Variables and Simple Data Types

Michael VanSickle

@vansimke

Introduction

- Simple data types
- Variables
- Expressions and math
- Constants
- Pointers

Simple Data Types

Strings

Numbers

Booleans

Errors

Strings

```
“this is a string”
```
interpreted string

```
`this is also a string`
```
raw string

```
“this is an escape character: \n it creates a newline”
this is an escape character:
 it creates a newline

`this is an escape character: \n it creates a newline`
this is an escape character: \n it creates a newline

`raw strings
ignore new lines`
raw strings
ignore new lines
```

Numbers

Integers	Unsigned integers	Floating point numbers	Complex numbers
99	0	6.02e23	1 + 2i
0	15	3.1415	0.833i
-937	7329	0.25	6.02e23 + 3.1415i
int	**uint**	**float32**	**complex64**
		float64	**complex128**

Booleans

true

false

error type

The error built-in interface type is the conventional interface for representing an error condition, with the nil value representing no error.

https://pkg.go.dev/builtin#error

Errors

```
type error interface {
    Error() string
}
```

Demo

Show where to find simple data types

pkg.go/dev/builtin

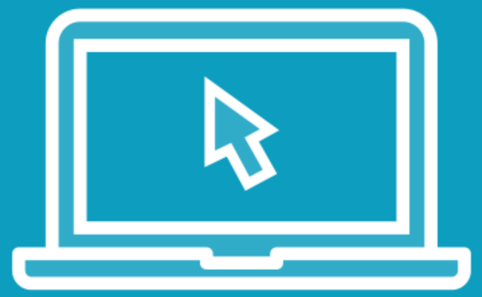
Variables

```
var myName string                // declare variable
var myName string = “Mike”       // declare and initialize


var myName = “Mike”              // initialize with inferred type
myName := “Mike”                 // short declaration syntax
```

Type Conversions

```
var i int = 32
var f float32

f = i                   // error! – Go doesn't support implicit conversions
f = float32(i)          // type conversions allow explicit conversion
```

Demo



variable declaration and type conversion

Arithmetic

```
a, b := 10, 5          // Go allows multiple variables to be initialized at once!

c := a + b             // 15 - addition
c = a - b              // 5 - subtraction
c = a * b              // 50 - multiplication
c = a / b              // 2 - division
c = a / 3              // 3 - integer division used for integers
c = a % 3              // 1 - modulus (remainder of integer division)
d := 7.0 / 2.0         // 3.5 - decimal results given for floating point numbers
```

Comparisons

```
a, b := 10, 5

c := a == b             // false - equality
c = a != b              // true  - inequality
c = a < b               // false – less than
c = a <= b              // false – less than or equal
c = a > b               // true – greater
c = a >= b              // true – greater than or equal
```

Demo

comparisons and arithmetic

Constants

```
const a = 42

const b string = “hello, world”

const c = a

const (
    d = true
    e = 3.14
)
```

- constant (implicitly typed)
- explicitly typed constant
- one constant can be assigned to another
- group of constants

```
const (
    a = “foo”
    b                   // “foo”
)
const c = 2 * 5         // 10
const d = “hello, “+“world”
const e = someFunction()
```

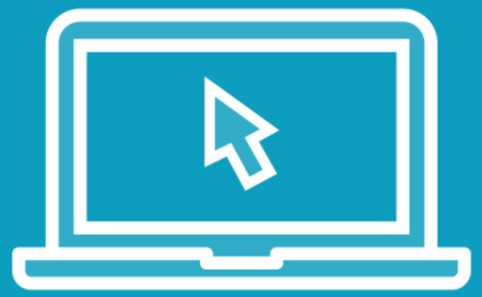
- unassigned constants receive previous value

- constant expression
- must be calculable at compile time
- this won’t work – can’t be evaluated at compile time

```
const a = iota          // 0


const (
    b = iota            // 0
    c                   // 1
    d = 3 * iota        // 6
)
const (
    e = iota
)
```

- iota is related to position in constant group
- iota starts at zero on first line
- constant expression copied, iota increments
- iota increments again
- iota resets to zero with each group

Demo



constants

constant expressions

iota

Pointers and Values

```
a := 42

b := a

a = 27
```

Pointers and Values

```
a  := 42

b  := &a

*b                  // 42

a = 27
*b                  // 27
```

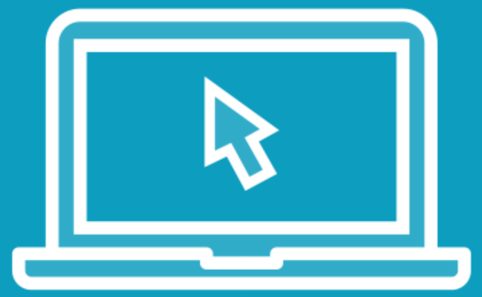
```
a := “foo”

b := &a

*b = “bar”

c = new(int)
```

- Create a string variable
- address operator returns the address of a variable
- dereference a pointer with asterisk
- built-in “new” function creates pointer to anonymous variable

Pointers are primarily used to share memory.

Use copies whenever possible.

Demo



Pointers

Create

Dereference

Summary

- Simple data types
- Variables
- Expressions and math
- Constants
- Pointers