


# Basil the Basil Programming Language

What first year students should learn.

What hobbyists should learn.

What professionals should learn.

What may be the only programming language you need.

Basil  is deliberately small, friendly, and all about helping new programmers start building strong fundamentals from day 1. It combines the approachability of classic BASIC with an optional, modern brace style that looks comfortable to students headed toward C, C#, Java, JavaScript, or Go.


The goal isn't to replace those languages; it's to meet beginners where they are, teach core concepts clearly, and give them a smooth runway to the rest of the ecosystem.


---

## Why first languages matter

Your first programming language shouldn't be a puzzle box. It should:

- Lower cognitive load while you're learning core ideas like variables, expressions, control flow, and functions.
- Offer clear, immediate feedback (short edit-run cycles, gentle error messages).
- Be consistent in how it uses syntax to express ideas.
- Build habits that transfer to the broader programming world.
- Provide a practical real world experience with an immediately applicable skill set.

Basil  was designed against these criteria. It keeps the classic readability of BASIC, but adds an alternate "modern" surface syntax so that what you learn today still looks familiar later.

At the same time, Basil  is powerful enough to build real projects, with a growing standard library and a modular "mod" system that adds out-of-box functionality like AI, AWS, SQL databases, HTTP, SMTP, JSON, CSV, cryptography, audio/MIDI/DAW support, and more.

Basil  is also made for the AI age, the first programming language designed for AI from the ground up.

Two ways to say the same thing (both syntaxes valid in Basil 🌿)

Classic BASIC style:

## Sample Basil 🌿 Code Example

Below are two equivalent ways to write an infinite loop with a break condition in Basil 🌿:

```
REM Infinite loop with BREAK (will break at 3)
LET i = 0
WHILE TRUE BEGIN
    INCR i
    IF i = 3 THEN BEGIN      REM Block IF
        BREAK
    END
    PRINT i
END
```

```
// Infinite loop with BREAK (will break at 3)
let i = 0;
while true {
    i++;
    if i == 3 {              // Block IF
        break;
    }
    print i;
}
```

Both styles are valid in Basil 🌿 and help learners transition smoothly from beginner to advanced programming concepts.

You can mix and match styles in one program. Internally, both forms compile to the same structures and run the same way.

## Why Basil works as a first learning language

- Gentle, explicit control flow
    - if ... then and if ... { ... } are both accepted; else/elseif read naturally.
    - while, for, and select case are straightforward and visible.
  - Clear block boundaries
    - You can choose BEGIN ... END or { ... }. Either way, blocks are explicit and obvious.
  - Low ceremony, fast feedback
    - Small surface area, immediate execution, simple I/O (print, println).
  - Case-insensitive keywords; readable by design
    - Beginners don't lose momentum over capitalization or minor formatting.
  - A bridge to mainstream languages
    - The brace form prepares students to read/write C-family languages without abandoning BASIC's clarity.
- 

## But what about Python?

Python's popularity in classrooms isn't an accident. Educators pick it because:

- It has a gentle reputation and "readable" look.
- It ships with many batteries included.
- There's a large community and abundant learning material.

Those are real advantages. Still, for many first-time programmers, Python can introduce avoidable friction:

- Significant whitespace as structure
  - Block structure is invisible punctuation. One stray space or a tabs-vs-spaces mismatch breaks the program in ways that are hard to see.
- Implicit behaviors and late surprises
  - Dynamic dispatch and truthiness rules show up early but take time to internalize; many mistakes surface only at runtime.
- Mixed metaphors in the beginner toolkit
  - Both len(x) and x.method(...) styles appear side by side; slicing, comprehensions, and decorators are powerful but conceptually dense for week one.

- Environment and packaging overhead
  - Virtual environments, package managers, and interpreter versions can crowd out the actual learning during the critical first weeks.

None of these are fatal, and Python remains a great second (or third) language. Basil 🌱’s design simply tries to remove these tripwires from the first-language experience.

---

## How Basil 🌿 addresses first-year pain points

- Visible structure
    - Choose braces or BEGIN/END. Students can literally “see the block.”
  - Predictable, explicit control flow
    - if/elseif/else, while, for/next, and select case have minimal hidden rules.
  - One concept at a time
    - You can start with the classic style and later migrate to braces without relearning the language.
  - Transferable skills
    - The modern style maps cleanly to C, C#, Java, JavaScript, and Go idioms.
  - Friendly diagnostics
    - Errors mention both classic and modern forms (e.g., “Expected THEN or ‘{’ after IF condition.”), guiding students instead of stopping them.
- 

## A suggested path for an intro course (e.g., COP-1000)

1. 🌱 Week 1–2: Variables, arithmetic, print/println, simple if/then.
2. 🌱 Week 3: Loops (while, for/next), break and continue.
3. 🌱 Week 4: Functions (func, return), parameters, local scope.
4. 🌱 Week 5: Decisions at scale: select case; string operations.
5. 🌱 Week 6: Modernization—introduce the brace style in parallel; show side-by-side translations.
6. 🌱 Week 7+: Objects and modules as applicable; project work.

Students leave with working mental models and syntax that looks familiar across the industry.

## Quick syntax map: classic to modern

- IF
  - Classic: IF cond THEN BEGIN ... END
  - Modern: if cond { ... }
- ELSE / ELSEIF
  - Classic: ELSE BEGIN ... END or single statement
  - Modern: } else if cond { ... } else { ... }
- WHILE
  - Classic: WHILE cond BEGIN ... END
  - Modern: while cond { ... }
- FOR / NEXT
  - Classic: FOR i = 1 TO 10 ... NEXT i
  - Modern: same control header; body can use { ... }
- SELECT CASE
  - Classic: SELECT CASE x ... END [SELECT]
  - Modern: select case x { ... }

Both forms are always valid; pick one or mix as you learn.

---

## Basil 🌿 is a cross-platform Interpreter and Native-Code Compiler

- Basil 🌿 is a cross-platform interpreter and native-code compiler. It runs on Windows, Linux, and macOS.
- Basil 🌿 can run source files, or compile to a single portable executable that runs without installing a runtime or dependencies.

## A Large Standard Extended Library with "Mods"

Basil 🌿 contains a robust standard library and pre-built "Basil 🌿 Feature Objects" or "Mods" that provide real-world functionality out of the box. These include:

- 🌿 File I/O
- 🌿 String manipulation
- 🌿 Date and time functions

- 🌱 Math functions
- 🌱 Artificial Intelligence (AI) integration
- 🌱 Networking (HTTP, SMTP, CURL)
- 🌱 Database access (SQLite, SQL, ORM Wrappers)
- 🌱 JSON and CSV handling
- 🌱 Web Development (Templating HTML with embedded like Php)
- 🌱 AWS integration
- 🌱 Advance Screen UI (CrossTerm)
- 🌱 AI/ML interfaces
- 🌱 Audio/MIDI/DAW support
- 🌱 Cryptography (Base64, PGP, Zip)
- 🌱 WebAssembly support
- 🌱 Example starter Mods
- 🌱 Tons of examples and documentation
- 🌱 AI Onboarding for Rust Developers
- 🌱 Community support
- 🌱 and more!

## Looking ahead

Basil 🌱 keeps backward compatibility while also adding new features. Recently added features include:

- 🌱 List and dictionary literals.
- 🌱 User-defined types (TYPE ... END TYPE).
- 🌱 Fixed-length strings where appropriate.
- 🌱 Game-capable graphics
- 🌱 Asterisk Integration (VoIP)
- 🌱 WebAssembly (WASM) support
- 🌱 Distributed processing (DPROC)
- 🌱 Interop with Rust, Go, and C#

At the time of this writing we are also working on:

- 🌱 An web-based IDE (Integrated Development Environment)
- 🌱 JetBrains integration
- 🌱 VS Code integration

... and we are open to suggestions!

These features will slot into the existing language without disrupting the core learning experience.

## Education and Community

Basil 🌱 is an open source project and is actively developed by a community of volunteers, built with education and community in mind.

We have built Basil 🌱 to be a great learning tool for beginners, while remaining robust and powerful for real-world use. We are committed to making it easy for you to learn the Basil 🌱 language and to contribute to the project.

---

## Summary

Basil 🌱 restores the simplicity many of us loved in our first encounters with BASIC, while offering a modern, brace-style path that aligns with today's mainstream languages. It's small enough to learn quickly, expressive enough to build real projects, and friendly enough to keep students in the game—so more learners finish the course confident, not frustrated.

## Resources

Github Repository: <https://github.com/blackrushllc/basil>

Complete Online Reference: <https://yobasic.com/basil/reference.html>

Basil is an Open Source BASIC compiler under the MIT License. See the LICENSE file for full details

Copyright (c) 2025 Blackrush LLC

Email: BlackrushDrive@Gmail.com

X/IG: @BlackrushWorld

<https://www.facebook.com/BlackrushBASIC>