Go 1.2 CHEATSHEET STAR WARS

installation

```
$ mkdir $HOME/go
$ export GOPATH=$HOME/go
$ export PATH=$PATH:$GOPATH/bin
```

configuration

```
$ mkdir $HOME/go
$ export GOPATH=$HOME/go
$ export PATH=$PATH:$GOPATH/bin
```

hello world

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("May the Force be with you.")
}
```

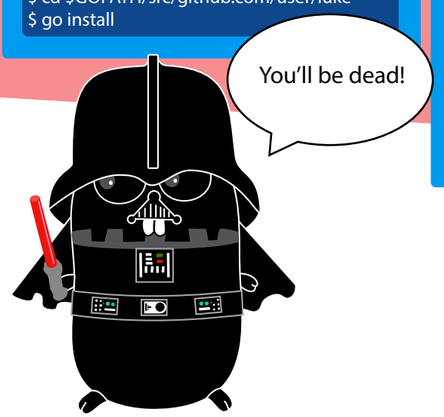
package

```
$ mkdir $GOPATH/src/github.com/user/luke
```

```
package main

import "fmt"

func main() {
    fmt.Printf("Use the Force, Luke.\n")
}
```

```
$ go install github.com/user/luke
```

```
$ cd $GOPATH/src/github.com/user/luke
$ go install
```



variable type

```
Abyssin     uint8  // integers (0 to 255)
Advoszec  uint16  // integers (0 to 65,535)
Anzat       uint32  // integers (0 to 4,294,967,295)
Aqualish   uint64   // integers (0 to 18,446,744,073,709,551,615)

Arcona       int8  // integers (-128 to 127)
Bith           int16  // integers (-32,768 to 32,767)
Brizzit        int32  // integers (-2,147,483,648 to 2147483647)
Chadra-Fan int64   // integers (-9,223,372,036,854,775,808 to
9,223,372,036,854,775,807)

Defel float32  // 32-bit floating-point numbers
Devaronian float64  // 64-bit floating-point numbers

Duros complex64  // float32 real and imaginary parts
Givin  complex128  // the set of all complex numbers with float64

Gotal byte  // alias for uint8

H'nemthe rune   // alias for int32

Human string  // text

Hutt      *[]int  // array elements of single type
Ithorian *[][]int  // array elements of single type
```

variable declaration

```
// single declaration
var Stormtrooper int

// multiple declaration
var firstname, lastname string
var (
    name string
    age unint
    rifleAmmunition uint
)

//constant
const Dance = "Tap-dance"

// affect a value while declaring
var wife = "Padmé Naberrie Amidala"

// simple declaration
darkVador := "Anakin Skywalker"
```

struct

```
package main

import (
    "fmt"
)

type Jedi struct {
    Force int
    Name string
    Fight() func
}

func Fight() {
    // todo
}

func main() {
    j := new(Jedi)
    j.Force = 4561
    j.Name = "Luke"
    j.Fight()
    fmt.Printf(%f, f)
}
```

method

```
// Internal function (private)
func love(j Jedi) string {
    return "I love " + j.Name
}
// external function (public)
func Fight(j Jedi) string {
    return "You will die  " + j.Name
}
// interface
func (* fr Friend) Make() int {
    return fr.num
}
```

interface

```
package main

import (
    "fmt"
)

type Food interface {
    Cook() string
}

type Burger struct {
    Name      string
    Baking    string
    ingredients []string
}

func (b *Burger) Cook() string {
    return "your " + b.Name + " is " + b.Baking
}

func main() {
    b := new(Burger)
    b.Name = "Bantha Burgers"
    b.Baking = "medium rare"
    fmt.Printf("Congratulation %s !\n", b.Cook())
}
```

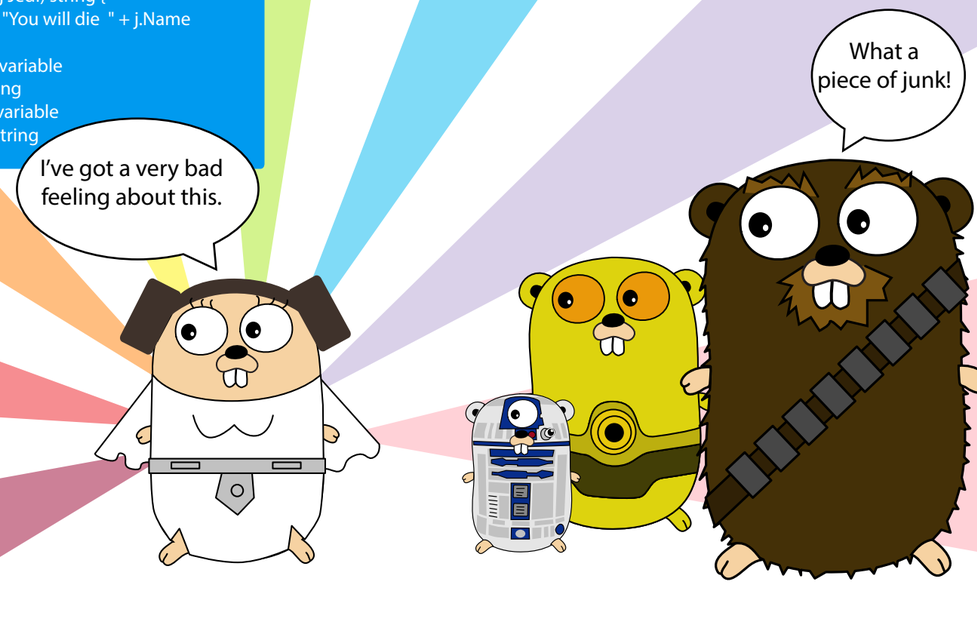
if, for, else...

```
if dark == light {
    return false
}

if dark != light {
    return true
} else {
    return false
}

force := 0
for i := 0; i < 10; i++ {
   force += i
}

sum := 0
for _, value := range myArray {
   sum += value
}
```

in/external

The first letter uppercase make the function visible outside the package

```
// Internal function (private)
func love(j Jedi) string {
    return "I love " + j.Name
}
// external function (public)
func Fight(j Jedi) string {
    return "You will die  " + j.Name
}
// external variable
var Sun string
// internal variable
var moon string
```



Go 1.2 CHEATSHEET STAR WARS

installation

```
$ mkdir $HOME/go
$ export GOPATH=$HOME/go
$ export PATH=$PATH:$GOPATH/bin
```

configuration

```
$ mkdir $HOME/go
$ export GOPATH=$HOME/go
$ export PATH=$PATH:$GOPATH/bin
```

hello world

```
package main

import (
	"fmt"
)

func main() {
	fmt.Println("May the Force be with you.")
}
```

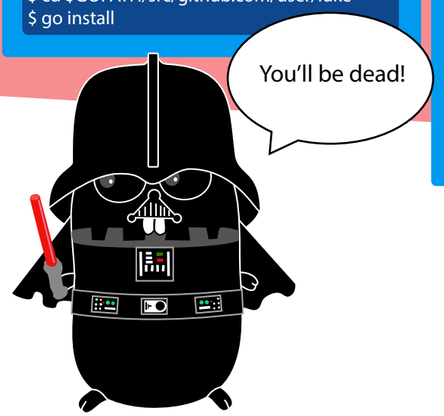
package

```
$ mkdir $GOPATH/src/github.com/user/luke
```

```
package main

import "fmt"

func main() {
	fmt.Printf("Use the Force, Luke.\n")
}
```

```
$ go install github.com/user/luke
```

```
$ cd $GOPATH/src/github.com/user/luke
$ go install
```



variable type

```
Abyssin     uint8  // integers (0 to 255)
Advoszec  uint16  // integers (0 to 65,535)
Anzat       uint32  // integers (0 to 4,294,967,295)
Aqualish   uint64   // integers (0 to 18,446,744,073,709,551,615)

Arcona       int8  // integers (-128 to 127)
Bith           int16  // integers (-32,768 to 32,767)
Brizzit       int32  // integers (-2,147,483,648 to 2147483647)
Chadra-Fan int64   // integers (-9,223,372,036,854,775,808 to
9,223,372,036,854,775,807)

Defel float32  // 32-bit floating-point numbers
Devaronian float64  // 64-bit floating-point numbers

Duros complex64  // float32 real and imaginary parts
Givin  complex128  // the set of all complex numbers with float64

Gotal byte  // alias for uint8

H'nemthe rune   // alias for int32

Human string  // text

Hutt      *[]int  // array elements of single type
Ithorian *[][]int  // array elements of single type
```

variable declaration

```
// single declaration
var Stormtrooper int

// multiple declaration
var firstname, lastname string
var (
	name string
	age unint
	rifleAmmunition uint
)

//constant
const Dance = "Tap-dance"

// affect a value while declaring
var wife = "Padmé Naberrie Amidala"

// simple declaration
darkVador := "Anakin Skywalker"
```

struct

```
package main

import (
	"fmt"
)

type Jedi struct {
	Force int
	Name string
	Fight() func
}

func Fight() {
	// todo
}

func main() {
	j := new(Jedi)
	j.Force = 4561
	j.Name = "Luke"
	j.Fight()
	fmt.Printf(%f, f)
}
```

method

```
// Internal function (private)
func love(j Jedi) string {
	return "I love " + j.Name
}
// external function (public)
func Fight(j Jedi) string {
	return "You will die  " + j.Name
}
// interface
func (* fr Friend) Make() int {
	return fr.num
}
```

interface

```
package main

import (
	"fmt"
)

type Food interface {
	Cook() string
}

type Burger struct {
	Name	string
	Baking	string
	ingredients []string
}

func (b *Burger) Cook() string {
	return "your " + b.Name + " is " + b.Baking
}

func main() {
	b := new(Burger)
	b.Name = "Bantha Burgers"
	b.Baking = "medium rare"
	fmt.Printf("Congratulation %s !\n", b.Cook())
}
```

if, for, else...

```
if dark == light {
	return false
}

if dark != light {
	return true
} else {
	return false
}

force := 0
for i := 0; i < 10; i++ {
   force += i
}

sum := 0
for _, value := range myArray {
   sum += value
}
```

in/external

The first letter uppercase make the function visible outside the package

```
// Internal function (private)
func love(j Jedi) string {
	return "I love " + j.Name
}
// external function (public)
func Fight(j Jedi) string {
	return "You will die  " + j.Name
}
// external variable
var Sun string
// internal variable
var moon string
```