



겉핥기식 Rust (Rust 소개)

**** Rust를 처음 공부하는 초보자가 적은 글 (많은 내용 없음)**

**** 틀린 내용 있을 수도 있음 (지적 환영)**

**** 러스트 개발을 위한 docs 가 아님!**



Why i use rust? → 속도, 안전성, 정확성

Rust is a language that provides a unique combination of performance, reliability, and safety.

It is designed to be fast and efficient, while also preventing common programming errors such as null pointer dereferences and data races.

Additionally, Rust has a growing and supportive community, making it a great choice for a variety of applications.

[What is Rust ?](#)

[Why Rust?](#)

[Memory Safety Issue??](#)

[vs C++?](#)

[Rust basic](#)

[Rust Tools](#)

[Rust Ownership](#)

[Rust-companies](#)

[Python? Rust?](#)

[Python과의 tool 비교](#)

[속도 비교](#)

[Rust의 단점?](#)

[Rust,, 적용할 수 있을까??](#)

[출처](#)

What is Rust ?

- **Low level 프로그래밍 언어**, 안전 / 빠름 에 초점이 맞추어져 있음.
- 2015년, 비교적 최근에 만들어진 언어
- C++과 비슷한 방식으로 컴파일하는 언어
 - `rustc` uses LLVM as its backend.
- Rust 는 많은 platforms and architectures 들을 지원함.
 - x86, ARM, WebAssembly, ...
 - Linux, Mac, Windows, ...
- Rust는 다양한 영역에 사용되는 언어
 - 네트워크
 - 웹어셈블리
 - 임베디드 시스템
 - servers 등
- “Portable”을 목표로 만들어져 platform-independent함.
- Rust's package manager, Cargo, includes built-in support for cross-compiling Rust code to run on different platforms.

Why Rust?

- 공식 docs에서 주장하는 내용 3가지

“Performance”

Rust is blazingly fast and memory-efficient: **with no runtime or garbage collector**, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.

“Reliability”

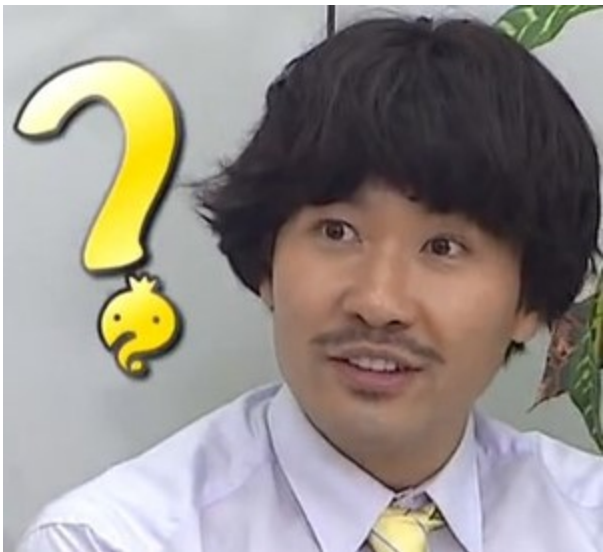
Rust’s rich type system and **ownership model guarantee memory-safety and thread-safety** — enabling you to eliminate many classes of bugs at compile-time.

“Productivity”

Rust has great documentation, a **friendly compiler** with useful error messages, and top-notch tooling — an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.

- Java 보다50% , 파이썬보다 98% 에너지 효율이 높음!

	Energy		Time
(c) C	1.00	(c) C	1.00
(c) Rust	1.03	(c) Rust	1.04
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(v) C#	3.14
(c) Swift	2.79	(v) Lisp	3.40



ESG 경영용 언어?

Memory Safety Issue??

Rust는 수동으로 할당하거나 할당해제할 필요가 없음! → Ownership

- C / C++ 언어는 개발자가 직접 메모리 관리를 해주어야함 → 가비지컬렉터 x
 - 예) char* 문자열 만들기 위해 메모리에 공간 할당 요청 후 free
- Memory Safety Issue는 보통 메모리 할당에서 발생함 (비할당 구역 접근, free된 공간 접근, 2번 할당 해제... 등)

vs C++?

- Memory 안전성 : buffer overflows, null pointer dereferences, and memory leak 등 개발자들에게 발생할 수 있는 문제를 ownership, borrowing system으로 메모리 접근이 안전하고 런타임 환경 에러 해결
- Type 안전성 : Rust는 매우 강력한 typed lang이기 때문에, 컴파일타임에서 발생할 수 있는 타입관련 버그를 해결
- Thread 안전성: C++에 비해서 다양한 concurrent 관련 표준 라이브러리를 제공

Rust는 C++보다 더 안전성에 초점을 두고 개발되었다,,!



대충 요약해보면, c와 c++ 보다 메모리 관리 / 보안 취약점이 덜하면서도 속도는 비슷하고, 파이썬과 JS 같은 High-level 언어들과 같은 개발 경험(메모리 직접 할당, 해제 필요 x)을 제공한다??

Rust basic

Rust Tools

- `rustup` : the Rust toolchain installer and updater
 - rust 버전관리 가능
 - rust 새로운 release 시, rustc, cargo 업데이트에 사용
- `rustc` : the **Rust compiler** which turns `.rs` files into binaries and other intermediate formats.
- `Cargo` : the Rust dependency manager and build tool
 - <https://crates.io/> 에서 dependency 다운로드
 - 프로젝트 빌드시 `rustc` 에 전달

Rust Ownership

```
fn say_hi(name:String){
    println!("HI {}!", name);
}
fn main(){
    let my_name = String::from("LSY");
    say_hi(my_name);
    println!("Nice to Meet you {}!", my_name);
}

// 예상 ouput)
// HI LSY!
// Nice to Meet you LSY!
```

Rust-companies

- Amazon(1) - Here at AWS, we love Rust, too, because it helps AWS write highly performant, safe infrastructure-level networking and other systems software.
- Atlassian - We use Rust in a service for analyzing petabytes of source code.
- Brave (GitHub) - Adblock engine for Brave Browser.
- Discord (1, 2) - Communication Platform designed for communities.
- Figma(1) - Our real-time multiplayer syncing server (used to edit all Figma documents) is written in Rust.
- Google (1) - In Android 13, about 21% of all new native code (C/C++/Rust) is in Rust. There are approximately 1.5 million total lines of Rust code in Android across new functionality and components such as Keystore2, the new Ultra-wideband (UWB) stack, DNS-over-HTTP3, Android’s Virtualization framework (AVF), and various other components and their open source dependencies. These are low-level components that require a systems language which otherwise would have been implemented in C++.
- Meta (1,2) - Facebook’s primary source control system is partially written in Rust.
- Mozilla (GitHub, Servo) - Building the Servo browser engine, integrating into Firefox, other projects.

Python? Rust?

Python과의 tool 비교


- Rust가 많은 양의 툴을 제공함

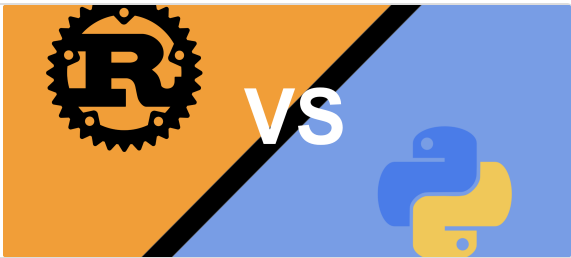
	Python	Rust
패키지 관리자	pip, conda	Cargo
포매퍼	black..	Cargo fmt
린터	flake8	Cargo clippy
테스트	pytest	Cargo test
프로젝트 환경관리	poetry, pipenv,,	Cargo new
문서화	sphinx	Cargo docs

속도 비교

파이썬과 러스트

안녕하세요, 추천팀 제이입니다. 저는 팀 내 프로젝트에 새롭게 러스트(Rust)를 도입하는 과정에서 동일한 애플리케이션을 파이썬(Python)과 러스트로 각각 개발하여, 비교 및 분석하는 과정을 경험했습니다. 이 경험을 토대로, 실무적 관점에서 파이썬과 러스트의 차이점을 말씀드리고, 러스트 도입 시 개발자들이 고려하면 좋을 부분들

 <https://tech.kakao.com/2023/03/02/python-and-rust/>



- 어플리케이션 환경
 - Kafka / MongoDB / API
 - 애플리케이션은 크게 카프카(Kafka) 메시지로 유입되는 사용자 피드백에서 데이터를 추출해 수집하는 수집 모듈과 웹 API로 요청된 사용자 쿼리를 해석해 사용자 지표를 실시간으로 계산해 응답하는 쿼리 모듈로 나뉩니다. 성능 비교는 더 중요하다고 생각하는 수집 모듈을 기준으로 진행했습니다.

언어	평균 시간 (나노초)	최대 시간 (나노초)	최소 시간 (나노초)	평균 메모리 사용량 (바이트)
파이썬	3,852,418	7,380,500	931,875	51,053
러스트	2,014,239	5,465,708	1,072,875	11,397

Figure 2. 성능 비교

이미지 출처 : kakao-파이썬과 러스트

- 러스트가 파이썬보다 작업을 1.9배 정도 빠르게 처리.
- 메모리 사용량은 파이썬이 러스트 보다 4.5배 정도 더 사용
- 표에는 없지만 파이썬의 CPU 사용량은 러스트보다 최대 3배, 평균 2배 정도 높았음,

Rust의 단점?

학습 비용이 많이 듦

- 소유권, 수명, 열거형, 매크로,,, 등 생소한 개념 많음
- 파이썬에 비해 매우 배우기 어려운 언어
- 높은 학습 비용으로 인해 개발 비용이 많이 들 수 있음
 - 러스트 개발자 구하기 어려움, 소수 인원이 유지보수 가능

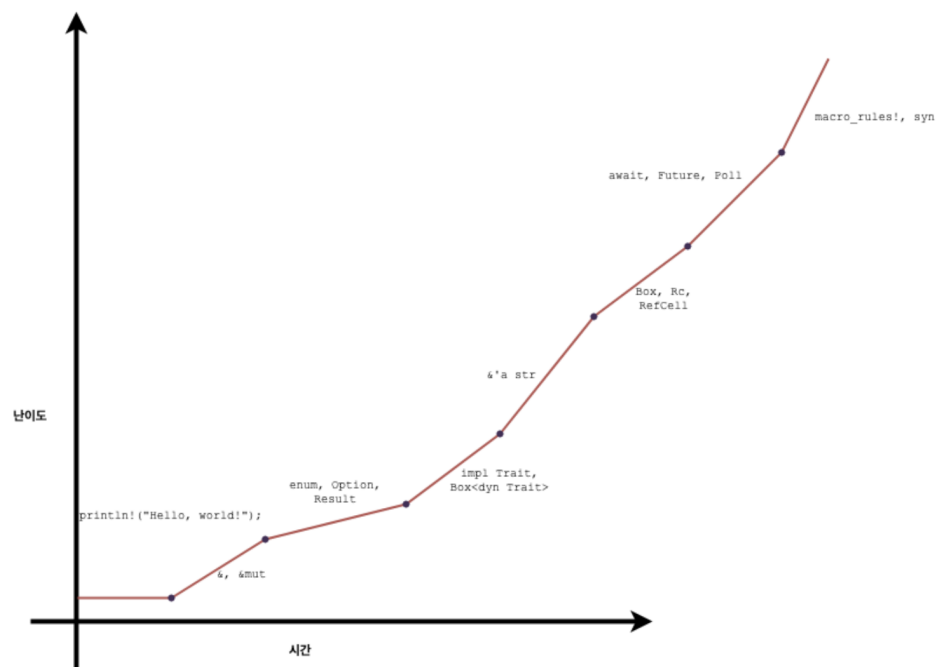


Figure 3. 글쓴이가 느낀 러스트 학습 난이도 곡선

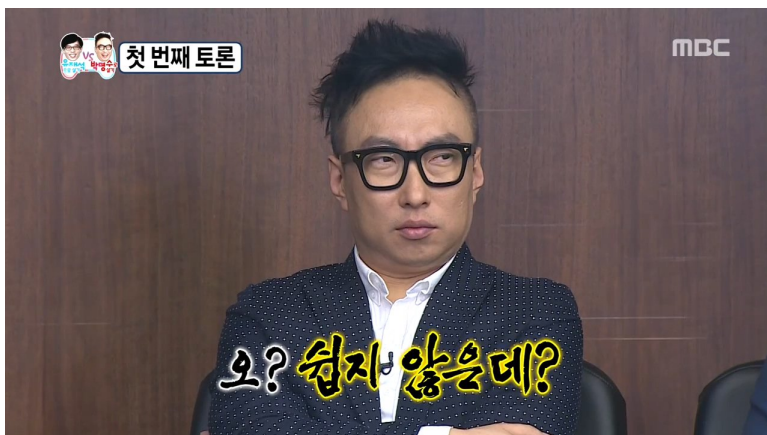
이미지 출처: kakao-파이썬과 러스트

- 러스트는 발생할 수 있는 모든 오류에 대해 어떻게 처리할지 명시하지 않으면 컴파일 실패.
 - 간단한 스크립트는 생산성 떨어질수도..?
- **Python에 비해 긴 컴파일 타임** → 러스트의 증분 컴파일에도 컴파일 시간이 오래 걸림

Rust,, 적용할 수 있을까??



- 간단한 프로그램이라면 오히려 개발하는데 더 많은 시간이 소요될 수도?
- Rust를 능숙하게 개발하기 위한 러닝커브가 높음
- Rust 개발 가능한 사람 찾기가 어려움
- CPU 연산이나 IO 등 러스트의 고성능 라이브러리는 충분히 매력적
- 컴파일러가 친절해서 이유는 몰라도 하라는대로 하면 됨



출처

- <https://www.rust-lang.org/>
- <https://google.github.io/comprehensive-rust/welcome.html>
- AWS re:Invent 2021 - Using Rust to minimize environmental impact
- <https://tech.kakao.com/2023/03/02/python-and-rust/>

author : yeoV <https://github.com/yeoV>

