Why programming languages?

A short polemic
on the value of a diverse linguistic diet

Common complaints

- "Everyone uses C++ or Java."
- "The languages we're learning are impractical and only good for ivory-tower academics with no connection to the real world."
- "Why don't you teach us how to build real programs?"
- "The only thing these languages are good for is building toys!"
- "I need to build enterprise-class solutions for information technology professionals! Now!!!!"

"Everyone uses C++, or Java."

- ...or, uh, in two years, C#.
- Eight years ago *nobody* used Java.
- And, well, twenty years ago people wrote a lot of COBOL and FORTRAN. (They still do.)
- Five years ago, Apple's next-generation OS was going to have its primary APIs in Objective-C.
- Java will have type-safe generics (bounded polymorphism) in a couple of years. So will C#.
- So what language do you want us to teach you now?
    - i.e., How long do you want to be employed?

"impractical", "academic" features

- **People used to believe that all these were impractical:**
  - garbage collection
  - inheritance/dynamic dispatch
  - typesafe generic polymorphism
  - pure object-oriented design
  - exceptions
- **All these features are in widely used languages today, or will be soon.**
- **So what language features do you want us *not* to teach you today?** (Lambdas? Python and Ruby have them.)

"I want to build real programs!"

- **What makes development of "real" programs hard?**
- **The inherent difficulty of building real programs has little to do with**
  - Wrestling with the slow write-compile-build-test development cycle of C, C++, or Java
  - Learning the bloated, complex APIs and IDEs that professional programmers put up with every day
- **Programming is inherently hard (partly) because of the *thought required to write correct programs*.**
- **Languages are excellent tools to teach different ways of thinking about problems.**
- **If you prefer to memorize API calls, then you're in the wrong place.**

"The only thing this stuff is good for is building toys!"

- **That's exactly what the suits at Xerox PARC said in the 70's when their researchers invented Ethernet, the graphical user interface, and object-oriented programming.**
- **Also, do not confuse libraries with languages.**
  - **Example:**
    - Perl used to have the best libraries for string munging.
    - In most other respects, Perl is a horrible language.
    - Today, when Python and Ruby (far nicer languages) both have Perl-like regular expression packages, a lot of people continue to cling irrationally to Perl.
    - They suffer.

"I need to build enterprise-class solutions for information technology professionals! Now!!!!"

- Do you ask your math teacher to teach you how to use Microsoft Excel so you can do "enterprise-class accounting solutions" instead of calculus?
- Do you ask your English teacher to teach you to write press releases and ad copy instead of essays?
- What makes Computer Science different?

A more positive take

- **Every language is a window into a way of thinking**
  - Knowing more languages helps you think about the organization of a system in different ways.
  - Languages are beautiful and interesting artifacts in their own right.

A more positive take

- **Also, on more concrete, direct, practical terms, broad understanding of languages will help you to:**
  - Cope with evolution of programming practice
  - Design/implement languages embedded within larger applications
  - Evaluate the suitability to task of competing programming technologies

Evolution of programming practice

- **Someday, the languages you use today are going to be obsolete.**
- **The features that new languages incorporate are almost always old features from other languages.**
- **Learning the concepts that form the foundation of all languages will enable you to easily pick up next year's language.**
- **Or this year's language...**
  - (By the time you get out of this course, you should have learned enough to teach yourself Java or C# easily.)

Embedded and domain-specific languages

- **"Every program attempts to expand until it can read mail. Those programs which cannot so expand are replaced by ones which can."**

-- Jamie Zawinski*

* Key developer: XEmacs and Netscape Navigator; owner/programmer/bartender, DNA Lounge nightclub, San Francisco)

Embedded and domain-specific languages

- **Likewise, every successful application grows until it becomes a *domain-specific programming environment*...**
  - office apps (MSOffice/VBScript),
  - web browsers (JavaScript) and servers (servlets, PHP, ASP, etc.)
  - game engines (UnrealScript/QuakeC/...),
  - desktop environments (AppleScript, KDE/DCOP),
  - graphics and multimedia (the GIMP, Shockwave/Flash),
  - and of course text editors (Emacs)...
- **...and those applications which cannot so grow are replaced by those which can.**
- **When it comes time for *you* to develop a "real" application, what are you going to do?**

Evaluating competing programming technologies

- **There's a lot of snake oil in programming tools.**
- **Choosing the right tools can make a huge difference in programmer productivity.**
- **When it comes time to build a large project, how can you evaluate programming technologies?**
  - Vendors' claims?
  - The ill-informed, fad-obsessed technology press?
  - The opinion of your friends?
- **A broad understanding of languages is crucial to enable you to judge for yourself.**