Computer Science 433
Programming Languages
The College of Saint Rose
Fall 2014

Topic Notes: Introduction and Overview

Welcome to Programming Languages!

What are PLs, and why study them?

Programming languages are the means by which we, as computer programmers, specify to the computer using a human-writable and human-readable notation, what our programs should do.

You might expect that a course entitled *Programming Languages* would consist of spending the semester learning a whole bunch of new programming languages.

While we will look at several programming languages you have not seen before, we will not study any of them to a depth where you will become an expert programmer in them.

So why are we studying these languages and the concepts behind them?

- Well, it's a required course...
- You will add to and refine the tools in your "programming toolbox."
- You will improve your ability to express ideas as programs.
- You will improve your ability to build *efficient* and *easy-to-maintain* software systems.

There are a *lot* of programming languages out there.

On the web: Programs that print the lyrics to *99 Bottles of Beer* in 1500 different languages and variations (as of August 2012) at
`http://www.99-bottles-of-beer.net/`

Why are there so many languages out there?

Different languages are most appropriate for different types of software systems.

- Business applications
  - often distributed, and increasingly so
  - process data, output reports, control transactions
- Manufacturing and control systems

  - reliability is essential
  - continuous operation

- Entertainment/Games/Web/Mobile applications
  - variety of device requirements
  - portability is a key
- Scientific computing
  - usually computationally intensive
  - increasingly reliant on parallelism
  - computational efficiency is important
- Artificial intelligence and research applications
  - often represent and manipulate symbols instead of numbers
  - can also be computationally intensive
- Systems and network programming
  - operating system and system programs
  - requires efficient non-stop operations
  - sometimes need to operate at a low level (specific machine instructions)

With so many competing goals for different applications, there will not be one language that is best for all purposes.

So a lot of our efforts will be to evaluate programming languages. The evaluation of any language will require that we consider the goals of the language – if it is not as good as some other languages in a particular respect, that might be by design and for good reason!

Key points for evaluation of a programming language include:

- Readability
- Writability
- Portability
- Reliability (of the language, *e.g.*, memory management in C++ *vs.* Java)
- Cost (price, training costs, compilation process)
- Ease of maintenance (of both compilers and source code)

Unfortunately, these goals are at times conflicting:

Readability *vs.* writability Perl is flexible and expressive, hence very writable. But as a result, Perl code can be very difficult to read.

For example:

```
$array[-1]
```

is a quick way to reference the last element in an array, but it is less obvious to the reader (in fact, completely confusing to the Perl non-expert) than the Java equivalent

```
a[a.length-1]
```

Writability *vs.* reliability C and C++ provide flexible use of pointers, enhancing its writability. Unfortunately, this is error-prone, potentially leading to invalid pointers, access to unallocated locations, and null pointer references, and, ultimately, less reliable programs.

Tradeoffs like these will come up throughout the course.