Enums

Learn to Code with Rust / Section Review

Enums

- An **enum** is a type that represents a set of possible values.
- Each possible value is called a **variant**.
- Use the **enum** keyword followed by a name in **PascalCase** and curly braces.
- Separate variants with a comma and a line break.
- By default, custom enums do not implement the **Copy** trait. Ownership rules apply.

Tuple Variants

- A variant can store associated data.
- A **tuple variant** stores one or more pieces of data based on order/position.
- In the variant definition, provide parentheses and the sequential types of the data.
- In the enum instance, provide parentheses and the sequential values for the declared types.

Struct Variants

- A **struct variant** stores one or more pieces of data by field name.
- In the variant definition, provide curly braces and a block. Inside the block, declare the fields with their associated types.
- For the enum instance, provide curly braces and the concrete values for the fields.

The **match** Keyword

- The **match** keyword is often paired with enums.
- The **match** keyword validates that every enum variant has an associated code block.
- A **match** pattern can extract the associated data from a tuple variant or a struct variant.
- A **match** pattern can match against an exact piece of associated data.

Methods

- Like structs, enums can define methods.
- Use the **impl** keyword followed by the enum name and a block.
- Methods can take ownership of the enum or borrow it through a reference.
- Methods can define additional parameters after **self**. Pass *those* arguments in during invocation.

The **if let** Construct

- The **if let** construct executes a block of code if there is a match against a specific enum variant.
- The **if let** construct declares variables for the associated data. The block has access to the variables.
- Declare the hardcoded enum variant on the left-hand side of the equal sign. Declare the dynamic value on the right-hand side.