Project Structure

Learn to Code with Rust / Section Review

Projects, Packages, and Crates

- A **project** is a folder with a **Cargo.toml** file.
- A **package** is a collection of one or more **crates**.
- A **crate** is a collection of Rust code that produces an executable or a library.

Types of Crates

- A **binary crate** is a program that compiles to an executable.
- Rust infers that the package has a binary crate if the compiler finds a **src/main.rs** file.
- A **library crate** exports functionality for other Rust programs (i.e., binary crates) to use.
- Rust infers that the package has a library crate If the compiler finds a **src/lib.rs** file.

Package Requirements

- A package can have both a library crate and a binary crate.
- A package must have at least one crate (i.e., at least a **main.rs** or **lib.rs** file).
- The crate name is determined from the name value in the **Cargo.toml** configuration file.

Modules

- A **module** is an organizational container that encapsulates related code.
- A **module** can store various Rust constructs -- constants, functions, structs, enums, submodules, and more.
- A **module** serves as a namespace. Different modules can hold items with the same name.

Privacy

- All module items are private by default.
- Use the **pub** keyword to allow external code to access a module item.

Privacy II

- If we use the **pub** keyword with an enum, all of its variants will be public.
- If we use the **pub** keyword with a struct, all of its fields will still be private.
- We can designate certain struct fields to be public and others private.
- If a field is private, the struct must support some associated function for creating an instance.

Options for Declaring a Module

- Declare a module with the **mod** keyword with a name.
- First option: Declare the module's contents in an inline block.
- Second option: Declare the module's content in a file with the module's name.
- Third option: Declare the module's content in a **mod.rs** file in a folder with the module's name.

Submodules

- A **submodule** is a module nested within another.
- Use the scope resolution operator (::) to navigate downwards into a submodule.
- Submodules require the creation of a folder for the parent module.
- Then, we use the same naming principles (file name matching the module or folder with **mod.rs**).

The **crate** Prefix

- An **absolute path** is the full path to a name starting from the crate root (**main.rs**).
- The **crate** keyword refers to the top-level of the crate.
- A **relative path** is the path to a name starting from the current location.

The **use** Keyword I

- The **use** keyword brings a name into the current scope, simplifying paths in the code.
- The **use** keyword can target any nested item including a module.
- For multiple values, provide curly braces and separate the names with commas.

The **use** Keyword II

- Use the **self** keyword to include a current module as a top-level name alongside an item within that module.
- The **as** keyword defines an alias for a name in the given file.
- The **glob operator** (*) imports all public items from the module.

The **super** Keyword

- The **super** keyword references the parent module from a submodule.
- The submodule can use **super** to pull in a name from a sibling.

The **pub use** Keywords

- The **pub use** keywords "re-export" an imported name from the current module.
- Library authors can simplify the complexity of their crate's module structure by exporting the most commonly used items from the crate root (**lib.rs**).

External Crates and the Standard Library

- An external crate is a dependency pulled into the current project.
- Declare dependencies in the **Cargo.toml** file.
- The **standard library** is a collection of modules built into Rust.
- The same **use** syntax applies to importing names from an external crate and the standard library.

Library Crates

- A **library crate** stores reusable code that other binary crates can use. It does not have a **main** function.
- Your project's binary crate can pull in the library crate's content.
- The library crate's name is set by the package name in **Cargo.toml**.
- Mark top-level modules as public with the **pub** keyword to allow outside code to access them.

Multiple Binary Crates

- The compiler will treat any Rust files in **src/bin** as binary crates.
- A binary crate needs a **main** function (the entrypoint).
- These binary crates can coexist alongside the primary binary crate in **src/main.rs**.
- Use **cargo run --bin name** to compile and run a specific binary.