Class 1: Introduction to Programming Languages

SI 413 - Programming Languages and Implementation

Dr. Daniel S. Roche

United States Naval Academy

Fall 2011

Roche (USNA)

SI413 - Class 1

Fall 2011 1 / 10

Phases of Programming

What does programming actually involve?

- Write a program
- Execute the program

Note: an **interpreter** essentially does compilation and execution simultaneously, on-the-fly.

Roche (USNA)

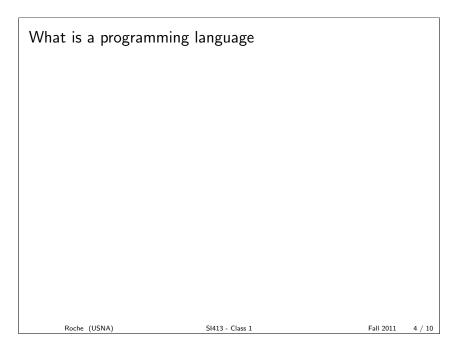
SI413 - Class 1

Fall 2011 2 / 10

Skill outcomes of SI 413

There are other goals on the course policy, but these are some things you will be able to do in a few months:

- (1) Choose a programming language well-suited for a particular task.
- 2 Learn a new programming language quickly and with relative ease.
- Understand the inner workings of compilers and interpreters and become a better user of both.



A multitude of PLs

Check out Wikipedia's list of PLs or the 99 Bottles of Beer site.

- There are a lot of PLs out there.
- Why so many? What features distinguish them?
- How can we talk about programming languages?

Roche (USNA) SI413 - Class 1 Fall 2011 5 / 10

Vocabulary for Programming Languages

Excerpt from the R6RS standard

Scheme is a **statically scoped** and properly **tail-recursive** dialect of the Lisp programming language invented by Guy Lewis Steele Jr. and Gerald Jay Sussman. It was designed to have an exceptionally clear and simple **semantics** and few different ways to form expressions. A wide variety of **programming paradigms**, including **functional**, **imperative**, and message passing styles, find convenient expression in Scheme.

Reading this should give you a good overview of what Scheme is about. But first we have to learn what the terms mean!

Roche (USNA) SI413 - Class 1 Fall 2011 6 / 10

Programming Language Paradigms

Most popular PLs fall into at least one of four classes:

- Imperative/procedural C, Fortran, Cobol
- Object-oriented
 C++, Java, Smalltalk
- Scripting Perl, PHP, Javascript
- Functional Lisp, Scheme, ML, Haskell

Roche (USNA)

SI413 - Class 1

Fall 2011 7 / 10

Imperative Programming Languages

Consider the following code fragment from C++:

```
int x = 0;
x = 3;
x = x + 1;
```

- Each statement is a command.
- Code specifies actions and a specific ordering.
- Expressions may produce values (these do), but *side effects* are often more important.

Roche (USNA)

SI413 - Class 1

Fall 2011 8 / 10

Functional Programming

Functional programming is *declarative*: the output is a mathematical function of the input.

Emphasizes describing what is computed rather than how.

Key features:

Referential transparency

The value of an expression does not depend on its context.

Functions are first-class

Functions can be passed as arguments, created on-the-fly, and returned from other functions. Functions are data!

Types are first-class

This is not true in Scheme (there are no types), but is in other functional PLs.

Other common properties of functional PLs

- Garbage collection
- Built-in list types and operators
- Interpreters rather than compilers
- Extensive polymporphism (again, not applicable to Scheme)

Roche (USNA) SI413 - Class 1 Fall 2011 10 / 10