Raku++

An interpreter, a compiler and a toolchain for Raku

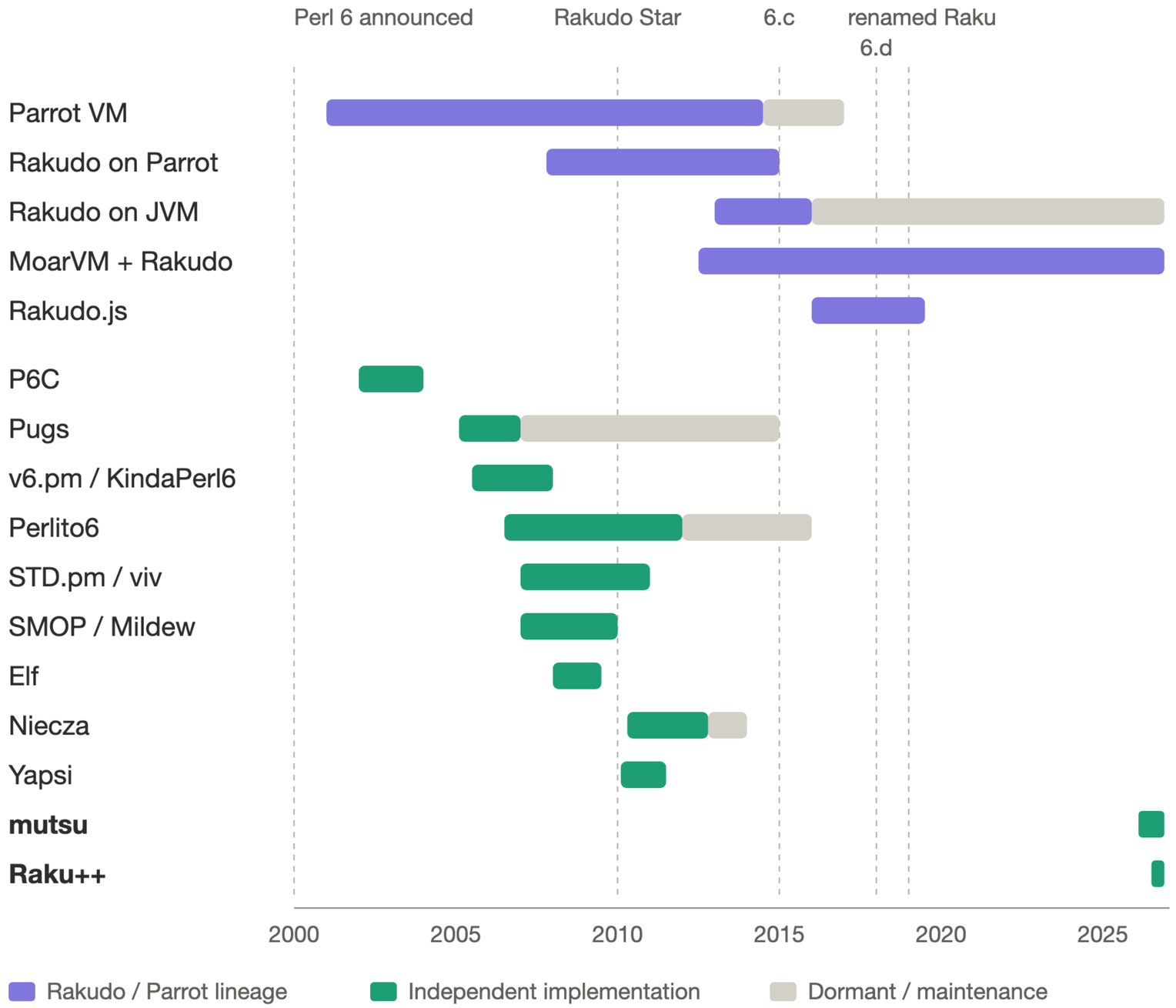
Andrew Shitov — London Perl Workshop — 21.11.2026

github.com/ash/rakupp

What it is

Raku++

A from-scratch implementation in C++


Perl 6 announced
Rakudo Star
6.c
renamed Raku
6.d
Parrot VM
Rakudo on Parrot
Rakudo on JVM
MoarVM + Rakudo
Rakudo.js
P6C
Pugs
v6.pm / KindaPerl6
Perlito6
STD.pm / viv
SMOP / Mildew
Elf
Niecza
Yapsi
mutsu
Raku++
2000
2005
2010
2015
2020
2025
Rakudo / Parrot lineage
Independent implementation
Dormant / maintenance



Vorwort

Building a compiler? But that can only be done by big companies like Microsoft, Google or Oracle.

Compiler bauen? Das machen doch nur große Firmen wie Microsoft, Google oder Oracle. Das stimmt, aber fast alle Informatikerinnen und Informatiker scheinen irgendwann einmal den Wunsch zu verspüren, eine eigene Programmiersprache zu entwerfen, und sei es nur eine domänenspezifische Sprache oder eine Kommandosprache für spezifische Zwecke. Natürlich möchte man dann auch einen Compiler dafür schreiben. Dieser Wunsch scheitert oft daran, dass das nötige Compilerbau-Wissen fehlt oder die in einschlägigen Büchern beschriebenen Techniken zu kompliziert sind und sich vor allem mit fortgeschrittenen Themen wie Optimierung, Registerallokation oder den Details der Codeerzeugung beschäftigen.

Dabei sind die Grundlagen des Compilerbaus einfach – jeder kann sie erlernen und zu seinem Methodenrepertoire hinzufügen. Während Compiler für Programmiersprachen wie Java oder C/C++ tatsächlich nur von großen Firmen entwickelt

Raku++

A *hand-written* lexer and parser

Raku++

A tree-walking evaluation over AST

Raku++

A full Raku compiler with grammars, Unicode, etc.

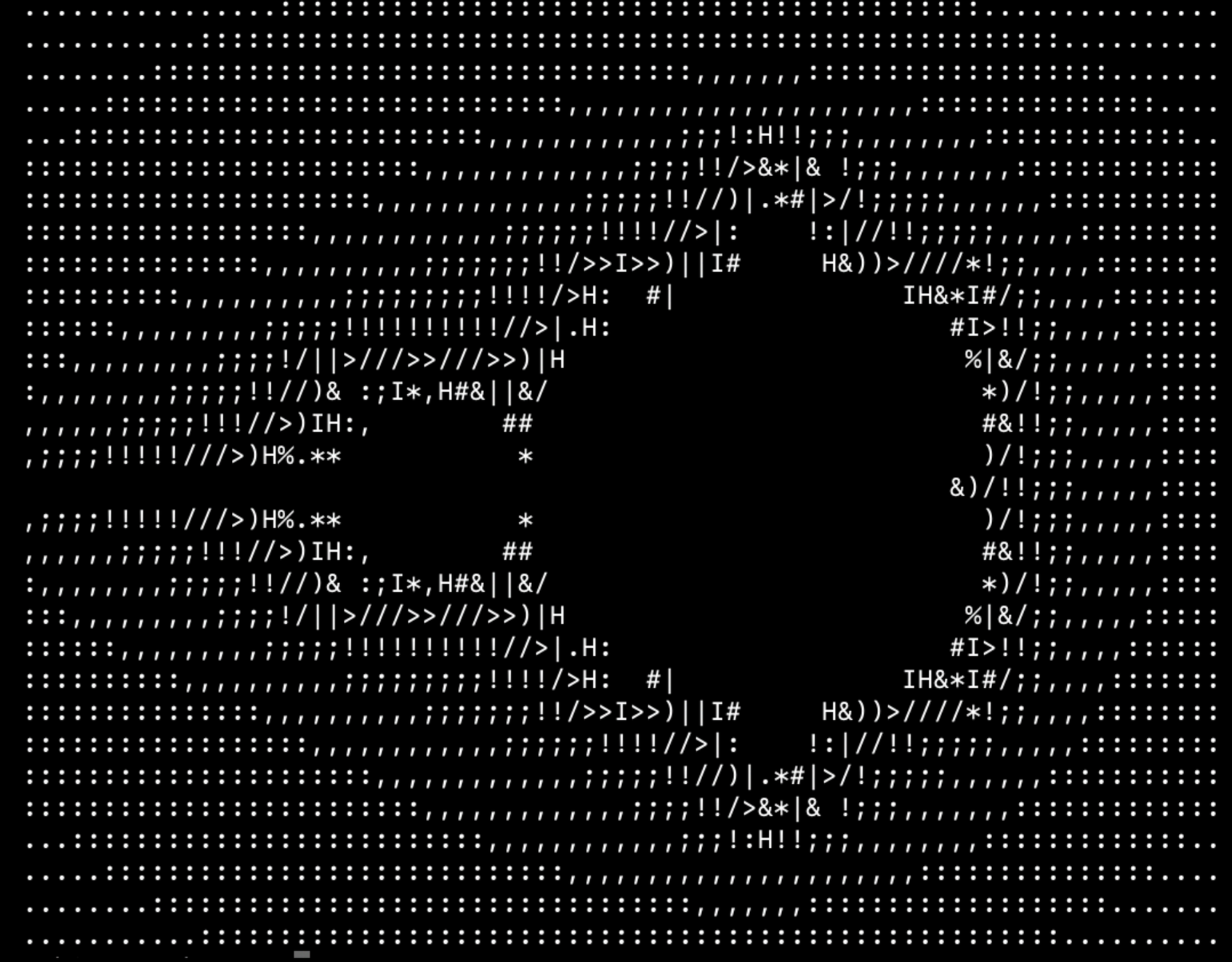
Raku++

An interpreter and
a compiler, and . . .

Raku++

… a toolchain, including a module manager

```
[ash@mac rakupp % raku++ examples/mandel.raku
```

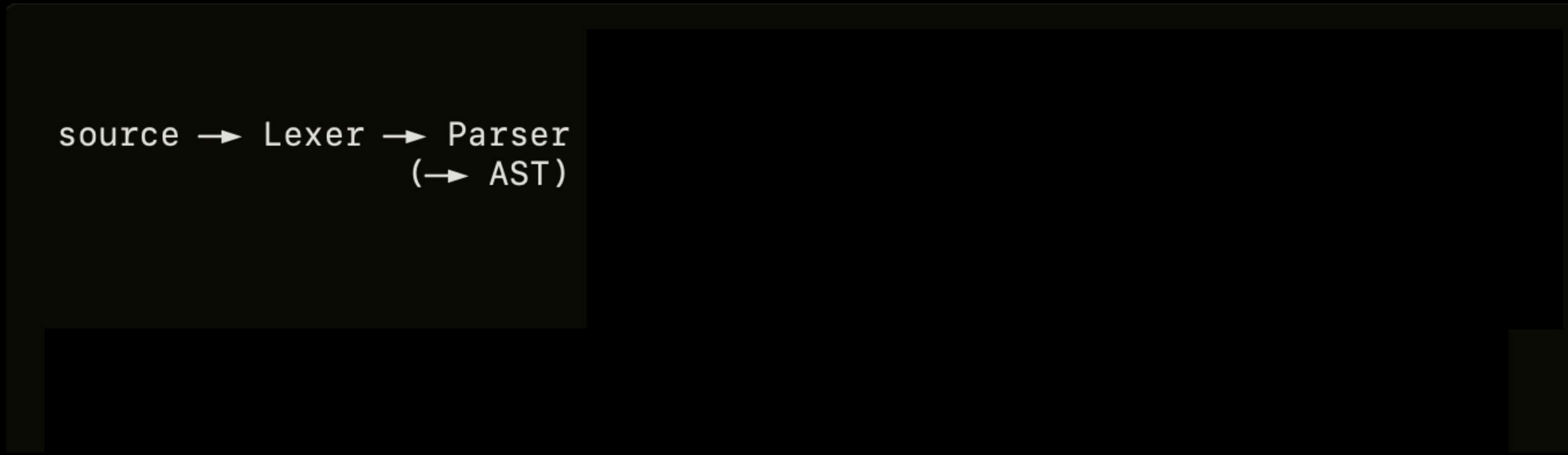


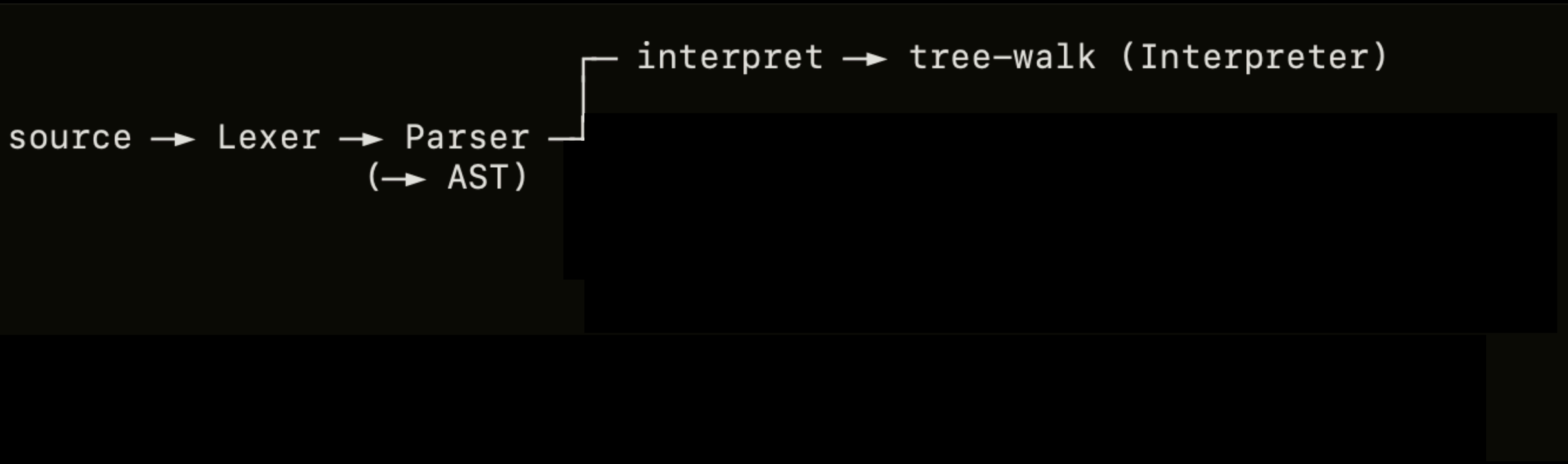
Architecture

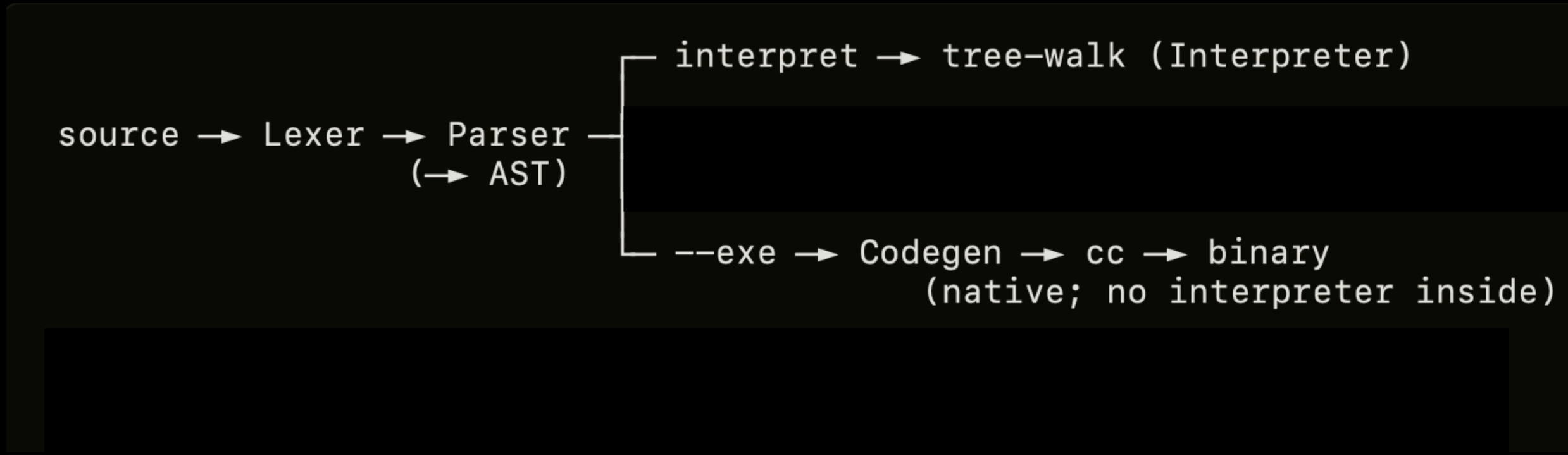
Goals

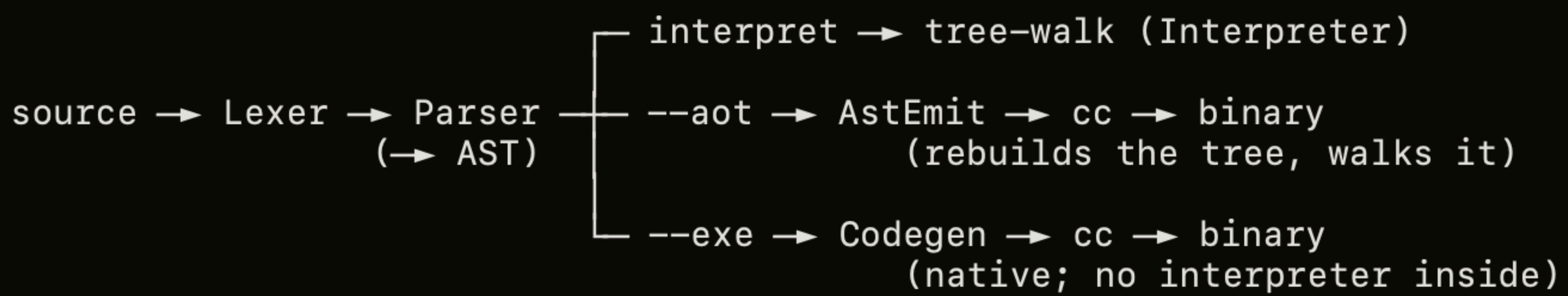
1. Speed
2. Compatibility

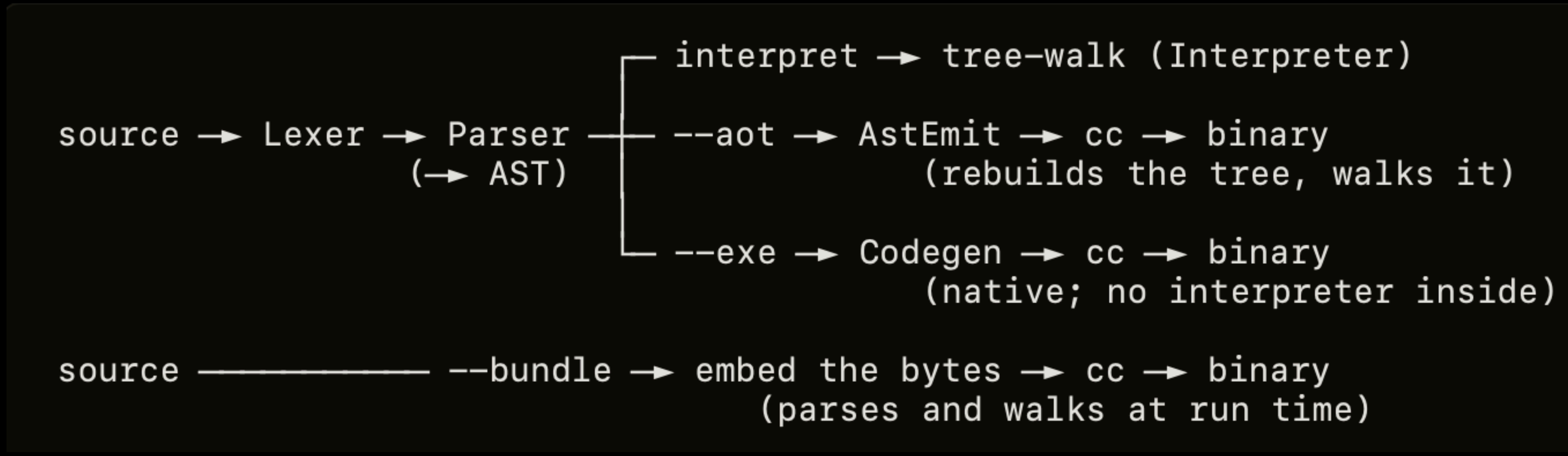
	Raku++	mutsu	Rakudo
Written in	C++17, no dependencies	Rust (~25 crates)	Raku + NQP
Execution	Walks the syntax tree directly	Bytecode VM + Cranelift JIT	MoarVM bytecode, with spesh and a JIT
Memory	Reference counting + cycle collector	Reference counting + cycle collector	Generational, moving GC
Startup (say)	4.4 ms (3.1 ms as a native binary)	4.2 ms	62.5 ms
Memory for say 1	1.2 MB	4.3 MB	92 MB
Standalone native binary	Yes (--exe)	No	No
Runs in a browser	Raku.js (WebAssembly)	WebAssembly build	—
Ecosystem	1,182 / 2,584 dists pass their tests	Bundles zef as mzef	1,791 / 2,529 dists pass their tests


source → Lexer → Parser
(→ AST)

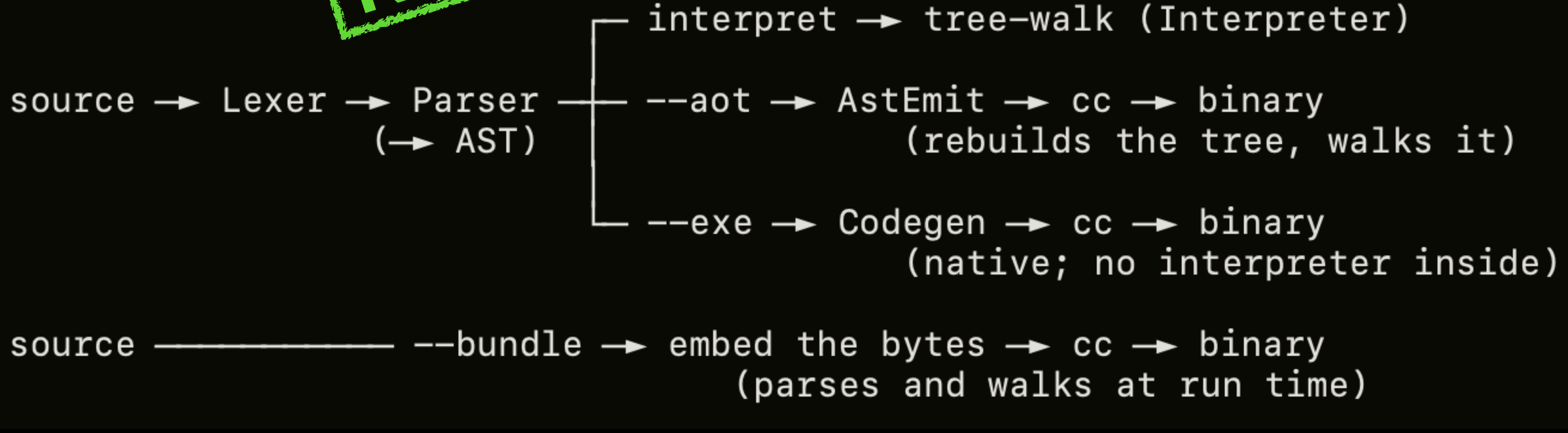

interpret → tree-walk (Interpreter)
source → Lexer → Parser
(→ AST)


source → Lexer → Parser
(→ AST)
interpret → tree-walk (Interpreter)
--exe → Codegen → cc → binary
(native; no interpreter inside)

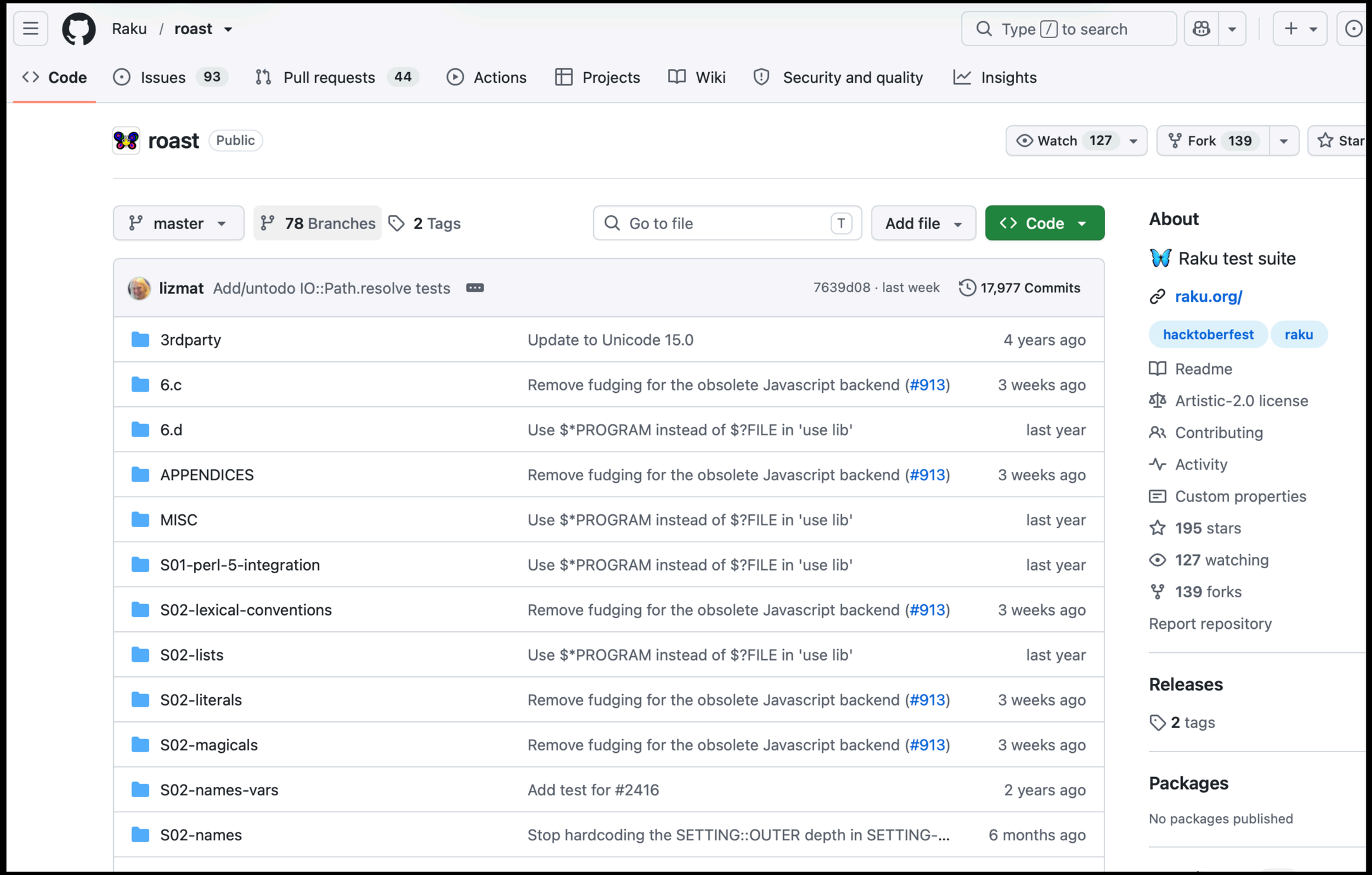

interpret → tree-walk (Interpreter)
source → Lexer → Parser
(→ AST)
--aot → AstEmit → cc → binary
(rebuilds the tree, walks it)
--exe → Codegen → cc → binary
(native; no interpreter inside)

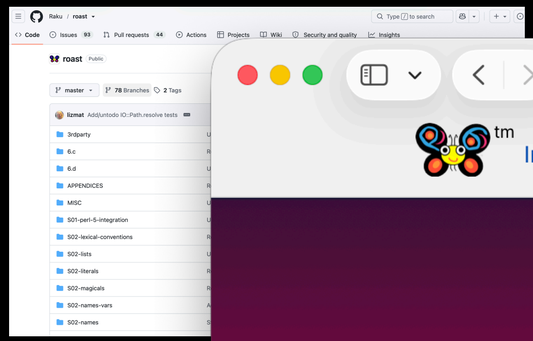

interpret → tree-walk (Interpreter)
source → Lexer → Parser
(→ AST)
--aot → AstEmit → cc → binary
(rebuilds the tree, walks it)
--exe → Codegen → cc → binary
(native; no interpreter inside)
source
--bundle → embed the bytes → cc → binary
(parses and walks at run time)

No VM


source → Lexer → Parser
(→ AST)
interpret → tree-walk (Interpreter)
--aot → AstEmit → cc → binary
(rebuilds the tree, walks it)
--exe → Codegen → cc → binary
(native; no interpreter inside)
source
--bundle → embed the bytes → cc → binary
(parses and walks at run time)

Test suite


Raku / roast
Type / to search
Code
Issues 93
Pull requests 44
Actions
Projects
Wiki
Security and quality
Insights
roast
Public
Watch 127
Fork 139
Star
master
78 Branches
2 Tags
Go to file
T
Add file
Code
lizmat Add/untodo IO::Path.resolve tests
7639d08 · last week
17,977 Commits
3rdparty Update to Unicode 15.0 4 years ago
6.c Remove fudging for the obsolete Javascript backend (#913) 3 weeks ago
6.d Use $*PROGRAM instead of $?FILE in 'use lib' last year
APPENDICES Remove fudging for the obsolete Javascript backend (#913) 3 weeks ago
MISC Use $*PROGRAM instead of $?FILE in 'use lib' last year
S01-perl-5-integration Use $*PROGRAM instead of $?FILE in 'use lib' last year
S02-lexical-conventions Remove fudging for the obsolete Javascript backend (#913) 3 weeks ago
S02-lists Use $*PROGRAM instead of $?FILE in 'use lib' last year
S02-literals Remove fudging for the obsolete Javascript backend (#913) 3 weeks ago
S02-magicals Remove fudging for the obsolete Javascript backend (#913) 3 weeks ago
S02-names-vars Add test for #2416 2 years ago
S02-names Stop hardcoding the SETTING::OUTER depth in SETTING-... 6 months ago
About
Raku test suite
raku.org/
hacktoberfest
raku
Readme
Artistic-2.0 license
Contributing
Activity
Custom properties
195 stars
127 watching
139 forks
Report repository
Releases
2 tags
Packages
No packages published



docs.raku.org

tm

Introduction Reference Miscellaneous Types Routines Raku® Chat More

Type f to search for

Raku documentation

Welcome to the official documentation of the Raku® programming language!

Getting started, Migration guides from other languages, & Tutorials

Documents introducing the language for various audiences.

Learn more

Language References

Documents explaining the various conceptual parts of the language.

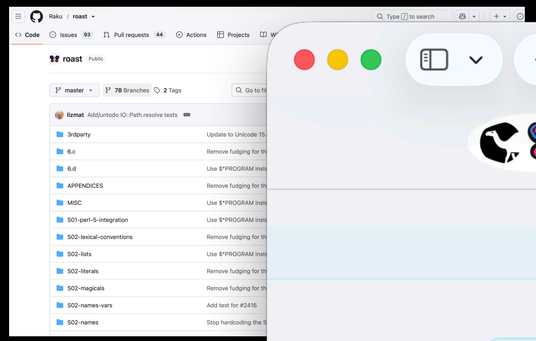
Learn more

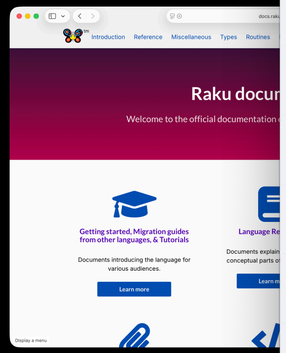
Type Reference

Index of built-in classes, roles and enums.

Learn more

Display a menu





theweeklychallenge.org

HOME ABOUT COMMUNITY ▾ THIS WEEK ▾ REVIEWS ▾ BLOGS SEARCH / INFO ▾

The London Perl & Raku Workshop: Registration open! Learn more

CHALLENGE ACTIVE

The Weekly Challenge

"You don't have to be *an expert*"



TASK #1

Largest Interval

You are given a positive integer. Write a script that generates a sequence of integers from 1 to given positive integer, filtering out any numbers that contain repeating digits (e.g., 11, 99, 100, and 101 are excluded) and return the biggest jump.

TASK #2

Minimum Genetic Mutation

You are given start and end gene string along with gene bank. Write a script to return the minimum number of mutations required to mutate from start to end gene. Return -1 if there is no such mutation. A gene must be in a bank to make it a valid gene string.

TAKE THE CHALLENGE | HOW TO CONTRIBUTE | SPONSOR US

GUEST CONTRIBUTIONS | LATEST CHAMPION | PERL REVIEW | RAKU REVIEW | YOUTUBE

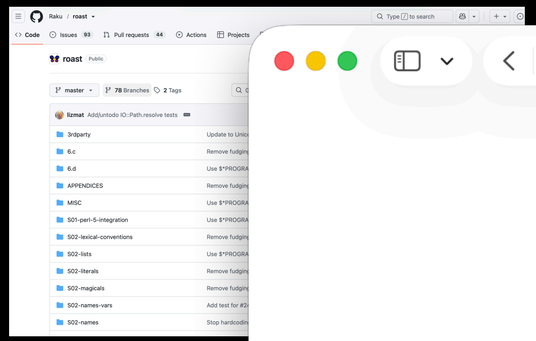


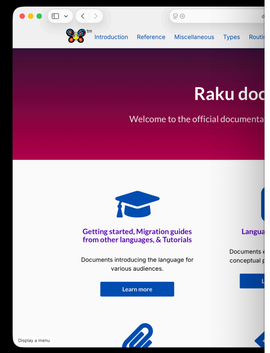
NEW BOOK BY MOHAMMAD SAJID ANWAR

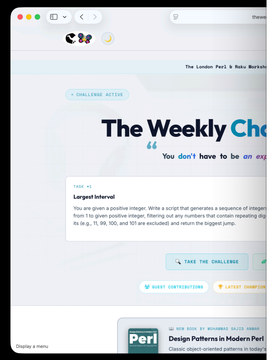
Design Patterns in Modern Perl

Classic object-oriented patterns in today's Perl - using core Perl, Moo and Object::Pad. Cleaner, more

Display a menu







raku.land

Rand | Stats | Search distributions

Raku Land is a directory of Raku distributions.

Written itself in Raku, licensed under Artistic 2.0, and available on GitLab; patches welcome!

Distributions

ZEF **_** — 26 Sep 2023 — ★ 21

A meta package for zero-dependency micro packages of under 70 lines of code

BUNDLE | LOWBAR | METAPACKAGE | MICRO

ZEF **Abbreviations** — 8 Jul 2026 — ★ 2

Provides abbreviations for an input set of one or more words

ABBREVIATIONS | OPTIONS | TEXT

ZEF **ABC** — 20 Mar 2021 — ★ 7

Toolkit for dealing with ABC music notation

ZEF **AccessorFacade** — 2 Jul 2023 — ★ 3

Provides a method trait to turn indivdual get/set subroutines into a single attribute

DEVEL | FACADE | UTIL

ZEF **AccountableBagHash** — 15 Sep 2026 — ★ 0

be an accountable BagHash / MixHash

ACCOUNTABLE | UTIL

ZEF **Acme::Anguish** — 29 Dec 2024 — ★ 17

Use Anguish programming language in your Raku programs

Recent Dists

Distribution	
Rakuest	0.8.2
Data::Reshapers	0.4.0
Duckula	0.18.0
Red	0.2.15
Red	0.2.14
Rakuish	0.106.2
Duckie	0.0.16
LLM::Functions	0.6.0
Elucid8::Build	0.13.5
Rakuish	0.106.1

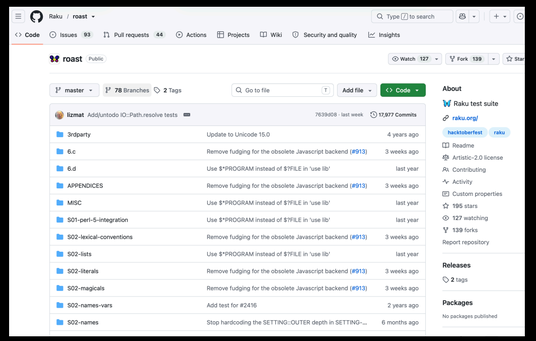
Popular Dists

Distribution	★
zef	225
zef	225
zef	225
Bailador	177
App::Rak	161
Inline::Perl5	97
cro	94
cro	94
DBlish	93
Sparrow6	87

Prolific Authors

Author	#

Display a menu



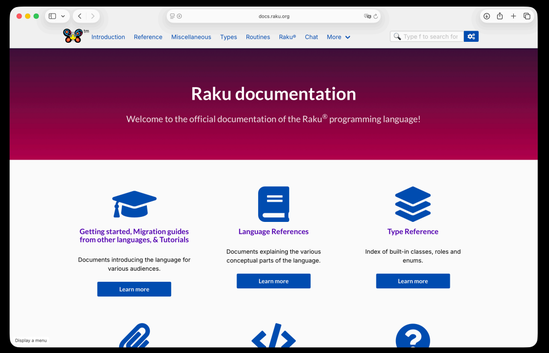
Roast

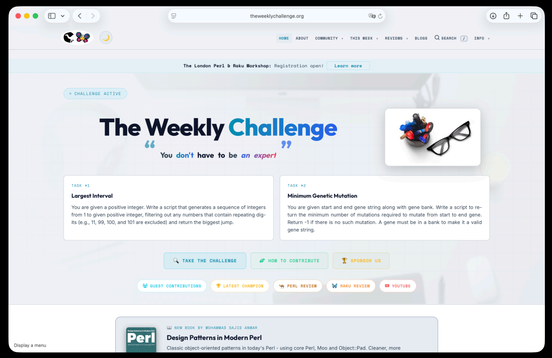
Koans

Docs

The Weekly Callenge

Raku Ecosystem





RakuGrid

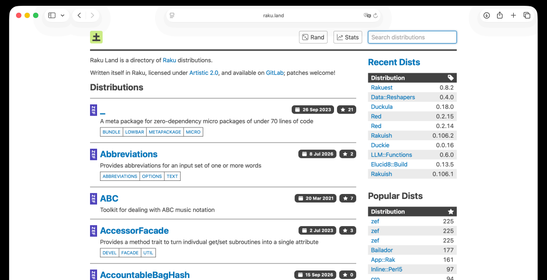
RakuGlaze

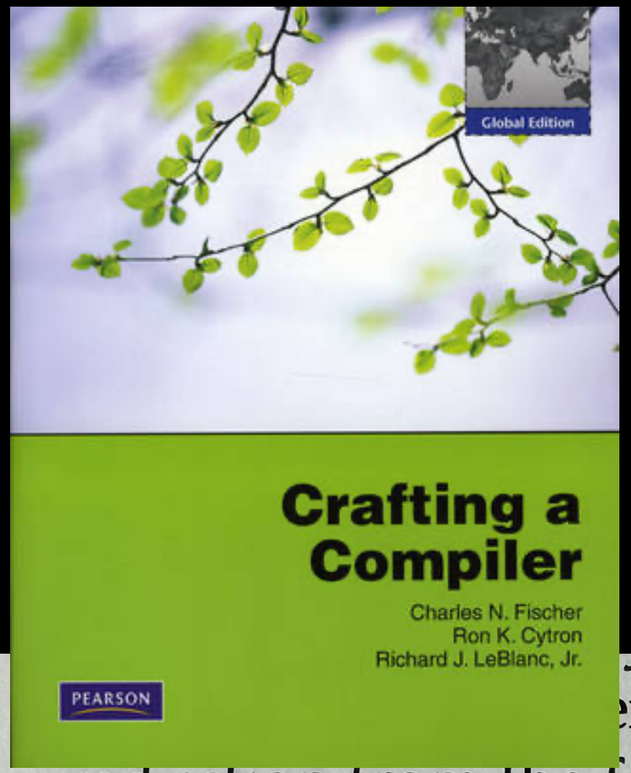
RakuMap

FatRoast

Corpus

Specs





ent in a program is reachable is **undecidable**, proved by reduction from the famous **halting problem** [HU79]. We certainly cannot ask our Java compiler literally to do the impossible!

In practice, a trusted **reference compiler** can serve as a de facto language definition. That is, a programming language is, in effect, defined by what a compiler chooses to accept and how it chooses to translate language constructs. In fact, the operational and natural semantic approaches introduced previously take this view. A standard interpreter is defined for a language, and the meaning of a program is precisely whatever the interpreter says. An early

Rakudo
is the reference compiler

Raku Behind the Docs

What the language actually does, one checked example at a time

THE RAKU++ PROJECT

Edited by Andrew Shitov

1209

corners

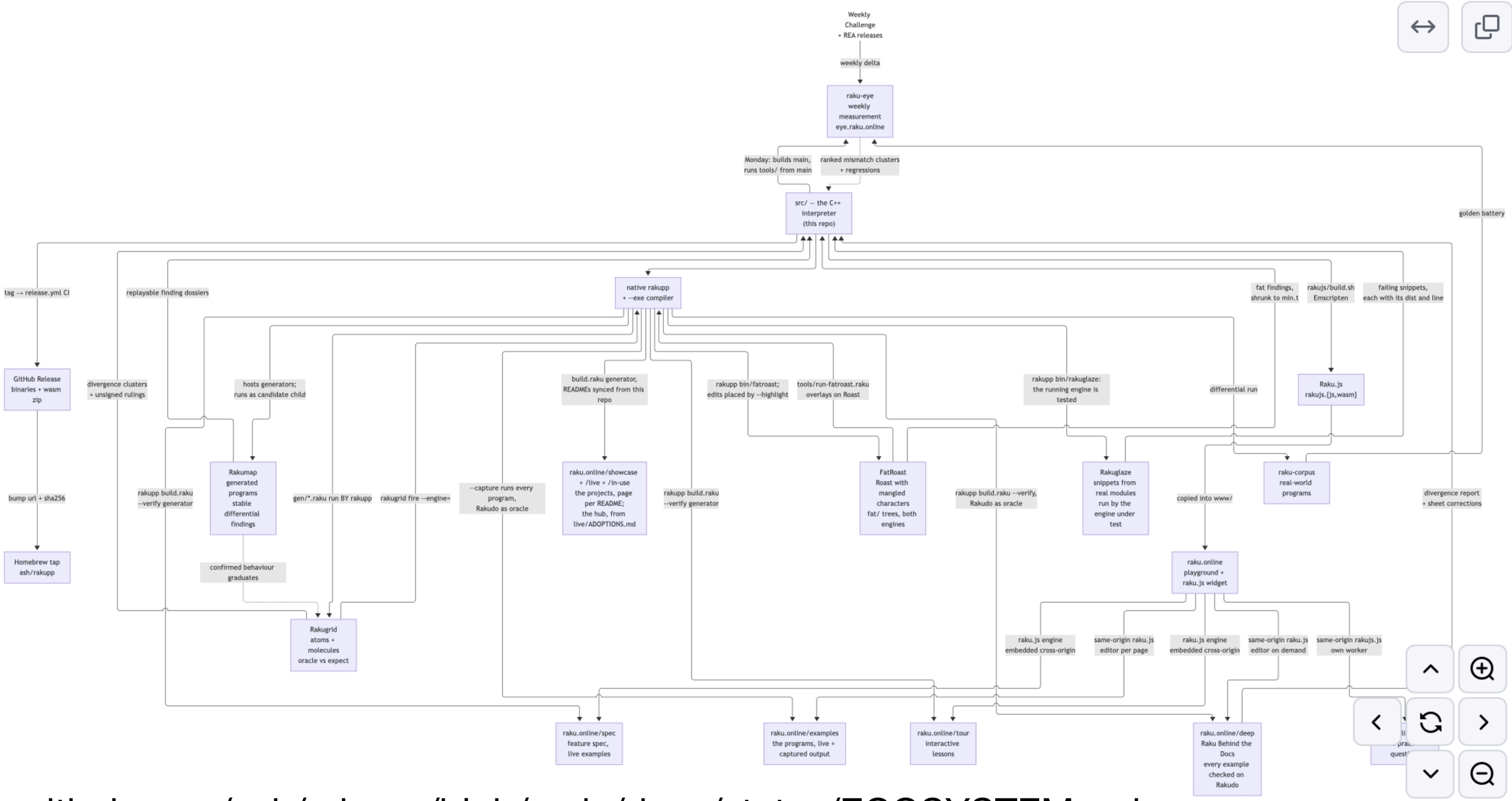
1631

examples, each checked against the reference compiler, Rakudo v2026.08

98

behaviours not in the official docs

learn.raku.online/deep



github.com/ash/rakupp/blob/main/docs/status/ECOSYSTEM.md

Roast

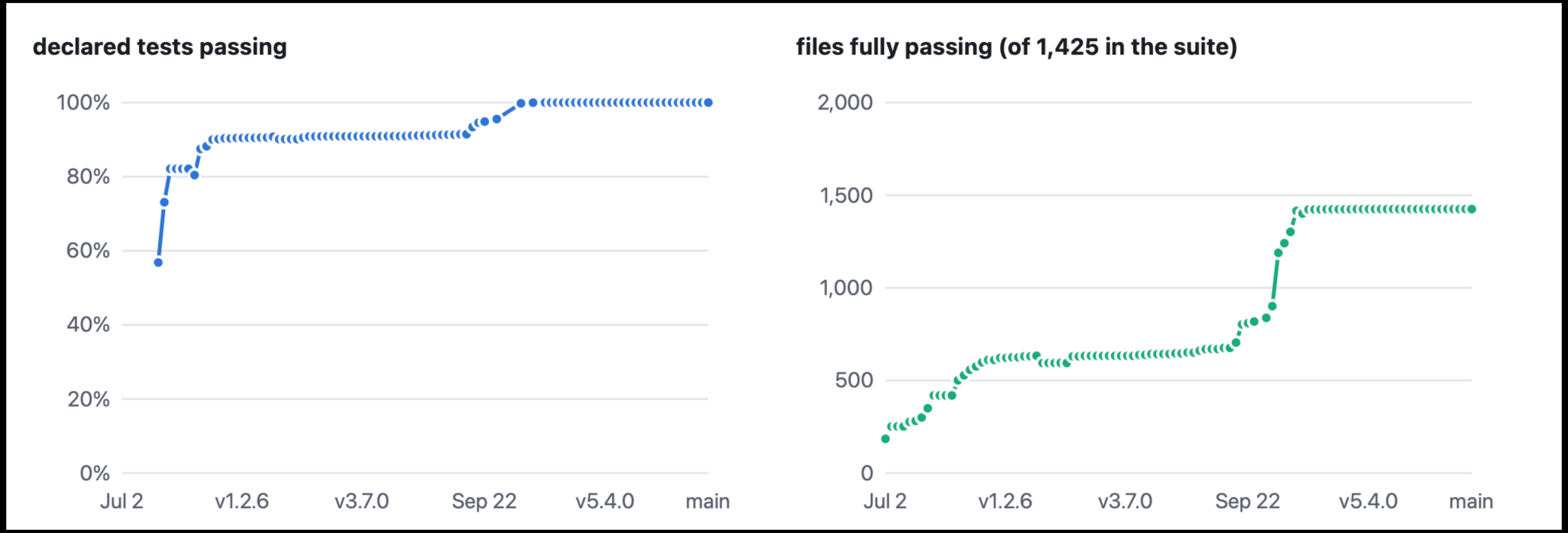
100.00%

Unicode 18.0

2-4 ms

startup time

100.00 % Roast passing



VERSION 5.10.0 · AN INDEPENDENT RAKU IMPLEMENTATION IN C++17

Raku++

An interpreter, a compiler and a toolchain for Raku

~3 ms
cold start of the native binary

100.00%
of the Roast tests an implementation is expected to pass

1,182 / 2,584
Raku ecosystem distributions passing their own test suites

0
servers, accounts or runtime dependencies

How to run

6 ways to run Raku

6 ways to run Raku

1. Interpreter

```
$ raku++ file.raku
```

6 ways to run Raku

2. Compiler

```
$ raku++ --exe -o file.bin file.raku
$ ./file.bin
```

6 ways to run Raku

3. Bundle

```
$ raku++ --bundle -o file.bin file.raku
$ ./file.bin
```

6 ways to run Raku

4. AOT

```
$ raku++ --aot -o file.bin file.raku
$ ./file.bin
```

6 ways to run Raku

5. via JavaScript (Bun)

bun.com

```
$ raku++ --target=js -o file.js file.raku
$ bun file.js
```

6 ways to run Raku

6. WASM online

raku.online/play

More ways to run Raku

7. C++

```
$ raku++ --cpp -o file.cpp file.raku
```

More ways to run Raku

8. Arduino

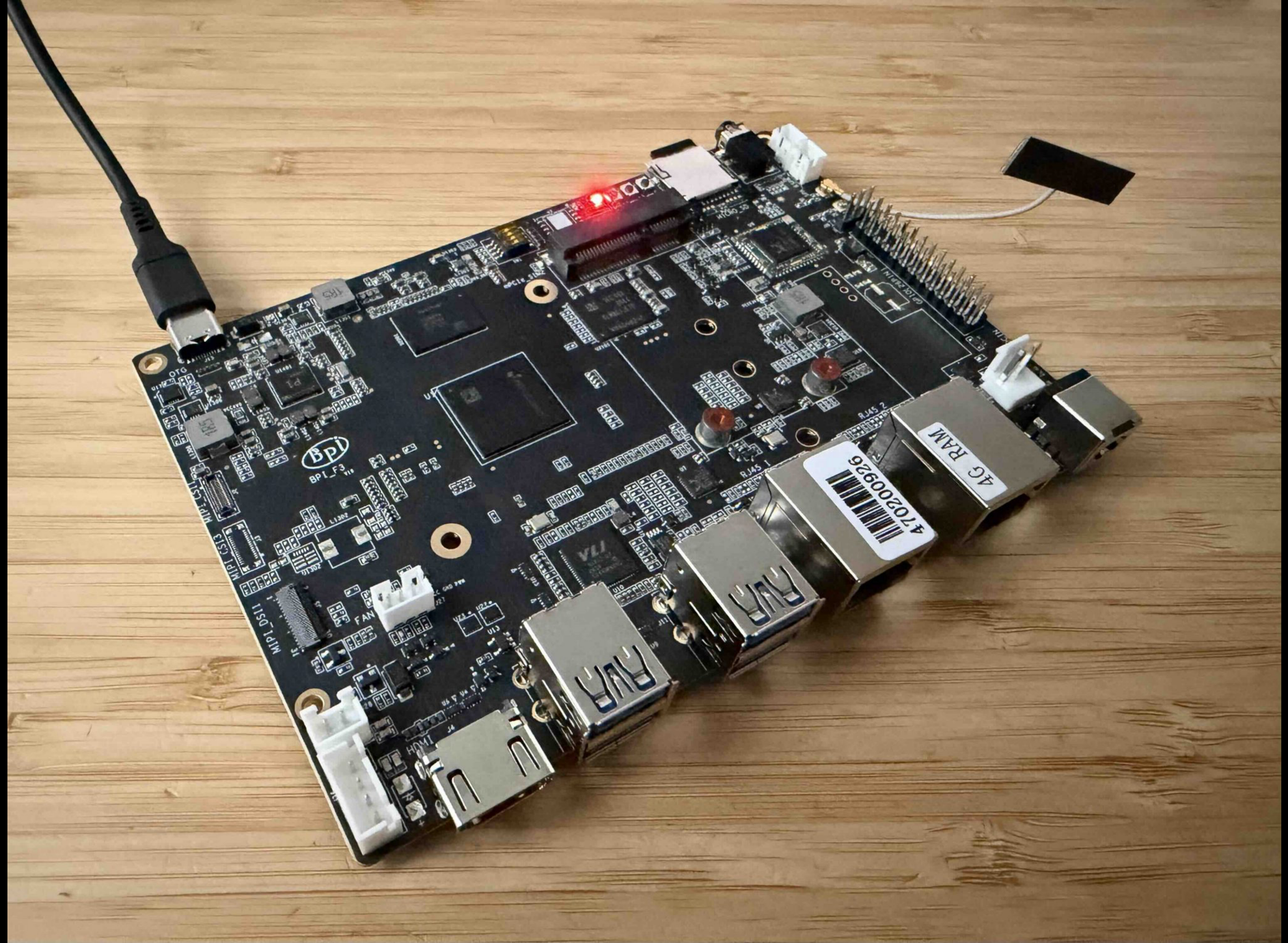
```
$ raku++ --target=arduino -o file.ino file.raku
```

More ways to run Raku

9. BBC micro:bit

```
$ raku++ --target=microbit -o file.cpp file.raku
```

PLATFORM	ARCHIVE	NOTES
macOS 11+, Apple Silicon and Intel	`rakupp-macos-universal.tar.gz`	one universal binary for both architectures — or Homebrew
Linux, x86-64	`rakupp-linux-x86_64.tar.gz`	static libstdc++ — no dependencies to satisfy
Linux, ARM64	`rakupp-linux-aarch64.tar.gz`	Raspberry Pi, Graviton, Ampere, Docker on an Apple Silicon Mac
Windows, x64	`rakupp-windows-x64.zip`	built with MSVC, static CRT — no redistributable
Windows, x64 (MinGW)	`rakupp-windows-x64-mingw.zip`	for a MinGW-w64 / MSYS2 toolchain
OpenBSD, x86-64	`rakupp-openbsd-x86_64.tar.gz`	base clang; built in a CI virtual machine each release
Linux, RISC-V (riscv64)	`rakupp-linux-riscv64.tar.gz`	static libstdc++, glibc 2.35+ (Ubuntu 22.04, Debian 13); cross-compiled in CI and tested under emulation. The first one ships with 5.3.0; from source it builds unchanged on the board too (a Milk-V Pioneer, for one)
The browser	`rakujs-*.zip`	the WebAssembly build — or just use the playground

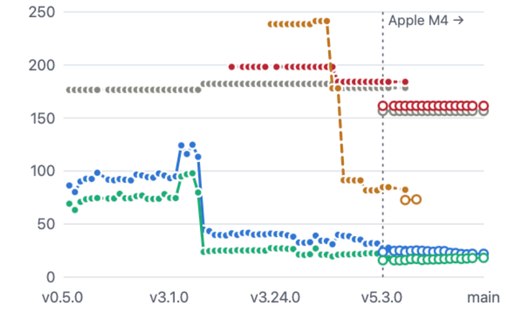

BPI_F3
4G RAM
4702009926
FAN
HDMI

Speed

regex

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu

hollow: Apple M4, from 2026-10-21

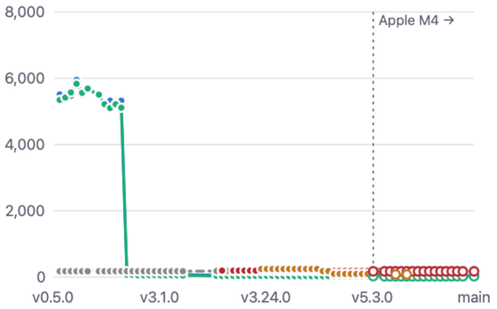


▶ Data table

textsplit

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu

hollow: Apple M4, from 2026-10-21

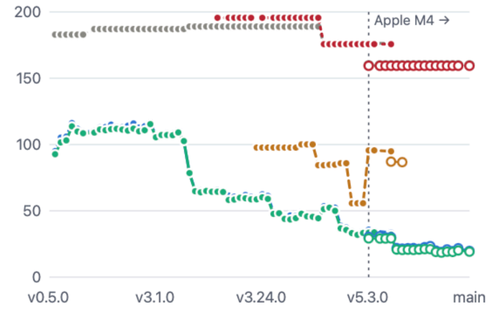


▶ Data table

arrayops

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu

hollow: Apple M4, from 2026-10-21

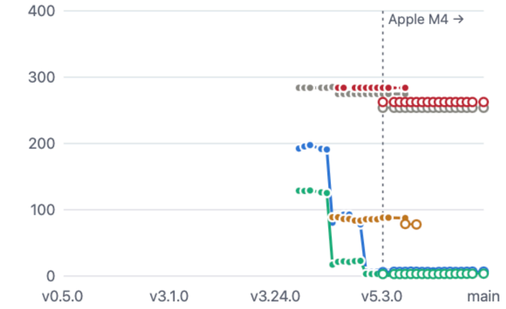


▶ Data table

mainwhen

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu

hollow: Apple M4, from 2026-10-21

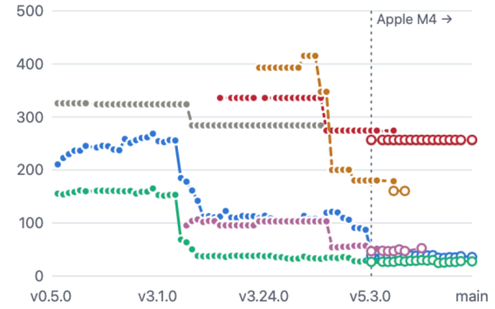


▶ Data table

hashfill · Perl twin

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu · perl

hollow: Apple M4, from 2026-10-21

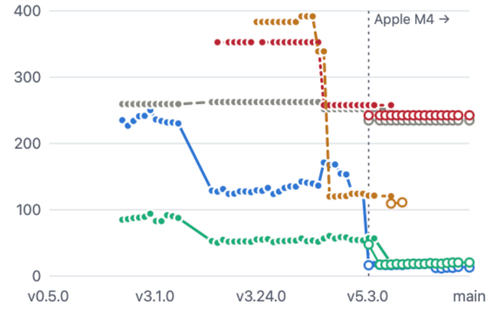


▶ Data table

arraypush

interpreter · native --exe · Rakudo · Rakudo (RakuAST) · mutsu

hollow: Apple M4, from 2026-10-21

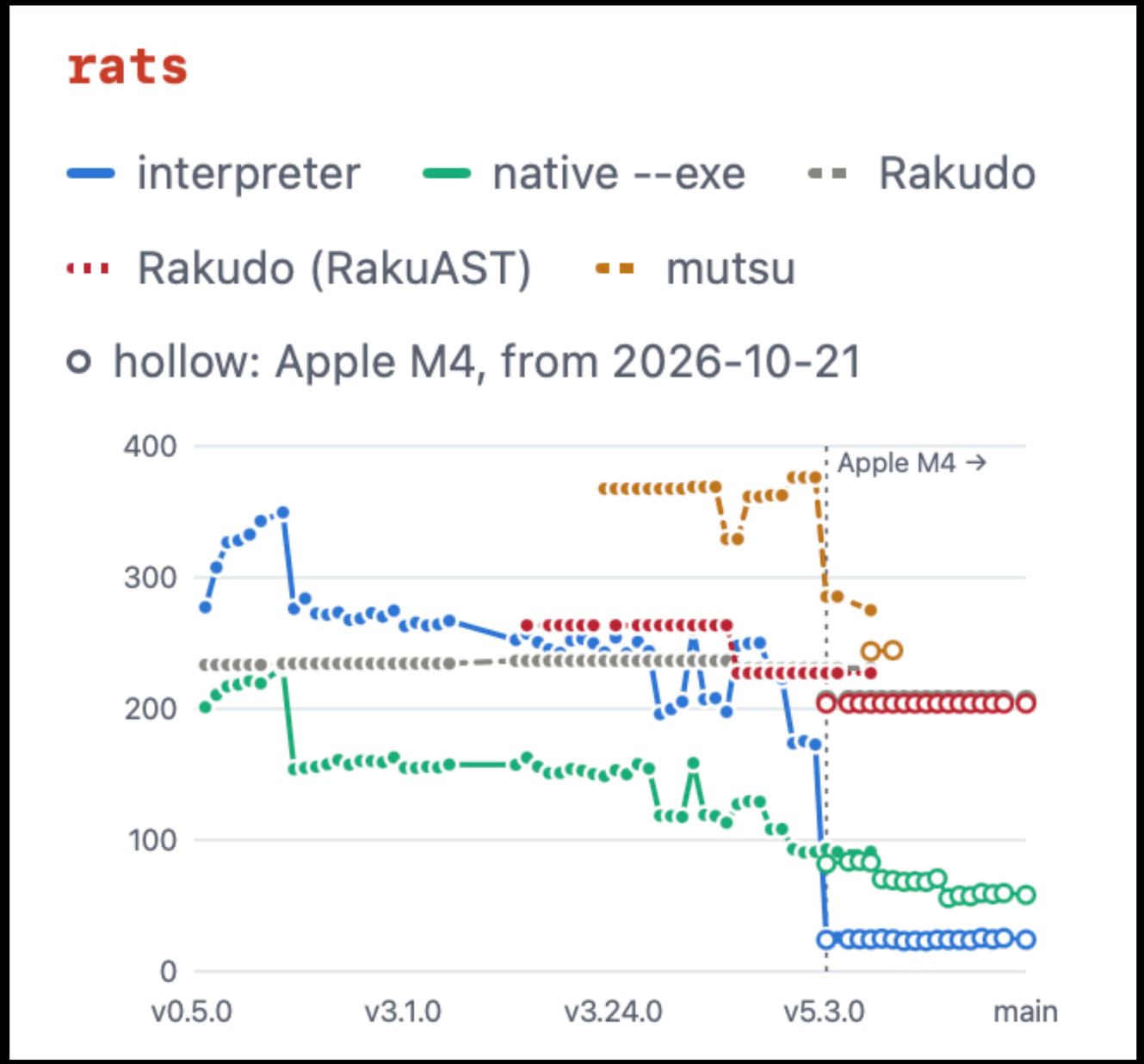


▶ Data table

raku.online/spec/dashboard

Speed

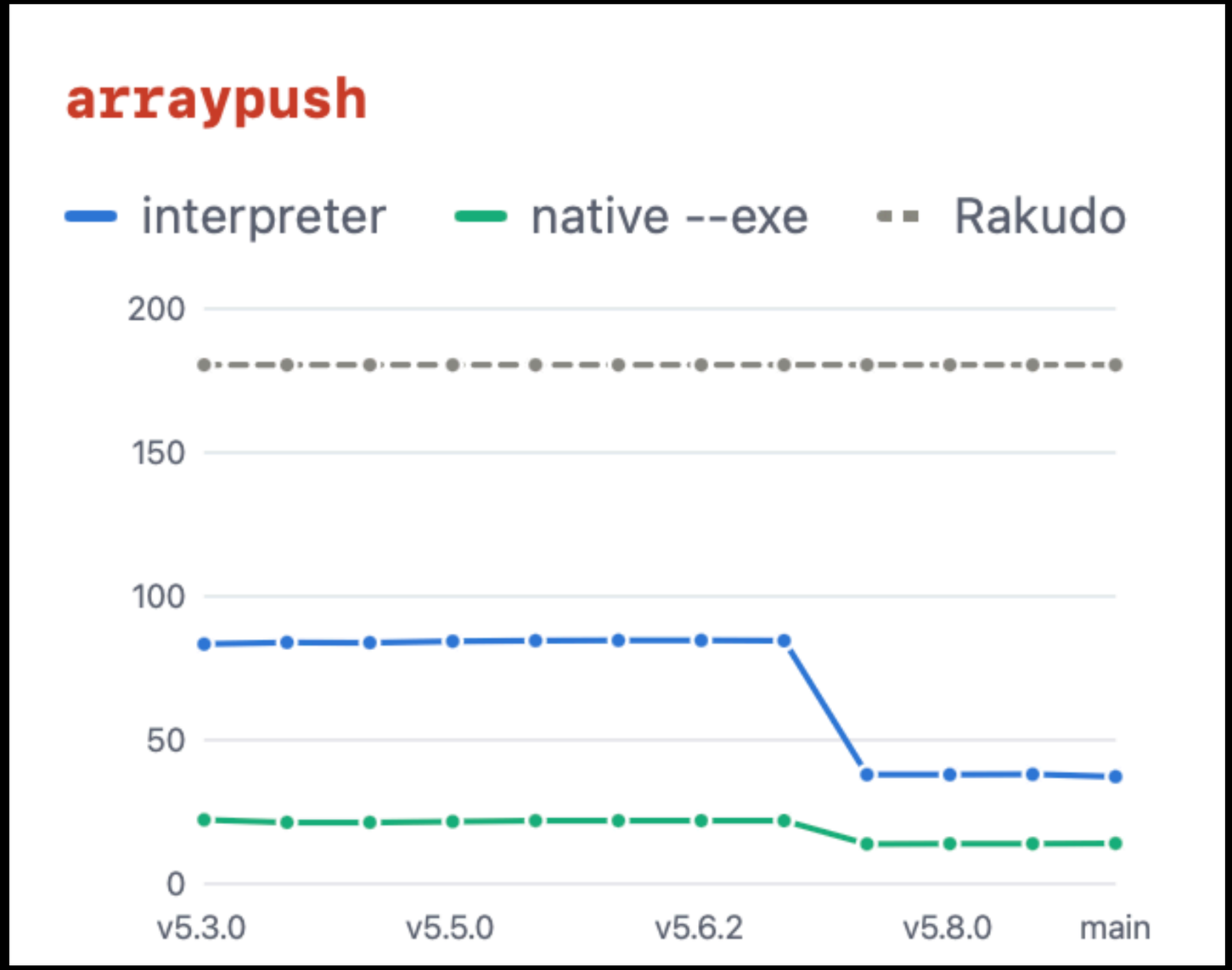
Execution time, ms



raku.online/spec/dashboard

Memory

RSS, MB



raku.online/spec/dashboard



Jovan Trujillo · 8h

I'm so excited you made this, it's making my TrueBASIC interpreter run so much faster on startup. Love it!

Reply Message 1

Compatibility

Rakudo internals

dist	what it calls	why
zef (the installer)	`::("Rakudo::Internals::JSON")` from-json/to-json; `::("Rakudo::Internals").?LL-EXCEPTION`	dependency-free JSON for every META it reads; verbose exception mode. Reached *dynamically*, so it compiles anywhere and explodes (or no-ops) at run time
OpenSSL	`Rakudo::Internals::JSON.from-json`	reads its `resources/libraries.json` at load time
Pakku (installer)	`::JSON.from-json/to-json`, `.IS-WIN`, `.DIR-RECURSE`	JSON, platform probe, directory walk
Pakku::Spec	`::JSON.from-json`	spec parsing
DBIish	`.IS-WIN`, `.REGISTER-DYNAMIC`	platform probe; registering a `$*…` dynamic
NativeHelpers::Blob	`.IS-WIN`	picks its libc
Base64::Native	`.IS-WIN`	same
Digest::SHA1::Native	`.IS-WIN`	same
IO::Socket::Async::SSL	`.NORMALIZE`	Unicode normalization of hostnames

JSON::Native

	100 records parse	100 records write	1 record parse	1 record write
Raku++, JSON::Fast	188–195 µs	223–231 µs	0.61–0.63 µs	0.67–0.69 µs
Raku++, **JSON::Native**	187–199 µs	223–228 µs	0.63–0.64 µs	0.67–0.69 µs
Rakudo, JSON::Fast	4.0 ms	4.5 ms	5.39–5.54 µs	9.72–10.0 µs
Rakudo, **JSON::Native**	4.0 ms	3.1–3.2 ms	6.63–6.84 µs	8.46–8.80 µs
mutsu, JSON::Fast	14.3–14.5 ms	103–106 ms	26.9–28.1 µs	237–240 µs
mutsu, **JSON::Native**	15.1–15.4 ms	107–109 ms	43.1–43.6 µs	259–261 µs

Data::Native

```
use Data::Native;


my %rec  = name => 'Ada', langs => <Raku C>, score => 9.5;
my $json = to-json(%rec);
say from-json($json)<langs>[1];      # C
say sha256-hex($json);

say json-backend();                  # what actually answered
```

The five tags — 29 functions and 5 backend subs

tag	names	reference interface
`json`	`from-json` `to-json`, and `from-jsonl` `to-jsonl` for JSON Lines	`JSON::Fast`; `JSON::Native` 0.2.0 for the JSON Lines pair
`csv`	`from-csv` `to-csv`	`CSV::Native` — no usable reference existed
`digest`	`md5 sha1 sha224 sha256 sha384 sha512`, their `-hex` twins, `hmac` `hmac-hex`	`Digest` + `Digest::HMAC`, with bduggan's `-hex` spelling
`zlib`	`compress` `uncompress` `gzslurp` `gzspurt` `crc32` `adler32`	`Compress::Zlib`
`random`	`crypt_random_buf` `crypt_random` `crypt_random_uniform`	`Crypt::Random`

github.com/ash/raku-modules/tree/main/Data-Native

NQP support

RakuAST (default in Rakudo 2026.09+)

Modules

```
$ raku++ install My::Module

$ raku++ install https://github.com/…

$ raku++ install ./my-dist

$ raku++ install -r lib


$ raku++ test X::Y

$ raku++ uninstall X::Y


$ raku++ install app.raku


$ raku++ --auto app.raku
```

raku++ install

is

zef-compatible

Showcases

raku.online/play/

Raku.online | Learn Raku ↗ | RAKU++ | Play | Install | In use | Integrate | Reference | Read | Support | Raku++

Raku.js Playground Raku++ 5.10.0 · compiled to WebAssembly · no server

example: sierpinski | Open... | Share | Docs | Check | Live | ▶ Run

RAKU SOURCE Copy

```
#!/usr/bin/env rakupp
# The Sierpinski triangle, drawn by an elementary cellular automaton.
#
# Rule 90 is the simplest rule that produces a fractal: each cell in the
# next row is the exclusive-or (+^) of its two upper neighbours. Start
# from a single live cell in the middle of an otherwise empty row and let
# the rule run, and the live cells trace out the Sierpinski gasket — the
# same self-similar triangle you get from Pascal's triangle taken mod 2.
#
# Features: infix bit ops (`+^`), array building, `.map`/`.join` for the
# row-to-text step. No randomness; the picture is identical every run.

my $rows  = 32;
my $width = 2 * $rows - 1;      # room for the triangle to spread each side

# One row of cells: 0 = dead, 1 = live. Seed a single live cell in the
# centre.
my @row = 0 xx $width;
@row[$width div 2] = 1;

for ^$rows {
    # Draw the current generation. '#' is a live cell, a space is a dead one.
    say @row.map({ $_ ?? '#' !! ' ' }).join;

    # Rule 90: next[i] = row[i-1] +^ row[i+1], with the world padded by
    # dead cells beyond both edges.
    my @next;
    for ^$width -> $i {
        my $left  = $i > 0          ?? @row[$i - 1] !! 0;
        my $right = $i < $width - 1  ?? @row[$i + 1] !! 0;
        @next[$i] = $left +^ $right;
    }
```

OUTPUT Copy

```
                               #
                              # #
                             #   #
                            # # # #
                           #       #
                          # #     # #
                         #   #   #   #
                        # # # # # # # #
                       #               #
                      # #             # #
                     #   #           #   #
                    # # # #         # # # #
                   #       #       #       #
                  # #     # #     # #     # #
                 #   #   #   #   #   #   #   #
                # # # # # # # # # # # # # # # #
               #                               #
              # #                             # #
             #   #                           #   #
            # # # #                         # # # #
           #       #                       #       #
          # #     # #                     # #     # #
         #   #   #   #                   #   #   #   #
        # # # # # # # #                 # # # # # # # #
       #               #               #               #
      # #             # #             # #             # #
     #   #           #   #           #   #           #   #
    # # # #         # # # #         # # # #         # # # #
   #       #       #       #       #       #       #       #
  # #     # #     # #     # #     # #     # #     # #     # #
 #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

— exit 0 · 77 ms —
```

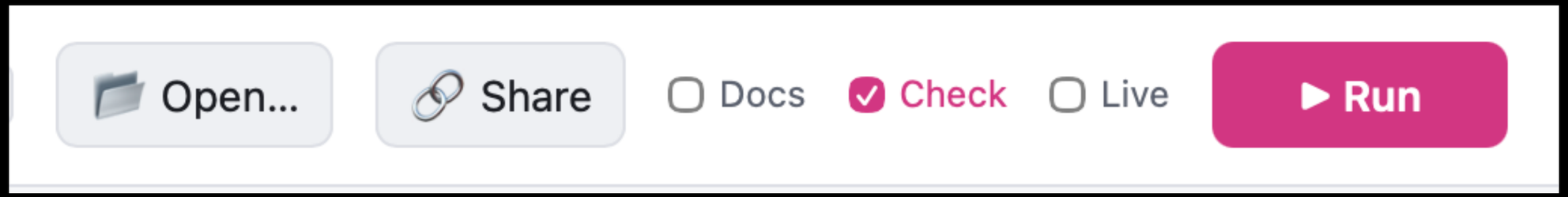
▸ STDIN — input for get / lines / prompt

exit code 0 · 77 ms

Display a menu ∨ Shitov (@ash), author of the Raku Course

Powered by Raku++ compiled to WebAssembly (Raku.js) — your code never leaves the browser | Raku | Spec | Site source

raku.online/play

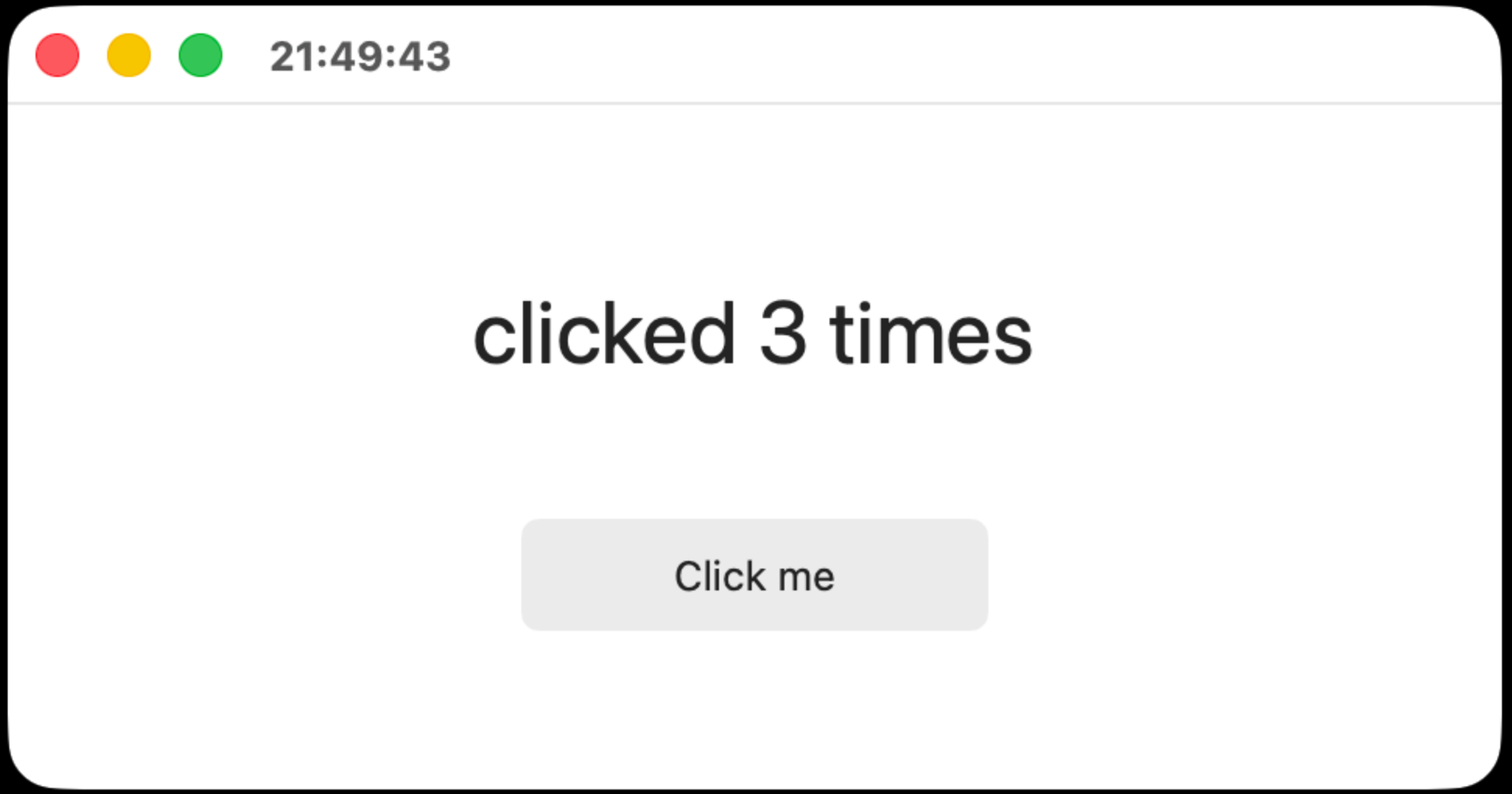


raku.online/play

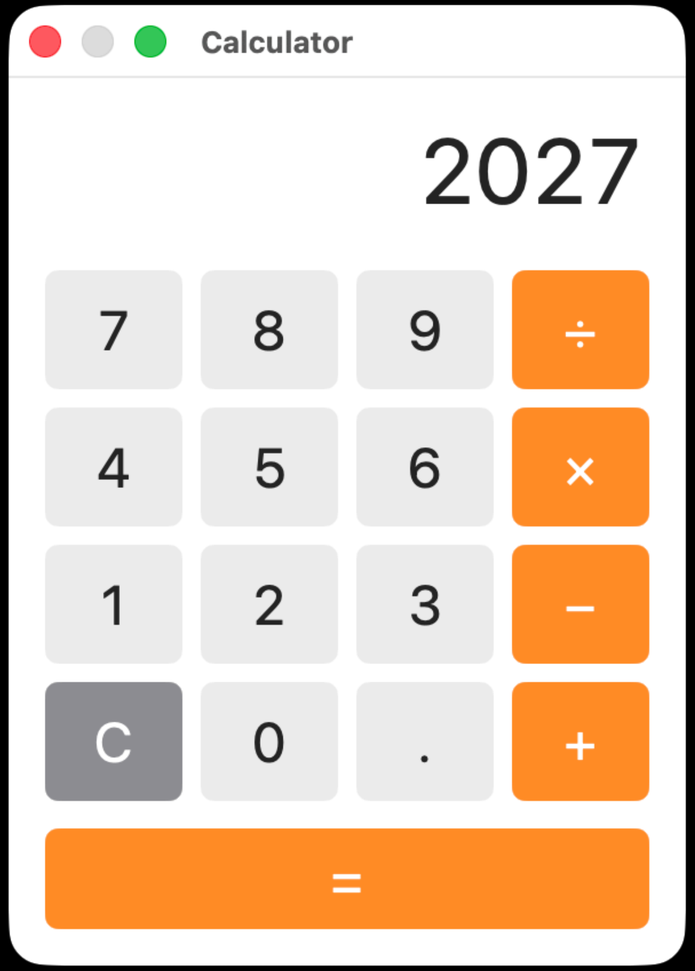
GUI with Raku

```
use GUI::Wings;

app 'Counter', {
    my $n = 0;
    window :title('Camelia counts'), :size(480, 220), {
        my $l = label 'clicked 0 times', :font(28);
        my $b = button 'Click me';

        react {
            whenever $b.clicks { $l.text = "clicked {++$n} times" }
            whenever Supply.interval(1) {
                window.title = DateTime.now.hh-mm-ss
            }
            whenever signal(SIGINT) { done }
        }
    }
}
```

GUI with Raku



GUI with Raku



raku.org

community learn tools docs modules source install ☀

Install Raku

Raku has three engines: Rakudo, the reference implementation; Raku++, an independent interpreter and native compiler written in C++; and mutsu, an independent interpreter written in Rust. They install side by side: the commands are `raku`, `rakupp` and `mutsu`.

Pick your system, then an engine.

Linux macOS Windows Docker Browser Other

Linux

Rakudo Raku++ mutsu

Raku++ runs as `rakupp`. Its installer offers to add the name `raku` too; say no if you want `raku` to stay Rakudo.

```
curl -fsSL https://raku.online/install.sh | sh
```

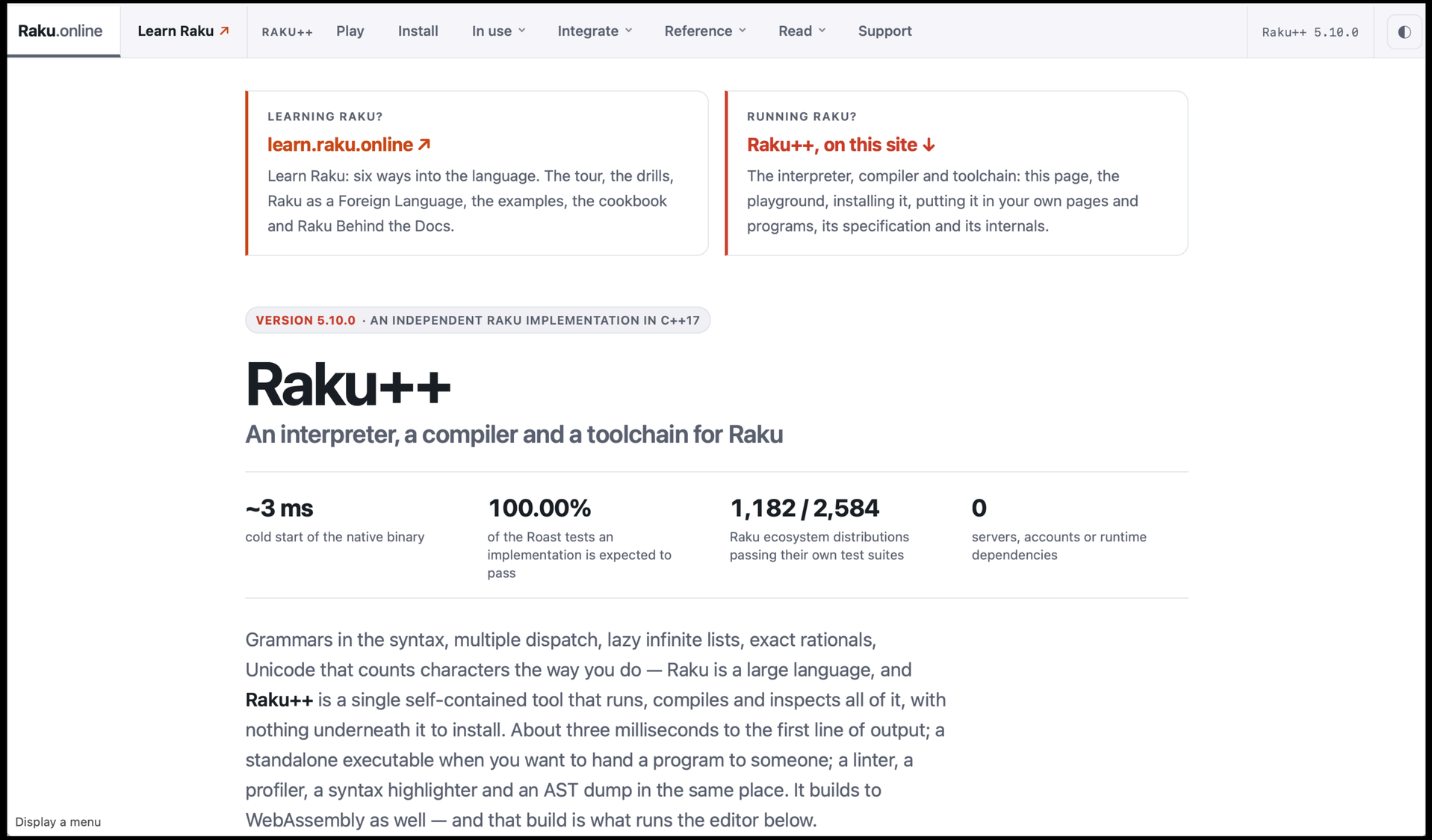
On Debian and Ubuntu there is also a `.deb` (`rakupp-linux-aarch64.deb` on ARM64):

```
curl -fLO https://github.com/ash/rakupp/releases/latest/download/rakupp-linux-x86_64.deb
sudo apt install ./rakupp-linux-x86_64.deb
```

Open a new terminal and check:

```
rakupp -e 'say 6 * 7'
```

raku.online


Raku.online
Learn Raku ↗
RAKU++
Play
Install
In use
Integrate
Reference
Read
Support
Raku++ 5.10.0
LEARNING RAKU?
learn.raku.online ↗
Learn Raku: six ways into the language. The tour, the drills, Raku as a Foreign Language, the examples, the cookbook and Raku Behind the Docs.
RUNNING RAKU?
Raku++, on this site ↓
The interpreter, compiler and toolchain: this page, the playground, installing it, putting it in your own pages and programs, its specification and its internals.
VERSION 5.10.0 · AN INDEPENDENT RAKU IMPLEMENTATION IN C++17
Raku++
An interpreter, a compiler and a toolchain for Raku
~3 ms
cold start of the native binary
100.00%
of the Roast tests an implementation is expected to pass
1,182 / 2,584
Raku ecosystem distributions passing their own test suites
0
servers, accounts or runtime dependencies
Grammars in the syntax, multiple dispatch, lazy infinite lists, exact rationals, Unicode that counts characters the way you do — Raku is a large language, and Raku++ is a single self-contained tool that runs, compiles and inspects all of it, with nothing underneath it to install. About three milliseconds to the first line of output; a standalone executable when you want to hand a program to someone; a linter, a profiler, a syntax highlighter and an AST dump in the same place. It builds to WebAssembly as well — and that build is what runs the editor below.
Display a menu

Adoptions

Running it

Sparrow6 is not shipped here — install it, with either installer. Both write the same store, so one co
direct dependencies (`JSON::Fast` , `Data::Dump` , `Hash::Merge` , `YAMLish` , `Terminal::ANSICol`
resolves to a plan of 17 distributions once theirs are counted; the installer handles all of it:

```
rakupp install Sparrow6      # or: zef install Sparrow6
```

Then just run it — no `RAKULIB` , no `-I` :

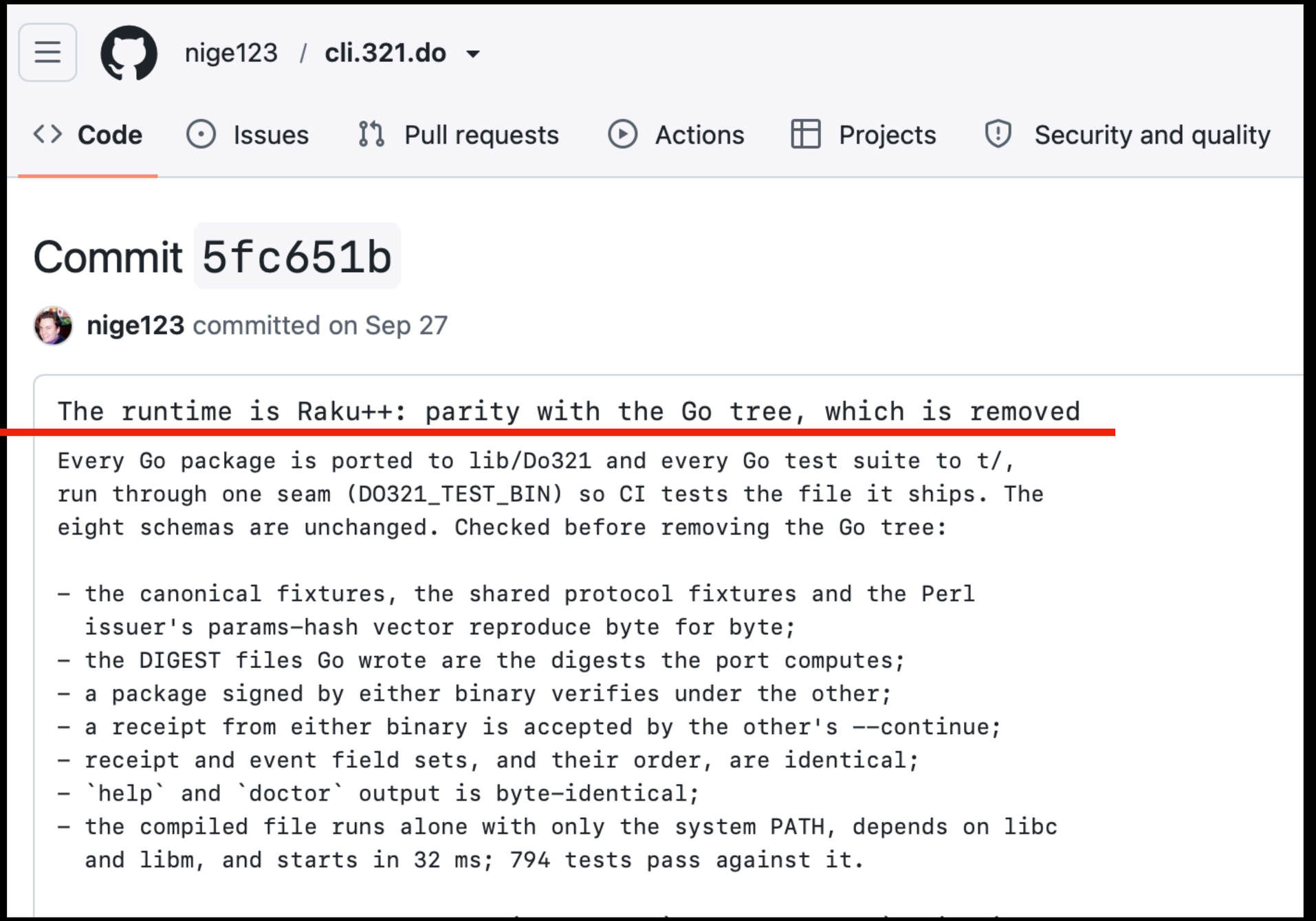
```
cd live/sparrow && rakupp scenario.raku
```

`compare.sh` runs the scenario under both engines and diffs stdout, with Sparrow's wall-clock line

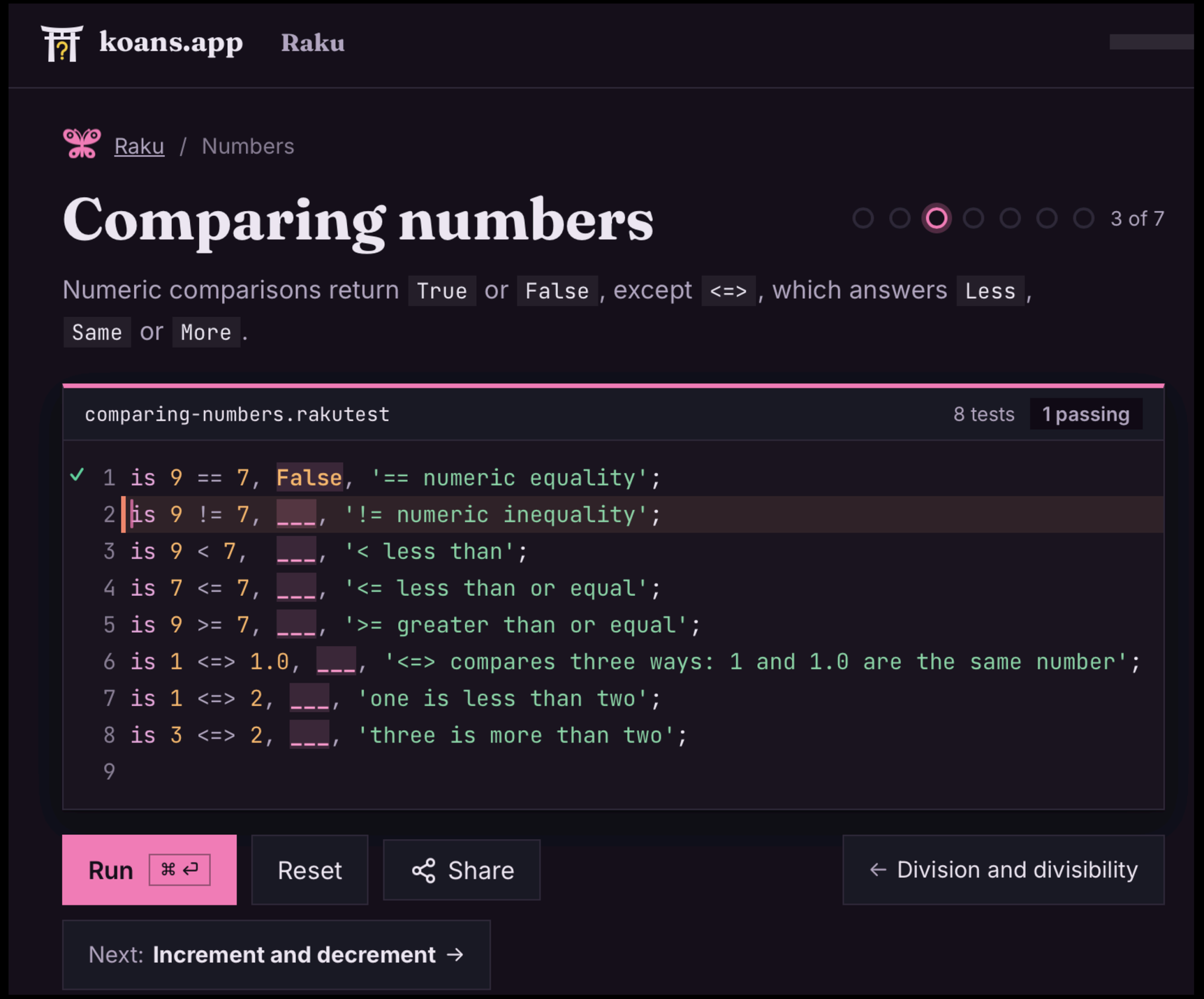
```
sh live/sparrow/compare.sh
```

Both take `rakupp` and `rakudo` from `PATH` . Running out of a build tree instead? Name it: `RAKUPP`
`live/sparrow/compare.sh` .

WHAT	WHO	WHAT IT DOES WITH RAKU++
RakuppLink	Anton Antonov	ships the Wolfram binding as a paclet in Wolfram's own repository
Raku Playground	Fernando Correa de Oliveira	offers rakupp as one of four runtimes, in the browser
koans.app	Naoum Hankache	grades 105 Raku koans with Raku.js in the browser, one of fourteen languages on a koans site that grew out of Raku Koans
rakupp-dsci	Alexey Melezhik	ports the Raku++ release matrix to another CI
sibl	4zv4l	packages rakupp in a personal Guix channel
321	Nigel Hamilton	compiles an agent runtime, written in Raku, into one standalone file per system
iz4	Nigel Hamilton	ships a tool for keeping a project's invariants as Raku++-built files that update themselves
xc	hjxkyw	compiles a business language to TL++, the language of the TOTVS Protheus platform, with a compiler written in Raku and run by rakupp
Raku full-stack skills	e. alvarez	teaches a coding agent to build Raku applications for Raku++, web backends included


nige123 / cli.321.do
Code
Issues
Pull requests
Actions
Projects
Security and quality
Commit 5fc651b
nige123 committed on Sep 27
The runtime is Raku++: parity with the Go tree, which is removed
Every Go package is ported to lib/Do321 and every Go test suite to t/,
run through one seam (DO321_TEST_BIN) so CI tests the file it ships. The
eight schemas are unchanged. Checked before removing the Go tree:
- the canonical fixtures, the shared protocol fixtures and the Perl
issuer's params-hash vector reproduce byte for byte;
- the DIGEST files Go wrote are the digests the port computes;
- a package signed by either binary verifies under the other;
- a receipt from either binary is accepted by the other's --continue;
- receipt and event field sets, and their order, are identical;
- `help` and `doctor` output is byte-identical;
- the compiled file runs alone with only the system PATH, depends on libc
and libm, and starts in 32 ms; 794 tests pass against it.

koans.app


koans.app
Raku
Raku / Numbers
Comparing numbers
3 of 7
Numeric comparisons return True or False, except <=>, which answers Less, Same or More.
comparing-numbers.rakutest
8 tests
1 passing
1 is 9 == 7, False, '== numeric equality';
2 is 9 != 7, ___, '!= numeric inequality';
3 is 9 < 7, ___, '< less than';
4 is 7 <= 7, ___, '<= less than or equal';
5 is 9 >= 7, ___, '>= greater than or equal';
6 is 1 <=> 1.0, ___, '<=> compares three ways: 1 and 1.0 are the same number';
7 is 1 <=> 2, ___, 'one is less than two';
8 is 3 <=> 2, ___, 'three is more than two';
9
Run ⌘ ⏎
Reset
Share
← Division and divisibility
Next: Increment and decrement →

CLI

```
$ raku
$ rakupp
$ raku++
```

```
-v
--info
```

```
# uppercase every line
perl   -ne 'print uc'            f.txt
rakupp -ne 'say $_.uc'           f.txt

# second whitespace-separated field
perl   -lane 'print $F[1]'       f.txt
rakupp -lane 'say @F[1]'         f.txt

# CSV second column (literal separator)
perl   -F, -lane 'print $F[1]'   f.csv
rakupp -F, -ane  'say @F[1]'     f.csv

# split on a run of digits (regex separator — note the explicit /…/)
perl   -F'[0-9]+'  -lane 'print join "|", @F'   f.txt
rakupp '-F/\d+/'   -ane  'say @F.join("|")'     f.txt

# sed-style in-place edit, with a backup
perl   -pi.bak -e 's/foo/bar/'                  *.txt
rakupp -pi.bak -e '$_ = $_.subst("foo", "bar")' *.txt

# whole-file (slurp) transform, in place
perl   -0777 -pi -e 's/\n\n+/\n/g'                        f.md
rakupp -0777 -pi -e '$_ = $_.subst(/\n\n+/, "\n", :g)'    f.md

# consume find -print0
find . -name '*.log' -print0 | perl   -0ne 'print "$_\n"'
find . -name '*.log' -print0 | rakupp -0ne 'say $_'
```

```
$ raku++ https://github.com/…
```

```
-c
--lint
```

$ raku++ --profile examples/quicksort.raku

```
Profile — wall time; builtins are attributed to their caller

  excl(ms)   incl(ms)      calls  routine
     8.009      8.033         10  quicksort (/Users/ash/rakupp/examples/quicksort.raku)
     1.632      2.061         19  merge-sort (/Users/ash/rakupp/examples/quicksort.raku)
     0.428      0.428          9  merge (/Users/ash/rakupp/examples/quicksort.raku)
     0.024      0.024         11  quicksort (/Users/ash/rakupp/examples/quicksort.raku)
```

```
$ raku++ --hints -e 'my $n = 10**4; say (1 + 1/$n) ** $n'

hint: line 1: an exact Rat raised to 10000: its numerator and
denominator run to about 40000 digits each, and the result becomes
a Num anyway; write the base with a Num (1e0 instead of 1) to
compute in Num directly

2.718145926825225
```

```
$ raku++ --seed=10 -e'say 100.rand'
```

```
--ast
--rakuast
```

```
[ash@MacBook-Air-4 rakupp % raku --ast tools/bench/loopsum.raku
Program
  Assign          |  =
    VarExpr       |   $total [decl my]
    IntLit        |   0
  For             |  ($_)
    list:         |
      Range       |    ..
        IntLit    |     1
        IntLit    |     1000000
    Assign        |   +=
      VarExpr     |    $total
      VarExpr     |    $_
  Call            |  say
    VarExpr       |   $total
```

```
[ash@MacBook-Air-4 rakupp % raku --rakuast tools/bench/loopsum.raku
 StatementList                          | my $total = 0; for 1 .. 1000000 { $total += $_ } say $total
   Statement::Expression                |  my $total = 0
     VarDeclaration::Simple             |   my $total = 0
       Name                             |    total
       Initializer::Assign              |    = 0
         IntLiteral                     |     0
   Statement::For                       |  for 1 .. 1000000 { $total += $_ }
     ApplyInfix                         |   1 .. 1000000
       Infix                            |    ..
       ArgList                          |    1, 1000000
         IntLiteral                     |     1
         IntLiteral                     |     1000000
     Block                              |   { $total += $_ }
       Blockoid                         |    { $total += $_ }
         StatementList                  |     $total += $_
           Statement::Expression        |      $total += $_
             ApplyInfix                 |       $total += $_
               MetaInfix::Assign        |        +=
                 Infix                  |         +
               ArgList                  |        $total, $_
                 Var::Lexical           |         $total
                 Var::Lexical           |         $_
   Statement::Expression                |  say $total
     Call::Name::WithoutParentheses     |   say $total
       Name                             |    say
       ArgList                          |    $total
         Var::Lexical                   |     $total
```

--doc-links

```
ash@MacBook-Air-4 rakupp % raku --doc-links tools/bench/loopsum.raku
tools/bench/loopsum.raku: 8 pages on docs.raku.org for 10 constructs (index of 2026-11-07)

Syntax
  my   2  https://docs.raku.org/syntax/my
  for  3  https://docs.raku.org/syntax/for

Types
  Int  2, 3  https://docs.raku.org/type/Int

Routines
  say  6  https://docs.raku.org/type/independent-routines#sub_say

Operators
  item assignment =  2  https://docs.raku.org/language/operators#infix_=_(item_assignment)
  infix ..           3  https://docs.raku.org/language/operators#infix_%2E%2E
  assignment +=      4  https://docs.raku.org/language/operators#Assignment_operators

Variables
  $_  4  https://docs.raku.org/syntax/%24_
```

```
--lsp
--mcp
```

```
$ raku++ --watch prog.raku
```

`--jit` just-in-time

`--cnp` copy-and-patch

(experimental)

--cpp




```
say "Hello, Raku!";
```




```
rtSinkCallStmt(RT, (rtUncontained(),
rtCallB(RT, __bfp0, "say",
ValueList{Value::str("Hello, Raku!")})));
```

Embed

▶ Run

```raku
class Circle {
    has $.radius;

    method area {
        3.14 * $.radius * $.radius;
    }

    method describe {
        'The area is ' ~ self.area;
    }
}

my $c = Circle.new(radius => 2);
say $c.describe; # The area is 12.56
```

OUTPUT

```
The area is 12.56

— exit 0 · 25 ms —
```

course.raku.org

Raku on WordPress

Raku Snippets is a WordPress plugin. Give a code block the class `raku`, and your readers can edit the program and run it, right in the post.

This is what a reader sees — change the code and press ▶ Run:

▶ Run

```raku
sub postfix:<!>($n) { [*] 1..$n }

say 5!;
say 30!;
```

Download raku-snippets.zip

version 1.0.0 · 1 KB · WordPress 5.0+, PHP 7.0+

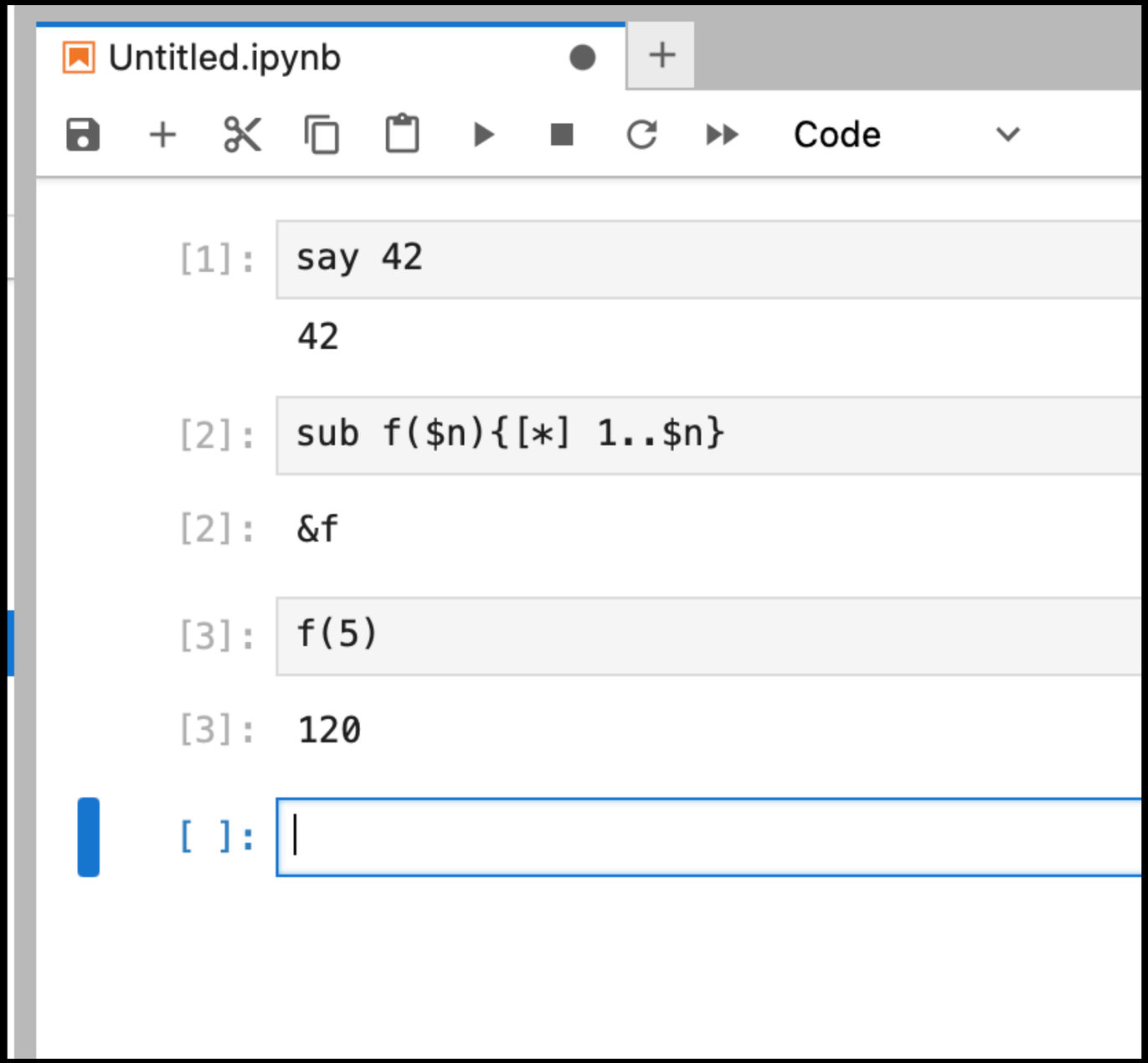
Raku code Class raku

`map`:

```
say (1 .. 999).grep(* %% (3 | 5)).sum;
```

Here, we compute the sum of all the numbers from 1 to 999 that are di

A similar example with strings and `first`:


Untitled.ipynb
Code
[1]:
say 42
42
[2]:
sub f($n){[*] 1..$n}
[2]:
&f
[3]:
f(5)
[3]:
120
[ ]:

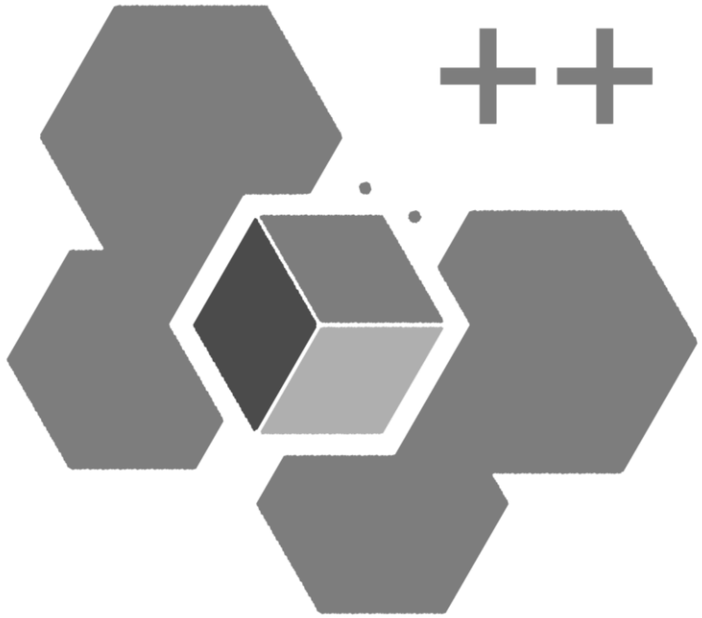
AntonAntonov/

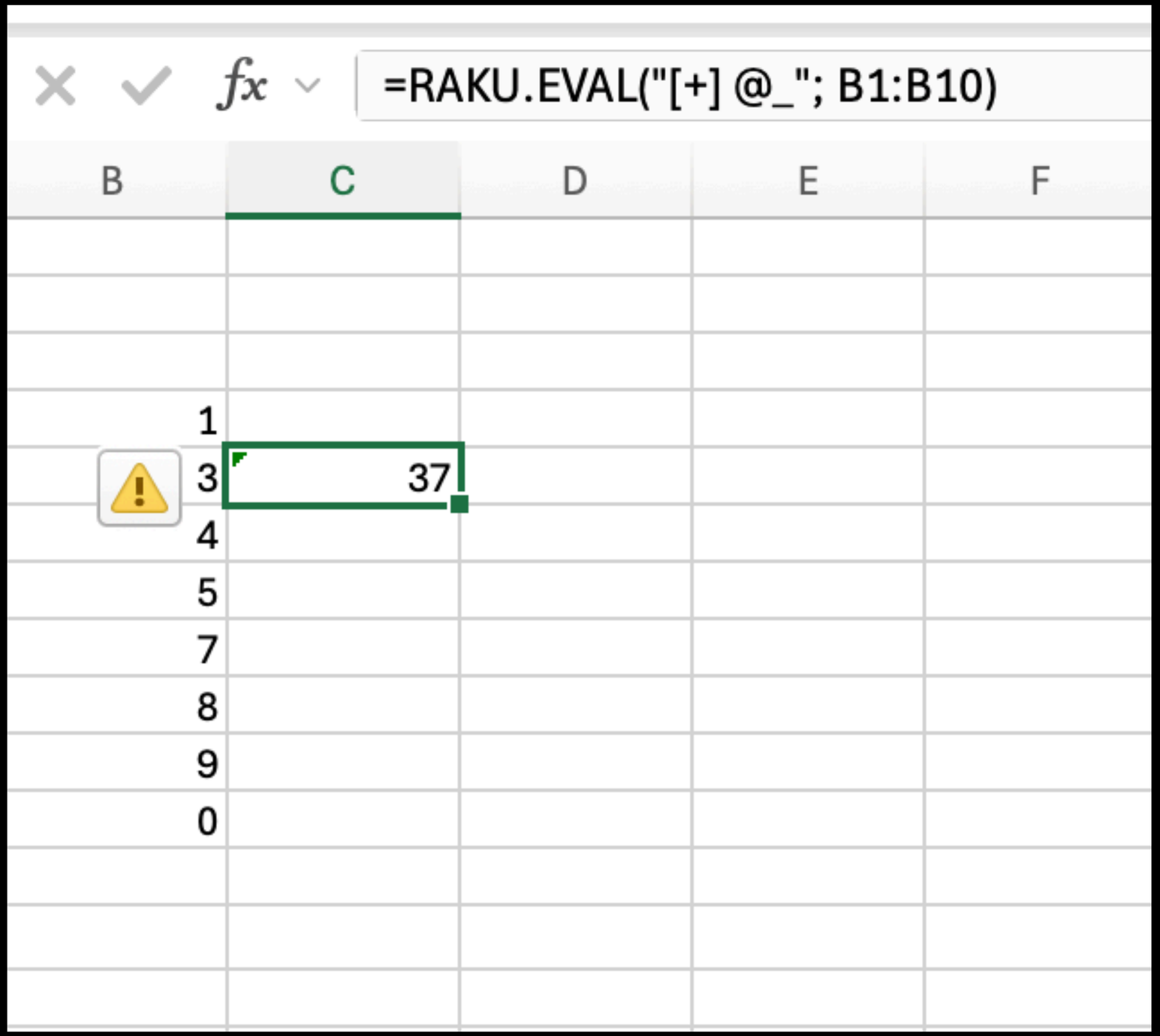
RakuppLink

Functions for interacting with Rakupp (Raku++)

Contributed by: Anton Antonov

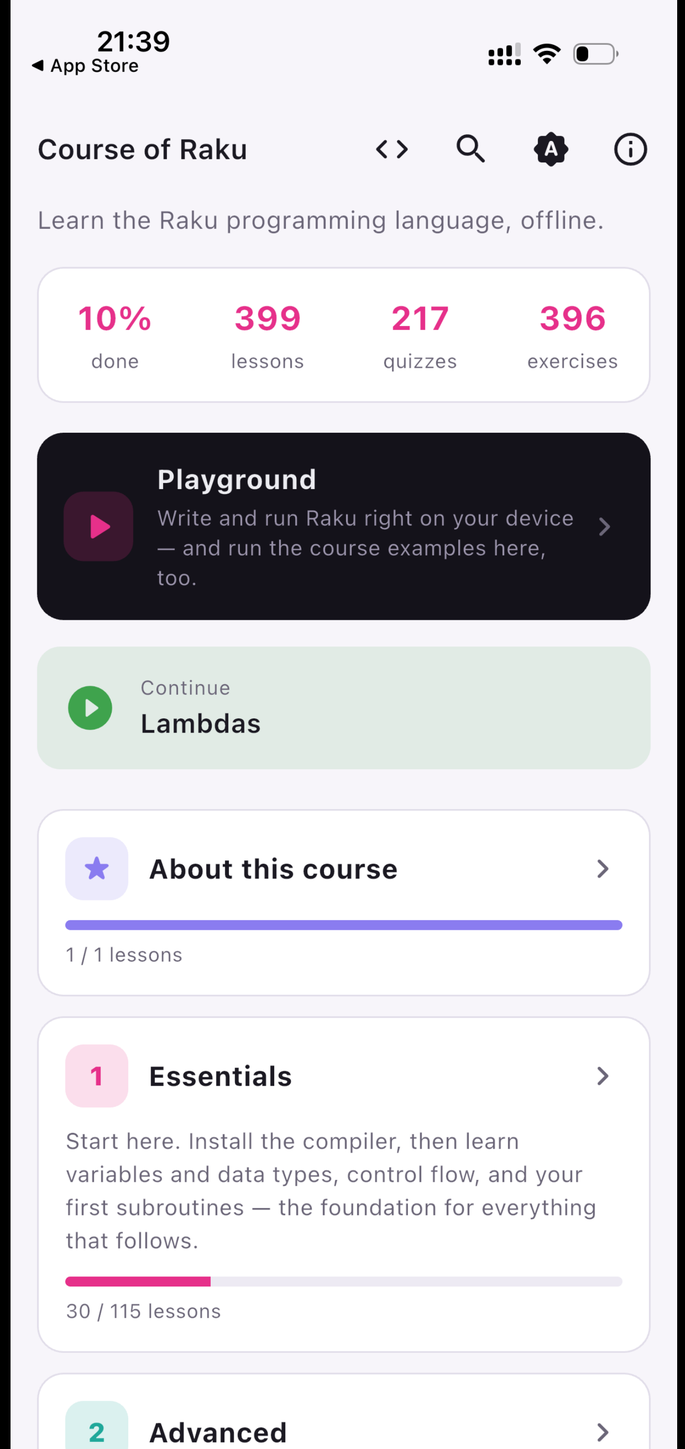
Rakupp interpreter interaction functions based on Foreign Function Interface.

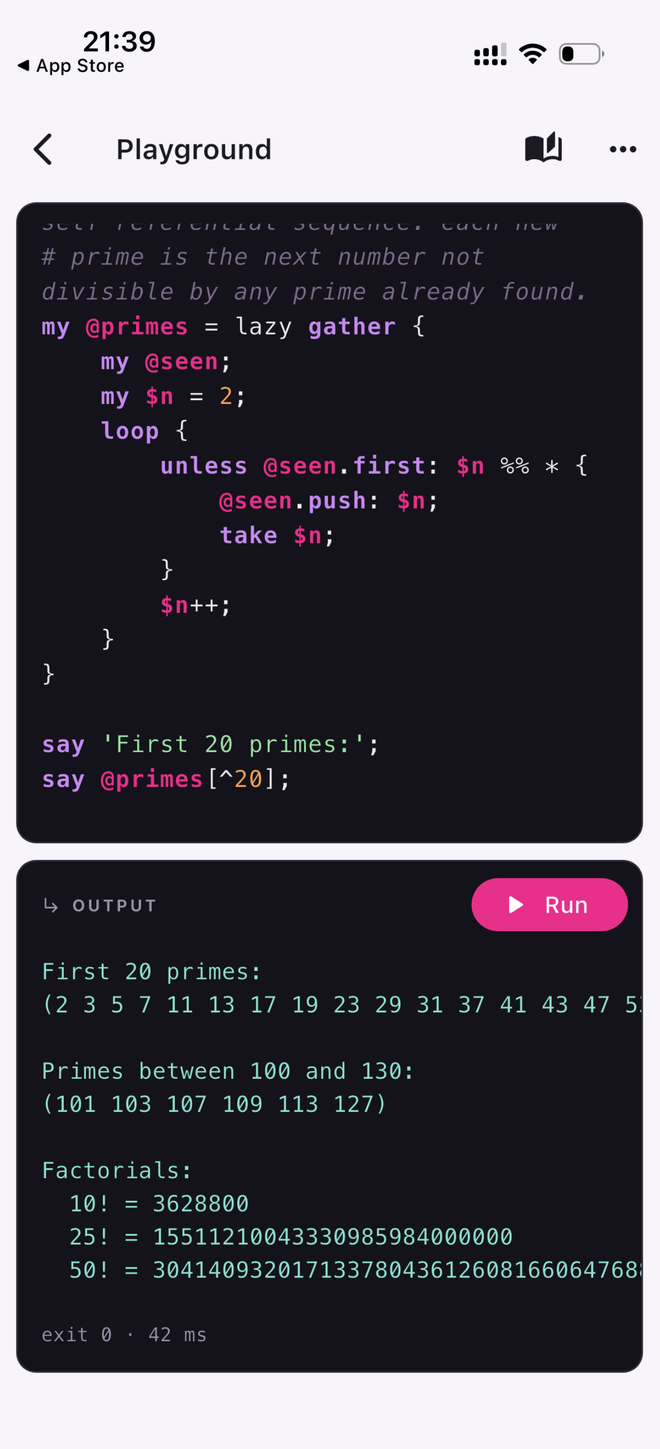



=RAKU.EVAL("[+] @_"; B1:B10)
B
C
D
E
F
1
3
37
4
5
7
8
9
0

The app: *Learn Raku*







pypi.org/project/rakulang/

Type '/' to search projects

Blog Docs Log in Register

rakulang 5.7.0

Raku grammars and an embedded Raku++ interpreter for Python, over librakupp

```
pip install rakulang
```


rakulang — Raku from Python

`rakulang` is based on Raku++, an implementation of the Raku language written in C++. The Raku++ engine comes inside the package.

Run Raku code and Raku grammars inside a Python program. Raku is good at text: its grammars turn messy input into structured data, and its numbers are exact (integers of any size, rationals). `rakulang` lets a Python program use that without leaving Python.

Install

```
pip install rakulang
```

That is all. You need Python 3.9 or later on macOS, Linux or Windows. There is nothing to compile and nothing else to install: the package carries its own Raku engine.

If pip answers `error: externally-managed-environment` (Ubuntu, Debian and Homebrew's Python do), your system Python does not take packages directly. Make a virtual environment for your project and install into that:

Project links

Data verified by PyPI on Nov 11, 2026

Documentation
Issues
Repository
Homepage

Key dates

PyPI data

Released: Nov 11, 2026

LATEST RELEASE

1 maintainer

PyPI data

andrewshitov

Display a menu

```
import rakulang

raku = rakulang.interpreter()

raku.eval('say "Hello from Raku!"')

print(raku.eval("(1..10).sum"))
print(raku.eval("2 ** 100"))
print(raku.eval("<apple banana cherry>.map(*.uc)"))
```

```
$ python3 1.py
Hello from Raku!
55
126765060022822940149670320537 6
['APPLE', 'BANANA', 'CHERRY']
```

```
import rakulang

source = """
grammar Shopping {
    rule  TOP  { <item>+ }
    rule  item { <name> '=' <qty> }
    token name { \\w+ }
    token qty  { \\d+ }
}

class ShoppingActions {
    method item($/) { make $<qty>.Int }
    method TOP($/)  { make $<item>.map(*.made).sum }
}
"""

shopping = rakulang.Grammar.from_source(source, name="Shopping",
                                        actions="ShoppingActions")

m = shopping.parse("milk=2  bread = 1\neggs=12")

for item in m["item"]:
    print(item["name"].str(), item["qty"].int())

print("total:", m.made)
```

Bindings

Raku inside your own program. Run Raku code, call Raku subs with your values, and parse text with Raku grammars — the results come back as your language's own types.

Python	**Ready.** `pip install rakulang` — macOS, Linux and Windows, the engine inside the package.
Wolfram Language	**Available** as the RakuppLink paclet in Wolfram's Paclet Repository.
JavaScript	Preview, for Bun. Built from the repository.
Go	Preview, with cgo. Built from the repository.
Rust	Preview, a crate built from the repository.
C++	Preview, two headers that ship with the engine.
iOS and Android	The engine compiled into an app and called through the C API from Swift, Kotlin or Flutter.



raku.online