PLD

KG

Course Title: Concepts, Techniques, and Models of Programming Languages
Credits: (3 credits)

Objective

This course explores the design space of programming languages via an in-depth study of two programming languages, one functional (e.g. Haskell/Scheme), and one object-oriented (e.g. C++/Java); it focuses on idiomatic uses of each language and on features characteristic for each language. This course is about how programs written in high-level programming languages are executed. This includes topics such as parsing, internal program representations, type checking, and interpreting. Students will learn about implementation approaches of common abstraction mechanisms and modern programming language constructs. The course explores the use of the functional programming approach to design and implement programs. And also, contrastingly, the main features of modern object-oriented languages.

Aims and Objectives

This course introduces the basic concepts that serve as a basis for understanding the design space of programming languages in terms of their constructs, paradigms, evaluation criteria and language implementation issues. It introduces concepts from imperative, object-oriented, functional, logic-based, constraint-based and concurrent programming. These concepts are illustrated by examples from varieties of languages such as C/C++, Java/C#, Smalltalk, Scheme, Haskell, Prolog, Erlang, Julia, Rust. The module also introduces various implementation issues, such as static and dynamic semantics, abstract machines, type inferencing, etc. This course uses a uniform notation to specify progressively the programming models of most programming languages. These models range from declarative models that cover functional programming and dataflow computing to stateful models for imperative and object-oriented programming, as well as concurrent models for message-passing systems and shared state.

Intended for: *core for CSAI, elective for other UG or for PG*

Prerequisites: *FM111+FM121, or equivalent*

Syllabus

Declarative models of computation and declarative programming, relational and functional models. Concurrency, data driven and lazy evaluation. Programming with state, component-based and object-oriented programming. The interplay between concurrency and state. Interpretation and virtual machines. Introduction to semantics of programming languages

Prerequisites: Knowledge of object-oriented programming in Java or C++, and knowledge of discrete mathematics.

Outline

Module 1: Brief Survey of some common languages (eg. C,C++, Python)

Module 2: Background theory (computability, language hierarchy, lambda calculus, rule-based grammars)

Module 3: Block structured languages and sketch/outlines of their implementation (parsing, compiling, code gen)

Module 4: List based languages and their impl

Module 5: Object Oriented Langs & brief outline of impl.

Module 6: Type systems. Info-flow based languages.

Module 7: Functional Languages

Module 8: Logic-based and constraint-based languages

Module 9: Case studies: Java, Rust, Julia, Haskell

Reference material:

1: Concepts of Prog. Langs. Robert Sebesta, 2019

2: Prog. Lang. Concepts. Peter Sestoft, 2012

3: Concepts, Techniques, and Models of Computer Programming. Roy and Haridi, 2004

Chapter 2 Evolution of the Major Programming Languages 37

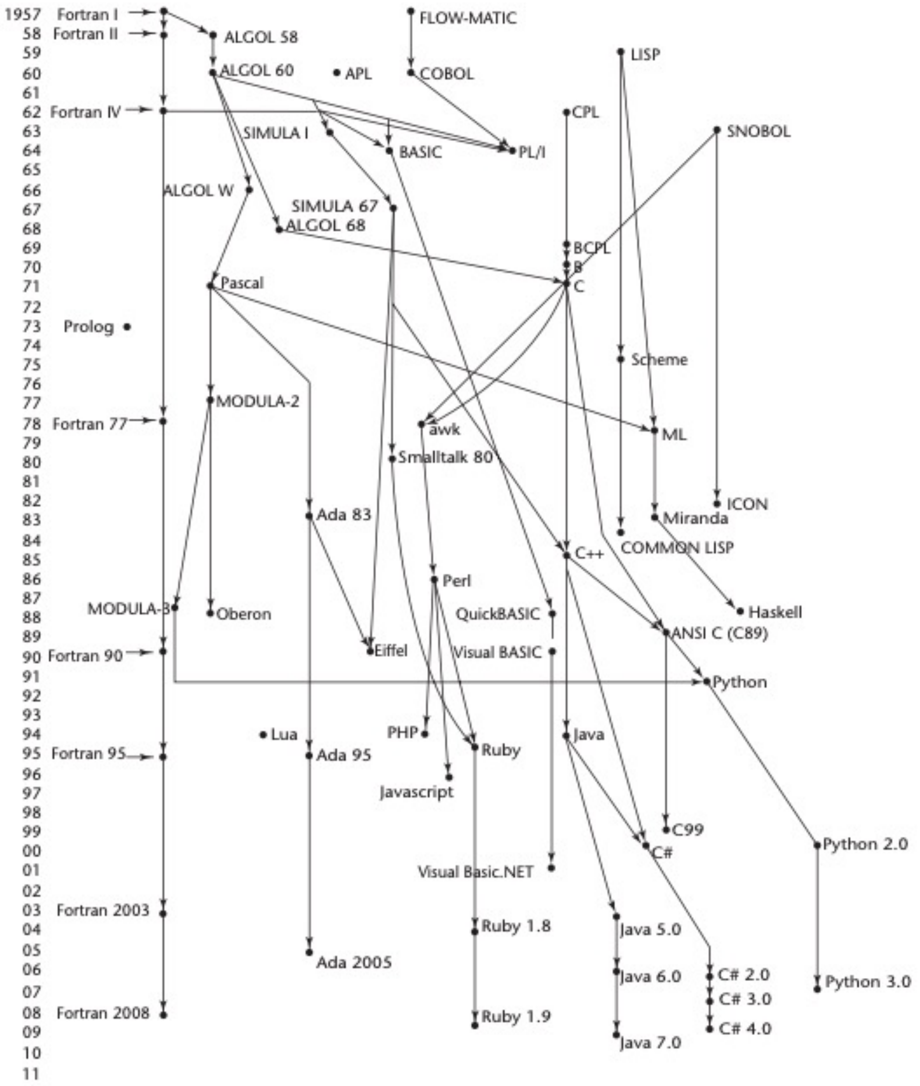


Figure 2.1

Genealogy of common high-level programming languages

sebesta

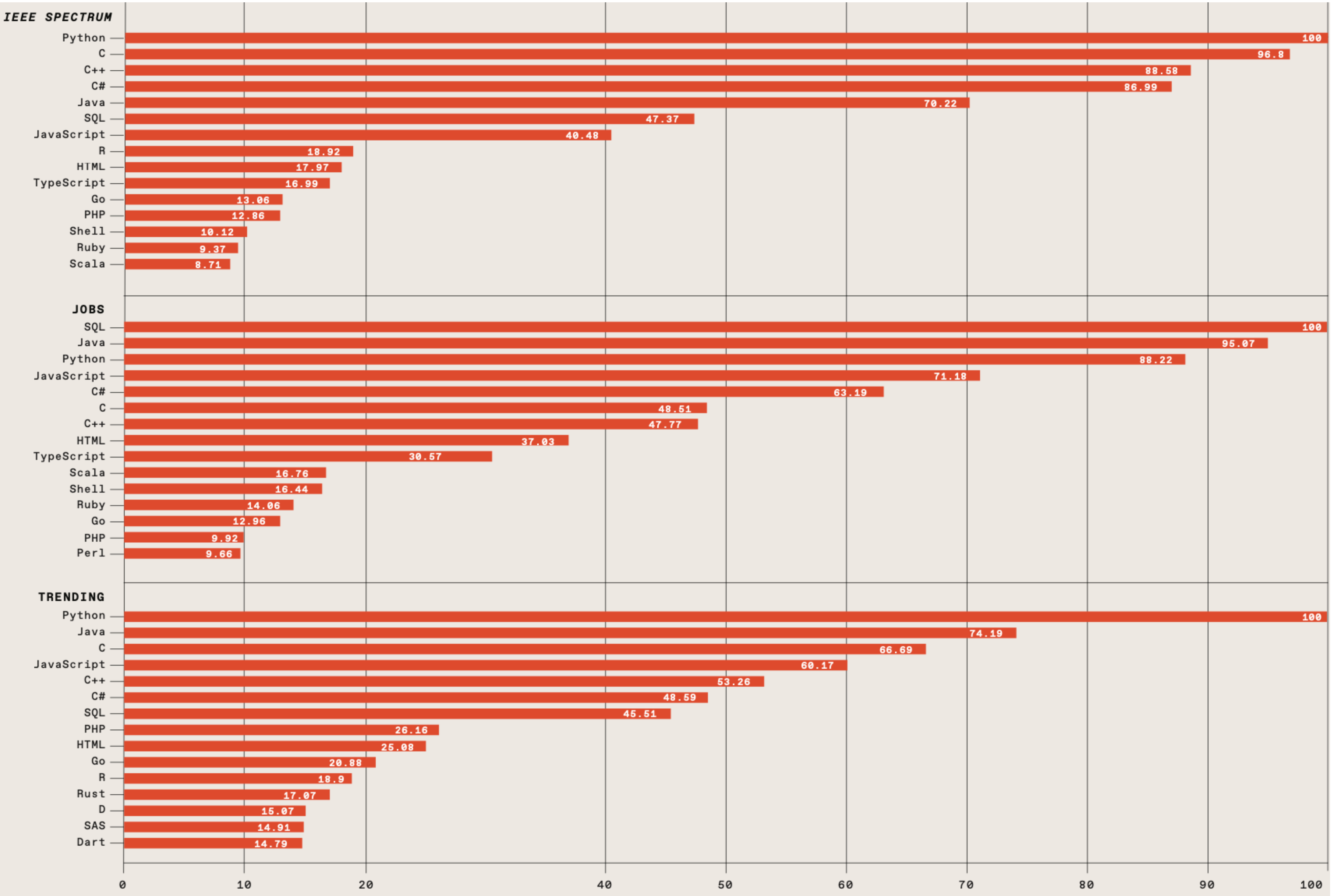
SQL Should Be Your *Second* Language

Welcome to ***IEEE Spectrum's*** annual ranking of the Top Programming Languages. This year we've revamped and streamlined our interactive ranking tool online and made other changes under the hood. But the goal remains the same—to combine metrics from different sources to estimate the relative popularity of different languages. Here we show the results for the highest-ranked languages, but you can find the full list of 57 languages online.

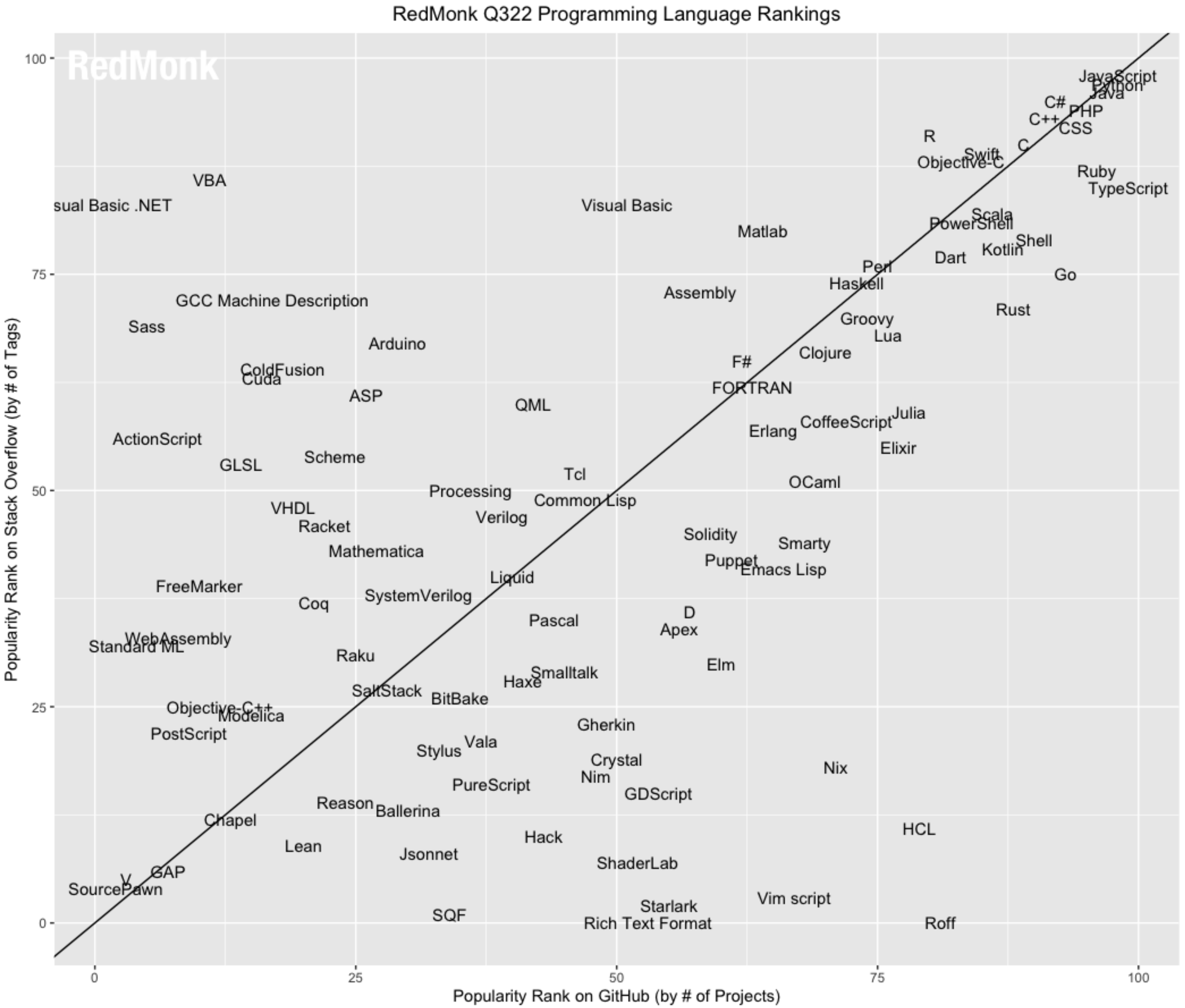
Sources include GitHub, Google, Stack Overflow, Twitter, and IEEE Xplore. The raw data is normalized and weighted according to the different rankings offered. For example, the *Spectrum* ranking is heavily weighted toward metrics that reflect the interests of IEEE members, while Trending puts more weight on forums and social-media metrics.

Not surprisingly, Python and C come out on top, followed by C, C++, C#, and Java. But among these stalwarts there is the rising popularity of SQL, which has become the de facto way to speak with databases. The strength of the SQL signal is not because there are a lot of employers looking for *just* SQL coders, in the way that they advertise for Java experts or C++ developers. They want a given language *plus* SQL.

This is likely because so many applications today involve a front-end or middleware layer talking to a back-end database. So it may not be the most glamorous language, or what you're going to use to implement the next Great Algorithm, but some experience with SQL is a valuable arrow to have in your quiver. ■



1 JavaScript: Dyn types,
runtime typechecking
2 Python: Dyn types, oo
3 Java: oo
4 PHP
5 C#   oo
6 CSS
7 C++ oo
7 TypeScript
9 Ruby
10 C
11 Swift
12 R
12 Objective-C
14 Shell
15 Scala
15 Go
17 PowerShell
17 Kotlin
19 Rust
19 Dart



- SQL: DB (1st order logic)
- Shell: Unix
- R, SAS: statistical langs
- HTML

- Geometry: also 1st order logic

- C# a general-purpose, high-level multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and **component-oriented programming disciplines**
- Java: a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers **write once, run anywhere** (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages.
- Javascript (JS), a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).
- Typescript: **a strict syntactical superset of JavaScript and adds optional static typing to the language**. It is designed for the development of large applications and transpiles to JavaScript. As it is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.
- Go: a statically typed, compiled programming language syntactically similar to C, but with **memory safety**, garbage collection, structural typing and CSP-style concurrency.

wiki

- PHP: a general-purpose scripting language that is especially suited to **server-side web development**, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere. It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. Used by 75% of all the websites whose server-side programming language known.
- Ruby: an **interpreted**, high-level, general-purpose programming language which supports multiple programming paradigms (including procedural, object-oriented, and functional programming) designed with an emphasis on **programming productivity and simplicity**. In Ruby, everything is an object, including primitive data types. Dynamically typed and uses garbage collection and just-in-time compilation.
- Scala: a **strong statically typed** general-purpose programming language that supports both object-oriented programming and functional programming. Designed to be concise, many of Scala's design decisions are aimed to **address criticisms of Java**. Scala source code can be compiled to Java bytecode and run on a Java virtual machine (JVM). Scala can also be compiled to JavaScript to run in a browser, or directly to a native executable. On the JVM Scala provides language interoperability with Java so that libraries written in either language may be referenced directly in Scala or Java code
- Perl: a family of two high-level, general-purpose, interpreted, dynamic programming languages. provide text processing facilities without the arbitrary data-length limits of many contemporary Unix command line tools. Perl 5 gained widespread popularity in the late 1990s as a **CGI scripting language**, in part due to its powerful regular expression and string parsing abilities. In addition to CGI, Perl 5 is used for system administration, network programming, finance, bioinformatics, and other applications, such as for GUIs. Nicknamed "the Swiss Army chainsaw of scripting languages" because of its flexibility and power

wiki

- Rust: a multi-paradigm, high-level, general-purpose programming language. Rust emphasizes performance, type safety, and concurrency. **Rust enforces memory safety**—that is, that all references point to valid memory—without requiring the use of a garbage collector or reference counting present in other memory-safe languages. To simultaneously enforce memory safety and prevent concurrent data races, Rust's "borrow checker" tracks the object lifetime of all references in a program during compilation. Rust is popular for systems programming but also offers high-level features including some functional programming constructs.
- D: a multi-paradigm system programming language that originated as a **re-engineering of C++**. D combines the performance and safety of compiled languages with the expressive power of modern dynamic and functional programming languages. Idiomatic D code is commonly as fast as equivalent C++ code, while also being shorter. The language as a whole is not memory-safe[14] but includes optional attributes designed to guarantee memory safety of either subsets of or the whole program. Type inference, automatic memory management and syntactic sugar for common types allow faster development, while bounds checking and design by contract find bugs earlier at runtime, and a concurrency-aware type system catches bugs at compile time.
- Dart: a programming language **designed for client development** such as for the web and mobile apps; can also be used to build server and desktop applications. It is an object-oriented, class-based, garbage-collected language with C-style syntax. It can compile to either machine code or JavaScript, and supports interfaces, mixins, abstract classes, reified generics and type inference.

wiki