

Ringkasan Lengkap Bahasa Pemrograman dan Ekosistem Tooling

Dokumen ini merangkum peta besar, kurikulum universal, ringkasan tiap bahasa, runtime, toolchain, jalur belajar, roadmap 30/60/90 hari, dan cheat sheet command run/compile untuk lingkungan Windows yang kamu pakai.

Gambaran Utama

- Kelompok utama: bahasa pemrograman, runtime/platform, engine/toolchain/package manager, dan database/query system.
- Target penguasaan yang realistis: pahami sintaks, cara menjalankan, paradigma, ekosistem, use case, kapan dipakai, dan jebakannya.
- Fokus besar dokumen ini: fondasi universal -> peta per bahasa -> peta paradigma -> urutan belajar -> roadmap -> proyek mini -> ringkasan inti.

Isi Dokumen

1. Kurikulum universal yang wajib dikuasai di semua bahasa
2. Peta besar per kelompok bahasa dan tool
3. Ringkasan 36 bahasa/runtime/tool satu per satu
4. Peta paradigma dan cara berpikir
5. Urutan belajar paling efisien
6. Roadmap 30/60/90 hari
7. Template belajar per bahasa
8. Proyek mini dan pemetaan use case
9. Tingkat kesulitan umum
10. Catatan lingkungan yang kamu punya
11. Kesimpulan dan lapisan kompetensi

Bagian 1 - Kurikulum Universal yang Wajib dikuasai di Semua Bahasa

Apa pun bahasanya, fondasi yang harus dikuasai tetap sama. Jika fondasi ini kuat, pindah dari satu bahasa ke bahasa lain akan jauh lebih mudah karena kamu hanya mengganti sintaks dan ekosistem, bukan mengganti cara berpikir dari nol.

Syntax dasar

Fokus inti yang wajib dipahami:

- komentar
- variabel
- tipe data
- operator

Control flow

Fokus inti yang wajib dipahami:

- if dan else
- switch atau match
- loop for dan while

Fungsi

Fokus inti yang wajib dipahami:

- parameter
- return value
- scope
- recursion

Struktur data

Fokus inti yang wajib dipahami:

- array atau list
- map atau dictionary atau hash
- set
- stack dan queue

String dan text processing

Fokus inti yang wajib dipahami:

- indexing
- slicing
- formatting
- parsing

Input dan Output

Fokus inti yang wajib dipahami:

- baca input user
- baca atau tulis file
- stdout dan stderr

Error handling

Fokus inti yang wajib dipahami:

- exception
- error value

- result type

Modularitas

Fokus inti yang wajib dipahami:

- module
- package
- import
- dependency

Paradigma

Fokus inti yang wajib dipahami:

- procedural
- OOP
- functional
- concurrent atau asynchronous

Build dan run

Fokus inti yang wajib dipahami:

- interpreted
- compiled
- bytecode
- JIT
- AOT

Testing

Fokus inti yang wajib dipahami:

- unit test
- integration test

Debugging

Fokus inti yang wajib dipahami:

- print debug
- stack trace
- step debugger

Performance

Fokus inti yang wajib dipahami:

- kompleksitas waktu
- memori
- bottleneck

Proyek nyata

Fokus inti yang wajib dipahami:

- CLI
- web app
- API
- automation
- data processing
- GUI
- systems programming

Bagian 2 - Peta Besar Per Kelompok Bahasa dan Tool

Bahasa pemrograman

Python, C, C++, Java, C#, Go, Swift, Kotlin, Ruby, Rust, TypeScript, R, Julia, Haskell, Scala, Clojure, Elixir, Erlang, OCaml, Scheme, Bash, PHP, Lua, Tcl, Nim, Crystal, dan Assembly (NASM). Kelompok ini adalah inti kemampuan coding.

Runtime atau platform

Node.js, .NET, Deno, dan Bun. Ini bukan bahasa baru, tetapi lingkungan yang menjalankan atau mengelola bahasa tertentu.

Engine, toolchain, dan package manager

V8, npm, npx, Vite, dan Py launcher. Ini mengatur proses build, install dependency, dev server, atau pemilihan interpreter.

Database atau query system

SQLite. Ini bukan bahasa umum, tetapi SQL di dalamnya sangat penting untuk hampir semua programmer modern.

Makna praktisnya adalah: untuk benar-benar menguasai semuanya, kamu tidak cukup menghafal sintaks. Kamu harus tahu kapan sebuah alat dipakai, di mana posisinya dalam workflow, dan bagaimana menghubungkannya dengan tool lain.

Bagian 3 - Ringkasan 36 Bahasa, Runtime, dan Tool Satu Per Satu

1) Python 3.9.13

- Apa itu: bahasa serbaguna, sintaks sederhana, sangat kuat untuk automation, scripting, web backend, data science, AI/ML, dan tooling.
- Ciri utama: dinamis, interpreted, sangat kaya library, dan cocok untuk belajar cepat.
- Materi inti: variabel (int, float, str, bool), list, tuple, dict, set, if, for, while, function, lambda, class, object, file I/O, try/except, module/import, virtual environment, dan pip.
- Kelebihan: paling cepat dipakai produktif, bagus untuk prototyping, ekosistem raksasa.
- Jebakan: performa mentah lebih lambat dari C/C++/Rust, perlu disiplin struktur proyek, beda versi Python bisa memengaruhi package.
- Cocok untuk: pemula, data analyst, automation engineer, dan backend cepat.

Command dasar:

```
py -3 main.py  
python main.py
```

2) Py launcher

- Apa itu: launcher resmi Windows untuk memilih versi Python.
- Fungsi: membantu memilih interpreter ketika ada banyak versi Python terpasang.
- Praktik terbaik: di Windows sering lebih aman memakai `py -3 script.py` daripada bergantung pada python yang mungkin menunjuk versi lain.

Command dasar:

```
py -3 script.py  
py -3.14 script.py
```

3) C (gcc)

- Apa itu: bahasa sistem klasik yang sangat dekat ke memori dan hardware.
- Dipakai untuk: sistem operasi, embedded, library performa tinggi, dan dasar memahami komputer.
- Materi inti: tipe primitif, pointer, array, function, struct, memory manual, header file, preprocessor, compile dan link.
- Kelebihan: cepat, dekat ke mesin, dasar terbaik memahami memori.
- Jebakan: raw memory, buffer overflow, pointer bug, undefined behavior.
- Catatan lingkunganmu: gcc yang kamu sebut sangat tua, jadi fokus aman adalah C dasar dan uji kompatibilitas.

Command dasar:

```
gcc main.c -o main.exe  
.\main.exe
```

4) C++ (g++)

- Apa itu: perluasan C dengan OOP, template, STL, generic programming, dan performa tinggi modern.
- Materi inti: semua dasar C, class/object, constructor/destructor, inheritance/polymorphism, references, templates, STL, smart pointers, RAII, exception.
- Kelebihan: performa tinggi, fleksibel, dipakai di engine, finance, systems.
- Jebakan: kompleks, fitur sangat banyak, error compile kadang panjang.
- Fokus belajar: mulai dari C dasar, `std::string`, `std::vector`, class, lalu modern C++.

Command dasar:

```
g++ main.cpp -o main.exe
```

.\main.exe

5) Java (OpenJDK 21 LTS)

- Apa itu: bahasa OOP kuat, stabil, enterprise-class.
- Dipakai untuk: backend enterprise, Android legacy/core, sistem bisnis besar, aplikasi skala besar.
- Materi inti: class/object, method, encapsulation, inheritance, polymorphism, collections, exception, generics, streams, concurrency, JVM.
- Kelebihan: stabil, tooling matang, sangat bagus untuk sistem besar.
- Jebakan: verbose dan butuh struktur file/class yang rapi.
- Fokus belajar: OOP murni, collections, exception, file I/O, multithreading, build tools.

Command dasar:

```
javac Main.java  
java Main
```

6) .NET 8.0

- Apa itu: platform, bukan bahasa. Umumnya dipakai dengan C#, juga bisa F# atau VB.NET.
- Makna praktis: kalau orang bilang .NET, biasanya maksud kerja hariannya adalah C# di atas .NET.
- Materi inti C#: sintaks mirip C/Java, OOP, properties, LINQ, async/await, exception, generics, collections, dependency injection dasar.
- Kelebihan: produktif, tooling bagus, kuat untuk desktop, web, API, enterprise.
- Jebakan: harus paham ekosistem .csproj, SDK, dan package.

Command dasar:

```
dotnet new console -n App  
cd App  
dotnet run
```

7) Go 1.26

- Apa itu: bahasa sederhana, compiled, sangat kuat untuk backend, microservices, CLI tools, networking, dan concurrency.
- Materi inti: package, struct, interface, slice, map, error as value, goroutine, channel, pointer sederhana, module.
- Kelebihan: sintaks kecil, compile cepat, concurrency enak, deployment mudah.
- Jebakan: fitur bahasa lebih sedikit dan style Go ketat.
- Cocok untuk: backend, devops tools, high-concurrency services.

Command dasar:

```
go run main.go  
go build
```

8) Swift 6.2

- Apa itu: bahasa modern Apple untuk iOS/macOS, app development, dan modern systems-safe programming.
- Materi inti: let vs var, optional, struct/class, protocol, enum kuat, closure, error handling, generics.
- Kelebihan: modern, aman, sintaks bersih.
- Jebakan: optional harus benar-benar dipahami; ekosistem banyak berpusat di Apple.

Command dasar:

```
swift main.swift  
swiftc main.swift -o main.exe  
.\main.exe
```

9) Kotlin 2.3

- Apa itu: bahasa modern di JVM, populer untuk Android, backend, dan aplikasi JVM modern.
- Materi inti: null safety, class/data class, function, collection API, extension function, lambda, coroutines.
- Kelebihan: lebih ringkas dari Java, modern, nyaman dipakai.
- Jebakan: perlu paham interop dengan Java untuk proyek JVM besar.

Command dasar:

```
kotlinc Main.kt -include-runtime -d Main.jar  
java -jar Main.jar
```

10) Ruby 3.4

- Apa itu: bahasa scripting yang sangat ekspresif dan nyaman dibaca.
- Materi inti: semuanya objek, block, method, class/module, mixin, enumerable, file I/O, gems.
- Kelebihan: sintaks enak, produktif, bagus untuk automation dan web (Rails).
- Jebakan: performa mentah bukan yang tercepat dan style idiomatik perlu dibiasakan.

Command dasar:

```
ruby main.rb
```

11) Rust 1.93

- Apa itu: bahasa systems modern yang fokus pada keamanan memori, performa, dan zero-cost abstraction.
- Materi inti: ownership, borrowing, lifetimes (dasar), struct, enum, match, Result, Option, trait, crate/module, iterator, async (lanjut).
- Kelebihan: sangat aman, cepat, cocok untuk systems/backend/tooling.
- Jebakan: kurva belajar curam; ownership terasa berat di awal.
- Urutan belajar: variable + mutability -> ownership -> references -> struct/enum -> error handling -> traits -> cargo.

Command dasar:

```
rustc main.rs  
.\main.exe  
atau:  
cargo run
```

12) Node.js 24

- Apa itu: runtime JavaScript di luar browser.
- Dipakai untuk: backend, tooling, CLI, automation, dan full-stack JS.
- Materi inti: CommonJS vs ES Modules, file system, path, process/env, HTTP server, async/await, event loop.
- Kelebihan: sangat populer, ekosistem besar, cocok untuk web/backend cepat.
- Jebakan: dependency bisa ramai; async flow harus rapi.

Command dasar:

```
node main.js
```

13) V8

- Apa itu: JavaScript engine, bukan bahasa.
- Fungsi: menjalankan JavaScript secara cepat; dipakai di Node.js dan browser Chromium.
- Kenapa penting: untuk memahami JIT, garbage collection, dan relasi dengan event loop.
- Praktisnya: kamu tidak menulis program 'di V8' secara langsung; cukup pahami bahwa bahasanya JavaScript dan salah satu engine utamanya adalah V8.

14) npm

- Apa itu: package manager untuk ekosistem Node.js.
- Fungsi: install package, run script, manage dependency.
- Materi inti: package.json, dependencies vs devDependencies, scripts, versioning semver.

Command dasar:

```
npm init -y
npm install express
npm run dev
```

15) npx

- Apa itu: tool untuk menjalankan package sementara atau CLI tanpa install global permanen.
- Kapan dipakai: scaffold project dan menjalankan package CLI dengan cepat.

Contoh:

```
npx create-vite@latest
```

16) TypeScript 5.9

- Apa itu: superset JavaScript dengan static typing.
- Materi inti: type annotation, interface, type alias, union/intersection, generic, enum/literal type, type narrowing, module.
- Kelebihan: codebase besar lebih aman; tooling editor jauh lebih kuat.
- Jebakan: harus bisa membedakan type system dan runtime JS biasa.
- Fokus belajar: kuasai JS dulu, lalu type/interface, generic, dan tsconfig.

Command dasar:

```
tsc main.ts
node main.js
```

17) Vite

- Apa itu: build tool dan dev server modern untuk frontend.
- Fungsi: scaffold project, dev server cepat, bundling frontend.
- Kapan dipakai: React, Vue, Svelte, atau vanilla TS/JS frontend.

Contoh:

```
npm create vite@latest
npm install
npm run dev
```

18) Deno 2.6

- Apa itu: runtime modern untuk JavaScript/TypeScript.
- Materi inti: runtime TS/JS, module URL / package support, permission model, built-in tooling.
- Kelebihan: TypeScript terasa native, tooling bawaan rapi, security model bagus.
- Jebakan: ekosistem beda dari Node; beberapa package perlu adaptasi.

Command dasar:

```
deno run main.ts
```

19) Bun 1.3

- Apa itu: runtime dan toolkit cepat untuk JavaScript/TypeScript.
- Fungsi: runtime, package manager, bundler, test runner.
- Kelebihan: cepat, praktis, all-in-one.
- Jebakan: kompatibilitas beberapa ekosistem masih perlu dicek.

Command dasar:

```
bun run main.ts
```

20) R 4.5

- Apa itu: bahasa statistik dan analisis data.
- Materi inti: vectorized operation, dataframe, factor, statistik deskriptif, visualisasi, model statistik, package.
- Kelebihan: sangat kuat untuk statistik akademik dan analisis.
- Jebakan: gaya sintaks berbeda dari bahasa umum; environment perlu disiplin.
- Cocok untuk: statistika, penelitian, analisis data.

Command dasar:

```
Rscript main.R
```

21) Julia 1.12

- Apa itu: bahasa scientific computing modern.
- Materi inti: multiple dispatch, array numerik, package, performance-oriented code, scientific computing, plotting.
- Kelebihan: bagus untuk numerik, sintaks nyaman, cepat untuk komputasi ilmiah.
- Jebakan: ekosistem lebih kecil dari Python/R di beberapa domain.

Command dasar:

```
julia main.jl
```

22) SQLite 3.51

- Apa itu: database engine yang menjalankan SQL; bukan bahasa umum, tetapi sangat penting.
- Materi inti SQL: CREATE TABLE, INSERT, SELECT, UPDATE, DELETE, WHERE, ORDER BY, GROUP BY, JOIN, index dasar.
- Kenapa penting: hampir semua programmer modern butuh SQL.
- Cocok untuk: database lokal, app desktop, testing, prototyping.

Command dasar:

```
sqlite3 data.db
```

23) Haskell (GHC)

- Apa itu: bahasa functional murni, sangat bagus untuk melatih logika.
- Materi inti: immutability, function as first-class value, type system kuat, pattern matching, recursion, list processing, type class, monad (lanjut).
- Kelebihan: melatih cara berpikir bersih dan konsep functional.
- Jebakan: paradigmanya sangat berbeda dari bahasa mainstream.

Command dasar:

```
ghc Main.hs -o main.exe  
.\main.exe
```

24) Scala

- Apa itu: bahasa JVM yang menggabungkan OOP dan functional.
- Materi inti: object/class, collection API, case class, pattern matching, immutability, higher-order function, functional style.
- Kelebihan: ekspresif, kuat untuk data/backend/JVM.
- Jebakan: kompleksitas bisa naik cepat dan ada banyak style.

Command dasar:

```
scalac Main.scala  
scala Main
```

25) Clojure

- Apa itu: bahasa Lisp modern di JVM, functional, data-oriented.
- Materi inti: list/code as data, immutable data structure, REPL-driven development, function composition, map/filter/reduce, macro (lanjut).
- Kelebihan: sangat kuat untuk berpikir data dan REPL sangat produktif.
- Jebakan: sintaks Lisp (banyak kurung) butuh adaptasi.

Command dasar:

```
clj
atau:
clj -M -m your.namespace
```

26) Elixir

- Apa itu: bahasa functional modern di BEAM (VM Erlang).
- Materi inti: immutable data, pattern matching, pipe operator, recursion, process ringan, concurrency, OTP dasar.
- Kelebihan: concurrency sangat kuat; bagus untuk sistem realtime/distributed.
- Jebakan: paradigma functional dan OTP butuh waktu dipahami.

Command dasar:

```
elixir main.exs
```

27) Erlang

- Apa itu: bahasa untuk sistem concurrency dan fault-tolerance tinggi.
- Materi inti: actor model, process ringan, message passing, fault tolerance, OTP, distributed system.
- Kelebihan: sangat tangguh untuk sistem telekom dan realtime.
- Jebakan: sintaks dan gaya lebih tua; perlu adaptasi mental model.

Command dasar:

```
erl
```

28) OCaml

- Apa itu: bahasa functional dengan dukungan imperative dan type system kuat.
- Materi inti: immutable data, pattern matching, algebraic data types, module, recursion, type inference.
- Kelebihan: cepat dan sangat bagus untuk konsep bahasa dan compiler.
- Jebakan: ekosistem tidak sebesar bahasa mainstream.

Command dasar:

```
ocamlc main.ml -o main.exe
.\main.exe
```

29) Scheme (Guile)

- Apa itu: bahasa keluarga Lisp yang sangat bagus untuk memahami evaluator, recursion, symbolic programming, dan dasar ilmu komputer.
- Masalah yang kamu punya: WSL error karena .wslconfig bermasalah, jadi Guile belum bisa dipakai normal lewat WSL.
- Perbaikan: cek C:\Users\ASUS\.wslconfig, pastikan format valid, lalu jalankan wsl --shutdown dan buka lagi WSL.
- Materi inti: list processing, recursion, lambda, lexical scope, symbolic evaluation.

Command dasar setelah normal:

```
guile script.scm
```

30) Bash

- Apa itu: shell scripting di Linux/WSL.
- Materi inti: variable, positional args, condition, loop, function, pipes, redirect, grep/sed/awk dasar, file permission.
- Kelebihan: sangat kuat untuk automation server/devops.
- Jebakan: quoting/escaping dan perbedaan shell environment.

Command dasar:

```
bash script.sh
```

31) PHP 8.4

- Apa itu: bahasa scripting backend web klasik yang masih sangat dipakai.
- Materi inti: variabel, array, function, OOP, superglobal, file include, request/response, database access.
- Kelebihan: mudah deploy dan sangat populer untuk web backend.
- Jebakan: style lama dan modern sering bercampur.

Command dasar:

```
php main.php
```

32) Lua

- Apa itu: bahasa scripting kecil, ringan, embeddable.
- Materi inti: table, function, closure, metatable, module sederhana.
- Kelebihan: kecil, cepat, bagus untuk embedded/game scripting.
- Jebakan: standar library minimal.

Command dasar:

```
lua main.lua
```

33) Tcl

- Apa itu: bahasa scripting lama namun praktis untuk automation dan tooling tertentu.
- Materi inti: command-oriented syntax, variable, procedure, string/list handling, automation script.
- Kelebihan: ringan dan berguna untuk legacy tooling tertentu.
- Jebakan: sintaks terasa tidak biasa bagi pemula modern.

Command dasar:

```
tclsh main.tcl
```

34) Nim

- Apa itu: bahasa compiled modern yang terasa seperti campuran Python dan sistem language.
- Materi inti: static typing, indentation syntax, proc, object, metaprogramming, performance-oriented code.
- Kelebihan: cepat, sintaks nyaman, cocok untuk native tooling.
- Jebakan: ekosistem lebih kecil.

Command dasar:

```
nim c -r main.nim
```

35) Crystal

- Apa itu: bahasa compiled dengan rasa Ruby.
- Materi inti: syntax ala Ruby, static typing, class/module, compile native.
- Kelebihan: enak dibaca dan performa native.

- Jebakan: ekosistem lebih niche.

Command dasar:

```
crystal run main.cr
```

36) NASM

- Apa itu: assembler untuk assembly x86/x64; bukan bahasa tingkat tinggi.
- Fungsi: low-level programming, bootloader, interop, optimasi sangat rendah level.
- Materi inti: register, instruction, memory addressing, stack, calling convention, syscall/interoperability dengan C.
- Kelebihan: paling dekat ke mesin.
- Jebakan: sangat detail, sulit, satu bug kecil bisa fatal.

Command dasar:

```
nasm -f win64 main.asm -o main.obj  
gcc main.obj -o main.exe  
.\main.exe
```

Bagian 4 - Peta Paradigma dan Cara Berpikir

Procedural / imperative

C, Bash, Lua, Tcl. Fokus pada langkah demi langkah dan state yang berubah.

OOP-heavy

Java, C#, Kotlin, Swift, Ruby, C++. Fokus pada class, object, encapsulation, dan struktur sistem yang modular.

Functional-heavy

Haskell, Clojure, Elixir, Erlang, OCaml, Scheme, dan Scala (hybrid). Fokus pada immutability, function composition, recursion, dan expression over statement.

Systems / performance

C, C++, Rust, Go, Nim, NASM. Fokus pada performa, memori, dan perilaku low-level.

Scripting / automation

Python, Bash, Ruby, PHP, Lua, Tcl. Fokus pada produktivitas cepat dan automasi.

Data / scientific

Python, R, Julia, SQL/SQLite. Fokus pada analisis data, statistik, dan komputasi ilmiah.

Web / JS ecosystem

JavaScript (via Node), TypeScript, Node.js, Deno, Bun, Vite, npm, npx. Fokus pada tooling web modern dan full-stack.

Makna terpentingnya: bahasa yang berbeda sering kali terasa sulit bukan karena sintaksnya jauh berbeda, tetapi karena paradigma dan ekspektasi desainnya berbeda. Semakin cepat kamu mengenali paradigmanya, semakin cepat kamu menguasai bahasanya.

Bagian 5 - Urutan Belajar yang Paling Efisien

Tahap 1 - Fondasi universal

- Python
- C
- SQLite (SQL)
- Bash

Python untuk produktivitas, C untuk memahami mesin, SQL untuk data, dan Bash untuk automation.

Tahap 2 - Bahasa kerja modern

- JavaScript + Node.js
- TypeScript
- Java
- C#/.NET
- Go

Ini membentuk fondasi kerja industri dan web/backend modern.

Tahap 3 - Native dan performa

- C++
- Rust
- Swift
- Kotlin

Tahap ini memperkuat kemampuan native, performa, dan sistem modern.

Tahap 4 - Data / scientific

- R
- Julia

Tahap ini memperkuat statistik, numerik, dan komputasi ilmiah.

Tahap 5 - Functional dan akademik

- Haskell
- Scala
- Clojure
- Elixir
- Erlang
- OCaml
- Scheme

Tahap ini memperkuat logika, paradigma, dan pemahaman desain bahasa.

Tahap 6 - Niche tapi berguna

- Ruby
- PHP
- Lua
- Tcl
- Nim
- Crystal
- NASM

Tahap ini memperluas jangkauan praktis dan spesialisasi.

Bagian 6 - Roadmap 30/60/90 Hari

Hari 1 - 30

Fokus pada fondasi universal: Python, C, SQL (SQLite), Bash, dan JavaScript via Node.js.

Target:

- Paham variabel, tipe data, kondisi, loop, dan fungsi
- Bisa input/output
- Bisa baca/tulis file
- Bisa query SQL dasar
- Bisa menjalankan script di Windows

Mini proyek:

- Kalkulator CLI
- Todo list file-based
- Parser teks sederhana

Hari 31 - 60

Fokus pada bahasa industri utama: TypeScript, Java, C#/.NET, dan Go.

Target:

- Paham OOP
- Menggunakan collections
- Error handling
- Module/package
- Build & run
- Project structure

Mini proyek:

- REST API sederhana
- CRUD CLI
- Log parser
- Data transformer

Hari 61 - 90

Fokus pada bahasa lanjut dan paradigma: C++, Rust, Kotlin, Swift, R, Julia, lalu dasar Haskell/Scala/Clojure/Elixir, dan survey praktis PHP/Ruby/Lua/Nim/Crystal.

Target:

- Pahami beda paradigma (OOP, functional, systems, scripting)
- Tahu kapan memilih bahasa tertentu
- Bisa compile/run tool utama di Windows

Mini proyek:

- Benchmark sederhana
- Parser lebih cepat (C++/Rust/Go)
- Analisis data kecil (R/Julia/Python)
- Functional exercise

Bagian 7 - Template Belajar Per Bahasa

Pakai pola ini untuk setiap bahasa. Dengan pola yang sama, kamu membangun memori belajar yang konsisten, jadi perpindahan bahasa menjadi lebih efisien.

Level 1 - Dasar

- hello world
- variabel
- tipe data
- operator
- kondisi
- loop
- fungsi
- input/output

Level 2 - Menengah

- array/list/map
- string processing
- file I/O
- module/package
- error handling
- testing dasar

Level 3 - Lanjut

- OOP / struct / class / trait / protocol
- generic / template / type system
- concurrency / async
- performance
- project layout
- dependency management

Level 4 - Mahir

- idiomatic style
- debugging
- profiling
- build/release
- packaging
- library ecosystem
- architecture per domain

Bagian 8 - Proyek Mini dan Pemetaan Use Case

Agar ilmu benar-benar nempel, tiap bahasa minimal punya proyek latihan yang nyata. Tujuannya bukan sekadar selesai, tetapi melatih I/O, struktur data, modularitas, dan debugging.

- Calculator CLI
- Todo list file-based
- Parser teks / log
- Data cleaner sederhana
- HTTP server kecil (jika cocok)
- CRUD sederhana (jika ada DB)
- Algorithm practice (sorting, search, recursion)

Pemetaan bahasa ke proyek paling cocok:

- Python: automation, data tool, scraper, CLI
- C: memory tool, parser cepat, dasar sistem
- C++: engine, high-performance app
- Java: enterprise backend
- C#/.NET: desktop, API, enterprise tool
- Go: web API, concurrent service, CLI
- Swift: Apple app, native modern
- Kotlin: Android, JVM app
- Ruby: script, Rails-style web
- Rust: safe systems tool, fast CLI/backend
- Node/TS: web backend, tooling, full-stack
- Deno/Bun: runtime modern JS/TS
- R: statistics/reporting
- Julia: numerik/scientific
- SQLite: local DB app
- Haskell/Clojure/OCaml: konsep, compiler, logic-heavy tools
- Elixir/Erlang: realtime/distributed system
- PHP: web backend
- Lua: embedded/game scripting
- Nim/Crystal: native tool modern
- NASM: low-level/interop

Bagian 9 - Tingkat Kesulitan Umum

Paling mudah masuk

- Python
- JavaScript
- Ruby
- Lua
- PHP
- Bash dasar

Menengah

- Java
- C#
- Go
- Kotlin
- Swift
- TypeScript
- R

Lebih sulit

- C
- C++
- Rust
- Haskell
- Scala
- Clojure
- OCaml
- Elixir/Erlang
- NASM

Maknanya: mulai dari bahasa yang mudah produktif, lalu naik ke bahasa yang sulit tetapi memperkuat fondasi. Strategi ini jauh lebih efisien daripada langsung melompat ke bahasa paling rumit.

Bagian 10 - Catatan Lingkungan yang Kamu Punya

- Python + Py launcher: sangat bagus. Untuk Windows, sering paling aman memakai py -3 agar interpreter yang dipilih konsisten.
- gcc untuk C: terlihat jauh lebih tua dibanding tool lain. Prioritaskan C dasar dan uji kompatibilitas sintaks.
- C++ / Rust / Go / .NET / Java / Node / TypeScript: ini sudah membentuk stack modern yang sangat kuat.
- Node + Deno + Bun: sangat bagus untuk eksperimen runtime JS/TS dan benchmarking sederhana.
- Scheme/Guile: saat ini tertahan masalah WSL, jadi konfigurasi WSL perlu dibenahi dulu.
- SQLite: sangat penting walau bukan bahasa umum. Jangan dilewatkan.
- NASM: jangan dijadikan titik awal, tapi sangat bagus kalau tujuanmu memahami level mesin.

Bagian 11 - Cheat Sheet Command Run/Compile untuk Windows

Ini adalah command dasar paling aman dan umum untuk menjalankan atau meng-compile tool yang kamu punya.

Python

```
py -3 main.py  
python main.py
```

C

```
gcc main.c -o main.exe  
.\main.exe
```

C++

```
g++ main.cpp -o main.exe  
.\main.exe
```

Java

```
javac Main.java  
java Main
```

.NET (C#)

```
dotnet new console -n App  
cd App  
dotnet run
```

Go

```
go run main.go  
go build
```

Swift

```
swift main.swift  
swiftc main.swift -o main.exe  
.\main.exe
```

Kotlin

```
kotlinc Main.kt -include-runtime -d Main.jar  
java -jar Main.jar
```

Ruby

```
ruby main.rb
```

Rust

```
rustc main.rs  
.\main.exe  
atau:  
cargo run
```

Node.js

```
node main.js
```

TypeScript

```
tsc main.ts  
node main.js
```

Deno

```
deno run main.ts
```

Bun

```
bun run main.ts
```

R

```
Rscript main.R
```

Julia

```
julia main.jl
```

SQLite

```
sqlite3 data.db
```

Haskell

```
ghc Main.hs -o main.exe  
.\main.exe
```

Scala

```
scalac Main.scala  
scala Main
```

Clojure

```
clj  
atau:  
clj -M -m your.namespace
```

Elixir

```
elixir main.exs
```

Erlang

```
erl
```

OCaml

```
ocamlc main.ml -o main.exe  
.\main.exe
```

Bash

```
bash script.sh
```

PHP

```
php main.php
```

Lua

```
lua main.lua
```

Tcl

```
tclsh main.tcl
```

Nim

```
nim c -r main.nim
```

Crystal

```
crystal run main.cr
```

NASM

```
nasm -f win64 main.asm -o main.obj  
gcc main.obj -o main.exe  
.\main.exe
```

npm / npx / Vite

```
npm init -y
```

```
npm install  
npm run dev  
npx create-vite@latest
```

Bagian 12 - Contoh Kode Awal Universal

Tes pertama yang paling sederhana untuk semua bahasa adalah Hello, World!. Tujuannya memastikan toolchain berjalan, command benar, dan encoding output normal.

Output uji awal:

Hello, World!

Python

```
print("Hello, World!")
```

C

```
#include <stdio.h>
int main(void){ printf("Hello, World!\n"); return 0; }
```

C++

```
#include <iostream>
int main(){ std::cout << "Hello, World!\n"; }
```

Java

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

C# (.NET)

```
Console.WriteLine("Hello, World!");
```

Go

```
package main
import "fmt"
func main(){ fmt.Println("Hello, World!") }
```

Swift

```
print("Hello, World!")
```

Kotlin

```
fun main() {
    println("Hello, World!")
}
```

Ruby

```
puts "Hello, World!"
```

Rust

```
fn main() {
    println!("Hello, World!");
}
```

JavaScript (Node)

```
console.log("Hello, World!");
```

TypeScript

```
console.log("Hello, World!");
```

Deno

```
console.log("Hello, World!");
```

Bun

```
console.log("Hello, World!");
```

R

```
print("Hello, World!")
```

Julia

```
println("Hello, World!")
```

Haskell

```
main = putStrLn "Hello, World!"
```

Scala

```
object Main extends App {  
  println("Hello, World!")  
}
```

Clojure

```
(println "Hello, World!")
```

Elixir

```
I0.puts("Hello, World!")
```

Erlang

```
-module(main).  
-export([start/0]).  
start() -> io:format("Hello, World!~n").
```

OCaml

```
print_endline "Hello, World!"
```

Scheme

```
(display "Hello, World!")  
(newline)
```

Bash

```
echo "Hello, World!"
```

PHP

```
<?php  
echo "Hello, World!\n";
```

Lua

```
print("Hello, World!")
```

Tcl

```
puts "Hello, World!"
```

Nim

```
echo "Hello, World!"
```

Crystal

```
puts "Hello, World!"
```

NASM sengaja tidak dijadikan titik awal contoh universal, karena untuk tahap pertama lebih praktis naik dari C dulu, baru masuk ke assembly.

Bagian 13 - Lapisan Kompetensi dan Kesimpulan Besar

Jika tujuanmu adalah 'menguasai semuanya', target realistisnya bukan menghafal seluruh sintaks, tetapi menguasai lapisan kompetensi berikut. Jika lima lapisan ini beres, bahasa lain akan jauh lebih mudah dipelajari.

Lapisan 1 - Fondasi

- Python
- C
- SQL
- Bash

Lapisan 2 - Industri umum

- JS/Node
- TypeScript
- Java
- C#

Lapisan 3 - Sistem modern

- Go
- Rust
- C++

Lapisan 4 - Data / science

- R
- Julia

Lapisan 5 - Paradigma lanjut

- Haskell
- Clojure
- Elixir/Erlang
- OCaml/Scheme

Ringkasan paling penting:

- Python = paling cepat untuk produktivitas.
- C = fondasi mesin dan memori.
- C++ = performa besar dan fleksibel.
- Java = enterprise stabil.
- C#/.NET = produktif dan kuat.
- Go = backend sederhana dan cepat.
- Swift/Kotlin = modern native/JVM.
- Ruby/PHP/Lua/Tcl = scripting pragmatis.
- Rust = systems aman dan modern.
- Node/TS = web dan tooling modern.
- Deno/Bun = runtime JS/TS modern.
- R/Julia = sains data dan numerik.
- SQLite = inti query data dan database lokal.
- Haskell/Scala/Clojure/Elixir/Erlang/OCaml/Scheme = memperkuat logika dan paradigma.
- Nim/Crystal = native modern niche.
- NASM = level mesin.

Kesimpulannya, tidak ada satu bahasa terbaik untuk semua hal. Yang ada adalah rantai kompetensi: fondasi -> produktivitas -> industri -> performa -> data -> paradigma -> spesialisasi. Semakin kuat fondasi dan semakin jelas peta belajarmu, semakin cepat kamu bisa berpindah bahasa tanpa merasa mulai dari nol.

Dokumen selesai.