Data Types

Data Types Explained

- All data in programs consists of binary numbers (0 or 1)
- A **data type** is a way that the program can interpret the binary numbers
- Numbers, letters, and words are all different types:
  - 15
  - y
  - hello

Data Types In Go

- Go is a statically typed language
  - Data types must be provided by the programmer
- Go uses **type inference** to determine what type of data it is working with
  - Data types only need to be provided in specific circumstances
  - Can always specify the type if desired
    - Compiler error if wrong type is used

Primitive Data Types

- All primitive data types in Go are numeric
- Type indicated in code is a convention
  - It's possible that the data is invalid for the given type
    - Only applies when working with user input or manually manipulating the binary data

Signed Integer Types

Data Type	Min Value	Max Value
int8	-128	127
int16	-32768	32767
int	-2147483648	2147483647
int32	-2147483648	2147483647
int64	-9223372036854775808	9223372036854775807

Unsigned Integer Types

Data Type	Min Value	Max Value
uint8	0	255
byte	0	255
uint16	0	65535
uint	0	4294967295
uint32	0	4294967295
uint64	0	18446744073709551615
uintptr	0	<pointer size>

Other Data Types

Data Type	Description
float32	32-bit floating point
float64	64-bit floating point
complex64	32-bit floating point real & imaginary
complex128	64-bit floating point real & imaginary
bool	true or false

Type Aliases

- Possible to create type aliases
- Same in every way to another type, just a different name
- Useful for providing indication of what kind of data is being utilized

```
type UserId int
type Direction byte
type Speed float64
type Velocity Speed
```

Type Conversions

- Converting between types can be done with parentheses

```go
type UserId int
type Speed float64

UserId(5)
Speed(88.3)
```

Recap

- A **data type** is a way to specify how data should be interpreted
- Go uses **static typing**, which is checked at compile time
  - The compiler uses **type infererence** which automatically determines which types to use
- **Type aliases** can be created to give new names to existing types

```go
type UserId int
```

- Converting between types requires parentheses

```go
UserId(5)
```