instance C++ is known as a popular language for video game development however it can be used to write code for many different things and thus is general purpose. An example of a specialized programming language (sometimes referred to as Domain Specific) would be R which is only used for statistical analysis.

Compiled vs Interpreted

Certain programming languages need to have their code compiled before it is executed. This is to say, transformed into an executable program so that a computer can understand and run the program without the need of the programming software used to create it. This process happens only once: when the file is being created. In the past you may have had to download a Java compiler/kit on your computer in order to run certain applications. While the time needed to make the executable file may be long, which is annoying for development purposes, once created, whenever the file is run the code runs very fast.

The alternative of this is an interpreted language sometimes known as scripting languages. These do not have to be compiled before they are executed however they require an interpreter to run. An interpreter takes the code and modifies it in such a way that it can be understood and then executed by the computer. This process of translation happens on the fly and must happen each time the code is run. The main programming languages that we are interested in: JavaScript and PHP are interpreted languages and their interpreter is the web browser. Therefore there's no need to install any additional software. Since there is no time needed to compile the code, interpreted languages are very good for development purposes. That said however since the code must be interpreted each time, it doesn't run as fast when running.

Compiled	Interpreted
Java, Objective-C, C, C++, C#	JavaScript, PHP, Python, Perl, Ruby
Needs Compiler	Need Interpreter
- Takes long to compile file which is annoying for development. + Run time is faster. + Translates one time.	+ No time to compile which is good for development. - Run time is slower.

Strongly-Typed vs **Loosely/Weak-Typed**

Strongly typed languages must specify the exact type of data that is being dealt with in the code. When you create a variable you must state whether the variable is a word or a number or a particular type of number (decimal or integer).

With Loosely or Weakly typed languages you do not need to specify the type of data that you are dealing with. Instead you can just go ahead and start using it. You will learn later in the course what variables and data types are.

Strongly-Typed	Loosely-Typed
Java, Scala, C, C++	JavaScript, PHP, Python, Perl,
- May take more code to write. + Less prone to error and easier to catch mistakes.	+ Less code to write. - More prone to errors due to ambiguity.

Enforced Programming Style vs **Non-Enforced Programming Style**

There are a number of different programming styles often referred to as programming paradigms. For the purpose of this Term 1 course we will not go into detail explaining these. However certain programming languages force a developer to use a particular style in which the language should be used.

Object Orientated Programming is a very popular programming style and certain languages will only work through using this. You will not see a language being listed as “Enforced Programming Style” however instead you will see “Object Orientated” language. The majority of languages allow you to program in whatever style you would like.

Object Orientated	Non-Object Orientated
Ruby, C++	JavaScript, PHP, Python

Closer look at some Programming Languages

JavaScript

JavaScript is a high-level, weakly-typed, interpreted client-side scripting language that allows a web designer the ability to insert code into their web page and make it more dynamic. It was created in 1995 and was originally called LiveScript. It is a programming language focused on web development. Certain sources call it the world’s most popular language.

```
document.write(“Hello World!”);
```

PHP

PHP (PHP: Hypertext Preprocessor) is a high-level, weakly-typed, server-side interpreted scripting language. It is used in web development to create dynamic pages and to communicate with backend databases.

```
echo “Hello World!”;
```

C++

C++ is a high-level, strongly-typed, object orientated compiled language. C++ was developed from C in an attempt to improve on certain aspects of that language. C++ is used extensively to write video game engines, graphics applications among many others.

```
#include <iostream>
int main()
{
      std::cout << “Hello World!\n”;
}
```

Java

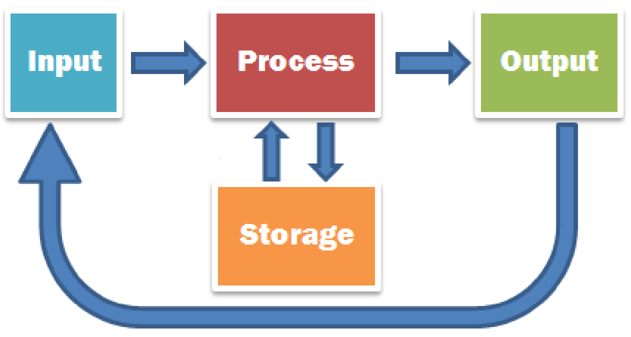
Java is a high-level, strongly typed, compiled language that is widely used to create internet applications and other software programs. It is also used to create Android applications hence its popularity in recent years.

```
public class Hello {
       public static void main(String []args) {
              System.out.println(“Hello World”);
       }
}
```

Input, Processing, Output, Storage

IPOS describes the general structure of a computer information processing program. For software engineers it essentially details the steps of how a user would interact and receive information from a piece of software.

To demonstrate the IPOS process I’m going to use a person writing a paper on Microsoft Word as an example.



Feedback

- **Input:** Any data from an external source that is going to be embedded into a system for some type of use. A variety of sources are used to input such as keyboard, mouse, microphone, buttons, input boxes, data picker etc.
    - John uses his computer's mouse to open MS Word and begins typing words with the keyboard. He alters the appearance of the document by pressing on screen buttons in the application.
- **Processing:** This takes place in the inner workings of the computer's CPU. It is the act of receiving inputted data and converting it to something useable.
    - Every time John types a key, MS Word determines what is pressed and deals with it. If the RETURN buttons is pressed the application moves onto the next line. If highlighted text is made italicized it is done so.
- **Output:** Is the result of the process. What you get after your inputted data is converted to "something useable". Output generally deals with any data that is leaving the system. Monitors and speakers are examples of hardware used by
    - The output in John's case is everything that appears on the screen. His new paper MS document is the combination of all the output.
- **Storage:** Data needs somewhere to go and is often kept on a storage medium such as a hard drive or disk. If data doesn't need to be output immediately after processing it may be saved for later use. Storage mediums tend to blur the lines between the IPOS process when handling data because if data is saved on a USB drive for instance it becomes an output. However if it is stored locally or in the cloud and is required for further processing then it becomes an input. This situation is referred to as **feedback**.
    - Periodically MS Word will temporarily save John's document as a safety net. When John is finished he may save the document to his USB drive.

- **Feedback:** When data is processed and outputted it may sometimes be fed back into the system. This cyclical nature produces a looping structure. There are 2 types of feedback loops:
    - Open Loop – If no user interaction is required when data is fed back into the system this is an open loop.
    - Closed Loop – If a user is prompted to do something when data is fed back into the system this is a closed loop.
  - In John's case an open feedback loop is created because every time he types a key, processing occurs which shows the result on screen and then allows John to type and press more keys to create more outputs.