universidade de aveiro

The Ruby Programming Language

Developed by:
- André Oliveira 107637
- Duarte Cruz 107359
- Hugo Correia 108215

Group 1 CSLP

What is Ruby?

Ruby is a dynamic, open-source programming language that focuses on simplicity and productivity.

It was created by Yukihiro “Matz” Matsumoto, blending elements of his favorite languages to form a new language that balanced functional programming with imperative programming.

Matz aimed to make Ruby natural, not simple, in a way that mirrors life.

Design Philosophy:

- Focus on programmer happiness
- Balance between simplicity and productivity
- Suitable for both beginners and experienced developers

"Trying to make Ruby natural, not simple," - Matz

Purpose and Use-Cases

Web Development

- Ruby on Rails, an opinionated web framework, powers popular platforms like GitHub, Shopify, Airbnb, and Twitch.

Scripting and Automation

- Used as a "glue" language for tools like Vagrant, Chef, Puppet, and Homebrew.

Prototyping and Data Analysis

- Suited for prototyping and data analysis tools like Jupyter, Metasploit, Cucumber, and Sass.

Strengths

Readability and Writability

- Intuitive syntax, mirroring natural language
- Appeals to new programmers

Flexibility and Productivity

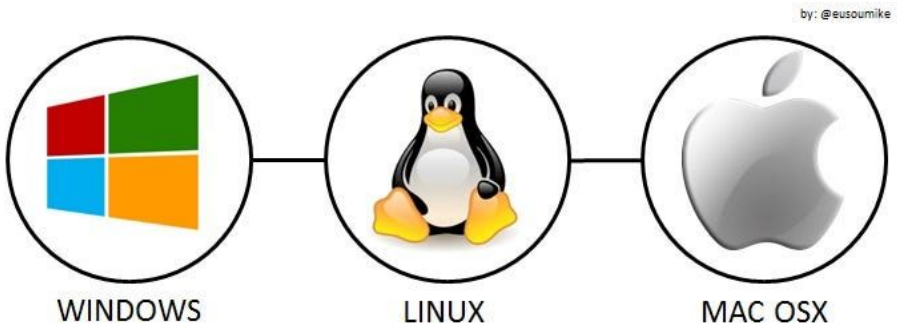
- Dynamic typing and duck typing
- Concise and expressive code

Community and Support and Ecosystem

- Large developer community
- Abundant libraries and tools
- Popular in web development

Portability and Compatibility

- Available on Windows, macOS, Linux, and BSD
- Compatible with C, C++, Java, Python, and Perl



Weaknesses

Performance

- Slower execution compared to C++ or Java

Scalability and Concurrency

- Challenges in managing multithreading effectively

Memory Usage

- Less efficient memory management

Type Safety

- Dynamic typing may lead to runtime errors

Ecosystem and Dependency Management

- Complex dependency management

Memory Management and Garbage Collection

Garbage Collector

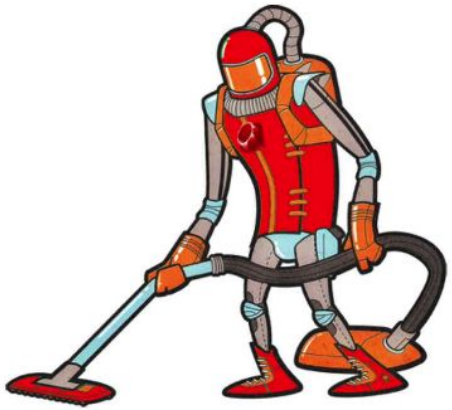
- Generational GC, Incremental GC, Memory Compaction

Memory Usage

- Historical criticism for higher usage
- Improvements in Ruby 3 for efficiency
- Development practices for efficiency

Memory Management

- Automation with GC
- Often seen as less efficient, leading to higher memory usage
- Manual interventions when needed
- Monitoring and diagnostics in Ruby 3



Programming Paradigms

Object-Oriented

- Everything is an object
- Supports inheritance, polymorphism, encapsulation

Functional

- Features from functional languages
- Supports first-class functions and closures

Imperative

- Uses statements to change program state
- Supports loops and conditionals

Procedural

- Supports procedural programming

Reflective

- Metaprogramming for manipulating structure at runtime
- Reflection for object inspection

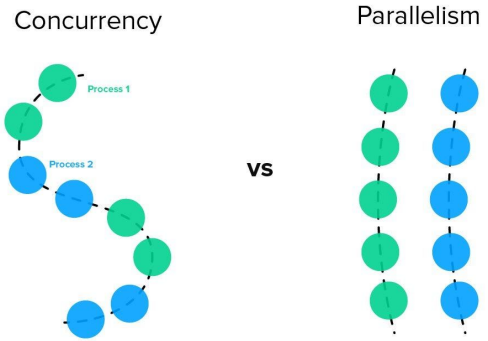
Metaprogramming

- Treating programs as data

Scripting

- Used for task automation and interaction with the OS

Parallelism and Concurrency



Concurrency

- Threads and fibers for concurrent execution
- GIL limits native thread execution

Parallelism

- Challenges due to GIL
- Achievable through processes (e.g., with the parallel gem)
- Alternative implementations (JRuby, Rubinius) for true parallelism

The future of Ruby

Matz's Vision and Ruby 3x3

- Three times faster by Ruby 3
- JIT compilation in MRI

Ractor for Concurrency

- Actor-like concurrency abstraction

Type Checking and Sorbet

- Static type checker for Ruby

Guilds for Parallelism

- Proposed for true parallelism

Active Development of Frameworks and Gems

- Continuous evolution of Ruby on Rails and other frameworks

Community Engagement and Conferences

- RubyConf, RailsConf shaping the language's future

Emerging Trends and Integration

- Integrating machine learning, AI, and serverless computing