Strings

UTF-8 and string literals

As we saw, indexing a string yields its bytes, not its characters: a string is just a bunch of bytes. That means that when we store a character value in a string, we store its byte-at-a-time representation.

Strings in Go are `UTF-8` encoded by default

Hence s[i] prints the decimal value of the byte held by the character. But to see individual characters, you can use %c format string in Printf statement. You can also use %v format string to see byte value and %T to see data type of the value.
1.Remember that a string is basically just a byte array

Strings in Go are UTF-8 encoded by default.
ASCII characters in UTF-8 occupies 8 bits or 1 byte.
decimal value of ASCII/UTF-8

Strings are a slice of bytes, simple as that. When we use for loop with range, we get rune because each character in the string is represented by rune data type. In Go, a character is represented between single quote AKA character literal. Hence, any valid UTF-8 character within a single quote (') is a rune and it’s type is int32