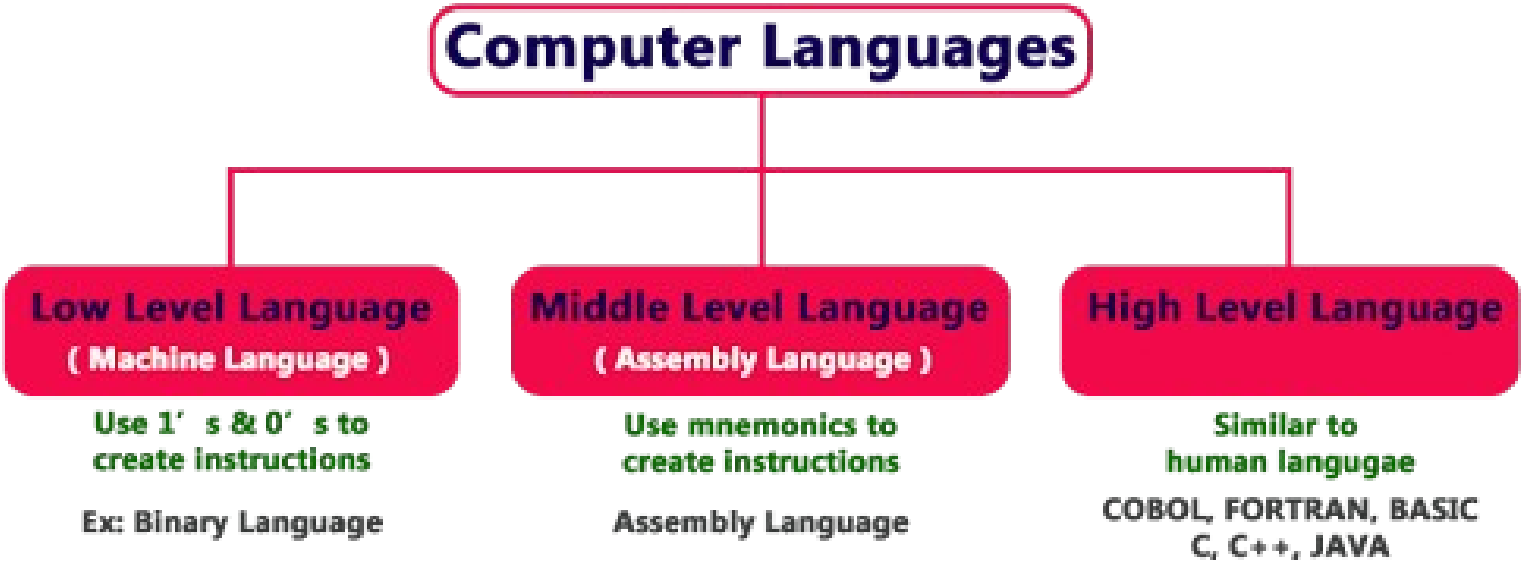
Introduction to Kotlin Language

Tushar B. Kute,
http://tusharkute.com

What is a Programming Language?


Computer Languages
Low Level Language
(Machine Language)
Use 1' s & 0' s to create instructions
Ex: Binary Language
Middle Level Language
(Assembly Language)
Use mnemonics to create instructions
Assembly Language
High Level Language
Similar to human langugae
COBOL, FORTRAN, BASIC
C, C++, JAVA

What is Kotlin?

Kotlin


Kotlin

What is Kotlin?

- Definition:
  - A modern, statically-typed, general-purpose programming language.
- Origin:
  - Developed by JetBrains (creators of IntelliJ IDEA).
- First Release:
  - 2011 (initial public release), 2016 (version 1.0).
- Purpose:
  - Designed to be a "better Java" – more concise, safer, and more productive.
- Core Principle:
  - Pragmatism – focusing on real-world developer needs.

The Philosophy Behind Kotlin

- Conciseness:
  - Less boilerplate code, more expressive syntax.
- Safety:
  - Strong emphasis on null safety to eliminate NullPointerExceptions.
- Interoperability:
  - 100% interoperable with Java (can seamlessly use Java libraries and frameworks, and vice-versa).
- Tool-Friendly:
  - Designed from the ground up to work well with IDEs, especially IntelliJ IDEA.
- Multi-Platform:
  - A single language for various environments.

Key Features (Conceptual Overview)

- Null Safety:
  - Explicitly distinguishes nullable and non-nullable types.
  - Safe call operator (?.), Elvis operator (?:), and !! (non-null assertion).
- Extension Functions:
  - Ability to extend existing classes with new functionality without inheritance.
  - Enhances readability and modularity.

Key Features (Conceptual Overview)

- Data Classes:
  - Concise way to create classes primarily used to hold data.
  - Automatically generates equals(), hashCode(), toString(), copy(), etc.
- Coroutines:
  - Simplified asynchronous programming.
  - Lightweight threads for concurrent operations.

Key Features (Conceptual Overview)

- Smart Casts:
  - The compiler automatically casts variables after type checks.
- Higher-Order Functions & Lambdas:
  - Functions that can take other functions as arguments or return them.
  - Enables functional programming paradigms.

Kotlin's Platform Targets

- Kotlin/JVM:
  - Compiles to JVM bytecode.
  - Used for server-side applications, desktop apps, and any Java-compatible environment.
- Kotlin/Android:
  - Official language for Android app development (since Google I/O 2019).
  - Leverages Android's existing ecosystem.
- Kotlin/JS:
  - Compiles to JavaScript.
  - Used for front-end web development.

Kotlin's Platform Targets

- Kotlin/Native:
  - Compiles to native binaries (no JVM required).
  - Targets iOS, macOS, Linux, Windows, WebAssembly, etc.
- Kotlin Multiplatform (KMP):
  - A framework for sharing business logic across multiple platforms (Android, iOS, Web, Desktop).
  - Aims for "write once, run everywhere" for core logic, while allowing native UI.

Advantages of Using Kotlin

- Increased Productivity:
  - Concise syntax reduces code lines.
  - Smart features (e.g., smart casts, data classes) save development time.
- Enhanced Code Quality & Safety:
  - Null safety minimizes common runtime errors.
  - Strong type system.
- Seamless Java Interoperability:
  - Easy migration from Java projects.
  - Can use vast existing Java libraries.

Advantages of Using Kotlin

- Strong Community & Ecosystem:
  - Active development by JetBrains and a growing open-source community.
  - Rich set of tools and libraries.
- Modern Language Features:
  - Incorporates best practices from other modern languages.

Who Uses Kotlin? (Adoption & Use Cases)

- Major Companies:
  - Google (Android development)
  - Netflix
  - Pinterest
  - Trello
  - Uber
  - Amazon

    (and many more)

Who Uses Kotlin? (Adoption & Use Cases)

- Common Use Cases:
  - Mobile Apps: Primary for Android, increasingly for cross-platform with KMP.
  - Backend Services: REST APIs, microservices.
  - Web Development: Front-end (Kotlin/JS) and full-stack.
  - Desktop Applications: With Compose Multiplatform.
  - Data Processing: Scripting, data analysis.

The Future of Kotlin

- Continued Evolution: JetBrains is committed to ongoing development and innovation.
- Kotlin Multiplatform Growth: Expected to be a major focus, enabling more code sharing.
- Performance Improvements: Ongoing compiler and runtime optimizations.
- Ecosystem Expansion: More libraries, frameworks, and tools.
- Community-Driven: The open-source community will continue to play a vital role.

Conclusion

- Kotlin is a powerful, modern, and pragmatic programming language.
- It addresses common pain points in software development (e.g., null pointer exceptions, verbosity).
- Its multi-platform capabilities make it a versatile choice for various development needs.
- With strong industry adoption and a vibrant community, Kotlin is a language with a bright future.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com