Rust

**Cargo** is the build system and package manager.

|  |  |
| --- | --- |
| Instruction | Description |
| Cargo new {name} | Generates the packet manager folder for Cargo to manage the rust project |
| Cargo build  --release | Compiles rust program.  Compiles with optimizations(superfast code however is slower compilation time) |
| ./target/debug/{CargoFile} | Creates executable of the cargo project on the target |
| Cargo run |  |
| Cargo check | Check correct compiling without producing an executable(speed-up process) |

Variables

|  |  |
| --- | --- |
|  |  |
| let | Create a variable |
| mut | Assigns mutable(modifiable, non-static content) attribute. Eg: let mut guess = 5; |
| instance::method() | ´´method´´ is an associated function of ´´instance´´ type (static method).  Eg: let guess = String::new(); |
| instance::method.submethod() | Calls submethod on method handle. Eg: io::stdin().read\_line() |
| {} | Format specifiers(%) of rust. In Rust they are just a placeholder.  Eg:println!("x = {} and y = {}", x, y); |

Reference

|  |  |
| --- | --- |
|  |  |
| &mut | Indicate that the variable is a reference, which allows code access of one variable through memory location. |

Handling potential failures

|  |  |
| --- | --- |
|  |  |
| .expect(“failed”); | Handles errors when returned values give err (Result types return either “ok” or “err”). this crashes the program when an error occurs; Right way to recover from an error, is to write an error handling function. |

Crate:collection of Rust source code files(Basically a library).

Version: you can add dependencies with **version** included.Ensuring **reproducible builds**: Rust will not automatically upgrade your dependencies until you explicitly upgrade them, this maintains code compatibility. To ignore manual versions, use cargo update which will update dependencies to latest versions.