세상에서 제일 쉬운 러스트 프로그래밍

1장 : 러스트 시작하기

윤인도

freedomzero91@gmail.com

- [파이썬과 비교하며 배우는 러스트 프로그래밍], 한빛미디어

가장 사랑받는 언어, 러스트

```python
print("Hello, Pythonista!")
```

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

- 러스트는 현재 스택오버플로우 개발자 설문조사에서 8년 연속 '가장 사랑받는 언어' 1위
- 국내에서도 2023 프로그래머스 설문조사에서 배우고 싶은 언어 6위

파이썬 개발자가 러스트를 배워야 하는 이유

1. 파이썬의 연산 속도를 개선한다
2. 멀티스레딩 구현이 훨씬 쉽다
3. 개발 도구가 매우 편리하다

파이썬과 러스트의 차이점

언어상의 차이

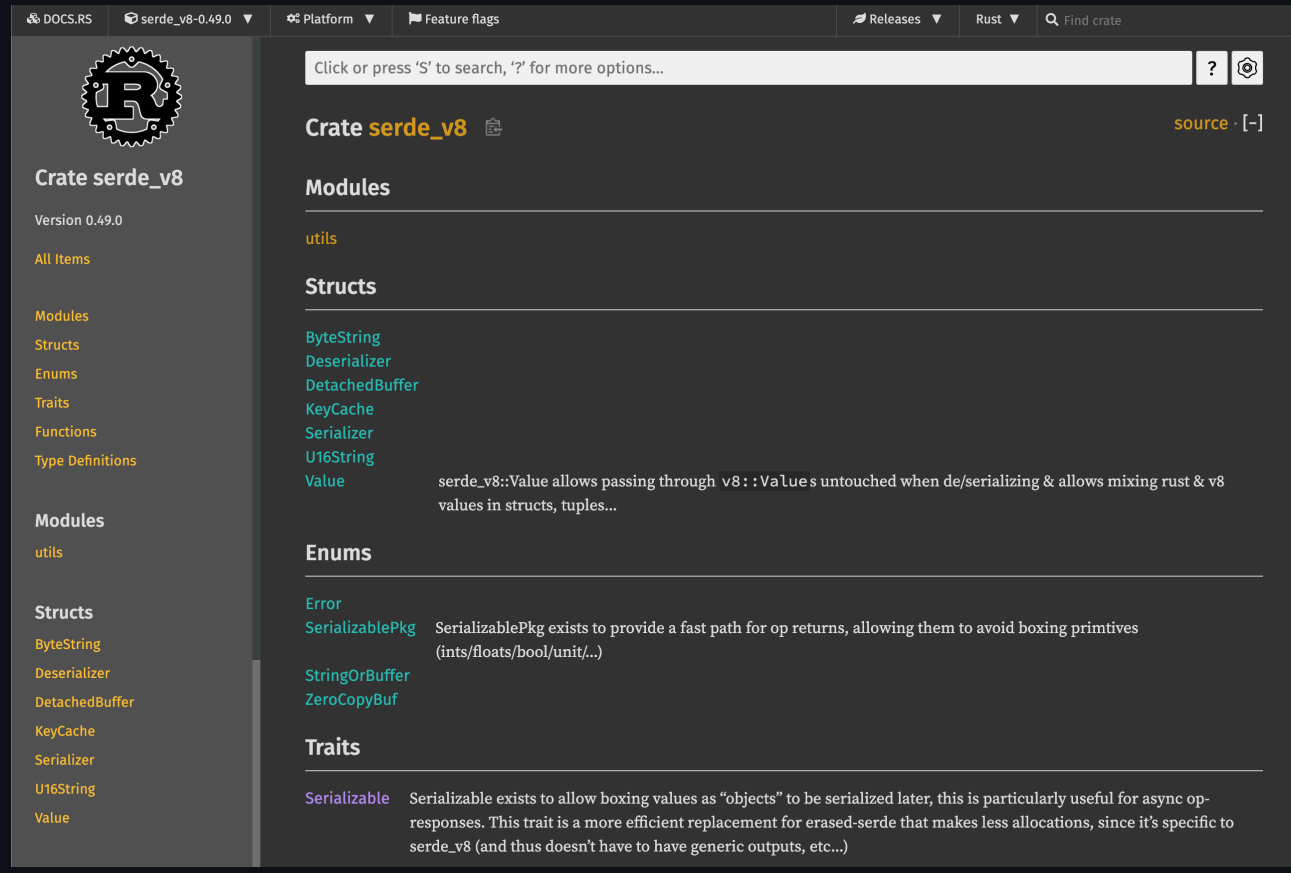
파이썬	러스트
인터프리터 언어	컴파일 언어
강타입 언어이면서 동적 타입 언어	강타입 언어이면서 정적 타입 언어
메모리 관리에 가비지 콜렉터 사용	메모리 관리에 소유권 모델 사용
대부분의 경우 객체지향 프로그래밍	함수형 프로그래밍
스타일 가이드가 유연함	명확한 스타일 가이드 존재

툴 비교

	파이썬	러스트
패키지 관리자	pip	cargo
포매터	black, yapf, autopep8	cargo fmt
린터	pylint, flake8	cargo clippy
테스트	pytest	cargo test
프로젝트 환경 관리	virtualenv, pipenv, pyenv, conda	cargo new
문서화	sphinx	cargo doc
벤치마크	cProfile, pyspy	cargo bench

```
cargo doc
```

https://docs.rs/serde_v8/0.49.0/serde_v8/



러스트의 경쟁 언어

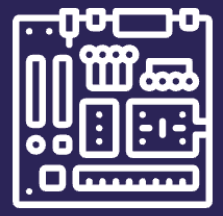
	스위프트	고	러스트
개발	Apple	Google	Mozilla
주 사용처	iOS, iPadOs, macOS 애플리케이션	네트워크 및 서버 프레임워크/ 애플리케이션	CPU 사용량이 많은 애플리케이션 혹은 시스템 소프트웨어
메모리 안전성	메모리 누수 문제가 아직 해결되지 않음	goroutine 사용 시 잠재적인 메모리 누수 발생 가능	메모리 안전성 보장

러스트로 뭘 할 수 있나요?

Command Line

WebAssembly

Networking



Embedded

러스트 사용 실제 사례들

Dropbox

코어 로직을 러스트로 재작성

Figma

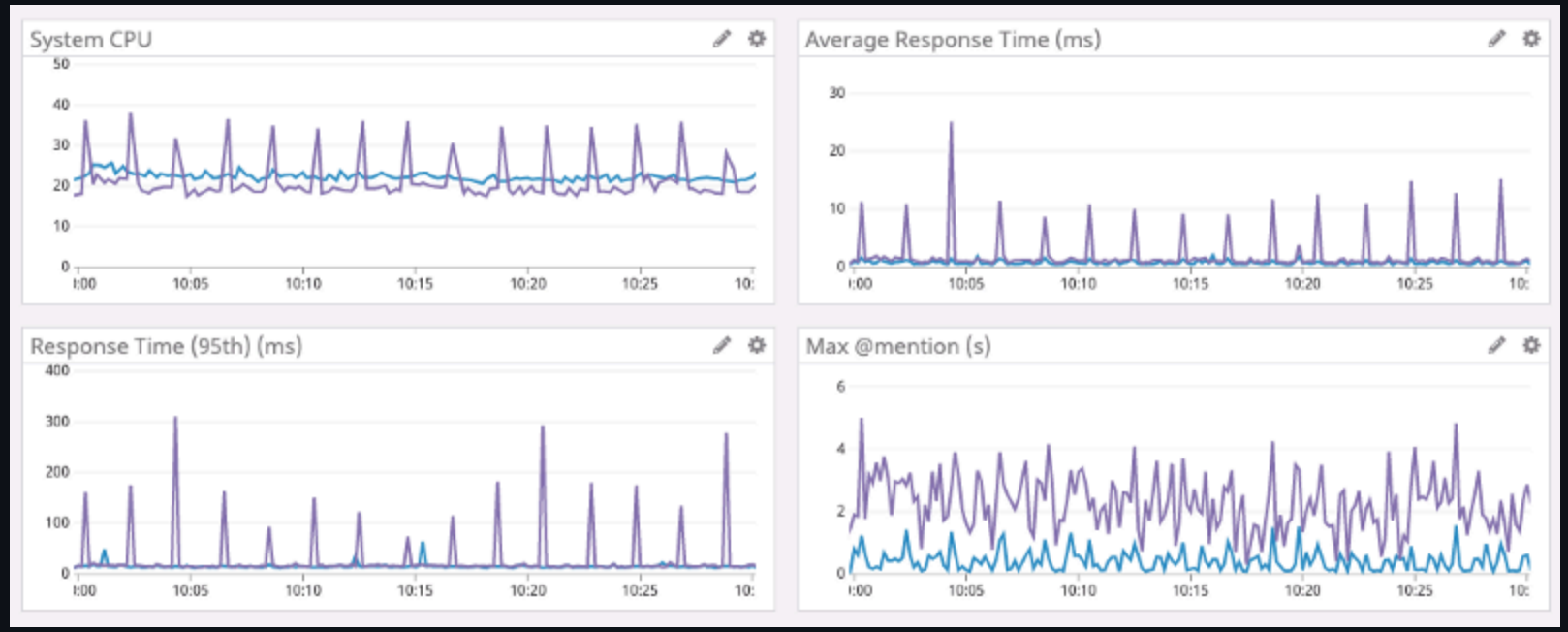
Metric	Old server		New server	Improvement
Peak average per-worker memory usage	4.2gb	⟶	1.1gb	3.8x smaller
Peak average per-machine CPU usage	24%	⟶	4%	6x smaller
Peak average file serve time	2s	⟶	0.2s	10x faster
Peak worst-case save time	82s	⟶	5s	16.4x faster

npm

레지스트리 서비스(registry service)의 병목 현상을 해결

Discord

Go에서 Rust로 이전



- 페이스북에서는 백엔드 서버를 작성하는 언어 중 하나로 러스트를 채택했습니다.
- 러스트의 후원 재단인 모질라에서 개발하는 파이어폭스 브라우저의 엔진(Servo Engine)은 러스트로 작성되었습니다.
- Next.js의 컴파일 엔진은 러스트로 재작성되었습니다.
- AWS(아마존웹서비스)의 Lambda에서 컨테이너는 FireCracker라는 러스트 툴 위에서 실행됩니다.
- Sentry 역시 파이썬의 낮은 퍼포먼스를 러스트를 도입해 해결했습니다.

러스트 개발 환경 설정하기

러스트 툴체인 설치하기

https://rustup.rs/#

macOS / Linux

맥(macOS) 또는 리눅스 사용자들은 아래 명령어를 통해 간단하게 설치가 가능합니다.

```
$ curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh
```

Windows

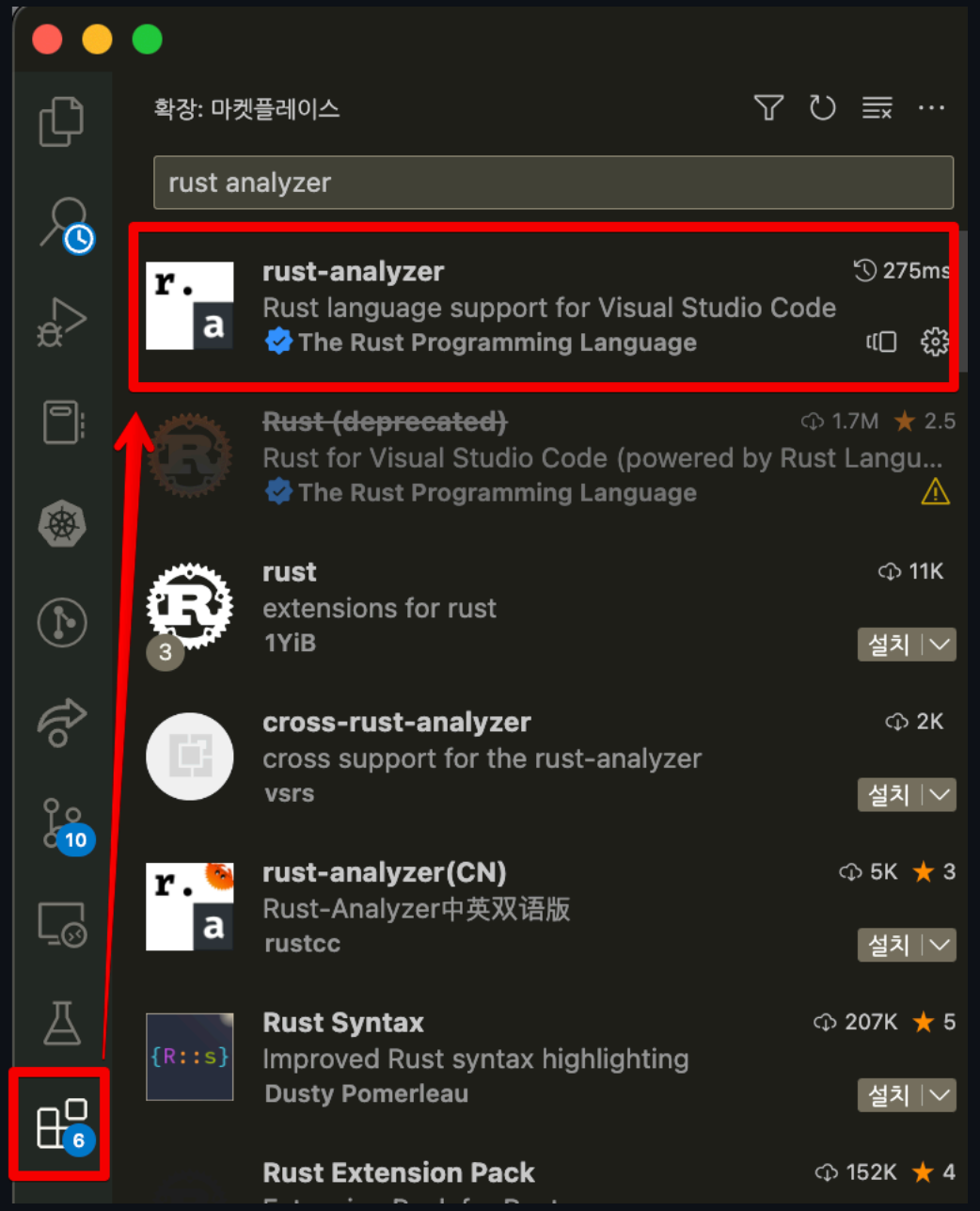
윈도우 사용자의 경우 위 홈페이지에서 34비트 또는 64비트 설치 파일을 다운로드 받습니다.

Visual Studio Code 설치 및 설정하기

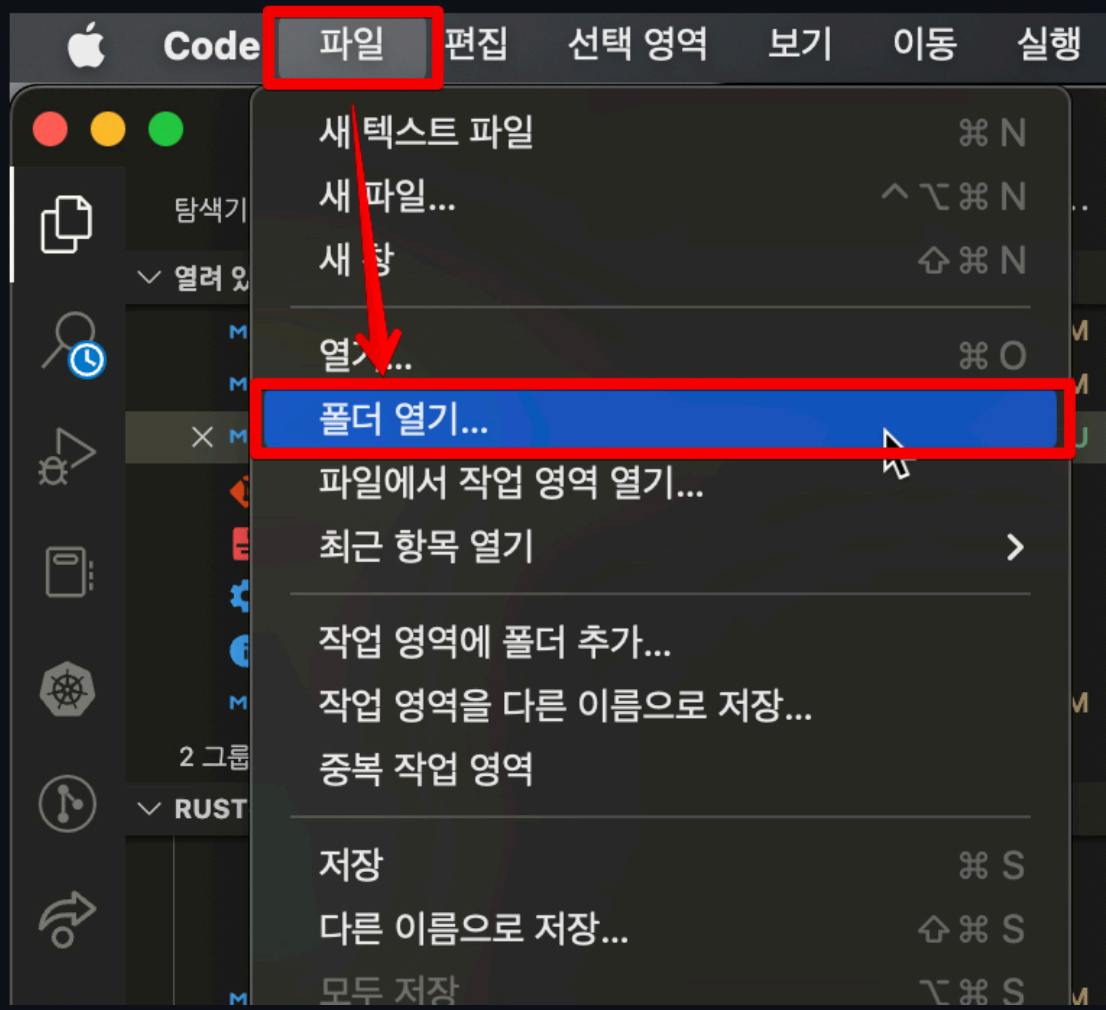
러스트에서 제공하는 컴파일, 디버깅, 언어 서버(Language server) 등의 기능을 쉽고 편리하게 사용 가능

VS Code를 설치해주세요.

rust-analyzer 설치하기



프로젝트 생성하기


Code
파일
편집
선택 영역
보기
이동
실행
탐색기
새 텍스트 파일 ⌘N
새 파일... ⌃⌥⌘N
새 창 ⇧⌘N
열기... ⌘O
폴더 열기...
파일에서 작업 영역 열기...
최근 항목 열기
작업 영역에 폴더 추가...
작업 영역을 다른 이름으로 저장...
중복 작업 영역
저장 ⌘S
다른 이름으로 저장... ⇧⌘S
모두 저장 ⌥⌘S
2 그룹
RUST

파이썬 폴더 만들기

```
.
└── python
    └── main.py
```

러스트 폴더 만들기

```
$ cargo new rust_part
```

최종 폴더 구조

```
.
├── rust_part
│   ├── Cargo.toml
│   └── src
└── python
└── main.py
```

러스트 폴더 구조

러스트의 프로젝트 폴더에는 다음과 같은 파일 구조가 만들어집니다.

```
.
├── Cargo.toml
└── src
    └── main.rs
```

`Cargo.toml` 파일은 프로젝트의 모든 설정값을 가지고 있는 파일

```
[package]
name = "rust_part"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
```

- [package] 부분에는 현재 프로젝트의 이름과 버전, 러스트 에디션 버전이 들어 있습니다. 러스트 에디션은 현재 연도보다 이전 연도가 들어 있을 수도 있는데, 이는 러스트 버전의 호환성을 위해서 버전을 에디션으로 구분하고 있기 때문입니다.
- [dependencies]는 현재 프로젝트에서 설치하는 크레이트의 이름과 버전이 들어갑니다. 나중 에 크레이트를 설치할 때 자세히 다루겠습니다.

`src` 폴더가 실제 러스트 소스코드가 들어가는 곳입니다. `main.rs`의 `main` 함수가 프로그램의 시작점이 됩니다.

러스트 코드 실행하기

`rust_part` 폴더로 이동

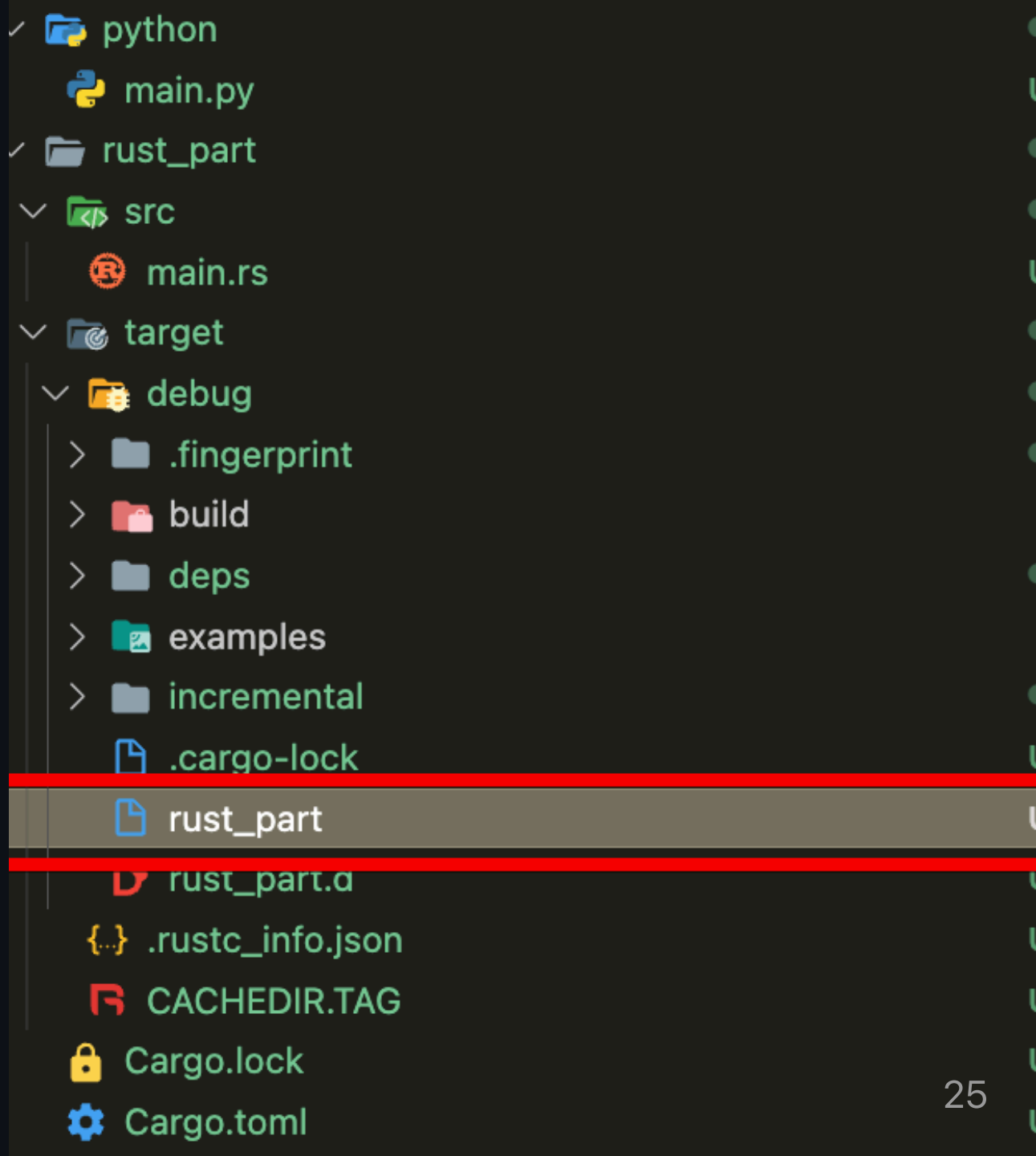
```
// main.rs
fn main() {
    println!("Hello, world!");
}
```

코드 컴파일하기

`cargo build` 명령어로 기본 코드를 컴파일합니다.

```
$ cargo build
   Compiling rust_part v0.1.0 (/Users/code/temp)
    Finished dev [unoptimized + debuginfo] target(s) in 0.31s
```

```
$ ./target/debug/rust_part
Hello, world!
```



최적화된 릴리즈 옵션으로 빌드하려면?

```
$ cargo build --release
Compiling notebook v0.1.0 (/Users/code/temp)
Finished release [optimized] target(s) in 1.34s
```

이때 바이너리 파일이 `target/release` 폴더에 생성되는 점을 주의하세요.

코드 실행하기

`cargo run` 명령어를 사용하면 코드를 컴파일하고 바이너리를 자동으로 실행합니다.

```
$ cargo run
Compiling temp v0.1.0 (/Users/code/temp)
    Finished dev [unoptimized + debuginfo] target(s) in 4.55s
     Running `target/debug/temp`
Hello, world!
```

> 릴리즈 모드는 `cargo run --release` 로 실행합니다.

rustfmt

- 윈도우 / 리눅스 : `Alt + Shift + F`
- 맥 : `Option + Shift + F`

`main.rs` 에 입력하고 포맷을 실행

```
fn main(    ){
    println! (
        "Please run 'rustfmt!'"
    );
}
```

실행 결과

```
fn main() {
    println!("Please run 'rustfmt!'");
}
```