# Slices

Learn to Code with Rust / Section Review

## Slices I

- A **slice** is a reference to a sequential portion of a collection type.
- Usually, slices target a chunk or fragment of the collection.
- The portion can technically be the complete collection.

## Slices II

- Use the borrow operator (**&**) with the collection and a pair of square brackets with a range.
- A string slice targets bytes. An array slice targets elements by index positions.

## Range Syntax

- A range consists of a lower and upper bound (**lower..upper**).
- The upper bound is exclusive. Rust will go up to that point but not include its value.
- Omit the lower bound to start from the beginning of the collection.
- Omit the upper bound to progress to the end of the collection.

## String Slices

- A string slice is a reference to a portion of a string.
- A string literal declared with double quotes is itself a **string slice**. Its portion is the complete text that is hardcoded into the binary executable.
- If we use the slice syntax ( [start..end] ) with an existing string slice or a **String**, we'll get another string slice. It's an independent reference.
- The length of a string slice is its count of bytes.

## Deref Coercion

- If a function defines a parameter's type as a string slice (**&str**), it can accept either a string slice (**&str**) or a String reference (**&String**).
- Rust will coerce a **String** reference (**&String**) argument to a string slice (**&str**).
- Deref coercion does not apply in reverse. Rust cannot coerce a string slice (**&str**) to a String reference (**&String**).
- The same rules apply to array slices (**&[i32]**) versus array references (**&[i32; n]**).