Vectors

Learn to Code with Rust / Section Review

Vectors

- A **vector** is an collection type for storing homogenous elements in order.
- A **vector** is similar to an array but it can grow and shrink in size.
- Each element is assigned an **index position** reflecting its place in line. The index starts counting at 0.
- Rust's vector type is **Vec**. It has one generic which represents the type of the stored elements (**Vec<String>**).

Create a Vector

- The **Vec::new** constructor function returns an empty vector.
- The **Vec::new** function requires a manual type annotation. Provide it with the variable or using the turbofish (**::<T>**) operator.
- As soon as code inserts an element into the vector, the compiler can infer its generic type.
- The **vec![]** macro creates a vector with pre-populated elements.

Vector Methods

- The **push** method appends an element to the end of the vector.
- The **insert** method adds an element at a specific index position.
- The **pop** method attempts to remove the last element from the vector. It returns an **Option** enum.
- The **remove** method removes an element by index position. It panics at runtime if the index is invalid.

Reading Vector Elements

- Use square brackets to extract a vector element by its index position.
- Rules of ownership apply. If a type does not implement the **Copy** trait, use the borrow operator (**&**) to avoid moving ownership.
- The **get** method accepts an index position and returns an **Option** enum. The **Some** variant will store a reference to the value.
- Borrow a slice with the borrow operator, the value, square brackets, and a range.

Writing Vector Elements

- Use square brackets to target an index position, then overwrite its value with an equal sign.
- Rust permits one mutable reference to a value at a time.
- Rust permits any number of immutable references to a value at a time.