

Introduction to Programming

Software Development 1 (F27SA)

Michael Lones

Week 1, lecture 2

Today's Lecture

- What is programming?
- Introduction to programming languages
- Introduction to software development in Java

What is Programming?

In a nutshell, it's a way of telling a computer **how** to achieve something useful

- By giving the computer a series of instructions
 - Written in a precise and detailed way
 - In a particular programming language
- ▼ Remember that computers are stupid; they need to be told exactly what to do and when to do it

What is a Computer?

A computer is a machine for processing instructions and thereby doing useful things

- Early computers were designed to do only one thing, e.g. Turing's computer for breaking WW2 codes
- Modern computers are general-purpose, meaning they can do anything, so long as you know how to write the appropriate program

What is a Computer?

Computers are basically large digital circuits, broken up into several parts, including:

- A central processing unit (CPU) which reads through a program and executes each instruction in turn
- Memory, which (amongst other things) stores the intermediate results of computations carried out whilst executing programs

You'll find out more about this when you study Introduction to Computer Systems in Semester 2.

Programming Languages

When programming, instructions have to be written in a programming language

- There are 1000s of these; people love inventing new programming languages.
- But there's usually less than 10 in widespread use at any particular time.
- Like human languages, these differ in their syntax and semantics, i.e. the “words” they use, the structure of “sentences”, and the way in which sentences are assembled together.

Programming Languages

There are various ways of classifying them

- Generations:
 - **1st**: Machine languages, punched cards, switches
 - **2nd**: Assembly languages, easier for humans to read and write, but very low-level
 - **3rd/4th**: High-level languages, designed with programmers in mind, includes Java, Python etc.
 - **5th**: Aim to make programming more accessible, tend to be inefficient, still a work in progress

Programming Languages

There are various ways of classifying them

- Generations:

- **1st**: Machine languages, punched cards, switches

- **2nd**: Assembly languages, easier for humans to read and write, but very low-level

CS Y2

- **3rd/4th**: High-level languages, designed with programmers in mind, includes Java, Python etc.

**First
year**

- **5th**: Aim to make programming more accessible, tend to be inefficient, still a work in progress

CS Y2

Programming Languages

There are various ways of classifying them

- Applications:
 - **Industrial-strength:** These are widely used in industry, well-defined standards, produce programs that are relatively fast to execute, and scale well. Well known examples are C++ and Java.
 - **Scripting languages:** These are widely used in the management of computer systems, and for writing smaller programs. Reduce development time, but are less efficient and scalable. Includes Python and PHP.

Programming Languages

There are various ways of classifying them

- Applications:

- **Industrial-strength:** These are widely used in industry, well-defined standards, produce programs that are relatively fast to execute, and scale well. Well known examples are C++ and Java.

SD1-3

- **Scripting languages:** These are widely used in the management of computer systems, and for writing smaller programs. Reduce development time, but are less efficient and scalable. Includes Python and PHP.

These are used in various courses from Year 2 onwards

Programming Languages

There are various ways of classifying them

- Paradigms:
 - **Procedural**: This is the first style of programming that most people come across. Includes C and BASIC.
 - **Object-oriented**: This make it easier to write larger programs that are more readable. Includes C++ and Java.
 - **Multi-paradigm**: Languages such as Python and Ruby allow you to write programs in a mixture of styles, including procedural, object-oriented, and others.

Programming Languages

There are various ways of classifying them

- Paradigms:

The first half of SD1

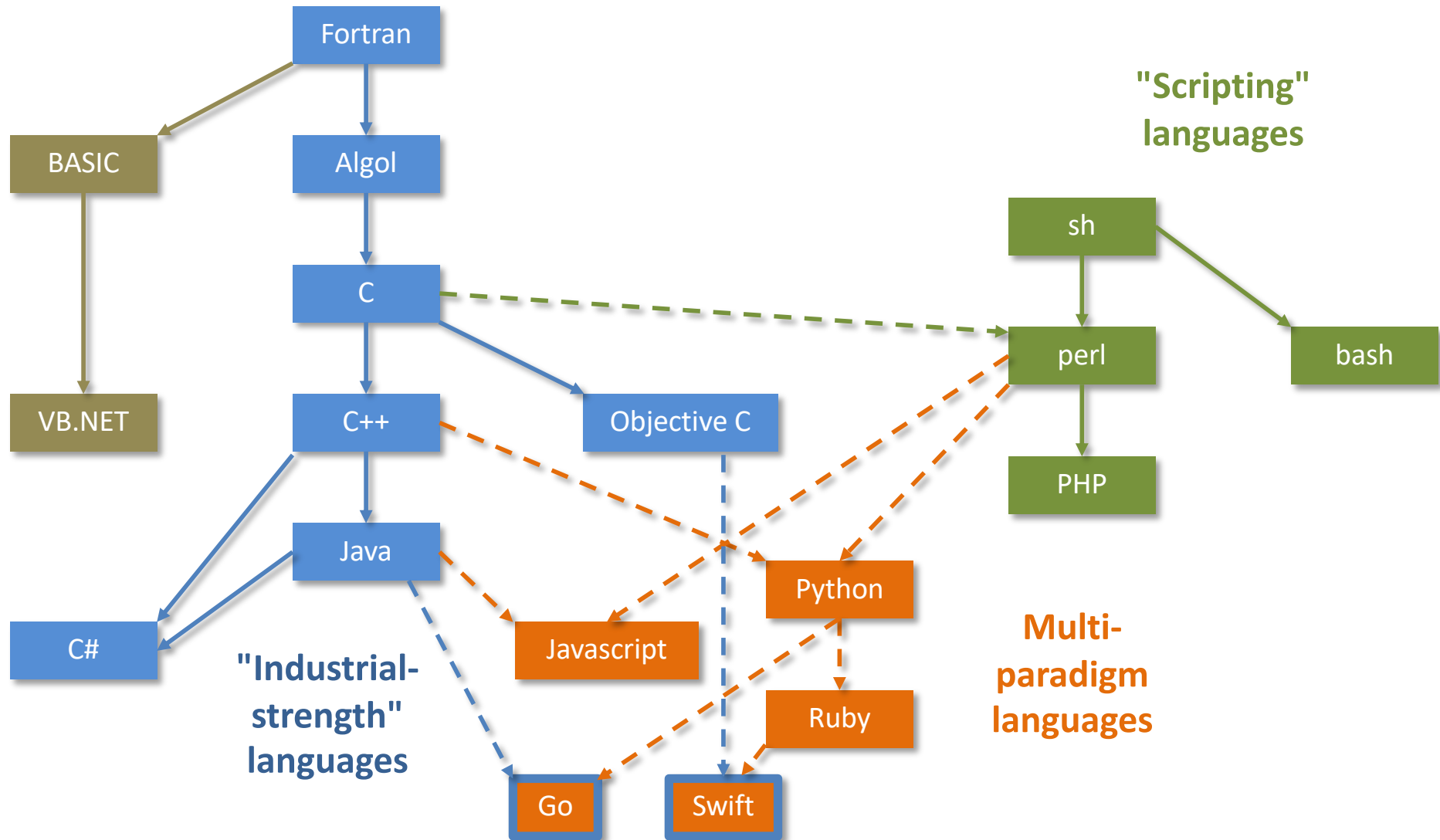
- **Procedural:** This is the first style of programming that most people come across. Includes C and BASIC.

- **Object-oriented:** This make it easier to write larger programs that are more readable. Includes C++ and Java.

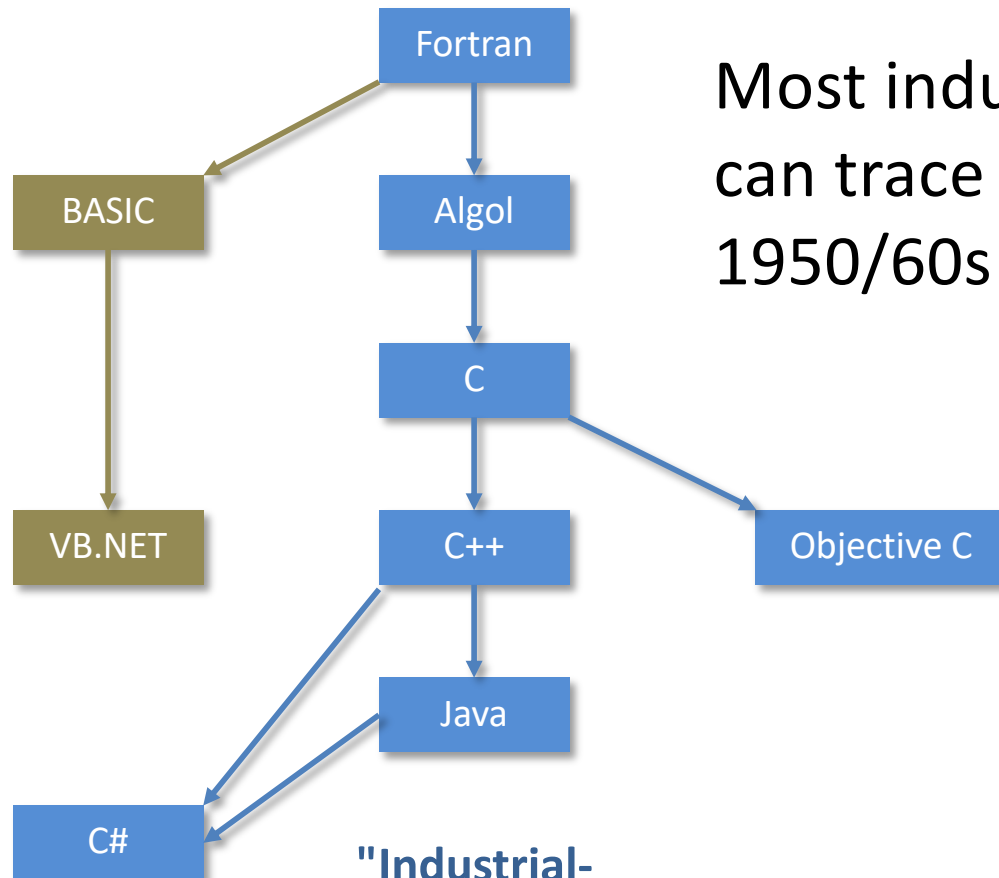
The second half of SD1, and in SD2

- **Multi-paradigm:** Languages such as Python and Ruby allow you to write programs in a mixture of styles, including procedural, object-oriented, and others.

Programming Languages



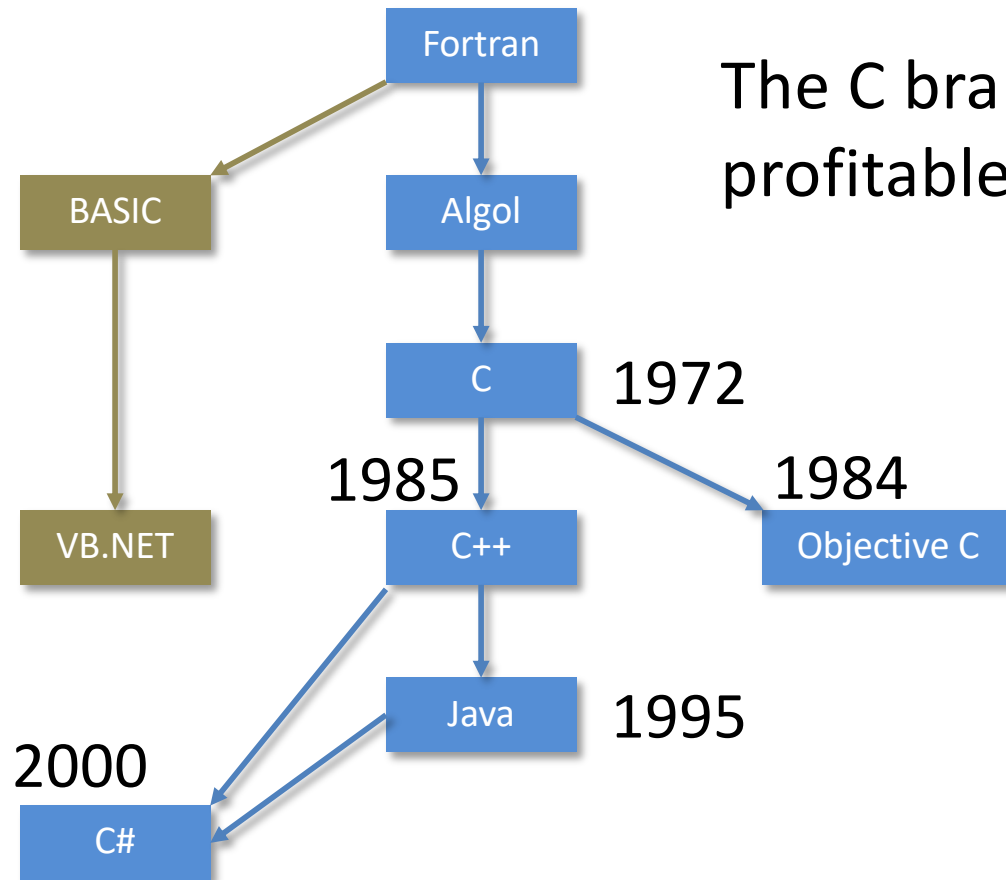
Programming Languages



Most industrial-strength languages can trace their roots directly to the 1950/60s languages Fortran and Algol

"Industrial-strength" languages

Programming Languages



The C branch has been the most profitable source of new languages

Programming Languages

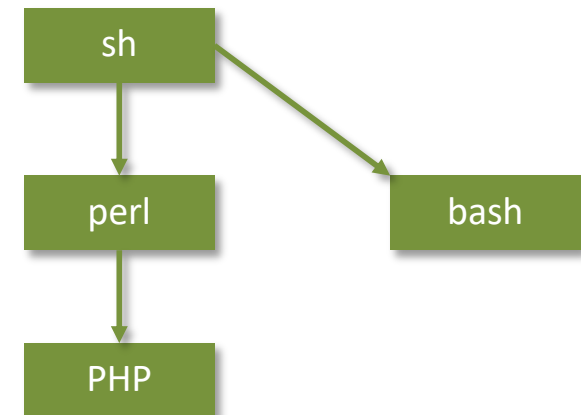
Scripting languages trace their roots to Unix command line languages designed for scripting computer management tasks

- **sh** = Unix shell
- **bash** = Bourne-again shell

Later scripting languages broadened the scope of programming activities

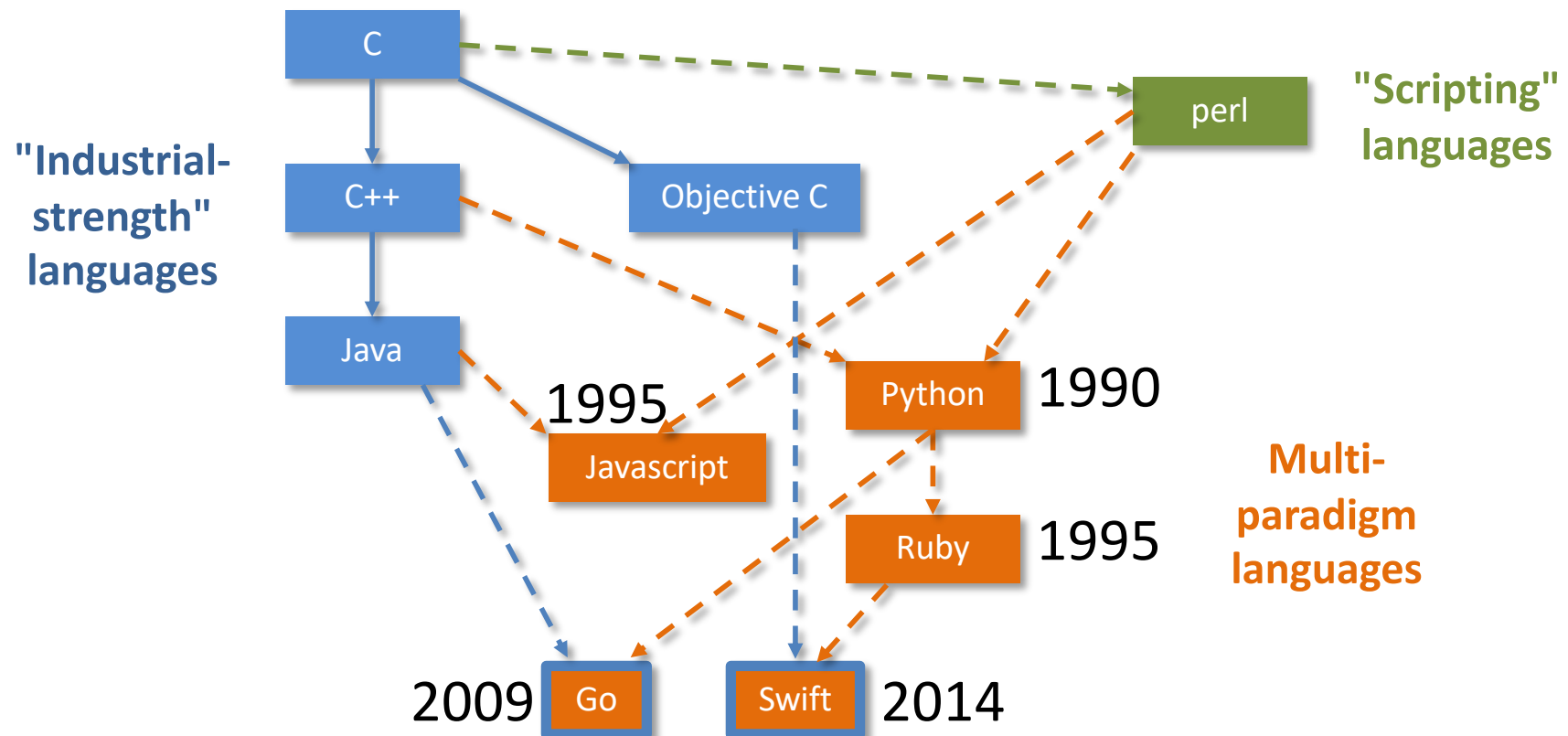
- **perl** became popular in the bioinformatics community
- **php** is used for web scripting

"Scripting"
languages



Programming Languages

Programming languages have increasingly complex roots, and often combine features of many existing languages



Hello Worlds

A program to print “hello world” to the screen.

In Python:

```
print("Hello World")
```

You can see why scripting languages like python are popular with programmers. They let you write programs very quickly.

Hello Worlds

A program to print “hello world” to the screen.

In C:

```
#include <stdio.h>

int main(void) {
    printf("Hello World\n");
}
```

Industrial-strength languages, such as C, involve writing a lot more **boiler-plate** code. This involves more typing, but helps to keep code structured and readable in larger programs.

Hello Worlds

A program to print “hello world” to the screen.

In Java:

```
public class HelloWorld {  
    public static void main(String[] a) {  
        System.out.println("Hello World");  
    }  
}
```

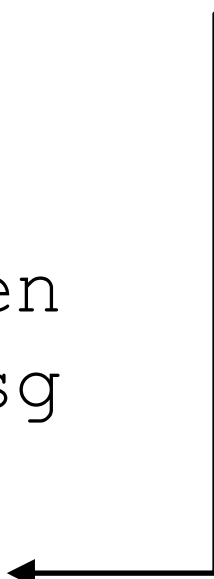
In Java, for example, you always have to make a **class definition** and declare the **main method**, which is where execution begins.

Hello Worlds

A program to print “hello world” to the screen.

In Assembly Language:

```
section .text
global _start
_start:
    mov edx, len
    mov ecx, msg
    mov ebx, 1
    mov eax, 4
    int 0x80
    mov eax, 1
    int 0x80
section .data
msg db 'Hello
      World', 0xa
len equ $ - msg
```



Things could be much worse than Java!

Programming Languages

Programs are written in many languages, yet a computer can only execute **machine language**

- This means that a program must be translated into machine language prior to execution:

```
public String count(int to)
{
    int i=0;
    String output;
    while(i < to) {
        output += i;
        i++;
    }
    return output;
}
```



```
01101010001010
10101010101110
10110010101011
11110010101110
11000111110010
01011110001010
11110010101010
00010101111001
11001011101011
```

Compilers and Interpreters

This process of turning a high-level program into a machine language program uses either:

- **An interpreter.** When the program is executed, it is translated into machine code on the fly. Languages such as Python and Javascript are always treated in this way, and are known as *interpreted languages*.
- This approach is flexible, but relatively slow, since the code has to be translated each time it is run.

Compilers and Interpreters

This process of turning a high-level program into a machine language program uses either:

- **A compiler.** Before the program is run, it is fully translated into machine code. The translated program is then stored as an executable file, e.g. a .exe file under Windows.
- This approach is less flexible, and often more complicated, but results in much faster programs.

Compilers and Interpreters

It is also possible to combine these approaches into something called a **just-in-time compiler**

- This dynamically compiles sections of code that are repeatedly executed, combining the flexibility of an interpreter with the speed advantage of a compiler.
- It is used in modern languages such as Java and C#

Any Questions?

Java

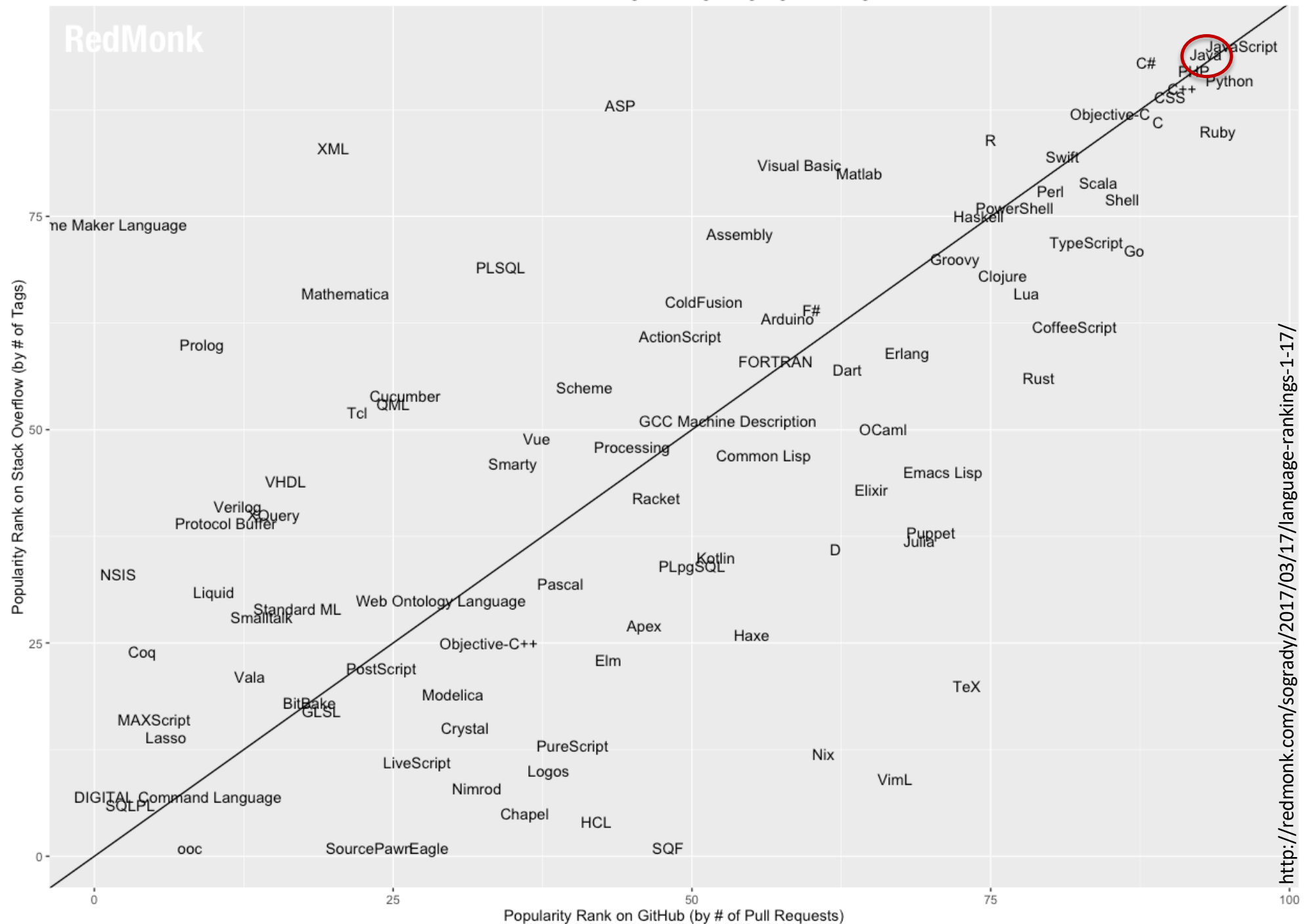
In this course, we use Java

- It is a widely used programming language, particularly within the software industry.
- It is a high-level language that supports both procedural and object-oriented programming.
- It is well documented, and there are lots of resources available on the internet for novice programmers.
- It is platform agnostic, meaning that the same programs will run on Windows, Linux and Mac OS.

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings.

Jun 2017	Jun 2016	Change	Programming Language	Ratings	Change
1	1		Java	14.493%	-6.30%
2	2		C	6.848%	-5.53%
3	3		C++	5.723%	-0.48%
4	4		Python	4.333%	+0.43%
5	5		C#	3.530%	-0.26%
6	9	⬆	Visual Basic .NET	3.111%	+0.76%
7	7		JavaScript	3.025%	+0.44%
8	6	⬇	PHP	2.774%	-0.45%
9	8	⬇	Perl	2.309%	-0.09%
10	12	⬆	Assembly language	2.252%	+0.13%
11	10	⬇	Ruby	2.222%	-0.11%
12	14	⬆	Swift	2.209%	+0.38%
13	13		Delphi/Object Pascal	2.158%	+0.22%
14	16	⬆	R	2.150%	+0.61%
15	48	⬆	Go	2.044%	+1.83%

RedMonk Q117 Programming Language Rankings



Java Development Kit (JDK)

This contains all the programs you need to write and run Java programs. Most important are:

- **java**, a program that runs Java programs.
- **javac**, the Java compiler, which translates a Java code file into something called a class file.

These are both command line programs. You will get to try them out in the first lab session. If you want to use them on your own computer, you can download from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Java Virtual Machine (JVM)

Java is weird in that Java code is first compiled into **Java byte code**, and then into machine code

- This is why, when you run `javac` on your `.java` code files, it generates `.class` files.
- The `java` program then executes these `.class` files using a Java Virtual Machine, which translates them into machine code using just-in-time compilation.

But you don't need to know about this in SD1, so don't worry if it sounds like gobbledegook at the moment!

Java Programming in Practice

To run a Java program, you need to:

1. Write the program's source code
2. Compile the source code using **javac**
3. Run the compiled code using **java**

MyClass.java

```
public class MyClass {  
    public void main(...) {  
        // my code  
    }  
}
```

javac
MyClass
.java



MyClass.class

```
0: iconst_2 1:  
istore_1 2: iload_1  
3: sipush 1000 6:  
if_icmpge 44 9:  
iconst_2 10: ...
```

java
MyClass



Welcome to my
program. Please
follow the
instructions.

Java Programming in Practice

1. Write the program's source code
 - You can use any text editor for this
 - e.g. Atom, Sublime (both cross-platform), see Tutorial 1 (on Vision) for more suggestions
 - **Don't use a word processor!**
 - The file must have the extension .java, e.g. MyClass.java

Java Programming in Practice

2. Compile the source code using **javac**

- You do this from the command line, known as *Terminal* on Linux and Mac OS, and *Command Prompt* on Windows
- First navigate to where the source code is stored:
`cd /Users/ml355`
- Then run: `javac MyClass.java`
- This will generate one or more **class files**, which contain the compiled code

Java Programming in Practice

3. Run the code using **java**

- Again, run this from the command line, in the directory where the class files are stored
- You need to specify the class name as the argument to java, e.g. `java MyClass`
- This class must have a `main` method
- In this half of the course, each program will consist of a single class with a main method.

My First Java Program

This program does nothing when run:

```
public class MyClass {  
    public static void main(String[] args) {  
  
    }  
}
```

MyClass.java

However, it does compile. It is in fact the least amount of code required for a program to compile.

My First Java Program

This is the program's (unimaginative) name

```
public class MyClass {  
    public static void main(String[] args) {  
  
    }  
}
```

MyClass.java

My First Java Program

This is the program's (unimaginative) name

```
public class MyClass {
```

MyClass.java

```
    public static void main(String[] args) {
```

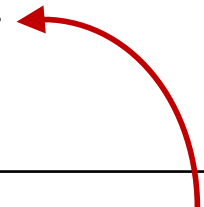
**This is the main method, where execution of a
program always begins**

```
    }
```

```
}
```

My First Java Program

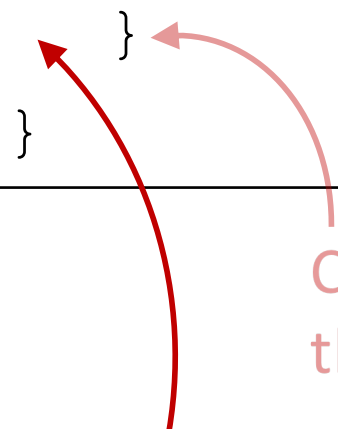
```
public class MyClass {  
    public static void main(String[] args) {  
  
    }  
}
```

MyClass.java

A pair of curly braces indicate a **block** of code. There are two pairs in this example: the class block and the main block.

My First Java Program

```
public class MyClass {  
    public static void main(String[] args) {  
  
    }  
}
```



MyClass.java

Curly braces indicate a **block** of code. There are two in this example: the class block and the main block.

Tabbed indents show hierarchy. Here, the indentation of the main block shows it is inside the class block.

My First Java Program

```
public class MyClass {
```

MyClass.java

```
    public static void main(String[] args) {
```

```
    }
```

```
}
```

These indicate the visibility of this part of the program to other parts of the program. You'll learn about this in Part 2 of SD1.

My First Java Program

```
public class MyClass {  
    public static void main(String[] args) {  
    }  
}
```

MyClass.java

Classes are the basis of object-oriented programming.
You'll have to wait until Part 2 to find out more. For
now, know that every program needs at least one class.

Java Hello World

Let's add something to the main method:

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

MyClass.java

```
$ javac MyClass.java  
$ java MyClass  
Hello World
```

Terminal

Java Hello World

Now let's remove something:

MyClass.java

Terminal

Errors

Programming languages are intolerant to errors, even small ones, and will tell you about them

- **Compile-time errors** are generated by `javac` when your program is not syntactically correct
- **Runtime errors** (known as exceptions) are generated by `java` when your program has semantic errors
- Read the error message; they are often informative, though unfortunately not always!

And a Little More

```
public class MyClass {  
    public static void main(String[] args) {  
        // count from 1 up to 20  
        for(int i=1; i<=20; i++) {  
            System.out.print(i+" ");  
        }  
    }  
}
```

```
$ javac MyClass.java
```


Terminal

```
$ java MyClass
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Comments

```
public class MyClass {  
    public static void main(String[] args) {  
        // count from 1 up to 20  
        for(int i=1; i<=20; i++) {  
            System.out.print(i+" ");  
        }  
    }  
}
```



This is a
comment.
These make
code more
readable, so
use them!

```
$ javac MyClass.java
```

```
$ java MyClass
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Terminal

Comments

```
public class MyClass {  
    public static void main(String[] args) {  
        /* And this is a multi-line comment,  
        which is handy when you want to write  
        something that doesn't fit conveniently  
        onto one line! */  
  
        // count from 1 up to 20  
        for(int i=1; i<=20; i++) {  
            System.out.print(i+" ");  
        }  
    }  
}
```


Readability

It is important to keep your code readable. This makes it easier to maintain and to spot errors.

- Always indent your code correctly. This makes it obvious how the code is structured
- Add comments whenever it isn't immediately obvious what a section of code does
- Give things meaningful names (i.e. not MyClass)



You'll also get much more useful feedback (and more marks!) if we find your code easy to read

Historical Notes

It's worth being aware that Java has changed a lot since it was first released in 1995.

- There's lots of old code on the internet and in old books, some of which is quite dated.
- Anything before Java 5 (also known as Java 1.5), released in 2004, is considered prehistoric.
- It's hard to keep up with all the new language features. Anything after Java 8 is confusing to us old-timers!

Quiz

Which of these does **not** describe Java?

- A. A language for writing large software systems
- B. It makes use of a compiler
- C. It is a descendent of the C language
- D. It is a relatively slow language to execute
- E. It is a 3rd/4th generation language

Quiz

Which of these is **not** true of Java programs?

- A. There must be a main method
- B. Statements are usually followed by a ";"
- C. Compile-time errors are known as exceptions
- D. A program must contain at least one class
- E. Comments can span multiple lines

Quiz

Assembly language is:

- A. Mostly used for high-level programming
- B. Less readable than machine language
- C. Specific to a particular system architecture
- D. A 1st generation programming language
- E. Easy to explain to elderly relatives

Next

Tutorial 1 (look at this)

- This contains information about how to set up your own computer to develop Java programs

Lab session (come to this)

- Becoming familiar with the computer labs and Linux
- Compiling and running a Java program

Lectures (next week)

- Types, variables, expressions, I/O, if, switch, ?: