Rust 101

Small start for big language

How to learn new language

- What do we need this?
- Where can I use it?
- Code structure (how to compile ,release , test, etc)
- Variables
- Functions
- Loops
- Structure (object-oriented ? , functional programming ?)
- What are the simplifications? How can I use it the language better ?

Why do we need this?

- Memory Safety: Rust's ownership model, along with its borrowing and lifetime rules, ensures memory safety without needing a garbage collector
- Concurrency: Rust provides powerful abstractions for dealing with concurrency safely. Its type system and ownership rules help you write concurrent code that is free from data races and other common pitfalls, making your applications more robust and responsive.
- Performance: Rust offers performance comparable to that of C and C++ because it does not have a runtime or garbage collector
- Cross-platform Development: Rust supports cross-compilation, allowing developers to compile programs for many different platforms from a single codebase.
- Tooling: Rust comes with Cargo, its package manager, which also serves as a build system. Cargo simplifies dependency management, building, testing, and documentation, making the development process more efficient.

When should I prefer rust

- Systems programming
- Backend developing
- Embedded systems
- Networking & Concurrency

Not for :

- Enterprise application (like java frameworks)
- Front end development
- Scripting

Compiler

- Rustc
- Cargo (Packaging and build)
- Rustup

Lets Create Hello world

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

Code structure

- /src for code
- /test can be used for tests or integreation tests. Test can be defined in the source code too

Data types

Integer

Length	Signed	Unsigned
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	u64
128-bit	i128	u128
arch	isize	usize

Float : f32 and f64

Boolean : bool

Character : char

String : it has Object model and pointer model str and String

Flow Controls

- If - else if – else
- Loop - forever while
- While
- For and