Programming Languages Pragmatics

WHY STUDY PROGRAMMING LANGUAGES?

DR. ERIC CHOU

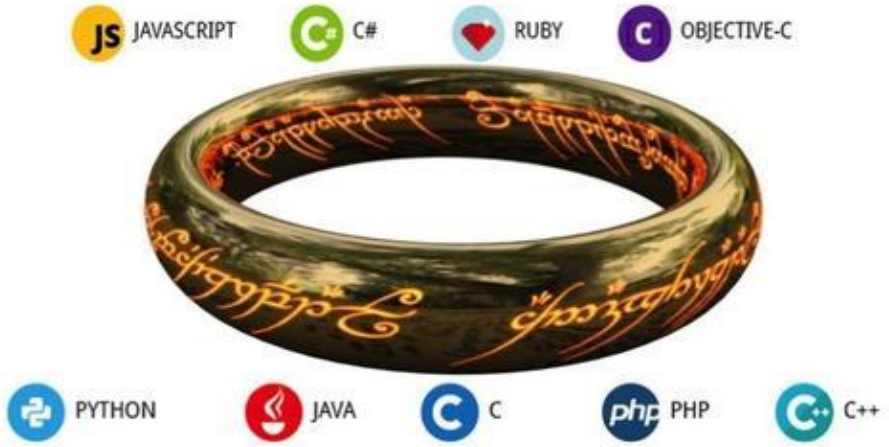
IEEE SENIOR MEMBER

Help you choose a language

1. C vs. Modula-3 vs. C++ for systems programming
2. Python vs. Fortran vs. APL vs. Ada for numerical computations
3. C/C++ vs. Ada vs. Modula-2 for embedded systems
4. Common Lisp vs. Scheme vs. ML for symbolic data manipulation
5. Java vs. C#(.Net) for networked PC programs

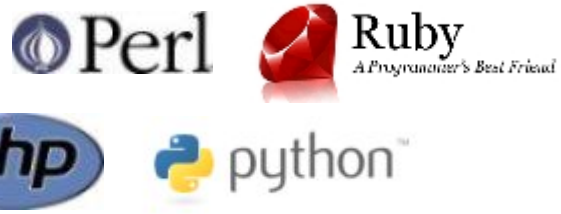
Modern Real-World Languages

Markup/Data Languages

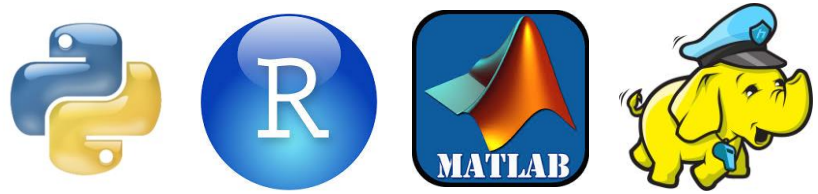


Network (Web-server) Languages

Mobile (App) Languages



Database Languages



Number and Data Processing Languages

Desktop (.exe) Languages

Hardware Description Languages

Electronics Languages

Make it easier to learn new languages

Some languages are similar; easy to walk down family tree

concepts have even more similarity; if you think in terms of iteration, recursion, abstraction (for example), you will find it easier to assimilate the syntax and semantic details of a new language than if you try to pick it up in a vacuum.

Think of an analogy to human languages: good grasp of grammar makes it easier to pick up new languages (at least Indo-European). East Asian Languages need to pick up Kanji Characters (CJKV, Sino-Tibetan and Altaic Languages)

Help you make better use of whatever language you use (I)

- understand obscure features:
  - In C, help you understand unions, arrays & pointers, separate compilation, varargs, catch and throw
  - In Common Lisp, help you understand first-class functions/closures, streams, catch and throw, symbol internals

Help you make better use of whatever language you use (II)

- understand implementation costs: choose between alternative ways of doing things, based on knowledge of what will be done underneath:
  - use simple arithmetic equal (use x*x instead of x**2)
  - use C pointers or Pascal "with" statement to factor address calculations
  - avoid call by value with large data items in Pascal
  - avoid the use of call by name in Algol 60
  - choose between computation and table lookup (e.g. for cardinality operator in C or C++)

Help you make better use of whatever language you use (III)

- figure out how to do things in languages that don't support them explicitly:
  - lack of suitable control structures in Fortran
  - use comments and programmer discipline for control structures
  - lack of recursion in Fortran, CSP, etc
  - write a recursive algorithm then use mechanical recursion elimination (even for things that aren't quite tail recursive)

Help you make better use of whatever language you use (IV)

- figure out how to do things in languages that don't support them explicitly:
  - lack of named constants and enumerations in Fortran
  - use variables that are initialized once, then never changed
  - lack of modules in C and Pascal use comments and programmer discipline
  - lack of iterators in just about everything fake them with (member?) functions

Study of Programming Languages

- What is available?
- What is not available?
- What is good?
- What is bad?
- What is the use?