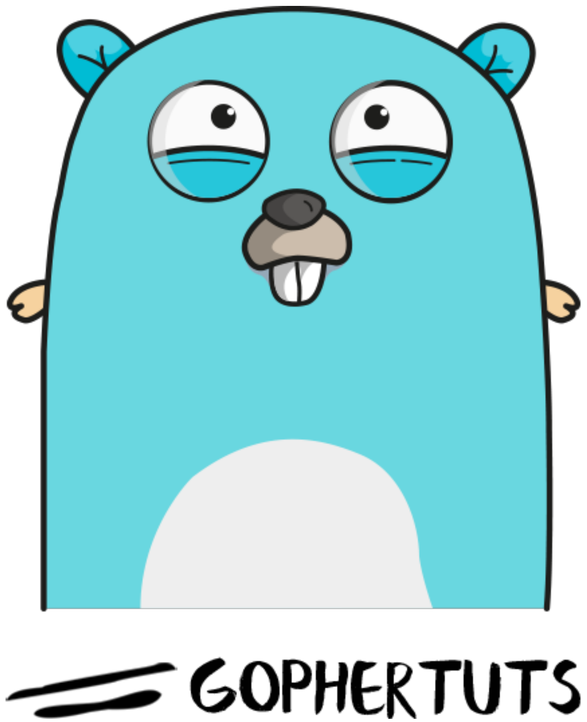
# VARIABLES & TYPES



# GO BASICS

Program anatomy

Go commands

Go workspace & project layout

Built-in types

Custom types

Basic types

Composite types

Variable declarations

Type conversions & inference

Zero values

# PROGRAM ANATOMY

main.go

```go
package main

import "fmt"

func main() {
    fmt.Println(a...:"Hello World!")
}

```

# GO WORKSPACE & PROJECT LAYOUT

```
├── $GOPATH/
│   ├── src/
│   │   ├── github.com/
│   │   │   ├── organization/
│   │   │   │   └── project/
│   │   │   ├── user1/
│   │   │   └── └── project/
│   ├── bin/
│   │   ├── goimports
│   │   └── my-bin
│   ├── pkg/
│   │   ├── darwin_amd64/
│   │   │   ├── github.com/
│   │   │   │   ├── uber-go/
│   │   │   │   │   └── atomic.a
│   │   │   │   ├── user1/
│   │   └── └── └── └── package/
│   └──
└──
```

# GO COMMANDS

```
go run

go build

go install
```

# TYPE SYSTEM

No ***type*** *hierarchy*

No ***classes*** or ***objects***

No type ***extension***

Just ***types*** & ***variables*** of a certain type

Just type ***aliasing/composition***

# BUILTIN TYPES

| | | | |
|---|---|---|---|
| bool | int32 | rune | (interface) |
| uint8 | int64 | uintptr | |
| uint16 | int | byte | |
| uint32 | float32 | (array) | |
| uint64 | float64 | (slice) | |
| uint | complex64 | (map) | |
| int8 | complex128 | (struct) | |
| int16 | string | (channel) | |

# CUSTOM TYPES

```
myType type builtInType
```

# BASIC TYPES

```
bool
uint8
uint16
uint32
uint64
uint
int8
int16
int32
int64
int
float32
float64
complex64
complex128
string
rune
uintptr
byte
```

atom

# BASIC TYPES LAYOUT

Type of ***values***

Values ***range***

Available ***operations***

***Relationships*** with ***other types***

# COMPOSITE TYPES

(array)

(slice)

(map)

(struct)

(channel)

(interface)

# VARIABLE DECLARATIONS

```
i1, i2, … := v1, v2, …

var i = value

var i type = value

var i type

const i = value
```

# VARIABLE DECLARATIONS

declaration

initialization

assignment

declaration & assignment

# ZERO VALUES

```
var x int
var x int = 0
var x = 0
var x int; x = 0
x := 0
```

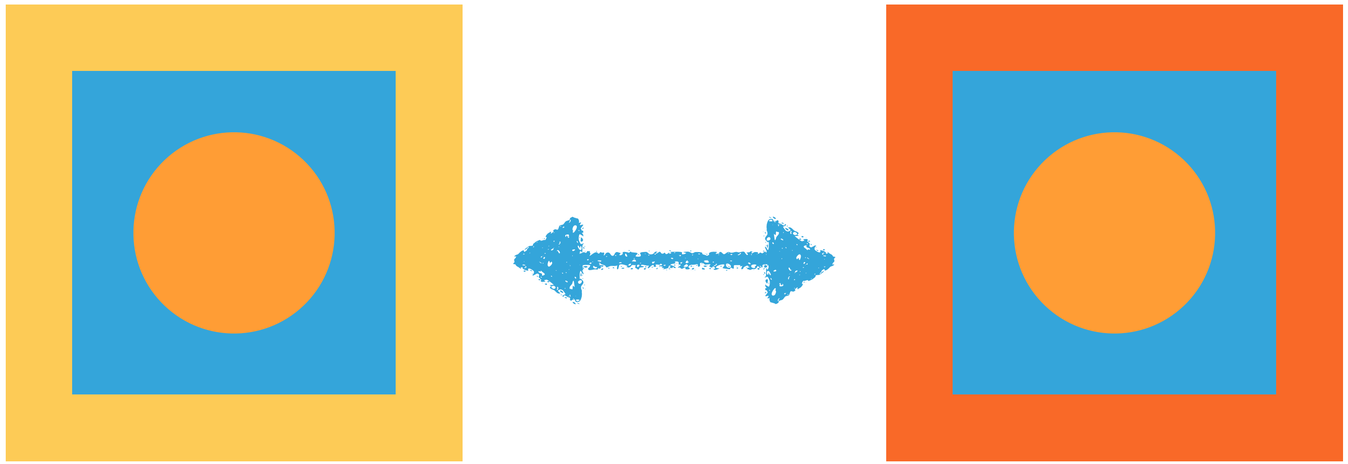
# ZERO VALUES

```
var x int
var x int = 0
var x = 0
var x int; x = 0
x := 0
```

# ZERO VALUE EXAMPLES

```
int 0
string “”
float32/64 0.0
struct{int, string} struct{0, “”}
[]string{} []string{}
map[string]bool nil
error nil
```

# TYPE CONVERSIONS

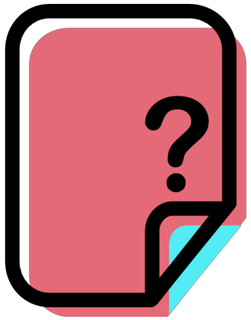


```
typeXVar = TypeX(typeYVar)
```

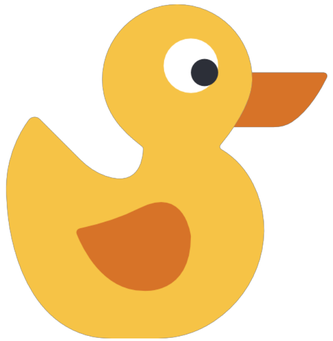
# TYPE INFERENCE

```
varInt := 20
myType := MyStructType{}
var p = People{}
const x = 10
```

# TYPE ASSERTION



Let's assume you're



```
known, ok = unknown.(KnownType)
```

# CHALLENGE TIME