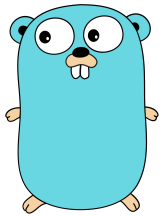
Go from scratch: Beginners-Friendly Guide

by Denis Shchuka

The Anatomy of a Go program



Basic building blocks of Go program

Variables

```
var i int = 10
```

name — data type — value

Statements

```
x := "Hello"
i++
```

Statements are executed

Expressions

```
3.1415 * r * r
x + getArea(y)
```

Expressions are executed

Control Structures

Change execution flow

```
for k < 100 {
  fmt.Println(k*2)
  k = k + 2
}
```

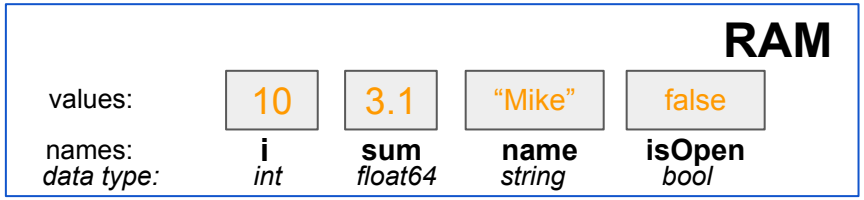
Functions and Methods

Block of code that performs a specific task and can return a value

```
func getArea(x, y int) int {
  return x * y
}
```

Programming is all about of combination of these blocks to produce result in form of working software

Variables and Fundamental Data Types




```
var i int = 10
var sum float64 = 3.1
var name string = "Mike"
var isOpen bool = false
```

Go basic data types

```
Integer     - int, int8, int16, int32,
int64,uint, uint8, uint16, uint32,
uint64, uintptr                    //-100,0,355,...
Boolean     - bool                 //true, false
String      - string               //"Hello, Go"
Rune        - rune                 //"⌘","語"
Byte        - byte                 //97
Float       - float32, float64     //3.1415, -0.00121
Comlex      - complex64, complex128 //(5+2i)
```

Data type specifies what kind of value can be stored in variable

Each variable has a name, a value and a data type.

Variables and Fundamental Data Types

You can declare variable in a various ways:

1. Explicit declaration with initialization

```
var i int = 10
```

2. Explicit declaration with deferred value assignment

```
var i int //i==0 - default value
...
i = 10
```

3. Implicit declaration

```
i := 10  //i has int type
```

4. Block declaration

```
var (
      i int = 10
      name string = “Michael”
)
```

Go doesn’t allow you to mix different date types in an expression.
You cannot add, subtract, multiply or compare values that have different data types.

```
x:= 10
y:= 10.0
fmt.Println(x == y) //Compilation error!
```

Working with strings

- Strings in **Go** are UTF-8 encoded
- Strings in **Go** are immutable

Useful packages

strings

Basic functions for working with strings

- Contains(“Hello”, “el”) *//true*
- Count(“Hello”, “ll”) *//2*
- ToLower(“HELLO”) *//“hello”*
- ToUpper(“Hello”) *//“HELLO”*
- Index(“Hello”, “o”) *//4*

fmt

Formatted Input and Output

- Println(“Hello Go”)
- Sprintf(“%d apples”, 5) *//“5 apples”*
- Printf(“%f”, 1.65) *//“1.65”*

utf8

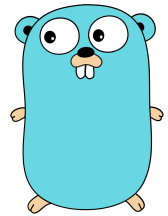
utf8 specific functions

- RuneCountInString(“Go”) *//2*
- ValidString(“("Hello, 世界")”) *//true*

Go from scratch: Beginners-Friendly Guide

by Denis Shchuka

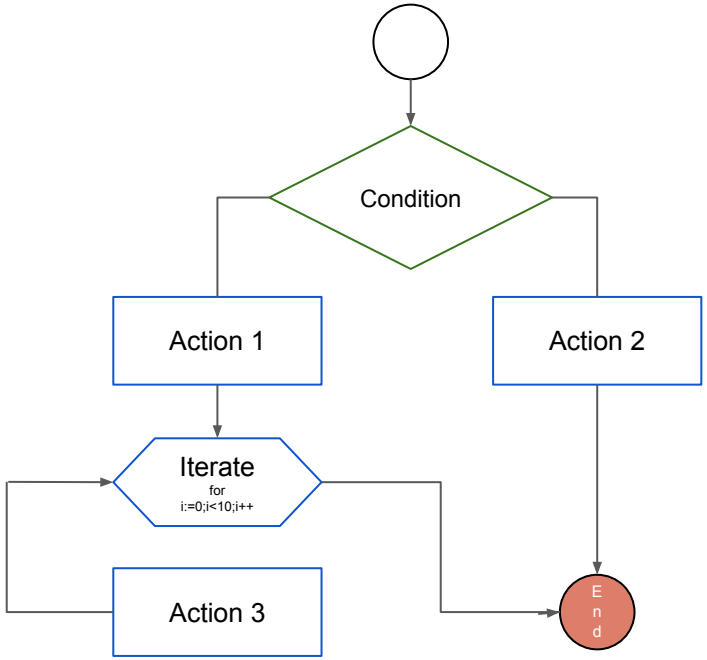
The Anatomy of a Go program.
Control structures.



Control structures

Control structures allow you to change execution flow of your program

IF ...
ELSE...



SWITCH ...

FOR ...

Control structures

IF STATEMENT

Example 1. IF

```
const speedOfLight = 299792458
if turtleSpeed < speedOfLight {
     turtleSpeed += speedOfLight
}
```

You can combine conditions as you wish

```
if condition1 {
      //do something..
} else if condition2 {
      //do something else..
} else {
      //first two conditions weren't catched
}
```

Logical operators

```
&& - logical AND
|| - logical OR
!  - logical NOT
```

Example 2. IF-ELSE

```
if withdrawAmont > 0  && balance > withdrawAmount
{
     withdraw(withdrawAmount)
}
else {
     fmt.Println("Sorry… Your  balance is
insufficient.")
}
```

Control structures

SWITCH STATEMENT

Example 1.

```
day := "Friday"
switch day {
case "Monday":
     fmt.Println("Go to work!")
case "Friday":
     fmt.Println("TGIF!")
default:
     fmt.Println("Today is a weekday")
}
```

Example 2. Fallthrough

```
switch 1 {
case 1:
     fmt.Println("We got 1")
     fallthrough
case 2:
     fmt.Println("We got 2!")
default:
     fmt.Println("Today is a weekday")
}

//result:
//We got 1
//We got 2
```

Control structures

FOR STATEMENT

! The FOR statement allows us to repeat a block of code as many times as its condition is true
! Unlike many other languages **Go** has the only one looping construct - the FOR statement

Example 1.

```
i := 1
for {
	fmt.Println(i)
	i++
	if i == 11 {
		break
	}
}
```

Example 2.

```
i := 0
for i<10 {
	i++
	fmt.Println(i)
}
```

Example 3.

```
for i:=0;i<10;i++ {
	fmt.Println(i+1)
}
```

Demo and First Exercise

EXERCISE

Try to modify Dice Roller program:

- Add second dice
- Limit user tries
- Change number of dice faces
- ...