GoLang Type Conversions Cheat Sheet

1. Built-in Conversions

Go requires explicit type conversions using Type(value).

```
// int → float64
var i int = 42
var f float64 = float64(i)

// float64 → int (truncates decimals)
var pi float64 = 3.14
var n int = int(pi)

// uint → int
var u uint = 100
var x int = int(u)

// string → []byte
s := "Hello"
b := []byte(s)

// []byte → string
s2 := string(b)

// string → []rune (handles Unicode)
r := []rune(s)
```

2. String ↔ Numbers (strconv package)

```
import "strconv"

// string → int
num, _ := strconv.Atoi("123")

// int → string
s := strconv.Itoa(456)

// string → float64
f, _ := strconv.ParseFloat("3.1415", 64)

// float64 → string
fs := strconv.FormatFloat(3.1415, 'f', 2, 64)
```

3. Custom Types

```
type Celsius float64
type Fahrenheit float64

func main() {
    var c Celsius = 100
    f := Fahrenheit(c*9/5 + 32)  // Explicit conversion
    println(f)
}
```

4. Important Notes

- Go does not allow implicit conversions (e.g., int + float64 requires explicit cast). - Converting float to int truncates, not rounds. - []byte is for raw bytes, []rune is for Unicode-safe characters. - Always check

errors when converting strings to numbers with strconv.