### 1. \*\*Assembly Language\*\*

- \*\*Introduction\*\*: Assembly language is a low-level programming language that is closely related to machine code. It is specific to a computer architecture and is used for writing programs that need to directly manipulate hardware.

- \*\*Features\*\*: Direct hardware manipulation, high performance, minimal abstraction from machine code.

- \*\*Use Cases\*\*: Systems programming, embedded systems, real-time systems, performance-critical applications.

### 2. \*\*FORTRAN (1957)\*\*

- \*\*Introduction\*\*: FORTRAN (FORmula TRANslation) is one of the oldest high-level programming languages, developed by IBM for scientific and engineering calculations.

- \*\*Features\*\*: Strong support for numeric computation and array handling, high performance for computational tasks, portability across systems.

- \*\*Use Cases\*\*: Scientific computing, numerical weather prediction, computational physics, computational chemistry.

### 3. \*\*LISP (1958)\*\*

- \*\*Introduction\*\*: LISP (LISt Processing) is a family of programming languages with a long history in artificial intelligence research.

- \*\*Features\*\*: Powerful symbolic computation capabilities, code-as-data (homoiconicity), macros, dynamic typing.

- \*\*Use Cases\*\*: Artificial intelligence, symbolic computation, academic research, rapid prototyping.

### 4. \*\*COBOL (1959)\*\*

- \*\*Introduction\*\*: COBOL (COmmon Business-Oriented Language) was developed for business, finance, and administrative systems for companies and governments.

- \*\*Features\*\*: Strong data processing capabilities, English-like syntax, extensive use in legacy systems.

- \*\*Use Cases\*\*: Business applications, financial systems, government databases, legacy system maintenance.

### 5. \*\*ALGOL (1960)\*\*

- \*\*Introduction\*\*: ALGOL (ALGOrithmic Language) was designed for scientific computations and introduced many concepts used in later languages.

- \*\*Features\*\*: Block structure, structured programming, lexical scoping, recursion.

- \*\*Use Cases\*\*: Algorithm research, mathematical computations, academic instruction.

### 6. \*\*BASIC (1964)\*\*

- \*\*Introduction\*\*: BASIC (Beginner's All-purpose Symbolic Instruction Code) was created to provide an easy-to-learn language for beginners and non-professionals.

- \*\*Features\*\*: Simple syntax, interactive development, wide platform availability.

- \*\*Use Cases\*\*: Education, early personal computing, hobbyist programming.

### 7. \*\*C (1972)\*\*

- \*\*Introduction\*\*: Developed at Bell Labs, C is a general-purpose programming language that has influenced many other languages.

- \*\*Features\*\*: Low-level memory access, simple and powerful syntax, portability, extensive standard library.

- \*\*Use Cases\*\*: Systems programming, embedded systems, operating systems, game development.

### 8. \*\*Prolog (1972)\*\*

- \*\*Introduction\*\*: Prolog (PROgramming in LOGic) is a logic programming language associated with artificial intelligence and computational linguistics.

- \*\*Features\*\*: Declarative programming paradigm, pattern matching, tree-based data structures, backtracking.

- \*\*Use Cases\*\*: AI research, natural language processing, theorem proving, expert systems.

### 9. \*\*SQL (1974)\*\*

- \*\*Introduction\*\*: SQL (Structured Query Language) is a domain-specific language used in programming and managing relational databases.

- \*\*Features\*\*: Declarative syntax for querying and manipulating data, transactional control, data definition, and manipulation capabilities.

- \*\*Use Cases\*\*: Database management, data analysis, data warehousing, web development.

### 10. \*\*ML (1973)\*\*

- \*\*Introduction\*\*: ML (MetaLanguage) is a general-purpose functional programming language known for its type inference and pattern matching.

- \*\*Features\*\*: Strong static typing, type inference, pattern matching, functional programming paradigms.

- \*\*Use Cases\*\*: Academic research, language development, theorem proving, complex algorithms.

### 11. \*\*Ada (1980)\*\*

- \*\*Introduction\*\*: Ada is a high-level programming language developed for the U.S. Department of Defense for embedded and real-time systems.

- \*\*Features\*\*: Strong typing, modularity, concurrency support, reliability, safety.

- \*\*Use Cases\*\*: Defense systems, avionics, transportation systems, real-time processing.

### 12. \*\*C++ (1983)\*\*

- \*\*Introduction\*\*: C++ is an extension of the C programming language that incorporates object-oriented features.

- \*\*Features\*\*: Object-oriented programming, generic programming, low-level memory manipulation, extensive standard library.

- \*\*Use Cases\*\*: Systems programming, game development, real-time simulation, large-scale software development.

### 13. \*\*Objective-C (1984)\*\*

- \*\*Introduction\*\*: Objective-C is an object-oriented programming language that adds Smalltalk-style messaging to C.

- \*\*Features\*\*: Dynamic runtime, object-oriented features, integration with C and C++ code.

- \*\*Use Cases\*\*: macOS and iOS application development (historically, before Swift).

### 14. \*\*Perl (1987)\*\*

- \*\*Introduction\*\*: Perl is a high-level, interpreted programming language known for its text processing capabilities.

- \*\*Features\*\*: Regular expressions, string manipulation, extensive CPAN library, flexibility.

- \*\*Use Cases\*\*: System administration, web development, network programming, text processing.

### 15. \*\*Python (1991)\*\*

- \*\*Introduction\*\*: Python is a high-level, interpreted programming language known for its readability and wide-ranging application.

- \*\*Features\*\*: Simple and readable syntax, extensive standard library, dynamic typing, support for multiple paradigms (object-oriented, procedural, functional).

- \*\*Use Cases\*\*: Web development, data analysis, artificial intelligence, scientific computing, automation, scripting.

### 16. \*\*Ruby (1995)\*\*

- \*\*Introduction\*\*: Ruby is a dynamic, open-source programming language with a focus on simplicity and productivity.

- \*\*Features\*\*: Object-oriented features, dynamic typing, duck typing, garbage collection.

- \*\*Use Cases\*\*: Web development (Ruby on Rails), scripting, prototyping.

### 17. \*\*Java (1995)\*\*

- \*\*Introduction\*\*: Java is a class-based, object-oriented programming language designed to have as few implementation dependencies as possible.

- \*\*Features\*\*: Platform independence via the JVM, extensive standard library, strong memory management, built-in concurrency support.

- \*\*Use Cases\*\*: Enterprise applications, Android app development, web applications, distributed computing.

### 18. \*\*JavaScript (1995)\*\*

- \*\*Introduction\*\*: JavaScript is a high-level, dynamic programming language primarily used for web development.

- \*\*Features\*\*: Event-driven, prototype-based, functional programming capabilities, asynchronous programming with promises and async/await.

- \*\*Use Cases\*\*: Web development (client-side and server-side), interactive web applications, server-side scripting (Node.js).

### 19. \*\*PHP (1995)\*\*

- \*\*Introduction\*\*: PHP (Hypertext Preprocessor) is a server-side scripting language designed for web development.

- \*\*Features\*\*: Dynamic typing, built-in web development capabilities, extensive library support.

- \*\*Use Cases\*\*: Web development, content management systems (e.g., WordPress), server-side scripting.

### 20. \*\*Swift (2014)\*\*

- \*\*Introduction\*\*: Swift is a powerful and intuitive programming language developed by Apple for iOS, macOS, watchOS, and tvOS app development.

- \*\*Features\*\*: Modern syntax, safety features (e.g., optionals), performance optimizations, interoperability with Objective-C.

- \*\*Use Cases\*\*: iOS and macOS application development, cross-platform mobile development (via frameworks like SwiftUI).

### 21. \*\*Kotlin (2011)\*\*

- \*\*Introduction\*\*: Kotlin is a statically typed programming language developed by JetBrains, fully interoperable with Java.

- \*\*Features\*\*: Concise syntax, null safety, interoperability with Java, support for functional programming.

- \*\*Use Cases\*\*: Android app development, server-side development, cross-platform development (Kotlin Multiplatform).

### 22. \*\*R (1993)\*\*

- \*\*Introduction\*\*: R is a language and environment for statistical computing and graphics.

- \*\*Features\*\*: Extensive statistical and graphical capabilities, strong support for data analysis and visualization, CRAN repository for packages.

- \*\*Use Cases\*\*: Data analysis, statistical modeling, bioinformatics, data visualization.

### 23. \*\*Go (2009)\*\*

- \*\*Introduction\*\*: Go (or Golang) is a statically typed, compiled programming language designed by Google.

- \*\*Features\*\*: Simple syntax, built-in concurrency primitives (goroutines), strong performance, garbage collection.

- \*\*Use Cases\*\*: System programming, cloud services, distributed systems, web servers.

### 24. \*\*Rust (2010)\*\*

- \*\*Introduction\*\*: Rust is a systems programming language focused on safety and performance, particularly safe concurrency.

- \*\*Features\*\*: Ownership model for memory safety, zero-cost abstractions, modern syntax, concurrency support.

- \*\*Use Cases\*\*: Systems programming, performance-critical applications, web assembly, game development.

### 25. \*\*TypeScript (2012)\*\*

- \*\*Introduction\*\*: TypeScript is a superset of JavaScript that adds static typing.

- \*\*Features\*\*: Static type checking, modern JavaScript features, tooling support, compatibility with JavaScript.

- \*\*Use Cases\*\*: Large-scale web applications, front-end development, server-side development (Node.js).

Continuing from the previous list, here are additional programming languages that have played significant roles in the development of computing and technology. This list includes languages that may not be as widely known but are important in various niches and specialized fields.

### 26. \*\*Pascal (1970)\*\*

- \*\*Introduction\*\*: Pascal was developed by Niklaus Wirth as a language suitable for teaching structured programming and data structuring.

- \*\*Features\*\*: Strong typing, structured programming, rich data structuring capabilities.

- \*\*Use Cases\*\*: Education, early software development, embedded systems.

### 27. \*\*Smalltalk (1972)\*\*

- \*\*Introduction\*\*: Smalltalk is an object-oriented, dynamically typed, reflective programming language created at Xerox PARC.

- \*\*Features\*\*: Pure object-oriented design, message passing, dynamic typing, reflective capabilities.

- \*\*Use Cases\*\*: Object-oriented research, educational purposes, prototyping, development environments.

### 28. \*\*Scheme (1975)\*\*

- \*\*Introduction\*\*: Scheme is a minimalist dialect of Lisp designed with a focus on functional programming and simplicity.

- \*\*Features\*\*: Minimalist design, lexical scoping, first-class continuations, tail-call optimization.

- \*\*Use Cases\*\*: Academic research, teaching programming concepts, scripting, language design experimentation.

### 29. \*\*Erlang (1986)\*\*

- \*\*Introduction\*\*: Erlang is a concurrent, functional programming language designed for building scalable and fault-tolerant systems.

- \*\*Features\*\*: Lightweight concurrency (processes), message passing, fault tolerance, hot swapping.

- \*\*Use Cases\*\*: Telecommunications, real-time systems, distributed systems, web development (e.g., WhatsApp).

### 30. \*\*Haskell (1990)\*\*

- \*\*Introduction\*\*: Haskell is a purely functional programming language with strong static typing and lazy evaluation.

- \*\*Features\*\*: Pure functions, lazy evaluation, type inference, strong static typing, monads.

- \*\*Use Cases\*\*: Academic research, data analysis, financial systems, compiler construction, education.

### 31. \*\*Lua (1993)\*\*

- \*\*Introduction\*\*: Lua is a lightweight, high-level, multi-paradigm programming language designed primarily for embedded use in applications.

- \*\*Features\*\*: Simple syntax, extensibility, embeddability, garbage collection.

- \*\*Use Cases\*\*: Game development (scripting), embedded systems, configuration scripting, rapid prototyping.

### 32. \*\*Delphi/Object Pascal (1995)\*\*

- \*\*Introduction\*\*: Delphi is an integrated development environment (IDE) for rapid application development using the Object Pascal language.

- \*\*Features\*\*: Strong object-oriented features, rich VCL (Visual Component Library), RAD environment.

- \*\*Use Cases\*\*: Desktop application development, enterprise applications, database applications.

### 33. \*\*Matlab (1984)\*\*

- \*\*Introduction\*\*: Matlab is a high-level language and interactive environment for numerical computation, visualization, and programming.

- \*\*Features\*\*: Extensive mathematical functions, easy matrix manipulations, powerful plotting capabilities.

- \*\*Use Cases\*\*: Numerical analysis, data visualization, algorithm development, engineering simulations.

### 34. \*\*Scratch (2007)\*\*

- \*\*Introduction\*\*: Scratch is a visual programming language aimed at children and beginners to introduce them to programming concepts.

- \*\*Features\*\*: Block-based programming, drag-and-drop interface, focus on simplicity and ease of use.

- \*\*Use Cases\*\*: Education, introductory programming courses, interactive stories, games, and animations.

### 35. \*\*F# (2005)\*\*

- \*\*Introduction\*\*: F# is a functional-first programming language developed by Microsoft, part of the .NET ecosystem.

- \*\*Features\*\*: Functional programming features, strong type inference, interoperability with .NET, conciseness.

- \*\*Use Cases\*\*: Financial modeling, data science, web applications, general software development.

### 36. \*\*Groovy (2003)\*\*

- \*\*Introduction\*\*: Groovy is a dynamic language for the Java platform, with features similar to Python, Ruby, and Smalltalk.

- \*\*Features\*\*: Dynamic typing, scripting capabilities, seamless Java integration, concise syntax.

- \*\*Use Cases\*\*: Scripting for the JVM, build automation (Gradle), web development (Grails), rapid prototyping.

### 37. \*\*D (2001)\*\*

- \*\*Introduction\*\*: D is a systems programming language designed to combine the power and high performance of C and C++ with the productivity of modern languages.

- \*\*Features\*\*: System-level access, garbage collection, contract programming, unit testing.

- \*\*Use Cases\*\*: Systems programming, application development, game development, numerical computing.

### 38. \*\*Scala (2003)\*\*

- \*\*Introduction\*\*: Scala is a general-purpose programming language that provides support for functional programming and a strong static type system.

- \*\*Features\*\*: Concise syntax, interoperability with Java, functional programming features, type inference.

- \*\*Use Cases\*\*: Web development (Play Framework), data processing (Apache Spark), concurrent programming, general software development.

### 39. \*\*Julia (2012)\*\*

- \*\*Introduction\*\*: Julia is a high-level, high-performance programming language for technical computing, with syntax familiar to users of other technical computing environments.

- \*\*Features\*\*: High performance, dynamic typing, multiple dispatch, rich mathematical function support.

- \*\*Use Cases\*\*: Numerical analysis, data science, machine learning, scientific computing, computational biology.

### 40. \*\*Elixir (2011)\*\*

- \*\*Introduction\*\*: Elixir is a functional, concurrent language built on the Erlang VM (BEAM) for building scalable and maintainable applications.

- \*\*Features\*\*: Functional programming, concurrency, fault tolerance, metaprogramming, polymorphism.

- \*\*Use Cases\*\*: Web development (Phoenix Framework), distributed systems, real-time applications, telecommunications.

### 41. \*\*Crystal (2014)\*\*

- \*\*Introduction\*\*: Crystal is a programming language that aims to combine the syntax and expressive power of Ruby with the speed and efficiency of a compiled language.

- \*\*Features\*\*: Ruby-like syntax, static type-checking, high performance, concurrency with fibers.

- \*\*Use Cases\*\*: Web development, CLI tools, performance-critical applications, microservices.

### 42. \*\*Clojure (2007)\*\*

- \*\*Introduction\*\*: Clojure is a modern, functional, and compiled dialect of Lisp on the Java platform.

- \*\*Features\*\*: Immutable data structures, functional programming, concurrency primitives, macros, JVM interoperability.

- \*\*Use Cases\*\*: Web development, data analysis, concurrency-intensive applications, general-purpose programming.

### 43. \*\*Nim (2008)\*\*

- \*\*Introduction\*\*: Nim is a statically typed, imperative programming language designed for efficiency, expressiveness, and elegance.

- \*\*Features\*\*: High performance, metaprogramming, garbage collection, Python-like syntax.

- \*\*Use Cases\*\*: Systems programming, web development, game development, scientific computing.

### 44. \*\*Racket (1995)\*\*

- \*\*Introduction\*\*: Racket, originally called PLT Scheme, is a general-purpose, multi-paradigm programming language in the Lisp-Scheme family.

- \*\*Features\*\*: Macro system, rich standard library, extensible syntax, support for creating new languages.

- \*\*Use Cases\*\*: Language research, education, scripting, general-purpose programming.

### 45. \*\*OCaml (1996)\*\*

- \*\*Introduction\*\*: OCaml is a general-purpose, functional programming language with an emphasis on expressiveness and safety.

- \*\*Features\*\*: Strong static typing, type inference, pattern matching, imperative features.

- \*\*Use Cases\*\*: Academic research, language development, systems programming, financial modeling.

### 46. \*\*Dart (2011)\*\*

- \*\*Introduction\*\*: Dart is a client-optimized programming language for fast apps on any platform, developed by Google.

- \*\*Features\*\*: Ahead-of-time compilation, garbage collection, strong type system, asynchronous programming.

- \*\*Use Cases\*\*: Web development (AngularDart), mobile app development (Flutter), desktop applications.

### 47. \*\*VHDL (1980)\*\*

- \*\*Introduction\*\*: VHDL (VHSIC Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems.

- \*\*Features\*\*: Strong typing, concurrency support, simulation, synthesis.

- \*\*Use Cases\*\*: Hardware design, FPGA programming, ASIC design, digital circuit modeling.

### 48. \*\*Verilog (1984)\*\*

- \*\*Introduction\*\*: Verilog is a hardware description language used to model electronic systems.

- \*\*Features\*\*: Simplicity, concurrency support, simulation, synthesis.

- \*\*Use Cases\*\*: Hardware design, FPGA programming, ASIC design, digital circuit modeling.

### 49. \*\*APL (1964)\*\*

- \*\*Introduction\*\*: APL (A Programming Language) is a language known for its concise syntax and array programming capabilities.

- \*\*Features\*\*: High-level array manipulation, concise and symbolic notation, interactive environment.

- \*\*Use Cases\*\*: Mathematical computation, data analysis, financial modeling, research.

### 50. \*\*RPG (1960)\*\*

- \*\*Introduction\*\*: RPG (Report Program Generator) is a high-level programming language for business applications, primarily on IBM systems.

- \*\*Features\*\*: Fixed-format coding, powerful built-in functions for business processing.

- \*\*Use Cases\*\*: Business applications, enterprise resource planning, report generation on IBM i systems.

### 51. \*\*Simula (1967)\*\*

- \*\*Introduction\*\*: Simula is considered the first object-oriented programming language, created for simulation tasks.

- \*\*Features\*\*: Classes, objects, inheritance, coroutine.

- \*\*Use Cases\*\*: Simulation, modeling, early object-oriented programming research.

Certainly! Here are additional programming languages with detailed overviews:

### 51. \*\*Ada (1980)\*\*

- \*\*Introduction\*\*: Ada is a high-level, statically typed programming language designed for safety and reliability in large, long-lived applications, particularly in the defense and aerospace industries.

- \*\*Features\*\*: Strong typing, modularity, real-time support, concurrency, and extensive standard libraries.

- \*\*Use Cases\*\*: Defense systems, avionics, critical systems, transportation systems, real-time processing.

### 52. \*\*APL (1964)\*\*

- \*\*Introduction\*\*: APL (A Programming Language) is known for its concise syntax and powerful array processing capabilities.

- \*\*Features\*\*: Array-oriented, symbolic notation, interactive environment, high-level operations on multi-dimensional arrays.

- \*\*Use Cases\*\*: Mathematical modeling, data analysis, algorithm design, financial analysis.

### 53. \*\*Kotlin (2011)\*\*

- \*\*Introduction\*\*: Kotlin is a modern, statically typed programming language developed by JetBrains, fully interoperable with Java.

- \*\*Features\*\*: Null safety, concise syntax, type inference, coroutine support for concurrency, seamless integration with Java.

- \*\*Use Cases\*\*: Android app development, server-side applications, web development, cross-platform development with Kotlin Multiplatform.

### 54. \*\*RPG (Report Program Generator) (1960)\*\*

- \*\*Introduction\*\*: RPG is a high-level programming language for business applications, originally developed by IBM for report generation.

- \*\*Features\*\*: Fixed-format code structure, powerful file handling and data manipulation capabilities, integration with IBM i (AS/400) systems.

- \*\*Use Cases\*\*: Business applications, report generation, data processing on IBM i systems.

### 55. \*\*AWK (1977)\*\*

- \*\*Introduction\*\*: AWK is a domain-specific language designed for text processing and typically used as a data extraction and reporting tool.

- \*\*Features\*\*: Pattern matching, associative arrays, string manipulation, built-in support for regular expressions.

- \*\*Use Cases\*\*: Text processing, data extraction, report generation, scripting in Unix-like systems.

### 56. \*\*J (1990)\*\*

- \*\*Introduction\*\*: J is a high-level, general-purpose programming language designed by Kenneth E. Iverson and Roger Hui, known for its array processing capabilities.

- \*\*Features\*\*: Array-oriented, concise syntax, tacit programming (point-free style), interactive environment.

- \*\*Use Cases\*\*: Mathematical and statistical analysis, data visualization, algorithm development, financial modeling.

### 57. \*\*Icon (1977)\*\*

- \*\*Introduction\*\*: Icon is a high-level programming language focused on string processing and non-numeric data manipulation.

- \*\*Features\*\*: Goal-directed evaluation, generators, powerful string scanning, dynamic typing.

- \*\*Use Cases\*\*: Text processing, scripting, experimental programming, prototyping.

### 58. \*\*Nim (2008)\*\*

- \*\*Introduction\*\*: Nim is a statically typed, compiled systems programming language designed for performance, expressiveness, and ease of use.

- \*\*Features\*\*: Manual memory management, metaprogramming with macros, cross-platform compilation, high performance.

- \*\*Use Cases\*\*: Systems programming, web development, game development, high-performance applications.

### 59. \*\*REBOL (1997)\*\*

- \*\*Introduction\*\*: REBOL (Relative Expression Based Object Language) is a lightweight, cross-platform data exchange language and scripting language.

- \*\*Features\*\*: Domain-specific dialects, concise syntax, built-in network protocols, easy data exchange.

- \*\*Use Cases\*\*: Scripting, web services, prototyping, data interchange, lightweight applications.

### 60. \*\*Forth (1970)\*\*

- \*\*Introduction\*\*: Forth is a stack-based, procedural programming language and environment designed for interactive execution and low-level hardware control.

- \*\*Features\*\*: Concatenative syntax, stack-based operations, extensibility, efficient execution.

- \*\*Use Cases\*\*: Embedded systems, real-time applications, hardware control, experimental programming.

### 61. \*\*OCaml (1996)\*\*

- \*\*Introduction\*\*: OCaml is a general-purpose, statically typed programming language from the ML family, known for its expressiveness and safety.

- \*\*Features\*\*: Strong type system, type inference, pattern matching, functional programming, object-oriented features.

- \*\*Use Cases\*\*: Compilers, theorem proving, academic research, financial modeling, data analysis.

### 62. \*\*Factor (2003)\*\*

- \*\*Introduction\*\*: Factor is a concatenative programming language inspired by Forth, designed for high-level application development.

- \*\*Features\*\*: Stack-based, dynamic typing, extensible syntax, powerful standard library, focus on developer productivity.

- \*\*Use Cases\*\*: Scripting, rapid prototyping, web development, experimental programming.

### 63. \*\*Io (2002)\*\*

- \*\*Introduction\*\*: Io is a pure object-oriented programming language inspired by Smalltalk, Lisp, and Lua, with a prototype-based inheritance model.

- \*\*Features\*\*: Prototype-based objects, dynamic typing, coroutines, message passing, minimal syntax.

- \*\*Use Cases\*\*: Scripting, rapid prototyping, experimental programming, educational purposes.

### 64. \*\*Vala (2006)\*\*

- \*\*Introduction\*\*: Vala is a programming language designed to bring modern programming features to GNOME developers without imposing any additional runtime requirements.

- \*\*Features\*\*: Modern syntax, GObject-based object-oriented programming, seamless integration with C libraries, automatic memory management.

- \*\*Use Cases\*\*: GNOME desktop environment development, application development on Linux, systems programming.

### 65. \*\*PostScript (1982)\*\*

- \*\*Introduction\*\*: PostScript is a page description language used primarily in the electronic and desktop publishing areas.

- \*\*Features\*\*: Stack-based, procedural, powerful graphics capabilities, device independence.

- \*\*Use Cases\*\*: Desktop publishing, graphic design, document rendering, printing.

### 66. \*\*ColdFusion (1995)\*\*

- \*\*Introduction\*\*: ColdFusion is a rapid application development platform designed for web development, integrating server-side scripting with database connectivity.

- \*\*Features\*\*: Tag-based syntax, integration with databases, built-in functions for web development, simplified handling of web technologies.

- \*\*Use Cases\*\*: Web application development, content management systems, e-commerce applications, enterprise applications.

### 67. \*\*Racket (1995)\*\*

- \*\*Introduction\*\*: Racket is a general-purpose, multi-paradigm programming language in the Lisp-Scheme family, designed for language research, scripting, and education.

- \*\*Features\*\*: Macro system, powerful module system, rich standard library, support for multiple programming paradigms.

- \*\*Use Cases\*\*: Academic research, language development, scripting, educational tools, web development (Racket web server).

### 68. \*\*Zig (2015)\*\*

- \*\*Introduction\*\*: Zig is a general-purpose programming language designed for robustness, optimality, and clarity.

- \*\*Features\*\*: Manual memory management, no hidden control flow, compile-time code execution, cross-compilation support.

- \*\*Use Cases\*\*: Systems programming, game development, embedded systems, high-performance applications.

### 69. \*\*Haxe (2005)\*\*

- \*\*Introduction\*\*: Haxe is a high-level, cross-platform programming language that can compile to several target languages.

- \*\*Features\*\*: Statically typed, multiple target language output, rich standard library, meta-programming capabilities.

- \*\*Use Cases\*\*: Game development, web development, mobile applications, cross-platform applications.

### 70. \*\*Red (2011)\*\*

- \*\*Introduction\*\*: Red is a programming language inspired by REBOL, designed for scripting, cross-platform development, and systems programming.

- \*\*Features\*\*: Homoiconicity, extensible syntax, built-in GUI support, low-level programming capabilities.

- \*\*Use Cases\*\*: Scripting, cross-platform applications, GUI development, systems programming.

### 71. \*\*Pike (1994)\*\*

- \*\*Introduction\*\*: Pike is a dynamic programming language with a syntax similar to C, used for rapid prototyping and scripting.

- \*\*Features\*\*: Object-oriented, garbage collection, built-in support for various data types and structures, powerful standard library.

- \*\*Use Cases\*\*: Web development, scripting, rapid application development, prototyping.

### 72. \*\*PureScript (2013)\*\*

- \*\*Introduction\*\*: PureScript is a strongly-typed, purely functional programming language that compiles to JavaScript.

- \*\*Features\*\*: Pure functions, strong static typing, type inference, functional programming, interoperability with JavaScript.

- \*\*Use Cases\*\*: Front-end web development, single-page applications (SPAs), functional programming in the browser.

### 73. \*\*Tcl (1988)\*\*

- \*\*Introduction\*\*: Tcl (Tool Command Language) is a dynamic scripting language often used for rapid prototyping, scripted applications, and GUIs.

- \*\*Features\*\*: Simple syntax, dynamic typing, extensibility, event-driven programming.

- \*\*Use Cases\*\*: GUI development (Tk), scripting, rapid prototyping, embedded applications.

### 74. \*\*V (2019)\*\*

- \*\*Introduction\*\*: V is a statically typed compiled programming language designed for simplicity, performance, and safety.

- \*\*Features\*\*: Simple syntax, fast compilation, safety features (e.g., no null, no global state), cross-compilation.

- \*\*Use Cases\*\*: Systems programming, web development, game development, cross-platform applications.

### 75. \*\*APL (1964)\*\*

- \*\*Introduction\*\*: APL (A Programming Language) is known for its concise syntax and powerful array processing capabilities.

- \*\*Features\*\*: Array-oriented, symbolic notation, interactive environment, high-level operations on multi-dimensional arrays.

- \*\*Use Cases\*\*: Mathematical modeling, data analysis, algorithm design, financial analysis.

**1. C**

A lot of people see C as the most influential programming language ever. It’s hard to argue with that, given how pivotal C has been to C++, C#, Objective C, Java, Python, JavaScript, and many other languages you’ll find below. All those languages borrowed many basic features from C (especially its control structures). C was the first language to work on a higher level, opening the door for the development of complex software with significantly less knowledge.

Fundamental to the reimplementation of Unix since its 4th version, C enjoys justified popularity, thanks to its almost endless possibilities. This general-purpose language is frequently used to build operating systems, language compilers and interpreters, drivers, and utilities.

**2. Java**

[Java](https://www.bairesdev.com/technologies/java/outsource-services/) often appears at the top of programming languages rankings thanks to its huge popularity, rooted mainly in one thing: its “write once, run anywhere” (WORA) principle: any compiled Java code can run on any platform that supports Java without having to recompile it first. That’s possible thanks to the Java Runtime Environment (JRE), which runs on a Virtual Machine to provide it with that cross-platform capabilities.

Combined with the fact that Java is a general-purpose language, that makes it perfect for virtually any development project. In fact, Java is used for a lot of things, from Android apps and web apps to games, big data tools, and even scientific platforms.

**3. Python**

A lot of beginners that have gone online to ask which programming language should they learn first hear [Python](https://www.bairesdev.com/technologies/python/) as the first answer. That’s because the language has a strong emphasis on code readability, ease of use, and flexibility. All of that means that beginners can quickly learn it and start working on pretty much any application they can think of.

Though Python can be used to build everything from business applications to games, there are some strong suits for it. For one, the language is frequently used for data analysis, scripting, and scientific research, mainly because it includes a robust set of tools for that. But Python is also great for [web app development](https://www.bairesdev.com/solutions/web-development/) and even for the creation of AI-based applications.

**4. C++**

A successor to C, the idea behind [C++’](https://www.bairesdev.com/blog/what-is-c-plus-plus/)s development was to add higher-level paradigms to C but also keeping the low-level qualities present in it. Since its inception, C++ has evolved quite a lot to include object-oriented, generic, and functional features. C++ is a compiled language that has many compilers to work on many different platforms, thus extending its usability.

Given its predecessor, C++ boasts notable computing performance, which is the reason why it’s mostly used for system programming and embedded systems. Thus, it’s often used to create device drivers, game engines, image and audio software, [telecommunications networks](https://www.bairesdev.com/industries/telecom/), and interpreters.

**5. C#**

Just Like C++, C# is another programming language built on top of C where the major addition was object-oriented capabilities. C# is closely related to Microsoft’s .NET framework since it was initially developed by the Redmond giant as part of its .NET initiative and has now become the primary language for [.NET programming](https://www.bairesdev.com/technologies/net/).

C# is another language that can be used to build close to everything you can think of but it’s perfectly suited to build Windows desktop applications and games. Additionally, you can use it to create web apps as well as mobile applications which can later be used on virtually any mobile device through cross-platform tools like Xamarin.

**6. JavaScript**

[JavaScript](https://www.bairesdev.com/technologies/javascript/) has been a fundamental part of our everyday lives for quite some time now, simply because it’s one of the essential pillars of most websites (along with HTML and CSS). Through JavaScript, web pages can dynamically apply changes to styles, animate menus, validate data entered into a form without refreshing, and much more. Javascript also works for web and [mobile app development](https://www.bairesdev.com/solutions/mobile-app/), as well as server applications and games.

Most modern websites use JavaScript in some sense or another to control client-side page behaviors. And though it’s mostly a programming language for front-end development, it can be used on the server side as well (with Node.js).

**7. PHP**

Speaking of the server side, here comes [PHP](https://www.bairesdev.com/technologies/php/), probably the most popular programming language designed for server-side development. This general-purpose scripting language can collect data from online forms, create dynamic elements for web pages, or handle cookies. Though that is its main goal, PHP can do more than that.

For instance, PHP is great for command-line scripting to create scripts that only need a PHP parser to run, something nice for simple processing tasks. PHP can also be used for building desktop applications, though you have to really know your way around it to do so. Mastering the language isn’t a utopia, though, as the language has a gentle learning curve which makes it a great option for any newcomer to the programming world.

**8. VB.NET**

Though a lot of .NET developers now favor C# for .NET programming, Visual Basic still remains a fairly popular option to do so as well. This multi-paradigm, [object-oriented language](https://www.bairesdev.com/blog/top-object-oriented-programming-languages/) is the successor of the original Visual Basic, but it offers so much more, especially because it provides full access to .NET libraries but also because it’s easy to understand, is reliable, and it’s highly scalable.

Since VB.NET relies on the .NET framework, you can use this language to quickly create a wide range of Windows and Office applications as well as web and mobile apps.

**9. R**

With the rise of data science and statistical software, it’s only natural that a language as laser-focus as [R](https://www.bairesdev.com/blog/r-python-best-data-science-language/) would grow among the development ranks. Thanks to its wide array of libraries, R can implement a lot of statistical and graphical techniques, such as linear and nonlinear modeling, classification, clustering, and many others.

This nature is the main reason why statisticians and data miners use it to build statistical analysis and to create high-quality data visualizations. In fact, R is one of the most popular alternatives for the [big data phenomenon](https://www.bairesdev.com/solutions/data-science/), especially because it’s highly expandable through the many packages that are available for it.

**10. SQL**

[Databases](https://www.bairesdev.com/solutions/database/) are an integral part of our tech-driven world – and SQL is among the most popular languages used to communicate with them. SQL stands for Structured Query Language and allows developers to create databases with their structures and handle all the data-related tasks that come with it, such as inserting, editing, and querying.

The popularity of SQL is related to how it gives engineers the ability to use a conventional language to handle relational databases in new IT environments, like cloud-native systems and virtual networks. Thus, SQL is often a default tool to handle the data contained in a conventional database.

**11. Go**

Also known as Golang, [Go](https://www.bairesdev.com/blog/best-languages-web-development/) is a Google-designed programming language that’s midway between C and C++. In fact, it’s syntactically similar to C but more sophisticated because of the inclusion of features like garbage collection and structural typing. This brings it closer to C++ although without this programming language’s complexities.

Originally created for systems programming, Go has evolved and it’s now used for web applications, [cloud-based platforms](https://www.bairesdev.com/solutions/cloud-application/), containerization tools, and even for certain systems within cryptocurrencies. This makes it a great alternative for seasoned developers, though its ease of use is appealing to newbies as well.

**12. Swift**

If you are interested in [developing applications for any Apple device](https://www.bairesdev.com/blog/mobile-app-development-steps/), then you’ll need to learn Swift. This programming language was developed by the Cupertino company for iOS, OS X, watchOS, tvOS, and Linux development. It’s primarily inspired by C and Objective-C and C but it has extended compatibility.

This allows it to work perfectly with Cocoa and Cocoa Touch, Apple’s own frameworks as well as with any existing Objective-C code developed for Apple systems. Apart from the fact that this is the programming language for creating Apple apps, it’s also a high-performance, general-purpose language that focuses on safety and (naturally) software design.

**13. Perl**

Frequently referred to as “the Swiss army knife” of scripting, Perl has been around for almost 4 decades – and it’s still going strong. This high-level, interpreted, dynamic language can run over 100 platforms and it’s a great fit for both prototyping and big development projects. In fact, Perl went beyond its initial purpose (text editing) and it’s now used for a lot of things, including Linux system administration, network programming, and web development, among others.

It’s important to note that sometimes Perl refers to a family of languages, a term that includes Raku (formerly called Perl 6). However, Raku is now a programming language with its own development team and schedule and the original Perl team keeps working on its development.

**14. Assembly language**

Assembly language is the most fundamental programming language around, as it’s a low-level language. This basically means that this language has little to no abstraction from the computer’s own architecture (meaning that the language’s functions are as close to the processor instructions as you can possibly get). That’s why it’s also known as the symbolic machine code.

Assembly code is converted to machine code through a processor called assembler, which runs the resulting code on the processor.  Since assembly depends on the machine instructions themselves, each assembler has its own language, specific to that particular computer architecture. There’s some discussion about assembly language’s value in the modern landscape, but it has a proven worth when optimizing for speed or size, for driver development, or for cases when strict time constraints are inhttps://www.bairesdev.com/ruby/ place, such as real-time simulators.

**15. Ruby**

According to its own official website, “[Ruby](https://www.bairesdev.com/technologies/ruby/) is a language of careful balance”. They can certainly claim that since the language combines parts of Perl, Smalltalk, Eiffel, Ada, and Lisp to come up with a balanced mix between functional and imperative programming. This interpreted, high-level, general-purpose programming language is highly flexible since developers can remove or redefine any part of Ruby.

Ruby is mostly used for web application development, especially because of [Ruby](https://www.bairesdev.com/technologies/ruby/hire-developers/) on Rails, a very popular development framework that provides an MVC (Model View Controller) architecture and encourages the use of web standards. Ruby’s uses don’t end there, though, as it can also be used for data analysis, prototyping, and proof of concepts.

**16. MATLAB**

MATLAB is another programming language designed with engineers and scientists in mind. It runs on the proprietary platform of the same name and it’s a matrix-based language that allows developers to express complex computational mathematics in the most natural way. Thus, you can work with matrix manipulations, data implementations, and plotting of functions.

All of that turns MATLAB into the perfect ally for anyone looking to analyze data, develop algorithms, and create math models and applications. In MATLAB’s environment, you’ll find computation and visualization tools that are easy to use and can provide complex outputs through familiar mathematical notation.

**17. Classic Visual Basic**

As incredible as it feels, the TIOBE Index still ranks Classic Visual Basic among its top 20 programming languages. That’s really strange, as the programming language was declared a legacy by Microsoft back in 2008. However, it seems that there are many developers out there that still use them to keep the necessary [legacy systems](https://www.bairesdev.com/blog/problems-with-legacy-systems/) running.

Visual Basic used to be a good way to create Microsoft-centered applications using the components provided by its platform (especially after the VB community started developing its own components). However, today, it’s only worth learning for developers that are already familiar with other languages and need this for a very particular project. We expect Classic VB to disappear in the coming years but, hey, we still include it out of respect.

**18. Apache Groovy**

This is a fairly new object-oriented language that boasts Java-syntax compatibility and that runs on the Java platform. In a certain sense, you could say that Apache Groovy is similar to Python, Ruby, and Smalltalk, especially because it’s considered both a static and dynamic language. What’s more, since its code is compiled through the Java Virtual Machine, it can interoperate with other Java code and libraries, which extends its power.

Among its most powerful features, there are scripting capabilities, domain-specific language authoring, metaprogramming, and functional programming. All of that can let us see it as a “Java enhancer” that provides more flexibility and power to Java developers. Thus, any Java project can be retouched through Apache Groovy to increase its performance.

**19. Objective-C**

Yet another high-level programming language based on C, Objective-C took it a little further by adding features and syntax from Smalltalk. Thus, we can say it’s a superset of C since any C code can run perfectly in an Objective-C compiler. Objective-C was initially developed as a way to add object-oriented programming to C but has taken a life of its own by now.

Thanks to its capabilities, Objective-C was the main language for developing Mac OSX and iOS applications (in fact, you needed to learn it to create any apps for those systems). With the introduction of Swift by Apple, Objective-C isn’t the only alternative anymore, though it’s still an interesting option.

**20. Rust**

If you regularly check [Stack Overflow’s Developer Surveys](https://survey.stackoverflow.co/2022/), you’ve certainly come across Rust at some point, especially in the “most loved programming language” section. That’s because this multi-paradigm programming language is mainly focused on performance and safety. Rust has a very strict compiler that checks all variables and memory addresses, which leads you to write more effective and idiomatic programs.

Developers have fallen in love with Rust because of its speed, memory safety, and parallelism. All of those allow engineers to work on a wide range of applications, from game engines and operating systems to browser components and file systems.

**Classics, Up-and-coming, and Niche Languages**



**21. SAS**

This is a specialized programming language whose main goal is to perform statistical data analysis with databases and spreadsheets. SAS compiles the data, analyzes it, and offers the output through several visualization options or text/web-based documents. It’s mainly used in academia and government, given the level of control and freedom it offers to handle huge data loads.

**22. Scratch**

Getting kids interested in programming has become a strategic goal for governments, academics, and businesses alike. That’s why a programming language such as Scratch is so important – this is a free, visual programming language based on Smalltalk targeted to children aged 8 to 16 to educate them in basic programming terms and teach them about systematic reasoning while creating stories, games, and animations.

**23. D**

D is a programming language designed to overcome the drawbacks of C (just as C was designed to sophisticate B’s offering). D is basically like C++ in that it’s object-oriented but unlike it, it’s not backward compatible, which helps remove the disadvantages present in C++.

**24. Dart**

Dart is another Google-designed language that the company used internally to create web, server, and mobile applications. In a sense, Dart compiles the source code in the same way as JavaScript, though the code has its Dart Virtual Machine. Though it has been around for quite some time, Dart is just starting to get the attention it deserves, mainly because Google introduced Flutter for native [Android development](https://www.bairesdev.com/technologies/kotlin/) – and developers that want to work with it need Dart to get started.

**25. PL/SQL**

SQL by itself won’t take you very far when working with databases. That’s why you need a language like PL/SQL, a procedural language that’s a superset of SQL. It builds on top of SQL adding procedural programming features, including basics like conditionals, loops, functions, objects, types, and more. Since it was created by Oracle, it works perfectly with their databases.

**26. Logo**

Developers of a certain age surely remember Logo, one of the first programming languages to teach children how to code. A lot of us learned how to write code on a console by providing commands to a small turtle that drew graphics. To some people, Logo is Lisp for kids, even when it has powerful features to create images, multimedia presentations, and games.

**27. Delphi**

Also known as Object Pascal, Delphi is both a programming language and an IDE for rapid application development. In fact, Delphi was designed as the successor of Turbo Pascal, a language already noted for its fast compiling times. Delphi, however, added full object-oriented programming to that foundation to become a fantastic alternative for desktop, mobile, web, and console [software development](https://www.bairesdev.com/solutions/custom-software/).

**28. COBOL**

Common Business-Oriented Language or COBOL as we all know it, is one of the first high-level programming languages that’s still around today. This imperative, procedural, and object-oriented language is still active mainly to maintain and support existing business, [finance](https://www.bairesdev.com/industries/finance/), and administrative systems. It’s highly likely that COBOL (like Visual Basic) is seeing its last days but it still has things to offer, especially a surprising mainframe processing speed.

**29. Kotlin**

One of the rising stars in the [software development](https://www.bairesdev.com/software-development-services/) sky, Kotlin is a cross-platform, general-purpose programming language that’s Google’s preferred language for Android development. That certainly has boosted its popularity but its benefits have also played an important part. Kotlin is like a more concise version of Java, is highly interoperable with it, has many features, is easy to learn, and is less error-prone.

**30. OpenEdge ABL**

This is a business application fourth-generation programming language that offers a high level of abstraction. In fact, the goal of OpenEdge ABL is to be extremely user-friendly, providing anything a developer needs for Rapid Application Development. That’s why this language is perfect for prototyping and creating systems with simple interfaces.

**31. Julia**

Another young language that’s seeing a rise in popularity, Julia is a high-level and dynamic programming language. It can work well for a wide array of projects but most of its features are a better fit for computational science. Additionally, Julia includes libraries for linear algebra, regular expression matching, and floating-point calculations, which makes it obvious that it’s a great choice for developers working with numerical analysis.

**32. ABAP**

The Advanced Business Application Programming language (or ABAP, for short) is a crucial language to create enterprise tools, applications, and systems or SAP software (named like that after the company that created ABAP in the first place).

**33. Scala**

Some people say that Scala is like an overhauled version of Java – and they might be just right. That’s because Scala is also object-oriented, compiles to Java bytecode to run on the JVM, and it’s even interoperable with Java, which means that it can use its libraries. Where it differs from Java is in the fact that Scala incorporates features of functional programming, it’s more concise and has a syntax that’s easier to read.

**34. Transact-SQL**

Also known as T-SQL, Transact-SQL is an extension of SQL developed by Microsoft and Sybase which is used to interact with relational databases. As such, T-SQL is key for developers using Microsoft SQL Server, the database management system created by the Redmond giant. All of the communications with this system are carried out through T-SQL statements, which explains why this language is popular among certain users – it’s practically mandatory!

**35. Scheme**

Some people might say that Scheme is just a Lisp dialect but, to us, it’s a little more than that. True, Scheme may be too minimalist an adaptation of Lisp with a small core and a set of interesting tools. However, this economic approach can also be its strength. In fact, many educators swear by it and some engineers see in Scheme the best possible ally to develop embedded systems and for scripting.

**36. Prolog**

Prolog is a very different language to the ones above, mainly because it’s a logic programming language. That means that, unlike a lot of other languages that use procedural logic, Prolog is more about first-order logic, where the underlying logic of a program is expressed in various relations of facts and rules. Thus, the software works by interpreting queries according to these relations. Such an approach is why Prolog is mostly associated with artificial intelligence and NPL projects.

**37. Ada**

Ada Lovelace (a.k.a. The Enchantress of Numbers) is considered to be the creator of the first computer program by a lot of people. Whether that’s true or not, The Enchantress certainly inspired a group of French developers that came up with Ada, an object-oriented, high-level programming language extended from Pascal. It might not be the most popular language out there but the French team that created it didn’t aim for popularity but security – especially because the language was commissioned by the US Department of Defense, which is where it finds most of its uses.

**38. Lisp**

We’ve mentioned a couple of its offspring before, so it was about time we talked about Lisp, one of the eldest high-level programming languages still in use. In reality, the name Lisp points to a family of programming languages that share the same roots – the pioneering ideas that Lisp brought to life, such as tree data structures, dynamic typing, recursion, and higher-order functions, to name a few. It has been used for many things throughout the years but its relevance today is explained by the adoption of many AI and machine learning developers as their language of choice.

**39. Apex**

Another proprietary programming language, Apex is used by Salesforce to execute flow and transaction control statements and calls to the API in Salesforce servers. It’s an object-oriented language that’s similar to Java and C# that allows Salesforce developers to add business logic to many system-related events. Thus, software engineers that use it do so exclusively because of what Apex can give them in Salesforce.

**40. Lua**

Originally designed as a language that pursued the extension of software applications for increased customization, Lua continues to be one of the best languages for that purpose. That’s because it’s a lightweight high-level language that focuses on speed, portability, and ease of use. One of the best things about Lua is that it’s cross-platform, which extends its range of use.

**41. Fortran**

It almost feels ridiculous to think that a language that first appeared in 1957 might be around today, more than 60 years later, and still be relevant. Yet, that’s precisely what happens with Fortran, a general-purpose, imperative language developed by IBM with a strong focus on numeric computation and scientific computing. The secret to its relevance and popularity is that Fortran’s foundations are so robust that no other language can show similar performance in computationally intensive tasks such as computational fluid dynamics, geophysics, and finite element analysis. All that explains why Fortran is also used to create programs for benchmarking.

**42. Haskell**

Another language widely known for the many languages that use it as a basis, Haskell is another general-purpose, multi-paradigm, purely functional language. The “purely functional” part means that Haskell’s functions only depend on their arguments, regardless of their global or local states. That makes it easier to understand, which is why Haskel is mostly used for academic reasons (though there it also has industry-wide uses as well).

**43. Hack**

This might not be the most popular language, but Hack was developed by Facebook as a way to extend PHP through the introduction of static typing, new type hints, and enhancing the use of type hints. In that sense, Hack is an implementation of PHP that lets developers use both dynamic and static typing through a system called gradual typing.

**44. VBScript**

Yet another proprietary language created by Microsoft that’s based on Visual Basic, VBScript was the company’s alternative to JavaScript for web developers. Naturally, JavaScript’s dominance left it without much room, so this scripting language found a niche in the development of applications and systems within Microsoft environments. Thus, VBScript is used for systems administrators, unit testers, developers of embedded applications, and macro development for Microsoft apps.

**45. Visual FoxPro**

Though Microsoft released its final version back in 2007, Visual FoxPro still has an active community that still sees the language as a valuable asset for database application development. True, you won’t find many new applications that use Visual FoxPro as their language of choice, but there are plenty of legacy platforms available written in it that perform well and need to be maintained.

**46. TypeScript**

Microsoft adds another language to this list with TypeScript, a superset of JavaScript that adds optional static typing to the language. That means that any existing application written in JavaScript can be worked through this programming language. The idea behind the development of this language is to provide developers with the possibility to build JavaScript applications for both client and server-side execution as well as to allow the development of large applications.

**47. AWK**

Here’s another language that was developed in the 70s that’s still around. AWK is a domain-specific language that consists of a set of actions that are carried out against streams of text. Its goal is to process those texts to extract information or transform them into a different output. This makes AWK a fairly limited programming language but it’s a nice addition to any toolkit, especially for those that work with large amounts of text.

**48. ActionScript**

The TIOBE Index shows that there still is some interest in ActionScript, a programming language that you could say it’s another superset of JavaScript. And that’s a surprising thing, mainly because ActionScript’s purpose was the development of website and software with Adobe Flash Player components. As you might recall, Flash was a crucial element of the web some time ago, as it provided animations and interactive elements that couldn’t be done otherwise. However, with the arrival of HTML5, Flash fell from grace and now it’s mostly shunned. Basically, this all means that ActionScript is another one of those languages that will be gone sooner rather than later.

**49. Tcl**

Pronounced “tickle”, Tcl is a high-level, general-purpose language that is as simple as you can possibly get. In this language, everything is a command, even control structures such as “for” and “if”. This means that a developer can tweak pretty much anything they want through several commands, which provides them with a lot of power and flexibility. Those traits are what turned Tcl into an ally for developers working on rapid prototyping, scripted applications, and [testing](https://www.bairesdev.com/solutions/qa/).

**50. Smalltalk**

Smalltalk is a very influential programming language that helped shape the model–view–controller (MVC) pattern for user interface design and fostered a new era in GUIs. However, even for all its pivotal relevance, Smalltalk has been gradually abandoned by most developers and it’s only used by a small community that swears by it. A significant reason why Smalltalk is used today is because of the release of development frameworks for it, especially Seaside, a framework that makes it easy for you to create web apps.

**50 Other Languages Worth Mentioning**



Aside from all the programming languages mentioned above (from the well-known to the classics, the newcomers, and the almost goners, there are many other languages out there that are worth knowing about. Sure, they might not have the biggest communities, a wide array of applications, or the biggest brands investing in them. However, they have interesting things for themselves, which is why we’re including them here.

**51. ABC**

A general-purpose programming language and IDE, ABC is an alternative to BASIC, Pascal, or AWK, developed for teaching and prototyping. ABC was highly influential in Python’s design.

**52. Alice**

A dialect of Standard ML, this visual programming language uses 3D objects and point-and-click logic to teach basic programming concepts to new students and kids.

**53. APL**

APL stands for “A Programming Language” but don’t let that simplicity in the name fool you. APL is one of the most peculiar languages in the market, mainly because it uses a wide range of special graphic symbols for its functions and operators. Today, it’s mostly used for commercial and scientific purposes.

**54. AutoLISP**

Another member of the Lisp family, AutoLISP is a dialect of that language developed specifically to work within AutoCAD and several of its derivative platforms. That’s why it’s almost exclusively focused on geometry.

**55. Bash**

Bash (or Bourne Again Shell) is an sh-compatible command language that runs on a window where the user inputs commands to execute actions. Since its release, it has been used as the default login shell for a majority of Linux distributions.

**56. bc**

bc stands for “basic calculator” (or “bench calculator”, depending on who you ask) and it’s an arbitrary-precision calculator language that’s used for mathematical scripting, given that its calculations are performed with numbers whose digits of precision are only limited by available memory.

**57. Bourne shell**

Remember how Bash stood for “Bourne Again Shell”? This is the Bourne shell on which that pun lies on. That means that this is very similar to Bash in that it interprets and executes commands and provides programming abilities based on them.

**58. C shell**

Here’s another shell, which means that C shell also runs from a window and executes user commands. The main difference is that the C shell is easier and faster to use, thanks to the language’s style, which resembles C, making it more readable.

**59. Clipper**

Clipper is a compiler that works to extend software that was originally created to work in MS-DOS through variants of xBase. A rather powerful general-purpose programming language, it was mostly used for database and business programs.

**60. Clojure**

Clojure is another general-purpose programming language that combines the approach of scripting languages with a solid infrastructure for multithreaded programming. A dialect of Lisp, it has been seeing a rise in popularity in recent years, thanks to its use in creative computing and the advocacy of several thought leaders.

**61. CoffeeScript**

CoffeeScript’s main rule is “it’s just JavaScript” (which might feel odd, because it isn’t precisely JavaScript). That means that CoffeeScript is a small language that compiles to JavaScript and that seeks to streamline the code while also getting rid of the interpretation at runtime.

**62. Common Lisp**

Yet another dialect of Lisp, Common Lisp is an improvement over its successor, Maclisp. Common Lisp’s objective was to unify and standardize different Maclisp dialects into a general-purpose programming language that combined procedural, functional, and object-oriented paradigms.

**63. Crystal**

A general-purpose, object-oriented language, Crystal was inspired by Ruby and developed to have its productivity and combine it with the speed and efficiency of compiled languages. Thus, it’s a good alternative for Ruby developers looking for increased performance.

**64. cT**

This is an algorithmic language similar to C, Fortran, and Pascal but with a stronger focus on multimedia. Thus, cT includes increased support for color graphics, mouse interactions, and video. It was a good alternative for developing multimedia programs across different platforms, but it was pushed aside in favor of VPython.

**65. Elixir**

Elixir is a general-purpose language that runs on BEAM, Erlang’s virtual machine. That’s because Elixir is based on Erlang and has the same abstractions. The difference is that Elixir provides you a more productive tooling and extensible design.

**66. Emacs Lisp**

As you might have guessed, Emacs Lisp is a Lisp dialect that’s used as a scripting language for Emacs, a text editor. In a sense, Emacs Lisp is close to Maclisp and even shows some influence from Common Lisp, which is why it’s used to customize and extend Emacs.

**67. Erlang**

Erlang is a general-purpose language originally created to develop telecom applications but that’s now used as a way to build concurrent applications. With this highly-scalable, functional language you can create telecom tools, web apps, distributed computing, and any other solution that requires handling multiple simultaneous inputs.

**68. Euphoria**

Euphoria is a general-purpose imperative-procedural language that focuses on simplicity, legibility, and rapid development. This allows Euphoria to handle dynamic sets of data easily, which is why it’s used for string and image processing, artificial intelligence, and mathematical studies.

**69. F#**

This general-purpose, cross-platform language was created by Microsoft with efficiency as its main goal. After reaching open-source status, F# gained certain popularity and you can now use it for [enterprise programming](https://www.bairesdev.com/solutions/enterprise/), data science, machine learning, cloud computing, math and stats, and other related fields.

**70. Forth**

This is a procedural language without type checking that features interactive execution of commands and offers the possibility to compile sequences of commands for later use. Forth is mostly used in embedded systems that require interaction with hardware.

**71. Genie**

Genie is a fairly new high-level language that was developed as an alternative to the Vala compiler in search of a simpler and cleaner dialect. In fact, you can say that both Genie and Vala are pretty much alike since their only differences lie on their syntax. Thus, Genie is closer to modern languages like Python and Delphi.

**72. Icon**

Labeled “a very high-level language” by some people, Icon is focused on making it easy to process strings and structures while providing you with high-level graphic possibilities. The syntax is extremely easy to use given its abstraction level, which makes it perfect for creating short and concise programs in the fields of text analysis, document formatting, and data laundry.

**73. IDL**

Standing for Interactive Data Language, IDL is a language mainly focused on data analysis and visualization. It shares syntax with PV-Wave and it’s somewhat of a niche language, as it’s mostly used in astronomy, atmospheric physics, and medical imaging.

**74. Inform**

Inform is a peculiar language that powers interactive fiction works. It’s an object-oriented and procedural language that provides the necessary structures to create objects that you can combine to create the narrative of your stories.

**75. Io**

Io is an object-oriented language partly inspired by Smalltalk, Lua, and Lisp. Its philosophy is to explore conceptual unification and dynamic languages, which is why Io is more about simplicity and flexibility rather than performance.

**76. Korn shell**

Here’s another Unix shell based on the Bourne shell but it also includes features from the C shell. Thus, Korn shell is a combination of the two most popular shells and it’s also the easiest one to use, which is why newcomers tend to use it, especially in commercial environments.

**77. LabVIEW**

LabVIEW is a graphical dataflow language, which means that you build applications by stringing together a set of routines represented by images. It was designed for interfacing with measurement and control devices through simple code, which allows you to build complex tools without having to write a single line.

**78. Ladder Logic**

The name “Ladder logic” was once used to refer to a method of documentation of the design and construction of relay racks through interconnected symbols. That principle evolved into this language, which presents an application with a graphical diagram. That’s why Ladder Logic is mostly used in industrial control applications.

**79. LiveCode**

LiveCode is a natural expressive language with a strong focus on readability. Its developers promise that you’ll write up to 90% less code with it, thanks to its English-like, object-oriented nature. That doesn’t mean it isn’t powerful, as LiveCode has been used to create AI, big data, and cryptography apps.

**80. Maple**

Maple is both a symbolic and numeric environment and a multi-paradigm language. Through it, you can design powerful applications capable of dealing with symbolic mathematics, numerical analysis, data processing, and visualization, among others.

**81. Mercury**

This is a functional logic language that has a purely declarative logic. In that sense, Mercury is related to Prolog and Haskell and, in fact, can be seen as a subset of the former, albeit with strong types and modes.

**82. ML**

ML stands for “Meta Language” a name that indicates its main focus – programming language research. That’s possible thanks to its use of the Hindley-Milner type system, which automatically assigns types without requiring explicit annotations, all while ensuring type safety.

**83. MQL4**

MQL4 is a high-level, object-oriented language based on C++ and a strong focus on flexibility. That’s what allows it to develop complex programs with a huge number of calculations. This, in combination with built-in functions for managing trading orders, is what explains why this language is mostly used for trading robots, indicators, scripts, and libraries for the MetaTrader 4 platform.

**84. NATURAL**

NATURAL is another proprietary fourth-generation language that is structured and less procedural than other conventional languages. It creates its programs in the system of the same name and can run them both interpretatively or executed as compiled objects. It was developed to work with Adabas, a database package that’s still widely used.

**85. NXT-G**

This is another graphical programming language that was built to program with the LEGO Mindstorms NXT robotics kit. Through a drag-and-drop environment, you can arrange function blocks and “sequence beams” to control the flow of the program. It’s a good way to introduce kids to programming and even for newcomers in the robotics field because it can manage components beyond LEGO-manufactured ones.

**86. OpenCL**

Some might say that OpenCL is a framework rather than a language, but we stand by the TIOBE Index selection. Through it, it’s possible to create programs that can be executed across many platforms consisting of CPUs, GPUs, DSPs, FPGAs, and other processors and accelerators. It provides a standard interface for parallel computing of accelerators found in cloud servers, mobile devices, embedded platforms, and more.

**87. Oz**

A multi-paradigm language, Oz is mostly used for teaching programming. It contains most of the aspects you’d find in the most popular programming paradigms, such as logic, functional, imperative, object-oriented, and concurrent programming. This, in combination with its simple semantics, make Oz a great educational alternative.

**88. PL/I**

PL/I or Programming Language One is another legacy language that’s become very niche. Designed for scientific, engineering, business, and system programming, this procedural, imperative language is now only used to maintain old systems or convert them to run in new hardware.

**89. PostScript**

A highly specialized language, PostScript is a page description language whose purpose is to describe graphics and page layouts. Thus, it’s mostly used for visual output: printing, font design, desktop GUIs, and individual images. Since PostScript describes elements in a resolution-agnostic way (making them highly scalable), it’s in the publishing and data visualization fields where it finds most of its use.

**90. Q**

Q is another proprietary language that was designed for array processing and to serve as a query language for kdb+, a column-based database. This database, in turn, is based on the language k, of which Q is a variation with a more readable interface.

**91. Racket**

Racket is a general-purpose language based on Scheme and mostly focused on programming language design and implementation (though it can also be used for scripting, research, and educational purposes). The language is a part of a platform of the same name, which includes a runtime system, libraries, and a JIT compiler.

**92. Red**

Red was first developed as a way to improve on Rebol but with the ultimate goal of becoming a full-stack language. Thus, Red is both imperative and functional and can be used for high-level and low-level programming through its two main components: Red/System and Red.

**93. Ring**

Ring puts innovation at the forefront of its logic. This general-purpose language supports many programming paradigms, including imperative, procedural, object-oriented, declarative, functional, meta, and natural programming. Simple and fast, Ring can be used in many projects, from web development to game development.

**94. RPG**

RPG (Report Program Generator) is a high-level language developed by IBM that originally served as an alternative to the punch card processing system on the IBM 1401. Today, it has evolved into a COBOL-like language that serves many businesses to create commercial applications on IBM’s iSeries minicomputer system.

**95. S**

The predecessor to R, S is a high-level language for data manipulation and analysis. Highly extensible, S is a functional system that treats functions written by users as first-class objects similar to those of the system. Its code is readable and the language is fairly easy to understand, which is why it’s used for statistical and graphical analysis in large applications.

**96. SPARK**

A variation of Ada, SPARK is a formally defined language whose main purpose is the development of software for high profile safety-critical systems such as those used in aviation and in space applications. To ensure your solution’s safety and security, you can use its contracts to dynamically verify the specification of its components.

**97. Stata**

Stata is more than a language – it’s a pair of powerful languages packed in a platform developed to support statistical analysis and specialist research. Through it, you can solve complex statistical problems and produce charts, graphics, and simulations. Both languages are great to write scripts and manipulate data.

**98. Vala**

As we mentioned before, Vala and Genie are fairly similar. Vala is an object-oriented language that generates C code and that’s syntactically similar to C#. In that way, anything you create with Vala should have a similar performance to anything you can create in C, though with the benefit of being easier to write and maintain.

**99. Verilog**

This is a hardware description language (HDL), a special type of language that describes the structure of electronic and digital logic circuits. Basically, this means that Verilog is used to design computer chips and digital circuitry in general.

**100. VHDL**

Another hardware description language, VHDL is another popular alternative for people involved in designing  CPUs, motherboards, FPGAs, and many other types of digital circuitry. VHDL can also work to simulate and test the performance of all designs.

There are countless computer programming languages, but there are some that are more widely used than others. Here are three of the most popular computer programming languages:

**HTML**

HTML stands for "Hyper Text Markup Language" and is utilized for coding web pages. It marks elements of a document, like headings and paragraphs, and tells a computer how they should be displayed.

**Java**

Java has been used since the beginning of the World Wide Web to improve websites and add interactive capabilities like buttons and app widgets. It is also commonly used for programming cell phones.

**C Language**

C Language works as a basic coding language and is widely used to make programs run faster. It remains a popular choice for video game developers who use C++ language as well since the two languages complement each other in terms of programming.**Related:**[**The 5 Best Programming Languages To Learn**](https://www.indeed.com/career-advice/career-development/best-programming-languages-to-learn)

[**Get interview-ready with tips from Indeed**](https://www.indeed.com/career-services/interview-help?collectorID=careerguidepromo-US&from=careerguidepromo-US)

Prepare for interviews with practice questions and tips

**50 types of programming languages**

If you're considering a job in computer programming or another technology field, being familiar with these programming languages can help set you apart from other candidates. Here are 50 types of programming languages you can learn:

**Procedural languages**

Procedural languages are based on the data viewing range of a code statement. Examples include Ada, BASIC, C/C++ and JavaScript.

**Functional languages**

Functional languages use stored data to perform recursive functions, which execute a process and then repeat it to solve any errors that arise during programming. Examples include Agda, Cuneiform, PureScript and APL.

**Machine languages**

Machine languages are made up of binary code, which is a series 0s and 1s that symbolize text or instructions for a computer program. One example of a machine language is Fortran.

**Assembly languages**

Assembly languages work in a similar way to machine languages by using short mnemonic codes to give the computer instructions. Examples include Lotus 1-2-3 and Turbo Pascal.

**Logic programming languages**

Logic programming languages add restrictions to statements made by developers that cause the computer to consider the possible outcomes of different actions. Examples include Prolog, ASP and Datalog.

**Data-oriented languages**

Data-oriented languages offer different ways to search and edit entity-relationship tables. Examples include Clarion, Gremlin, WebDNA and Wolfram Language.

**Business-oriented languages**

Companies use business-oriented languages to work with large quantities of data across a variety of different systems. Examples include SQL and COBOL.

**Education-oriented languages**

Education-oriented languages can help to teach computer programming and coding to novices who are unfamiliar with the processes. Examples include BASIC, Logo and HyperTalk.

**Object-oriented languages**

Object-oriented language identifies everything it encounters as objects that have internal and external data and then it performs based on moving these "objects" to where they need to be. Examples include Java, Visual Basic .NET, Ruby and Python.

**Scripting languages**

Scripting languages solve smaller programming issues and can be used to write operating system utilities. Examples include Perl, PHP, JavaScript and Python.

**Declarative languages**

Declarative languages tell a computer what needs to be done without including instructions for how to complete the task. This type of language can be classified as a logic programming or functional language, as they all help a computer to solve problems in programming. Examples include Prolog, Lisp, ML and Haskell.

**Document formatting languages**

Document formatting languages organize printed text and graphics, and some function similarly to a [word processor](https://www.indeed.com/q-word-processor-jobs.html). Examples include TeX, PostScript and SGML.

**World Wide Web display languages**

World Wide Web display languages are used to design web pages and provide them with the desired functions, such as page retrieval through links. Examples include HTML, XML and CGI.

**Front end coding languages**

Front end development languages are used to code the visual aspects of websites, games, software and apps. Examples include HTML, CSS and JavaScript.

**Database programming languages**

Database programming languages help to create databases and manipulate the way data is stored inside them. Examples include C++, COBOL, Java and Perl.

**Rule-based languages**

Rule-based languages implement rules once they are activated by certain conditions in a data set. Examples include AWK, CLIPS, Prolog and Wolfram Language.

**Compiled languages**

Compiled languages have been translated by computer programs from one programming language to another and convert information directly to code, which streamlines the programming process. Examples include ActionScript, Ballerina, C++ and ALGOL.

**Back end coding languages**

Back end coding languages code program servers so that web pages appear and function correctly. Examples include Python, Java and Ruby.

**System languages**

System languages can complete tasks like memory management or task management when programming an entire system. Examples include Swift, Rust, C++ and Nim.

**Algorithmic languages**

Algorithmic languages convey mathematical or symbolic computations and can use algebraic operations to convey information. Examples include Fortran, ALGOL, Lisp and C.

**Command-line interface languages**

Command-line interface languages use lines of text to send commands to computer programs. Examples include Batch, CLIST, TACL and 4DOS.

**Computational languages**

Computational languages declaratively communicate the logic and functions of a computer command. Examples include Wolfram Language, Gremlin, QML and Mercury.

**Visual languages**

Visual languages specify programs in two-dimensional ways through different types of graphic layouts. Examples include Grasshopper, GameMaker Language, XOD and ToonTalk.

**XML-based languages**

XML, or Extensible Markup Language, encodes documents in a way that can be read and understood by both people and computers and exists across the internet. Examples include Apache Ant, MXML, ECMAScript for XML and XQuery.

**Syntax handling languages**

Syntax handling languages generate analyzers that convert character sequences to token sequences that have specified meanings for context-free grammar. Examples include ANTLR, lex, Prolog and JavaCC.

**Interpreted languages**

Interpreted languages enable programs to be executed from source code with the help of an interpreter, rather than being compiled. Examples include Apache Ant, JavaScript, PostScript and Windows PowerShell.

**Little languages**

Little languages help to manage a specialized problem domain, such as text formatting, combinations and resource allocation. Examples include AWK, Comet, sed and SQL.

**Metaprogramming languages**

Metaprogramming languages write programs that write and edit other programs. Examples include C++, META II, Python and TREEMETA.

**Esoteric languages**

Esoteric languages test unorthodox methods of programming language design, such as using a fictional language as a basis for a program. Examples include Beatnik, INTERCAL, Piet and Whitespace.

**Non-English-based languages**

Non-English-based programming languages take keywords from languages other than English to create commands. Examples include Chinese BASIC, Lexico, Rapira and ezhil.

**Curly-bracket languages**

Curly-bracket languages use curly brackets or brace characters to define statement blocks. Examples include AWK, Ballerina, C++ and ECMAScript.

**Off-side rule languages**

Off-side rule languages designate blocks of code based on their indentation. Examples include ISWIM, ABC, Python and Elixir.

**Iterative languages**

Iterative languages are built around generators that solve smaller instances of a certain problem to inform how they can solve the original problem and generate a certain outcome. Examples include Aldor, Eiffel, Julia and Python.

**Constraint programming languages**

Constraint programming languages fall under declarative programming language and express relationships between their variables as constraints. Examples include MiniZinc, Oz and Kaleidoscope.

**Numerical analysis languages**

Numerical analysis languages are mainly used for technical computing. Examples include Wolfram Language, Analytica, Fortran and MATLAB.

**Multiparadigm languages**

Multiparadigm languages allow a program to use multiple programming styles to work at once in a single program and combine constructs from different programming languages. Examples include ALF, C++, ECMAScript and Python.

**Embeddable languages**

Embeddable languages are used in source code, for servers and by clients to embed code into free-form text. Examples include PHP, VBScript, ActionScript and JavaScript.

**Imperative languages**

Imperative languages convey information to computers through serial orders and large amounts of detail. They might also fall under other classifications as multiparadigm programming languages. Examples include MATLAB, ECMAScript, Perl and Python.

**Dataflow languages**

Dataflow languages use a representation of the exchange of data to specify programs and process streams of data. Examples include Analytica, Lucid, Oz and Ballerina.

**Authoring languages**

Authoring languages help to create interactive computer programs, such as tutorials or websites. Examples include Lasso, PILOT, TUTOR and Authorware.

**Concurrent languages**

Concurrent languages pass messages and offer language constructs for executing multiple processes at the same time. Examples include Ada, ChucK, Java and Oz.

**Array languages**

Array languages use scalars to apply operations to vectors, matrices and other high-dimensional arrays. Examples include Analytica, BASIC, MATLAB and Fortran 90.

**Extension languages**

Extension languages are embedded into other programs to use their features in extension scripts. Examples include JavaScipt, Perl, Squirrel and CAL.

**Hardware description languages**

Hardware description languages describe the design, structure and operation of electronic and digital logic circuits. Examples include Verilog, VHDL, Java and Ruby.

**Macro languages**

Macro languages can be for application or textual substitution to change one source code file into another, often in order to preprocess source code. Examples include C++, m4 and ML/I.

**Shading languages**

Shading languages use real-time rendering and offline rendering to create images, such as 3D computer graphics. Examples include AGAL, PSSL and RenderMan Shading Language.

**Reflective languages**

Reflective languages allow programs to examine and edit their high-level structure. Examples include Cobra, ECMAScript, Prolog and Ruby.

**Fourth-generation languages**

Fourth-generation languages are high-level languages built around database systems and are often used for managing databases and generating reports. Examples include ABAP, FOCUS, OpenEdge ABL and DataFlex.

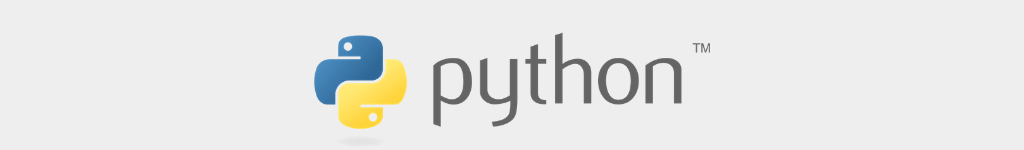
**Decision table languages**

Decision table languages clarify the logic of a program before writing it in another programming language. Examples include Filetab and FORTAB.

**Interactive mode languages**

Interactive mode languages allow for expressions to be entered and show their evaluation right away. Examples include BASIC, ECMAScript, Wolfram Language and Python.

### 1. Python



* Popularity: Very High
  + Salary expectations: Typically between USD$89,271 - $105,820 in the US
* Ease of Learning: Easy to Moderate
* Use Cases: General Use and Specialty
  + Web Applications
  + Artificial Intelligence

Python is a relatively new programming language, first introduced in 1989, that has surged in popularity with the emergence of new fields of application. It is an interpreted language that supports automatic memory management and object-oriented programming. It heavily prioritizes developer experience.

Python is very popular for general-purpose programming, including web applications. It has recently become known for specialty use in machine learning applications.

Python jobs are very plentiful, so it’s easy to find a job using Python, and there’s still plenty of room for growth.

#### Pros

* Readability and flexibility make Python suitable for a huge range of applications.
* Dynamic typing and asynchronous code help to speed up the development process.
* Can be learned very quickly by newbie developers.

#### Cons

* Python’s performance isn’t as good as some of its peers
* Executes a single thread at a time because of Python’s GIL.
* No native compatibility with iOS or Android is a big disadvantage for mobile developers.

### 2. C



* Popularity: Medium
  + Salary expectations: Typically between USD$71,433 and $95,168
* Ease of Learning: Moderate
* Use Cases: General Use and Specialty
  + Embedded systems
  + Hardware drivers
  + Local Applications

First introduced in 1972, C is well-established and enduring. Until Java was introduced, C was the dominant high-level language. The first versions of Unix, written in Assembly language, were ported to C. It was then used in the development of other early operating systems, including IBM System/370.

C has a long history of development on older systems with slower processors and little memory. Programs written in C had to be very efficient, so C has a reputation for high performance in cases where speed matters.

C is still very popular due to its use in systems development, including operating systems, embedded devices, and as firmware. The C standard library has been ported to many platforms, so it is viable in many use cases. However, the low-level systems programming it is typically used for is a more specialized skill than general application programming. This explains why C tends to have relatively few job openings as compared to other languages in the top 10.

There is likely to be some overlap in the jobs market with C++ (see the C++ listing below.)

#### Pros

* C is the basis for many other popular languages, so knowledge of C will serve you well
* Highly portable, readable and maintainable
* Fairly easy to learn

#### Cons

* Usually limited to “bedrock” code, like systems coding
* No safety nets, for example exception handling, to catch any errors as they are coded

### 3. C++



* Popularity: High
  + Salary expectations: Typically between USD$70,077 and $82,457
* Ease of Learning: Difficult
* Use Cases: General Use, Specialty
  + Local Applications
  + Web Services
  + Proprietary Services

C++ extends C with object-oriented features. The “double-plus” comes from the increment operator from C. C++ was developed to bring features from older languages to faster, more powerful platforms.

C++ occupies a similar area in the market to C, including systems programming and low-level hardware development. Over the years, the C++ standard libraries and specifications have been expanded considerably, leading to criticism that it has become overcomplicated and difficult to learn.

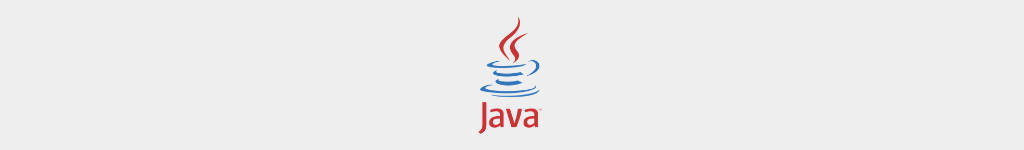
#### Pros

* Templating and inheritance make it easy to flexibly reuse design components
* A reputation for being very stable

#### Cons

* Often accused of being “bloated”
* C++’s complexity and abundance of features can compromise performance

### 4. Java



* Popularity: Very high
  + Salary expectations: typically between USD$84,316 - $112,332/yr in the US
* Ease of Learning: Moderate to Difficult
* Use Cases: General Use and Specialty
  + Web applications
  + Mobile
  + Embedded systems

Java is the leading general-purpose application development language and framework. It was introduced in 1991 by Sun Microsystems as a high-level, compiled, memory-managed language.

Java’s syntax is similar to C/C++, with curly braces for closures and semicolons to end statements. Automatic memory management is one of the features that made Java so popular after its initial release. Before Java was introduced, languages that required manual memory management, such as C and C++, were dominant. Manual memory allocation is tedious and error-prone, so Java was hailed as a major step forward for application developers.

The promise of Java, beyond memory management, was its cross-platform capability. This was marketed as “write once, run anywhere.” The Java Virtual Machine (JVM) runs Java bytecode, which is compiled from the Java language. JVMs are available for most major operating systems, including Linux, Mac, and Windows. It doesn’t always work perfectly, but when it does, a program written in Java can run on any platform with a compatible JVM.

Java is used for business, web, and mobile applications and is the native language for Google’s Android OS. Java also powers millions of set-top boxes and embedded devices. Java development skills are highly sought after.

If you’re considering a career in software development, you should strongly consider learning Java.

#### Pros

* Write Once, Run Anywhere: One version of Java code will run on any machine.
* Backwards compatibility: the newest versions of Java are still (mostly) compatible with even the oldest, making migrations painless.
* Because Java has been so big for so long, there’s a huge ecosystem of frameworks, libraries, and community support.

#### Cons

* The backwards compatibility principle is sometimes taken too far, extending the life of outdated and flawed features that should be retired.
* Greedy with memory and is a relatively verbose language, especially compared to the modern syntax of competitors like Python.

### 5. C#



* Popularity: High
  + Salary expectations: typically falls between USD$97,446 and $120,439
* Ease of Learning: Moderate
* Use Cases: General Use
  + Web Applications
  + Local Applications
  + Services/Microservices

C# was developed and introduced by Microsoft in 2000, along with the overall .NET framework. Syntactically, C# is very similar to Java and C/C++. It is a compiled, object-oriented language that compiles to .NET Intermediate Language. Originally, C# was used for Microsoft-focused development of Windows Forms and web development with ASP.NET. The .NET ecosystem has evolved recently with the introduction of the .NET Standard and .NET Core. These new frameworks and standards are cross-platform, running on Windows, Linux, and Mac.

C# is popular for local and web application programming, often (but not necessarily) in systems developed primarily based on Microsoft technology. Microsoft’s Xamarin framework allows developers to write Android and iOS applications in C#. It is suitable for systems programming in some cases, and has libraries available for embedded systems.

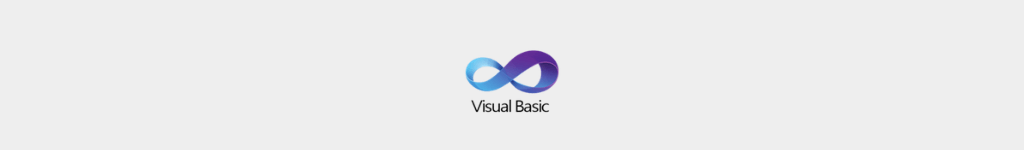
#### Pros

* Simple syntax makes C# more readable than predecessors
* Combined with .NET, powerful cross-platform flexibility
* Maintained by Microsoft, with strong documentation, updates, and community support

#### Cons

* Dependent on .NET for cross-platform and performance capabilities
* Slow and steep learning curve

### 6. Visual Basic .NET



* Popularity: Low
  + Salary expectations: approx. USD$84,000 on average
* Ease of Learning: Easy
* Use Cases: General Use
  + Web Applications
  + Local Applications

Visual Basic.NET (VB.NET) is Microsoft’s implementation of the Visual Basic language that compiles to .NET Intermediate Language. This allows developers to write .NET applications using Visual Basic and is a favorite for beginners. Applications written in VB.NET are just as capable as any. However, VB.NET was never very popular for business applications, with serious application developers preferring C, C++, and C#. Most applications written in VB.NET tend to be older and written for specific purposes, and are likely to be considered to be “legacy” applications destined for decommission or redevelopment.

#### Pros

* Beginner-friendly
* Ideal for spinning up a quick prototype

#### Cons

* Bloated syntax
* Code written in VB .NET tends to have poor performance

### 7. JavaScript



* Popularity: Very High
  + Salary expectations: typically between USD$97,858 and $120,182
* Ease of Learning: Moderate
* Use Cases: General Use
  + Local Applications
  + Web Applications

JavaScript is a high-level, dynamically typed, interpreted language. It uses Java-like syntax, hence the name JavaScript. JavaScript was first introduced in the early days of the public Internet, 1995. JavaScript is used to write code that runs in web browsers, on the client side. If you’ve been using the Web long enough to remember the introduction of Google Maps, you witnessed some of the first magic: the “infinite scrolling” in Maps is done using JavaScript.

Since its first introduction, JavaScript support has been added to all major web browsers. JavaScript frameworks including React, Angular, and Vue offer a Model-View-Controller application development paradigm, running entirely in the browser. JavaScript now supports the visual, browser-run elements of most modern web applications, which is why most Real User Monitoring tools cater for JavaScript.

JavaScript can also be combined with HTML to make cross-platform mobile applications. NodeJS is a web server that runs JavaScript on the server side. NodeJS applications are written entirely in JavaScript.

Given all these use cases and support, JavaScript is both popular and in high demand. It’s not very difficult to learn, though there are advanced programming techniques that take time to master. If you’re more comfortable with object-oriented languages, consider looking into TypeScript. TypeScript “overlays” object-oriented features and syntax, and transpiles to native JavaScript.

#### Pros

* Easy to learn and in wide demand
* Immense versatility (as evidenced by being so widespread)
* Rapid development and debugging

#### Cons

* JavaScript isn’t “strict” and won’t enforce good habits, leading to bad code from inexperienced developers
* Running client-side opens landmines in browser interpretation and security vulnerabilities

### 8. SQL



* Popularity: Very High
  + Salary expectations: typically between USD$81,506 and $101,426
* Ease of Learning: Easy to Moderate
* Use Cases: Specialty
  + Database Queries

SQL stands for Structured Query Language. SQL is used to query and modify data in a Relational Database Management System (RDBMS.) Vendor-specific implementations, such as PL/SQL (Oracle) and T-SQL (Microsoft) offer product-specific features.

SQL isn’t a general-purpose language that can be used to write applications. However, it is at least a useful, if not required skill of most developers. The term “full-stack developer” refers to a developer with a well-rounded skill set that includes all aspects of an application. This almost always includes accessing and saving data to a database. SQL is not hard to learn initially, though there are advanced use cases in Big Data and data analysis that require significant experience.

SQL is very popular with both developers and Database Administrators, so jobs that require SQL skills are plentiful. However, it is not a complete standalone skill set. SQL experience is a big plus on a resume, but it’s really only one of the skills required for any given job.

#### Pros

* Concise syntax and range of functions
* Invaluable for enterprise systems
* A valuable supplementary skill

#### Cons

* Reliant on a database engine to run
* Limited scope and capabilities

### 9. Assembly language



* Popularity: Low
  + Salary expectations: typically between USD$70,000 - $98,000
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Systems Programming
  + Hardware / Firmware development

“Assembly language” is a generic term for low-level code that closely represents the native machine instructions for a given microprocessor. Most of the languages on this list are “high-level” languages that are closer, syntactically, to English. High-level language code must be compiled down to an intermediate bytecode, or directly to machine instructions. Assembly code is assembled, (hence the name) not compiled.

The intent of a line of code written in C or Ruby is relatively easy to understand, just by reading it. Assembly, by contrast, is very difficult to understand without a careful reading of the entire program. Each operation, including math operations and moving data in and out of registers, is a complete statement. This means that it takes a lot more assembly code than C code to do the same amount of work.

Assembly code is most useful when performance is the most important goal. It is used for very low-level systems programming, or in some cases may be combined with application code for a performance boost. Jobs that require knowledge of assembly will include systems programming and hardware development, but it’s not a common core requirement.

#### Pros

* Minimal execution time and memory use
* Helps the developer to better understand machine-level computer architecture

#### Cons

* Very difficult to read and write
* A good supplementary skill, but not a first language

### 10. PHP



* Popularity: High
  + Salary expectations: typically between USD$77,239 - $102, 413 in the USA
* Ease of Learning: Easy
* Use Cases: General Use
  + Web Applications

PHP originally stood for “Personal Home Page” as part of its first name, PHP/FI (Forms Interpreter.) The official acronym is now PHP: Hypertext Processor. Its primary role is as a web application server-side scripting system. It was originally developed to extend a CGI program to support HTML forms and database access. The code of a PHP program is mixed in with the HTML, making it similar to Microsoft’s classic (pre-.NET) Active Server Pages. The interpreter reads the HTML and code, and executes the code portions of the page.

PHP is popular because it’s easy to learn. It is also the basis of popular web-based applications such as WordPress and Joomla. However, PHP also has a mixed reputation relating to software quality. Early versions lacked security controls and features that made it difficult to develop highly-secure applications. Recent developments in PHP frameworks and libraries have made improvements in security.

There are plenty of PHP jobs available for content-focused web applications like WordPress, and proprietary systems developed in PHP.

#### Pros

* A powerful backend language
* Stable and maintainable
* Excellent community support and mature ecosystem

#### Cons

* Better suited for web applications than desktop
* Some criticisms that PHP lacks security

### 11. R



* Popularity: Low
  + Salary expectations: typically between USD$74,619 - $95,401
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Statistical Computation and Analysis

The programming language R is primarily used by statisticians and researchers to perform statistical analysis of datasets. Demographers, insurance actuaries, and other jobs focused on statistics use R. Because of these practical fields of application, its code mass is considerable. As with MATLAB, most of the knowledge needed to work with R relates to statistics. R programmers combine statistics knowledge with the essentials of programming and software development.

Not many programmers are proficient in R, and the number of jobs also isn’t huge due to the specialized nature of the work. However, R knowledge can be a big advantage for data analysts doing statistical work. If that work sounds like something you want to look into, you should strongly consider adding R to your toolbox.

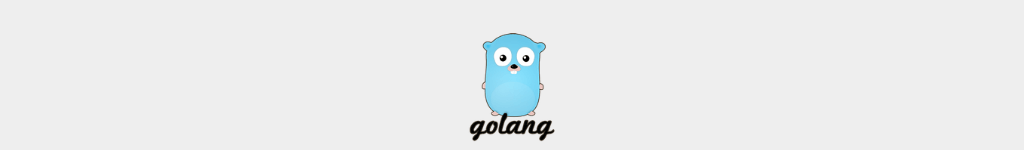
#### Pros

* Powerful specialized tool for academia and industry

#### Cons

* Limited applications

### 12. Go



* Popularity: Low
  + Salary expectations: approximately USD$102,329 on average
* Ease of Learning: Moderate
* Use Cases: General
  + Web Applications
  + Local Applications

Go (also known as Golang) is a relatively new kid on the block. It was introduced by two Google engineers in 2009. Go syntax borrows heavily from C and Java. The design goals for Go included cross-platform compatibility, simplicity, and support for modern processors.

Go is relatively easy to learn. It has some of the complexities of C/C++ (such as pointers) but its syntax and conventions are simpler. There’s a rapidly growing following in engineering and DevOps circles, and Go is a popular language in growing fields like machine learning. And because knowledge of Go is comparatively rare, roles that require proficiency in Go can be more lucrative.

#### Pros

* Growing in popularity driven by Google’s endorsement
* Fast and powerful
* Experienced programmers can learn relatively quickly

#### Cons

* Not yet as firmly established as its peers

### 13. Visual Basic



* Popularity: Low
  + Salary expectations: typically between USD$62,403 and $78,169
* Ease of Learning: Easy
* Use Cases: General
  + Local Applications

Visual Basic (VB) was introduced by Microsoft as a variant of the BASIC programming language. It is an event-driven language and Integrated Development Environment, primarily used to develop Windows applications. VB was designed to be easy to learn and to rapidly produce usable software. Visual Basic for Applications (VBA) is embedded in older versions of Microsoft Office applications, such as Access. VBA was used to provide programmatic manipulation of Office documents. Access databases used VBA to compose mini-applications.

Microsoft deprecated Visual Basic 6.0, the last version of Visual Basic, in 2008. While much code remains in the wild, it is no longer supported and jobs that require Visual Basic are dwindling. It is likely that any remaining jobs are focused on maintenance and/or porting to a modern platform.

#### Pros

* A good tool for beginners
* Extremely accessible and simple

#### Cons

* No longer relevant
* Microsoft-specific

### 14. MATLAB



* Popularity: Medium
  + Salary expectations: approx. USD$83,000 on average
* Ease of Learning: Moderate to Difficult
* Use Cases: Specialty
  + Mathematical Research

MATLAB is not a programming language per se. Strictly speaking, it’s an application used to calculate and model complex mathematical computations. It is used primarily in research settings, at universities and labs. MATLAB can handle complex matrix manipulations, and supports extensions to use complex mathematical notation. Functions written in C, C#, and FORTRAN can be called from MATLAB.

The knowledge needed to use MATLAB is more related to mathematical concepts and skills than knowledge of programming. If you’re already an advanced math student, for example working on a PhD in mathematics, MATLAB is relatively easy to learn. MATLAB is a valuable skill for those seeking a job in engineering or mathematics.

#### Pros

* Ideal for visualizing and manipulating technical data
* Many libraries and functions to sort common use cases

#### Cons

* Narrow specialty applications
* Requires mathematical background

### 15. Swift



* Popularity: Medium
  + Salary expectations: Typically between USD$74,406 - $99,129
* Ease of Learning: Moderate to Difficult
* Use Cases: Apple Mobile and Desktop applications
  + MacBook
  + iPhone
  + iPad

Apple introduced Swift in 2014 as a modern alternative to Objective-C. Its goals were to be easier to debug than Objective-C. Swift syntax is easier to read, and requires less code to do the same amount of work. However, poor version compatibility introduced with new versions may have stunted its adoption.

Despite a complicated history, there are a fair number of jobs available for Swift, and it is likely that it’s here to stay. As mentioned in the Objective-C listing below, if you want to develop for the Apple ecosystem, hedge your bets and learn both.

#### Pros

* Popularity is stabilizing after a rocky beginning
* Less verbose and more readable, fostering better developer experience

#### Cons

* Lingering backwards compatibility issues

### 16. Delphi/Object Pascal



* Popularity: Low
  + Salary expectations: typically between USD $80,000 - $150,000
* Ease of Learning: Moderate
* Use Cases: General
  + Local ppplications (mobile and desktop)
  + Web applications (frontend: transpilation to JS, backend: REST middlewares)

Delphi is a programming language and Integrated Developer Environment (IDE) for the Object Pascal language. Modern Object Pascal is an object-oriented, multi-paradigm derivative of procedurally oriented Pascal. Pascal was originally intended as a teaching tool, appeared in 1970.

This remains on the TIOBE list despite its age because there’s a lot of software out there written in Object Pascal with Delphi. New code is still being written in the language to this day. If you want to write software as a profession, Delphi and the Modern Object Pascal language is worth considering adding to your toolbelt.

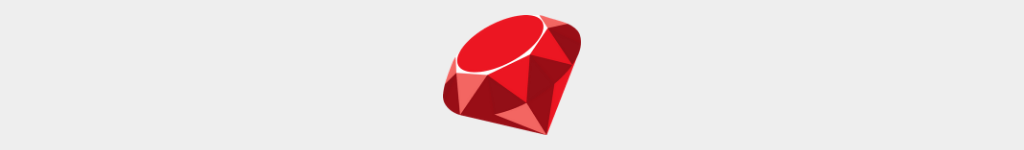
#### Pros

* Concepts like OOP and structuring maintainable code are transferable to other languages
* Supports multi platform development for mobile, desktop and the web.
* Efficient turnaround from idea to product.
* Modern, fast compilers.

#### Cons

* Job market fluctuates and isn’t as robust as distant cousin C#.
* The IDE and compilers are proprietary, which open-source advocates don’t like.

### 17. Ruby{ruby}



* Popularity: High
  + Salary expectations: typically between USD$71,811- $89,431
* Ease of Learning: Easy to Moderate
* Use Cases: General
  + Web Applications
  + Scripting

Ruby is an interpreted, dynamically typed, object-oriented language first introduced in the mid-1990s. It was inspired by several other languages, including Lisp, Perl, and Ada. Ruby is very popular for web application development. The Ruby on Rails framework (now known simply as “Rails”) is a model-view-component server-side framework written in Ruby.

Ruby is fairly easy to learn. Its common use in web applications and steady growth makes job opportunities easy to find.

#### Pros

* Easy to learn
* Healthy job prospects
* A leader in the web dev world

#### Cons

* Not a high-performance language
* Difficulty of debugging

### 18. Perl



* Popularity: High
  + Salary expectations: typically approx. USD$77,714 on average
* Ease of Learning: Easy to Moderate
* Use Cases: General
  + Local Applications
  + Web Applications

Perl was introduced in 1987 as a utilitarian scripting language, evolving from CGI scripting. Recent releases of Perl are quite different from early releases.

Perl is fairly easy to learn, but it has its detractors. The development of Perl was somewhat haphazard, leading to criticism that it is not well-organized. This has given Perl a reputation for being less than robust.

A lot of software has been written in Perl, and that continues to this day. Perl jobs are not hard to find. Having said that, it would be a stretch to say that Perl is a “modern” language. Perl may be a good language to learn early in a career as a way to get started, but it shouldn’t be the only one.

#### Pros

* Features like dynamic typing are designed to make development easier
* A practical and utilitarian language

#### Cons

* Use of special characters makes Perl an “ugly” language
* High CPU consumption

### 19. Objective-C



* Popularity: High
  + Salary expectations: typically approx. USD$104,000/yr in the USA
* Ease of Learning: Difficult
* Use Cases: Mobile Applications
  + Apple iOS devices: iPhone, iPad

Objective-C is a general purpose, compiled, object-oriented language. Its syntax is derived from Smalltalk. Until 2014, when Apple introduced Swift, it was the primary language used by Apple to develop applications for MacOS and iOS.

Objective-C is still relatively popular due to the large number of applications available that were written using it. Now that modern MacOS and iOS development is primarily in Swift, it’s likely that this popularity will eventually fall off as the number of supported applications tapers over time. Objective-C is not easy to learn. It uses syntax and language conventions that are not common to other languages, so experience with other languages does not apply well to Objective-C.

If you want to focus on software development for the Apple ecosystem, it’s a good idea to pick up both Objective-C and Swift. This will give you the ability to work on older applications written in Objective-C, and write new applications in Swift. Between the two, jobs are readily available.

#### Pros

* Still indispensable for iOS development

#### Cons

* Non-transferrable principles
* Will likely eventually be replaced
* Challenging to learn

### 20. Rust



* Popularity: Niche
  + Salary expectations: approx. USD$140,000/yr on average
* Ease of Learning: Difficult
* Use Cases: Specialty and General Use
  + Application/Software frameworks and components
  + Systems programming

Rust is a relatively new and powerful language that is popular for systems programming. Its design goals include safety, reliability, and speed. These features come at the cost of complexity and a high learning curve.

Rust is not a good first language, and is still a niche language. However, it’s become famous for having higher salaries associated with Rust than any other language, and the Rust community is vocal and passionate. Rust is great for a learning challenge, and there are jobs out there that use it, but if you’re just starting out, build your confidence in another language first.

#### Pros

* An extremely high-performance language
* Lucrative job market

#### Cons

* Difficult to learn
* Not as widespread as titans like Java or C#

### 21. Scratch



Popularity: Niche Salary expectations: n/a Ease of Learning: Easy Use Cases: Education

Scratch is a language designed to teach programming to beginners, developed by the Lifelong Kindergarten group at MIT. It’s designed to be simple and appealing, and allows users to create interactive stories, animations, games, and other interactive programs using a simple block-based visual interface. It’s free and open-source. Scratch is not suitable for professional use, and unless you want to teach programming, it has limited value. Jobs coding in Scratch are virtually non-existent, and its increased “popularity” is probably owing to growing interest in adding coding to early childhood curriculum.

#### Pros

* Simple, free, and accessible

#### Cons

* Not a professional programming language or career path

### 22. SAS



* Popularity: Low
  + Salary expectations: typically between USD$68,347 - $85,064
* Ease of Learning: Difficult
* Use Cases: General
  + Local Applications

SAS originally stood for “Statistical Analysis System” and was first developed in 1966 on mainframe computers. As the name suggests, it was used for statistical data analysis.

SAS is less common now as modern statistical analysis tools have taken the lead, though there are still some jobs available and certain industries still favour SAS over alternatives like R.

#### Pros

* Still relevant in certain industries

#### Cons

* Diminishing relevance, has been superseded by newer alternatives

### 23. Kotlin



* Popularity: Low
  + Salary expectations: typically $82,000-$113,000/yr in the USA
* Ease of Learning: Moderate
* Use Cases: Mobile Development
  + Android Applications

Kotlin runs on the Java VM and compiles to JavaScript. Its syntax is very similar to Java. Google supports Kotlin for Android OS development.

Kotlin has risen dramatically in popularity in the last 5 years, but it remains a niche language overall. It’s likely to remain a key player in the Android landscape, so if your interests lie there, Kotlin is a good option.

#### Pros

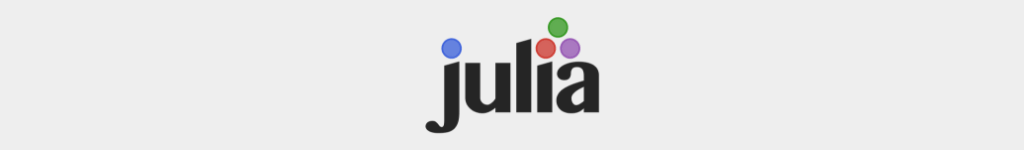
\*A strong asset for Android developers

* A fast-growing language

#### Cons

* Still a niche language with limited demand

### 36. Julia



* Popularity: Niche
  + Salary expectations: typically between USD$88,000 - $94,000 in the US
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Data Science

Julia is a relatively new language, introduced in 2012. Its syntax is a descendant of C. Julia was designed for data science and analytics applications.

Julia is another language that is closely associated with the work it is designed to support. If you are already in a data analytics job, there’s a good chance you have learned Julia.

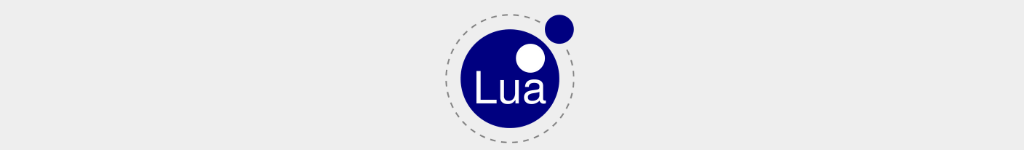
#### Pros

* Simple syntax and rapid development
* Jobs can be well-paid owing to specialty language requirements

#### Cons

* Slow to compile
* Poor interoperability with other languages

### 25. Lua



* Popularity: Niche
  + Salary expectations: Typically between USD$59,000 - $87,000
* Ease of Learning: Easy
* Use Cases: General Use
  + General Programming

Lua was developed in Brazil, during a time when trade policies made it prohibitively expensive to purchase licensed software from elsewhere in the world. It was designed to be portable and easy to learn for non-technical users. Lua has become popular among novice game developers as a scripting language.

Lua is a niche language with few jobs available. Learning Lua may make you a better, more well-rounded developer and help you get a foothold in the game dev world, but it isn’t likely to land you a job.

#### Pros

* Ease of learning suitable for beginners
* Portability

#### Cons

* Not in hot demand

### 26. Fortran



* Popularity: Niche
  + Salary expectations: typically USD$$65k - $86k on average
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Mainframe Programming

FORTRAN was developed by IBM for scientific and research-focused programming. It was originally introduced in 1957, making it one of the oldest languages on this list.

While FORTRAN’s popularity has waned over the decades, it has remained in use in specialty fields like physics and astronomy and in supercomputing. It’s risen in the TIOBE index in recent years as the demand has increased for massive volumes of complex data processing.

#### Pros

* Uniquely powerful for scientific computing purposes

#### Cons

* Still an obscure language with very specific applications

### 27. COBOL



* Popularity: Niche
  + Salary expectations: typically falls between USD$70,523 - $88,475
* Ease of Learning: Moderate to Difficult
* Use Cases:
  + Mainframe Application Development

COBOL is a very old language used primarily for mainframe development. It’s fairly difficult to learn in comparison to more modern languages.

Programmers that have been using COBOL for decades are enjoying high employability, due to the scarcity of COBOL programmers that are working and not retired. However, this is not a good reason to learn COBOL if you don’t already know it. Most COBOL code is being rewritten or replaced, so you’re better off investing in new skills for a new generation of languages and platforms.

#### Pros

* Still popular in government systems
* Relatively readable

#### Cons

* Becoming completely obsolete
* English-like syntax results in bloated and verbose reading

### 28. Lisp



* Popularity: Niche
  + Salary expectations: approx. USD$94,000 on average
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Systems programming

LISP is only one year younger than FORTRAN, making it one of the “great grandaddy” languages of this list. LISP has inspired the design of many younger languages on this list, including JavaScript, Scala, Python, and many others.

LISP isn’t very easy to learn, and has a very verbose syntax (it’s great if you LOVE typing parentheses!). However, LISP is another niche language that’s enjoying a resurgence because of the explosion of ML and AI programming, and programmers with established experience in LISP are suddenly at a premium. That doesn’t necessarily mean that it’s a good idea to start learning it as a beginner now, but LISP is also valuable for teaching a deep understanding of the languages that came after it.

#### Pros

* Understanding of LISP helps speed up learning of its descendant languages

#### Cons

* The revival of LISP for AI applications doesn’t necessarily mean future growth

### 29. Visual FoxPro



* Popularity: Niche
  + Salary expectations: typically between USD$58,000 - $79,000 in the USAe
* Ease of Learning: Moderate
* Use Cases: General

Visual FoxPro is a data-centric, object-oriented language and development environment created by Microsoft in 1992. It’s best suited for building database-driven applications, with support for SQL queries, data manipulation, and report generation.

The visual development environment makes it easier to design and build user interfaces and debug and test code.

#### Pros

* Easily and rapidly build Microsoft applications.

#### Cons

* No built-in support for web technologies such as HTML, CSS, and JavaScript.
* Only compatible with Windows, so can’t be used for building cross-platform applications.
* No longer actively developed or supported by Microsoft.

### 30. Ada



* Popularity: Niche
  + Salary expectations: approx. USD$89,000 on average
* Ease of Learning: Moderate
* Use Cases: General Use

Ada is a middle-aged language, first introduced in 1980. It’s most closely related to Pascal. Ada has many typical high-level language features, including static typing and object-orientation. It’s designed to encourage careful and secure programming, as evidence by its popular use in defense systems.

Ada is a niche language with a few jobs available. Most of that work is likely to be maintenance and porting to newer platforms.

#### Pros

* Simple syntax
* Relatively easy to learn
* Reliable and secure

#### Cons

* Not widely used outside of certain niche sectors
* Being replaced by Rust

### 31. Dart



* Popularity: Niche
  + Salary expectations: approx USD$83,000/yr on average
* Ease of Learning: Moderate
* Use Cases: General
  + Web Applications
  + Mobile Applications

Dart was introduced in 2011 by engineers at Google. It is a statically-typed, compiled, object-oriented language with a syntax similar to C. Dart transpiles to JavaScript.

Dart is still young and finding a foothold, but Flutter, a popular Dart framework, has helped boost demand and is especially popular for mobile development.

#### Pros

* A high-performance language
* Developer-friendly syntax
* Efficient cross-platform development with Flutter

#### Cons

* A new language with limited community support and demand

### 32. Scala



* Popularity: Moderate - High
  + Salary expectations: typically between USD$75,046 - $118,352
* Ease of Learning: Moderate to Difficult
* Use Cases: General Use
  + Software Frameworks
  + Web Applications

Scala was designed to fill gaps in Java, such as functional programming features. It has gained traction in the development of software libraries and applications.

Scala has been criticized for its steep learning curve. However, many of the principles learned in Scala apply neatly to other languages, including functional programming. There are plenty of Scala jobs available. It may not make a good starter language, but if you have some experience it’s worth looking into.

#### Pros

* Safe multi-threaded development
* Flexible and reliable

#### Cons

* Slow to compile
* Difficult and slow to get started with

### 33. Prolog



* Popularity: Niche
  + Salary expectations: typically between USD$84,000 - $114,000/yr in the USA
* Ease of Learning: Difficult
* Use Cases: Specialty
  + Artificial Intelligence

Prolog is a declarative language based on fairly advanced mathematical and logical theories. It is used for complex systems and can process sophisticated outputs in just a few lines of code.

However, to get there you have to learn Prolog’s syntax, which can be very difficult. Unless you happen to be in a field that uses Prolog, or want to get into AI or logic programming, it’s probably safe to skip it.

#### Pros

* Can do much more with fewer lines of code
* Well suited for complex database queries

#### Cons

* A steep and high learning curve
* Limited to specialized applications

### 34. D Lang



* Popularity: Niche
  + Salary expectations: D is currently too obscure to produce much reliable salary data
* Ease of Learning: Difficult
* Use Cases: Specialty

D is a general-purpose programming language intended to combine the performance of compiled languages with the ease and flexibility of interpreted languages. It’s statically typed and supports both imperative and object-oriented programming.

D’s expressive and readable syntax, garbage collection, and strong static type system all contribute to a high standard of developer experience. It has good support for concurrent and parallel programming, which makes it popular for building high-performance applications. However, D’s ecosystem lacks maturity and there’s been major inconsistency in standard libraries and poor packages. Overall, D is a solid language but growth prospects are uncertain.

#### Pros

* Suited to building high-performance applications
* Static typing will catch errors at compile time to improve reliability.

#### Cons

* Still fairly obscure
* Can be challenging to learn

### 35. PL/SQL



* Popularity: Low to Medium
  + Salary expectations: approx. USD$89,000/yr on average
* Ease of Learning: Moderate
* Use Cases: Database Queries
  + Oracle Databases

PL/SQL is the vendor-specific implementation of the SQL language. The syntax and features of PL/SQL align with features of Oracle databases. All dialects of SQL are moderately difficult to learn. Simple data querying and updating is fairly easy to learn. Joins, aggregation, and advanced concepts such as cursors require more understanding of database theory.

Oracle is a dominant database vendor, so PL/SQL jobs aren’t hard to find. If you are an Oracle Database Administrator, PL/SQL is a must-learn. Full-stack developers that work at the data “layer” should consider learning PL/SQL and other dialects.

#### Pros

* Backed and maintained by heavyweight Oracle
* A powerful tool for application development

#### Cons

* High memory consumption
* Niche uses

### 36. Bash



* Popularity: Low (see details)
  + Salary expectations: approx. USD$67,000 on average
* Ease of Learning: Easy to Moderate
* Use Cases:
  + Linux scripting and automation

Bash is a command processor, where scripts are used to automate tasks on Unix and Linux platforms. The syntax is fairly simple and easy to pick up.

Bash skills are required primarily for Unix/Linux administration, and in roles that need to write scripts for those platforms. Developers can use Bash to help automate software delivery pipelines. Bash is an important skill to develop for those use cases and others, but it’s not truly a full programming language alone.

#### Pros

* Very efficient within its specific purposes
* A powerful automation tool

#### Cons

* Only a component skill used to supplement knowledge of fully-fledged programming languages

### 37. Powershell



* Popularity: Medium
  + Salary expectations: typically falls between USD$68,355 - $101,697/yr
* Ease of Learning: Moderate - Difficult
* Use Cases: Windows
  + Scripting

PowerShell is a command-line shell and scripting language developed by Microsoft in 2006 based on Microsoft’s popular .NET framework. It’s designed for automating and managing tasks in Windows environments, although it can also be used on other platforms like Linux and macOS.

Powershell’s built-in cmdlets (command-line tools) provide access to a wide range of functions like file and folder management and system administration. It features support for object-oriented programming, event-driven programming, and a powerful scripting language.

PowerShell can be challenging to start out with for users who aren’t familiar with command-line interfaces or scripting languages.

#### Pros

* Included with all recent versions of Windows and is widely used in Windows development
* Has seen a heathy surge in popularity since becoming open source a few years ago

#### Cons

* In terms of performance, PowerShell isn’t your best option

### 37. Haskell



* Popularity: Niche
  + Salary expectations: typically fall between USD$68,172 - $98,607 in the USA
* Ease of Learning: Difficult
* Use Cases: General/Specialty

Haskell is one of the older purely functional, high-level languages, dating back to 1990. Functional languages focus on expressing intent through mathematical statements, rather than imperative subroutines that “act” on data. Many other functional languages derive from Haskell.

Haskell is somewhat difficult to learn, however, it is a good opportunity to learn functional programming. If you learn Haskell, you’ll learn this method of writing software, which is transferable to other languages and contexts. You’ll also be valuable to a niche market of jobs that need this specialized method of software development.

#### Pros

* Teaches valuable functional development skills
* Extremely maintainable
* Fast development

#### Cons

* Hard to learn
* Still fairly obscure and growth has been flat

### 39. Logo

* Popularity: Niche
  + Salary expectations: n/a
* Ease of Learning: Easy
* Use Cases: Education

Logo is a high-level programming language that was designed in the 1960s as a tool for teaching programming to kids. It uses turtle graphics, where a “turtle” moves around the screen while the user commands it draw lines and shapes.

Because it’s designed for kids, it’s visual-based and easy to learn and use, and more engaging and interactive for learners than conventional languages. With Logo being geared towards educational use, and especially towards young users, applications are extremely limited outside of this.

#### Pros

* Simple and accessible for learners

#### Cons

* Not a professional programming language or career

### 40. Transact-SQL

* Popularity: Niche
  + Salary expectations: approx. USD$72,000/yr in the USA
* Ease of Learning: Difficult
* Use Cases: General/Specialty
  + Microsoft database management

Transact-SQL (T-SQL) is a proprietary programming language developed by Microsoft for managing and querying data in Microsoft SQL Server and Azure SQL Database. It is an extension of the ANSI SQL standard and is used to create and modify database objects, insert and update data, and query and analyze data.

It integrates with other Microsoft technologies like Visual Studio and .NET, making it easier to build applications that interact with a SQL Server database. It may have a steep learning curve for users who are not familiar with SQL or database concepts.

#### Pros

* Works seamlessly with the rest of the Microsoft development suite

#### Cons

* Only supported by Microsoft SQL Server and Azure SQL Database
* a
* Vim script
* Viper (Ethereum/Ether (ETH))
* Visual DataFlex
* Visual DialogScript
* Visual FoxPro
* Visual J++ (Visual J plus plus)
* Visual LISP
* Visual Objects
* Visual Prolog
* WATFIV, WATFOR (WATerloo FORtran IV)
* WebAssembly
* WebDNA
* Whiley
* Winbatch
* Wolfram Language
* Wyvern
* X++ (X plus plus/Microsoft Dynamics AX)
* X10
* xBase++ (xBase plus plus)
* XBL
* XC (targets XMOS architecture)
* xHarbour
* XL
* Xojo
* XOTcl
* Xod
* XPL
* XPL0
* XQuery
* XSB
* XSharp (X#)
* XSLT
* Xtend
* Yorick
* YQL
* Yoix
* Z notation
* Z shell
* Zebra, ZPL, ZPL2
* Zeno
* ZetaLisp
* Zig
* ZOPL
* ZPL
* Z++

## 1. Python

**Purpose:**Python is a general-purpose dynamic programming language created in 1991.

**Popularity:**It’s one of the most popular coding languages today and is used for server-side web and system development.

**Pros and Cons:**The language has a simple syntax and can work on multiple platforms, making it far easier to learn and execute. It is one of the first programming languages that most coders learn. However, it has a lower execution speed compared to other languages.

**ALSO READ**: [What is Python Coding and Why it’s the Ticket to a Great Career](https://emeritus.org/blog/career-guide-what-is-python-coding/)

## 2. Java

**Purpose:**Created in 1995, the language is general purpose and object-oriented, allowing coders to develop everything from web and mobile applications to embedded servers. The language can be used on any platform and is one of the simplest languages to learn as it does not use features such as operator overloading and multiple inheritances. These factors make Java a far more user-friendly programming language.

**Popularity:**Java is one of the most popular programming languages worldwide.

**Pros and Cons:**Unlike most other languages, Java can be written on one device and run across various devices, making it portable. However, unfortunately, the language does not provide a backup facility and must be stored on the device’s storage.

## 3. C and C++

**Purpose:**The two programming languages are used interchangeably in daily conversation, with C’s creation in 1972 and C++ in 1985. Both are general-purpose programming languages, with their code used to implement operating systems such as Oracle and Intel.

**Popularity:**C++ is known as the ‘superset’ of C, with slightly more comprehensive grammar.

**Pros and Cons:**The languages are simple, compiled, and have comprehensive library support. However, both languages have similar issues in terms of excessive memory usage.

## 4. C#

**Purpose:**Similar to C++, C# is a general-purpose, object-oriented programming language. Developed by Microsoft in 2000, the language is popular among game developers and dynamic website creators.

**Popularity:**It has featured regularly in the top five [Popularity of Programming Languages Index](https://pypl.github.io/PYPL.html?src_trk=em66582394e1dc30.353875772089440585).

**Pros and Cons:**As it was initially created to rival Java, C# is highly versatile and can be easily understood by those with prior knowledge of C and C++. As a result, however, the performance of C# has been described as lackluster due to comparatively slower runtime.

## 5. JavaScript

**Purpose:** JavaScript is one of the primary languages used to code the World Wide Web.

**Popularity:**In addition to allowing for the development of web applications, JavaScript is the preferred language for many businesses as engineering teams only have to use just one programming language, making it simpler to debug and reducing costs throughout the development process.

**Pros and Cons:**JavaScript is easy to learn, has a simple structure, and can integrate well with other programming languages. However, unfortunately, users can view the code of websites, making it viable to compromise the security of website data.

## 6. SQL

**Purpose:**Created in 1974 at IBM, SQL (short for Structured Query Language) is a domain-specific programming language to access and manipulate databases to handle structured data.

**Popularity:**The International Organization for Standardization (ISO) made it the standard coding language in 1987.

**Pros and Cons:**It is fast and efficient, able to retrieve and process large amounts of data in short periods. Additionally, it is elementary to learn. However, the interface can be challenging to maneuver around.

## 7. PHP

**Purpose**: Described as specially suited for web development on their official [website](https://www.php.net/?src_trk=em66582394e1dc30.353875772089440585), the Hypertext Preprocessor language, popularly known as PHP, came into existence in 1994. It is an open-source language, free for downloading, and can be used on any platform, from Linux to Windows.

**Popularity:** A [survey](https://w3techs.com/technologies/details/pl-php?src_trk=em66582394e1dc30.353875772089440585) by the W3Techs shows that PHP is used by 77.3 percent of all the websites whose server-side programming language is known.

**Pros and Cons:** Its open-source nature is not as secure as other programming languages and unsuitable for coding larger content-based applications.

## 8. Go

**Purpose:**Go is an open-source programming language supported by Google, developed in 2009. It is a procedural language for everything from Web Development to Cloud and Network Services.

**Popularity:**As it is concise and easy to read, it is far simpler to learn. Further, the language’s ability to run multiple processes simultaneously has led popular websites to use it, including Uber, Netflix, Dropbox, and of course, its parent company, Google.

**Pros and Cons:**It has an extensive standard library that allows the creation of packages. However, it should be noted that the language lacks a User Interface (UI) tool kit.

## 9. Kotlin

**Purpose:**JetBrains developed Kotlin as a statically-typed, general-purpose programming language. It is interoperable with Java code with a concise syntax.

**Popularity:**As of 2019, Kotlin holds the title ‘[Preferred Language for Android App Developers](https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?src_trk=em66582394e1dc30.353875772089440585).’

**Pros and Cons:**Kotlin’s code has fewer bugs and is easier to learn than other languages due to its similarity with Java. However, as seen in other newer programming languages, the lack of an extensive user database also contributes to fewer resources and experts in the field.

## 10. MATLAB

**Purpose:**MathWorks developed this programming language in the late 1970s for engineers and scientists as a tool to establish a numeric computing environment. The typical language usage includes data analysis, algorithm modeling, and scientific computations.

**Popularity:**The language is platform-independent and can be used on multiple operating systems, adding to its popularity.

**Pros and Cons:**The software is not expensive and hence, more accessible than other coding languages. However, as MATLAB is an interpreted language, it takes longer to execute when compared to compiled languages such as C and C++.

## 11. R

**Purpose:**R serves statistical computing and graphics requirements. It was built in 1993 and has been used widely by business analysts. R is platform-free and open source. It allows integration of the language with others, such as C++.

**Popularity:**It is the eighth-most used programming language worldwide as of August 2020, according to the TIOBE Programming Community index.

**Pros and Cons:** It is a platform-independent language and is highly useful in machine learning operations. However, using R requires large amounts of storage as objects are stored in the physical memory. Hence, coding with significant data points would be more difficult.

## 12. Swift

**Purpose:** The Swift programming language, released in 2014, was developed by Apple Inc. to create iOS and macOS applications.

**Popularity:** The rise in the development of iOS applications mirrors Swift’s popularity.

**Pros and Cons:** Swift is an easy language to learn due to its open-source nature and ability to integrate C and C++ codes into the Swift application. However, as the language is still young, its user database is limited.

## 13. Rust

**Purpose:**Rust is a compiled programming language developed in 2010 that is general purpose and focuses on memory safety and reliability.

**Popularity:**It’s used across large companies such as Amazon, Facebook, and Microsoft.

**Pros and Cons:**It has a user-friendly compiler and strong editor with features such as auto-completion. However, the language is complex to learn and is slow in its compilation process.

## 14. Ruby

**Purpose:**Ruby is a general-purpose dynamic programming language designed in 1995 with an emphasis on the simplicity of the code.

**Popularity:**It is a high-level, interpreted language marketed as easy to write.

**Pros and Cons:**It is an open source code which may lead to reduced security. However, the code is difficult to debug and has slower processing when compared to other languages.

## 15. Dart

**Purpose:**Developed by Google in 2011, Dart is a client-optimized language for websites and mobile apps. It’s an open-source programming language, resembling Java and C.

**Popularity:**Its popularity is on the rise due to its similarity with Java and C, making Dart easy to use and adding to its charm.

**Pros and Cons:**It has high performance and runs faster than JavaScript. However, the lack of developer communities means fewer resources online, and the novelty of the language adds to the lack of support.

## 16. Scala

**Purpose:**Created in 2004, Scala is an object-oriented and functional programming language designed to address the disadvantages faced by users coding with Java.

**Popularity:**[According to Forbes](https://www.forbes.com/sites/forbestechcouncil/2021/12/22/why-scala-is-seeing-a-renewed-interest-for-developing-enterprise-software/?sh=43e4ce946c1f&src_trk=em66582394e1dc30.353875772089440585), programmers and developers are revisiting Scala and looking at it objectively for its advantages.

**Pros and Cons:**Scala has been marketed as a language that will avoid bugs in complex applications and can be compiled with JavaScript. However, it has a minimal developer pool, which decreases the resources around the code significantly.

Procedural Programming Languages

A programming paradigm that uses procedures or functions to abbreviate and categorize the code into reusable blocks is a procedural programming language. C, Pascal and FORTRAN are the most in-demand programming languages supporting this paradigm.

In this programming, the program is divided into functions or procedures. They are primarily insular sub-programs that perform a specific task. These procedures can be called from other program parts, allowing for flexible programming and code reuse. The priority is the systematic execution of a program. Further, it emphasizes the series of instructions and influences data stored in variables.

These are the best programming languages to learn as they are widely used in the fields like engineering, gaming and finance. Procedural programming can be less flexible though this does not stop it from being one of the most significant programming paradigms taught in high-ranking computer science courses.

Functional Programming Languages

Unlike procedural languages, functional programming languages are more flexible. It is composed of a series of functions. This programming paradigm stresses the use of model computations and data transformation. Haskell, Clojure, Lisp and Scala are the languages supporting this paradigm.

Functional programming makes programs easier to reason with and increases their reliability. The functions solely operate on their input arguments. They are less popular, but they have experienced a colossal boom from the educational point of view. They are assigned to variables, passed as arguments to other functions and return results from other functions.

Functional programming is efficient parallel programming. They have no mutable state. You can program functions and parallelly work as instructions. These codes support nested functions and consist of independent units that run coherently. Hence, this is more efficient.

Object-oriented Programming Languages

In an object-oriented programming language, objects define the data and the behavior of objects. These objects typically include data attributes representing the object's state and method. This language enables users to make a complex system with interconnected objects.

This language hides implementation components from the outside world through encapsulation. This makes it possible to build large intricate systems without stressing about the internal workings of respective objects. The other benefit that makes this language so in demand is the feature of inheritance. It creates a ranking of classes that share common features while still allowing customization.

Some popular object-oriented programming languages are Java, Python, C++, and Ruby. All these languages are top programming languages, but they share the principle of being object-oriented.

Scripting Languages

Simple to learn with easy syntax and dynamic typing, the scripting language is the type of language that is interpreted rather than compiled. The two types of this language are server-side scripting languages and client-side scripting languages. These languages make communication possible with other programming languages.

* Python - The easiest programming language used among developers is Python. It is an object-oriented programming language. The language has a high-Level data structure, and built-in libraries, that make it easy to use and suitable for rapid application development. It is easy, decoded and has a dynamic semantic language.
* Perl - The language is dynamic with innovative features that make it popular and different from what is available on Linux and Windows Server. Websites with high traffic usually use Perl, including IMDB, as it helps in text manipulation tasks.
* Bash - Bourne -Again, SHell is a scripting language that is the default command interpreter on most Linux/GNU operating systems. This language is easier than most of the other programming languages. Bash makes it easier to create script store documentation for others and provides useful reusable scripts.

Logic Programming Languages

As the name suggests, this is computer programming based on formal logic. This programming language program consists of a cluster of logical statements or rules that determine relationships among objects. It allows the system to extrapolate new information.

Artificial intelligence and expert systems commonly use this language where reasoning and conjecture are required. This language allows a concise and expressive program which is easier to reason about and maintain than programs that return to other paradigms.

To summarize, logical programming is a secure and flexible approach to solving problems in computer programming. It is suitable for every type of problem. It is a valuable tool for a few applications as well. One of the most popular logical languages is Prolog which consists of a set of facts and rules to describe a problem and reason about it.

Imperative Programming

In imperative programming, the programmer provides a set of instructions that the computer follows to manipulate the state of the program and the information structure within it. This paradigm describes the steps that a computer needs to take to solve a program rather than defining the mathematical function. C, C++, Java, and Python are some of the imperative programming languages,

The Imperative is the most popular programming language in software development for system programming and low-level programming tasks, which includes direct level control over hardware resources.

Front-end vs. Back-end Languages

Front-end languages are called client-side languages, which are used to create virtual and interactive website elements that users can see. The front-end languages include HTML, CSS and JavaScript.

Backend languages are known as server-side languages. They create logic and functionality behind the scenes of the website, like processing, storing and managing the user's account and authentication. Python, Ruby, and Java are some of the backend languages.

High-level vs. Low-level Languages

High-level languages are easy to read and write since they are understandable. Developers can write code at a higher level of abstraction without worrying about the underlying hardware or operating system, as it is close to natural languages. They have built-in libraries and frameworks to simplify ordinary programming tasks like managing data structures or executing calculations. Examples of high-level languages are Python, Java or Ruby.

On the other hand, level languages are much more complex to understand and write. The code written in languages is very efficient and directly manipulates the hardware and memory of the computer. Yet the developer needs to rely on manual efforts and understanding. Assembly language and C are examples of this type of language.

Interpreted vs. Compiled Languages

The interpreter directly executes and reads the interpreted code line by line without compiling it into machine code. It can be executed immediately without requiring separate compilation steps, such as Python, Ruby and JavaScript.

In contrast, the compiler first translates the compiled languages into machine code. Further converts the human-readable code into a structure that computers can execute. Languages like C, C++, and Java fall into this category.

What Programming Language Should You Learn?

With rigorous updates in the tech market, a candidate should be up-to-date with new languages. Though, other factors play a significant role in the decision of what programming languages to learn.

* Firstly, one needs to understand their goals and what career one wants to pursue after learning the programming language.
* Next, conduct thorough market research and narrow down the field you want to enter.
* After that, you need to understand the industry demand and the job portfolio of the desired profession. Learn the skills and languages that are demanded in the chosen field.
* You might want to start one of the easiest programming languages and gradually elevate your difficulty level. Python and Ruby are great languages, to begin with.
* Look for simply available material where you can learn quickly. Join communities and network with people at the top level of the field.
* Lastly, keep reinventing and learning as the market is constantly changing.

What Are the Best Programming Languages to Learn in 2024?

What coding and programming language should i learn? JavaScript and Python, two of the most popular languages in the startup industry, are in high demand. Most startups use Python-based backend frameworks such as Django (Python), Flask (Python), and NodeJS (JavaScript). These languages are also considered to be the best programming languages to learn for beginners.

Below is a list of the most popular and best programming languages that will be in demand in 2024.

1. Javascript  
2. Python  
3. Go  
4. Java  
5. Kotlin  
6. PHP  
7. C#  
8. Swift  
9. R  
10. Ruby  
11. C and C++  
12. Matlab  
13. TypeScript  
14. Scala  
15. SQL  
16. HTML  
17. CSS  
18. NoSQL  
19. Rust  
20. Perl

1. Javascript

JavaScript is a high-level programming language that is one of the core technologies of the World Wide Web. It is used as a client-side programming language by [97.8 percent](https://w3techs.com/technologies/details/cp-javascript/) of all websites. JavaScript was originally used only to develop web browsers, but they are now used for server-side website deployments and non-web browser applications as well.

Javascript was created in 1995 and was initially known as LiveScript. However, Java was a very popular language at that time, so it was advertised as a “younger brother” of Java. As it evolved over time, JavaScript became a fully independent language. Nowadays, JavaScript is often confused with Java, and although there are some similarities between them, the two languages are distinct.

Also Read: [Top 10 Reasons to Learn JavaScript](https://www.simplilearn.com/reasons-to-learn-javascript-article)

Javascript is the most popular programming language in the world and is in high demand among various organizations. The average Java developer earns around [$112,152](https://www.talent.com/salary?job=javascript+developer) each year.

|  |  |
| --- | --- |
| Level: | Beginner to Intermediate |
| Skills Needed: | HTML and CSS to define the content and layout of web pages |
| Platform: | Cross-platform (desktop, mobile, web) |
| Popularity Among Programmers: | The most popular programming language in the world |
| Benefits: | * Easy to learn and implement * Used everywhere on the web * Can run immediately within the client-side browser * Reduces the demand on the website server |
| Downsides: | Can sometimes be interpreted differently by different browsers which makes it difficult to write cross-browser code. |
| Degree of Use: | Widely used; highly applicable |
| Annual Salary Projection: | $112,152 |

. Python

Python is one of the most popular programming languages today and is easy for beginners to learn because of its readability. It is a free, open-source programming language with extensive support modules and community development, easy integration with web services, user-friendly [data structures,](https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-data-structure) and GUI-based desktop applications. It is a popular programming language for [machine learning](https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-machine-learning) and [deep learning](https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm) applications.

Python is used to develop 2D imaging and 3D animation packages like Blender, Inkscape, and Autodesk. It has also been used to create popular video games, including Civilization IV, Vegas Trike, and Toontown. Python is used for scientific and computational applications like FreeCAD and Abacus and by popular websites like YouTube, Quora, Pinterest, and Instagram. Python developers earn [average annual salaries of about $109,092](https://www.glassdoor.com/Salaries/python-developer-salary-SRCH_KO0,16.htm).

|  |  |
| --- | --- |
| Level: | Beginner – Python enables a beginner to become productive quickly |
| Skills Needed: | Problem-solving, abstract thinking |
| Platform: | Web, Desktop |
| Popularity Among Programmers: | Becoming continuously more popular |
| Benefits: | * Flexible * Naturally/Intuitively readable * Highly regarded official tutorials and documentation * Scripted as opposed to compiled |
| Downsides: | Doesn’t start with programming basics (known to abstract too many important basic concepts) |
| Popularity: | Becoming continuously more popular both in technical education and business uses |
| Degree of Use: | Coding skills widely used; popular in both technical education and business use |
| Annual Salary Projection: | $109,092 |

Also Read: [Introduction To Python Basics](https://www.simplilearn.com/learn-the-basics-of-python-article)

3. Go

Go was developed by Google in 2007 for APIs and web applications. Go has recently become one of the fastest-growing programming languages due to its simplicity, as well as its ability to handle multicore and networked systems and massive codebases.

[Go, also known as Golang](https://www.simplilearn.com/go-programming-language-article), was created to meet the needs of programmers working on large projects. It has gained popularity among many large IT companies thanks to its simple and modern structure and syntax familiarity. Companies using Go as their programming language include Google, Uber, Twitch, Dropbox, among many others. Go is also gaining in popularity among data scientists because of its agility and performance.

Go developers can earn an[average annual salary of $141,654](https://www.glassdoor.com/Salaries/golang-software-engineer-salary-SRCH_KO0,24.htm), and hitting up to $261,000.

|  |  |
| --- | --- |
| Level: | Beginner to intermediate |
| Skills Needed: | Easy to learn if you already know at least one other programming language; otherwise, you just need patience and a willingness to learn. |
| Platform: | Cross-platform, mainly desktop |
| Popularity Among Programmers: | Not as popular as Java or Python, but it has been increasing in popularity |
| Benefits: | * Widely considered a “minimalist” language * Easy to learn * Transparent code * Compatible * Fast |
| Downsides: | No generics (which means you may end up writing more code than you would in other languages); very little library support; Go developer community is not very robust or supportive |
| Degree of Use: | Widely used, particular for Google applications (created at Google) |
| Annual Salary Projection: | $141,654 |

4. Java

[Java](https://www.simplilearn.com/tutorials/java-tutorial/what-is-java) is one of the most popular programming languages used today.

Owned by Oracle Corporation, this general-purpose programming language with its object-oriented structure has become a standard for applications that can be used regardless of platform (e.g., Mac, Windows, Android, iOS, etc.) because of its Write Once, Run Anywhere (WORA) capabilities. As a result, Java is recognized for its portability across platforms, from mainframe data centers to smartphones. Today there are more than 3 billion devices running applications built with Java.

Java is widely used in web and application development as well as big data. Java is also used on the backend of several popular websites, including Google, Amazon, Twitter, and YouTube. It is also extensively used in hundreds of applications. New Java frameworks like Spring, Struts, and Hibernate are also very popular. With millions of Java developers worldwide, there are hundreds of ways to learn Java. Also, Java programmers have an extensive online community and support each other to solve problems.

Java is a more complex language to learn, but experienced developers with Java coding skills are in high demand. The average Java developer earns around [$109,225 each year](https://www.glassdoor.com/Salaries/java-developer-salary-SRCH_KO0,14.htm).

|  |  |
| --- | --- |
| Level: | Intermediate |
| Skills Needed: | Problem-solving, knowledge of the object-oriented structure |
| Platform: | Web, Mobile, Desktop |
| Popularity Among Programmers: | One of the world’s most popular; high demand |
| Benefits: | * Regarded as a good start for learning to think like a programmer and gain coding skills * You’ll be able to access/manipulate the most important computer functions, like the file system, graphics, and sound for any fairly sophisticated and modern program that can run on any operating system. |
| Downsides: | Lots of new vocabulary to learn; a higher-level language |
| Degree of Use: | Widely used; highly applicable |
| Annual Salary Projection: | $109,225 |

5. Kotlin

Kotlin is a general-purpose programming language originally developed and unveiled as Project Kotlin by JetBrains in 2011. The first version was officially released in 2016. It is interoperable with Java and supports functional programming languages.

Kotlin is used extensively for Android apps, web application, desktop application, and server-side application development. Kotlin was built to be better than Java, and people who use this language are convinced. Most of the Google applications are based on Kotlin. Some companies using Kotlin as their programming language include Coursera, Pinterest, PostMates among many others.

Kotlin developers earn an[average of $136,000 a year](https://www.ziprecruiter.com/Salaries/Kotlin-Salary), with the potential to earn up to $171,500.

|  |  |
| --- | --- |
| Level: | Intermediate to advanced |
| Skills Needed: | Prior experience with programming languages, particularly Java |
| Platform: | Web, Mobile, Desktop, Server |
| Popularity Among Programmers: | Increasingly popular; used for Android applications |
| Benefits: | * Less code-heavy than Java and other languages; * Relatively easy to adopt * Fully compatible with Java |
| Downsides: | Fluctuating compilation speed; no static keyword in Kotlin |
| Degree of Use: | Widely used; highly applicable |
| Annual Salary Projection: | $136,000 |

6. PHP

PHP is an open-source programming language created in 1990. Many web developers find it essential to learn PHP, as this language is used to build more than 80% of websites on the Internet, including notable sites like Facebook and Yahoo.

Programmers mainly use PHP mainly to write server-side scripts. But developers can also use this language to write command-line scripts, and programmers with high-level PHP coding skills can also use it to develop desktop applications.

PHP is considered a relatively easy language to learn for beginning developers. PHP professionals have access to several dedicated online communities, making it easy to get support and answers to questions.

On average, PHP programmers earn [average annual salaries of about $107,409.](https://www.glassdoor.com/Salaries/php-developer-salary-SRCH_KO0,13.htm)

|  |  |
| --- | --- |
| Level: | Beginner to intermediate |
| Skills Needed: | Simple if you have a background in programming languages, but relatively easy to learn for newcomers |
| Platform: | Cross-platform (desktop, mobile, web) |
| Popularity Among Programmers: | Used by nearly 80% of all active websites, so it’s very popular with web developers |
| Benefits: | * Open-source * Easy to develop and may be integrated with many different tools * Cost effective * Flexible with database connectivity |
| Downsides: | Not very secure; not well-suited for large applications; poor error-handling; cannot support a large number of apps |
| Degree of Use: | While it’s used by the vast majority of websites, its use is declining |
| Annual Salary Projection: | $107,409 |

Also Read: [Top 8 Practical Applications of PHP and Steps to Carve a Career in the Field](https://www.simplilearn.com/why-learn-php-article)

7. C#

Developed by Microsoft, C# rose to fame in the 2000s for supporting the concepts of object-oriented programming. It is one of the most used programming languages for the .NET framework. Anders Hejlsberg, the creator of C#, says the language is more like C++ than Java.

C# is best suited for applications on Windows, Android, and iOS, as it takes the help of the integrated development environment product, Microsoft Visual C++. C# is used on the back end of several popular websites like Bing, Dell, Visual Studio, and MarketWatch. C# developers earn around $[111,760 per year](https://www.glassdoor.com/Salaries/c-net-developer-salary-SRCH_KO0,15.htm).

|  |  |
| --- | --- |
| Level: | Intermediate |
| Skills Needed: | A basic understanding of how to write code |
| Platform: | Cross-platform, including mobile and enterprise software applications |
| Popularity Among Programmers: | Relatively popular, but not as popular as Java (its closest rival) |
| Benefits: | * Fast * Simple to use * Object-oriented * Scalable and updateable * Interoperable with other codes * Open source * Extensive library |
| Downsides: | Less flexible, as it depends on Microsoft’s .Net framework; poor x-platform GUI |
| Degree of Use: | Used by roughly one-third of all software developers, particularly relevant for Microsoft developers |
| Annual Salary Projection: | $111,760 |

8. Swift

A few years ago, Swift made the top 10 in the monthly TIOBE Index ranking of popular programming languages. Apple developed Swift in 2014 for Linux and Mac applications.

An open-source programming language that is easy to learn, Swift supports almost everything from the programming language Objective-C. Swift requires fewer coding skills compared with other programming languages, and it can be used with IBM Swift Sandbox and IBM Bluemix. Swift is used in popular iOS apps like WordPress, Mozilla Firefox, SoundCloud, and even in the game Flappy Bird. Professionals who develop iOS applications take home [average annual salaries of around $106,652](https://www.glassdoor.com/Salaries/ios-developer-salary-SRCH_KO0,13.htm).

|  |  |
| --- | --- |
| Level: | Beginner to intermediate |
| Skills Needed: | Willingness to spend time reading tutorials (which are widely available); no prior experience with programming languages required |
| Platform: | Mobile (Apple iOS apps, specifically) |
| Popularity Among Programmers: | Gaining in popularity, especially among Apple iOS application developers |
| Benefits: | * Relatively easy to learn * Clean syntax * Less code * Faster than comparable programming languages * Open source |
| Downsides: | Still a young language; poor interoperability with third party tools; lacks support for earlier versions of iOS |
| Degree of Use: | Used extensively for creating iOS apps used on the iPhone, iPad, and Apple Watch |
| Annual Salary Projection: | $109,652 |

9. R

[R is an open-source language](https://www.simplilearn.com/r-programming-language-business-analytics-quick-guide-article) that is essentially a different version of the S language. Much of the code that developers write for S runs on R without modification.

Applications built in R are used for processing statistics, including linear and nonlinear modeling, calculation, testing, visualization, and analysis. Applications coded using R can interface with a number of databases and process both structured and unstructured data.

R has a moderate learning curve and is not as easy for beginners to pick up as some other languages in this article. However, like other open-source programming languages, R boasts an active online community of developers, which is always a plus when learning new coding skills.

On average, R developers earn [average annual salaries of about $90,940](https://www.ziprecruiter.com/Salaries/R-Programmer-Salary).

|  |  |
| --- | --- |
| Level: | Intermediate |
| Skills Needed: | Easier if you already know Javascript or Python, but it’s recommended for people with a firm grasp of mathematics |
| Platform: | Mainly desktop |
| Popularity Among Programmers: | Not nearly as popular as it once was, mainly due to rival language Python’s soaring popularity |
| Benefits: | * Open source * Great support for managing data * Wide variety of packages available * Cross-functional with Linux, Windows, and Mac OS * Ideal for machine learning applications |
| Downsides: | Objects are stored in physical memory, which can strain resources; lacks basic security; slower than Matlab or Python |
| Degree of Use: | Widely used for analytics |
| Annual Salary Projection: | $90,940 |

10. Ruby

If you want to start with a language that is known for being relatively simple to learn, consider Ruby. Developed in the 1990s, it was designed to have a more human-friendly syntax while still being flexible from the standpoint of its object-oriented architecture that supports procedural and functional programming notation. A web-application framework that is implemented in Ruby is Ruby on Rails (“RoR”). Ruby developers tout it for being an easy language to write in and also for the relatively short learning time required. These attributes have led to a large community of Ruby developers and a growing interest in the language among beginning developers. The average salary for a Ruby developer is around [$121,000](https://www.glassdoor.com/Salaries/ruby-on-rails-developer-salary-SRCH_KO0,23.htm) per year.

|  |  |
| --- | --- |
| Level: | Beginner – Ruby and Ruby on Rails have evolved to become extremely popular for web developers. |
| Skills Needed: | Problem-solving, abstract thinking (ability to visualize what application users want to see) |
| Platform: | Web |
| Popularity Among Programmers: | On the rise |
| Benefits: | * Flexible * The syntax is considered easy to read and to write (no specialized “vocabulary” to get started) * Enforces good programming style |
| Degree of Use: | Least broadly used |
| Annual Salary Projection: | $121,070 |

11. C and C++

C is probably the oldest and popular programming language and is the root of other programming languages such as C#, Java, and JavaScript. C++ is an enhanced version of C. Many developers today skip learning C on its own, while others think learning C first provides a valuable foundation for C++ development. Both languages are widely used in computer science and programming.

C and C++ developers can make use of compilers for a wide variety of platforms, making applications developed in these languages largely transportable. Both C and C++ are considered high-performance languages. As such, they are widely used in developing applications where performance is a critical issue, such as client/server applications, commercial products like Firefox and Adobe, and video games. C and C++ developers earn [an average of $118,366 each year](https://www.glassdoor.com/Salaries/c-developer-salary-SRCH_KO0,11.htm).

|  |  |
| --- | --- |
| Level: | C – Intermediate to Advanced  C++ – Beginner to Intermediate |
| Skills Needed: | Problem-solving, basic computer knowledge. As C and C++ are geared toward low-level management of computer resources,  knowledge of computer functions such as memory management is beneficial. |
| Platform: | Mobile, Desktop, Embedded |
| Popularity Among Programmers: | * C – There has been a higher migration from C to C++ * C++ – One of the world’s most popular languages |
| Benefits: | * C – Used to learn the fundamentals of programming at the lowest (hardware) level * C++ – allows for a much higher “control” than other languages |
| Downsides: | * C – Coding in C is stricter, not very beginner-friendly language, the steeper learning curve * C++ – A bit more challenging to pick up and become productive with than C (and even more so than Java) |
| Degree of Use: | * C – One of the most widely used * C++ – Widely used |
| Annual Salary Projection: | $118,366 |

12. Matlab

Matlab is a proprietary programming language owned by MathWorks and originally released in the mid-1980s. It is built specifically for use by scientists and engineers.

Programmers use Matlab to build machine learning and deep learning applications. Matlab-based programs enable users to analyze data, create algorithms, process images, and verify research.

Generally, Matlab is easier to learn than other programming languages on our list. MathWorks’ website has an extensive section dedicated to answering questions about Matlab.

The average Matlab developer takes home an [average salary of $100,760 each year](https://www.ziprecruiter.com/Salaries/Matlab-Developer-Salary).

|  |  |
| --- | --- |
| Level: | Beginner |
| Skills Needed: | Basic knowledge of programming is recommended, but not required |
| Platform: | Mostly desktop |
| Popularity Among Programmers: | Not as popular as Python (its closest rival), but its popularity is increasing for hardware engineering and running visualizations |
| Benefits: | * Can be used to easily run and test algorithms * Easily debugged * Enables extensive data analysis and visualizations * Relatively simple to learn |
| Downsides: | As an interpreted (vs. compiled) language, it’s relatively slow; installation files take a significant amount of space on the computer |
| Degree of Use: | Not used extensively, outside of the science and research domains |
| Annual Salary Projection: | $100,766 |

13. TypeScript

TypeScript is a newcomer to top programming language lists, but it’s making headway. It was developed in 2012 by Microsoft and is a typed version of JavaScript that is well suited for large code bases. TypeScript is used to create JavaScript-based projects with typing in both client-side and server-side development, making useful for catching errors and preventing systemic issues.

TypeScript was named “most adopted tech” in the[2021 State of JavaScript Survey](https://stateofjs.com/). TypeScript developers typically earn an annual average of[$75,199](https://www.glassdoor.com/Salaries/typescript-developer-salary-SRCH_KO0,20.htm).

|  |  |
| --- | --- |
| Level: | Intermediate to Advanced |
| Skills Needed: | Basic knowledge of JavaScript required |
| Platform: | Web |
| Popularity Among Programmers: | Rapidly gaining popularity among programmers, especially for large-scale programs. |
| Benefits: | * Great for large-scale projects * It has the fastest growing GitHub community * Makes JavaScript development more efficient |
| Downsides: | You need to know JavaScript first. |
| Degree of Use: | Gaining heavy traction among web developers |
| Annual Salary Projection: | $75,200 |

14. Scala

Scala is a general-purpose, type-safe Java virtual machine language that combines the best oop and functional programming languages into one special high language. Scala is ideal for reducing and removing bugs in large, complex applications. It supports both object-oriented and functional programming.

Programmers can use Scala for any task that they normally would use Java for. Scala is a complex language, but that complexity gives it a lot of flexibility. Companies that use Scala include Netflix, Twitter, and the New York Times.

Scala developers earn a yearly average of[$127,005.](https://www.glassdoor.com/Salaries/scala-developer-salary-SRCH_KO0,15.htm)

|  |  |
| --- | --- |
| Level: | Complex |
| Skills Needed: | Basic knowledge of Java, C, C++, or Python recommended, but not required |
| Platform: | Cross-platform |
| Popularity Among Programmers: | Scala’s popularity has taken several hits, probably due to its complexity. It enjoys a good reputation in the fields of big data and machine learning. |
| Benefits: | * Faster than Python * Ideal for data analytics * Highly functional |
| Downsides: | ·       It’s very complex  ·       The SBT and compiler are clunky  ·       Limited developer pool |
| Degree of Use: | Extensive use in many companies’ data engineering infrastructure, especially data processing, web development, and distributed computing. |
| Annual Salary Projection: | $127,005 |

15. SQL

SQL is a standard database query language. It is used to access and manipulate data in databases. SQL is a declarative language that specifies the desired results, but not the steps to achieve those results. SQL is a powerful tool for accessing and manipulating data, and it is the world's most widely used database query language.

Benefits of SQL

* SQL is a standard database query language that enables users to manipulate and query data in a database quickly.
* SQL is widely used in many applications and environments, such as web applications, data warehouses, and e-commerce applications.
* SQL provides many benefits over other database query languages, such as improved performance, better data integrity, and more accessible data manipulation.

Cons of SQL

* If you are unfamiliar with programming or database concepts, SQL can be challenging to learn.
* SQL can be slow compared to other languages, mainly when working with large databases.
* And SQL is not well suited for certain types of tasks, such as complex mathematical calculations or machine learning.

16. HTML

[HTML](https://www.simplilearn.com/tutorials/html-tutorial/what-is-html)(HyperText Markup Language) is the standard language for creating web pages and applications. HTML is used to create web pages. You can use HTML to add images, links, and other types of content to your web page. HTML is a simple programming language; you don't need to know much about it to create a basic web page.

Images and other objects, such as interactive forms, can be embedded within the produced page using HTML structures. It enables the creation of structured documents by indicating structural semantics for text elements including as headings, paragraphs, lists, links, quotations, and other objects. HTML elements are delineated by tags, which are written in angle brackets.

Benefits of HTML

* Benefits of using HTML include creating well-structured, standards-compliant web pages that are easy to maintain and update.
* HTML is also easy to learn and is a great starting point for those new to web development.
* HTML is used to structure and present content on the web and is typically used alongside CSS and JavaScript.
* Additionally, HTML5 provides better multimedia and interactive content support, making it a powerful tool for creating engaging web experiences.
* HTML5 is the latest version and includes new features like video and audio elements, local storage, and 2D/3D graphics. HTML5 is designed to be more user-friendly and efficient than previous versions of HTML.

Cons of HTML

A few potential drawbacks exist to using HTML as a web development language.

* First, because HTML is a markup language, it is not as expressive as a programming language like JavaScript or PHP, which means that complex web applications or pages can be more challenging to develop in HTML.
* Additionally, HTML is not a very secure language, so web developers must carefully encode any user input to prevent security vulnerabilities properly.
* Finally, HTML can be somewhat challenging to learn for newcomers, as it has a lot of different elements and syntax rules

17. CSS

[CSS](https://www.simplilearn.com/tutorials/css-tutorial) (Cascading Style Sheets) is a style sheet language used to describe how a page that was produced in a markup language is presented. A style sheet, which is a set of rules for web browsers, can control an HTML or XML.

All HTML tags, including the text in the document's body, headings, paragraphs, and other text elements, are styled using CSS. The display of grid components, table elements, and picture

Benefits of CSS

* CSS is used to style all HTML tags, including the body of the document, headings, paragraphs, and other material. CSS can also be used to style how table components, grid elements, and images are displayed.
* Web developers use CSS to create responsive and accessible websites. CSS can make it easier for web developers to create websites that look good on all devices, including mobile phones and tablets.
* CSS can also help make websites more accessible to people with disabilities.
* CSS is easy to learn and use. Many tutorials and resources are available online, and anyone can start using CSS to style their web pages.

Con of CSS

* CSS can be challenging to debug. When there are errors in a CSS file, it can be difficult to track down the source of the problem.
* And it can be time-consuming to write. CSS files can be large and complex, and it can take a lot of time to create and maintain them.

Overall, the CSS is a powerful tool that can be used to style web documents. However, it is vital to know the potential drawbacks before using it.

18. NoSQL

NoSQL databases are non-relational databases designed to provide high performance and scalability. And NoSQL databases are often used in big data applications, where data is distributed across many nodes.

The four key categories of NoSQL databases are as follows:

* key-value stores,
* columnar stores,
* document stores,
* And graph databases.

Benefits of NoSQL

* NoSQL is a database system that does not use the traditional relational model.
* NoSQL databases are often used for big data applications that need to scale quickly.
* NoSQL databases can be faster and more scalable than relational databases.

Cons of NoSQL

* NoSQL databases are generally less mature than SQL databases and, as such, may lack some of the features and functionality that SQL databases offer.
* Additionally, because NoSQL databases are less standardized than SQL databases, it can be more challenging to find skilled personnel who can work with them.
* And NoSQL databases may be less compatible with existing applications and infrastructure than SQL databases.

19. Rust

Rust is a programming language designed to be safe, concurrent, and practical. It is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety. Rust is also memory-efficient: it uses minimal memory, making it ideal for embedded systems.

Benefits of Rust

* Rust is a fast and efficient language used to create high-performance applications.
* Rust is also a safe and reliable language, which makes it perfect for developing mission-critical software.
* Additionally, Rust is easy to learn and use and has a great community of developers who are always willing to help.

Cons of Rust

While Rust has many features that make it an attractive language, there are some drawbacks to using it.

* There needs to be more support or documentation available for other languages because it is a new language, making it difficult to learn and use Rust.
* Additionally, Rust only supports some platforms, and it can be challenging to integrate with existing codebases.

20. Perl

Perl is an interpreted, high-level, general-purpose programming language. Although Perl is not officially an acronym, various backronyms exist, including "Practical Extraction and Report Language." Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier. Since then, it has undergone many changes and revisions.

Perl is widely regarded as the " Swiss Army knife" of programming languages because of its versatility and power. It is used for various tasks, including web development, network programming, system administration, and more.

Benefits of Perl

* Perl is a powerful programming language with many features and applications. It is widely used in system administration, web development, network programming, and many other fields.
* Perl is easy to learn, and its concise syntax makes it an excellent choice for beginners. It is also very versatile, allowing you to write programs in various styles.
* Perl has excellent support for many databases, making it a good choice for database-driven applications. It also has excellent support for graphics and multimedia, making it a good choice for media-rich applications.

Cons of Perl

Although Perl is an interesting and feature-rich programming language, it also has some cons.

* First of all, it is more challenging to learn than some other languages like Python, and it has a very steep learning curve and can confuse beginners.
* Additionally, Perl is not as widely used as other languages, so there is not as much support available.
* Finally, it can be pretty slow compared to some other languages.

Conclusion

The domain of programming languages continues to evolve to meet the dynamic demands of technology and industry. It is expected that Python will remain a cornerstone for developers around the world given its versatility, scalability, and relevance to emerging technologies in the year 2024. Our analysis of the 20 most popular programming languages demonstrates that each language has unique strengths and opportunities, from established giants like Java and Python to newer entrants like Rust and Swift. Python, with its user-friendly syntax and extensive libraries, remains a preferred choice for beginners and seasoned professionals alike. And therefore, it's highly essential for developers to gain a detailed understanding of Python concepts through a comprehensive [Python training course](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCoursepages).