A Very Brief Intro to Golang

joshua@hauptj.com

Notable projects that use it

- Kubernetes
- Kubernetes – Helm
- Docker
- Packer
- Terraform
- Tekton

Why it exists

- Google needed a simple programming language that scaled and performed well for distributed systems at a large scale.

Designed with simplicity in mind

- All loops are declared with for
- Composition over inheritance
- Python like array operations
- Encapsulation declared by case
- Built in garbage collection
- Package manager, linter, auto formatter, and unit testing framework all packaged together

Composition over Inheritance

- Inheritance is not supported

```
type Building struct(){
    Door
    Name string
}
Type Door struct(){
    Location string
}
```

Loops

- For, while and do while are all declared with for

For Loop

```
for i := 0; i < 10; i++ {
    sum += i
}
```

Range

- Iterates over a slice or map
    - Like for each in Python

```
for i range slice {
    fmt.printf(i)
}
```

While loop

```
for sum < 1000 {
    sum += sum
}
```

Variables

- A list of variables can be declared with var
  - Like Javascript
- Can be set automatically, life python and JS

```
Number = balance(input int) int {
    Return input
}
```

Slices

- A dynamically sized flexible view of the elements in an Array
- numbers := [6]int{1, 2, 3, 4, 5, 6}
- A[low / left : high / right]
- **Like python**
- var s []int = numbers[1:4]
- Output: 2, 3, 4

Short Variable Declarations

```
number := balance(input int) int {
    Return input
}
```

Loops

- For, while and do while are all declared with for

For Loop

```
for i := 0; i < 10; i++ {
    sum += i
}
```

While Loop

```
for sum < 1000 {
    sum += sum
}
```

Range

- Iterates over a slice or map
- Like “for each” in Python

```
for i range slice {
   fmt.printf(i)
}
```

Slices

- A dynamically sized flexible view of the elements in an Array
- numbers := [6]int{1, 2, 3, 4, 5, 6}
- A[low / left : high / right]
- Like python
- var s []int = numbers[1:4]
- Output: 2, 3, 4

Composition over Inheritance

- Inheritance is not supported
- Like in React JS

Encapsulation

- Encapsulation is case sensitive
- Capitalized - public
- non capitialized - private

Encapsulation Example

```
type Building struct() {
    Door
    Name string
}

type Door struct() {
    Location string
}
```

Pointers

- Holds the memory address of a value
  - Like C
- However, Go does not support pointer arithmetic
  - var *ptr int
  - ~~prt++~~

Defer

- Defers the execution of a function until the surrounding function returns.

```
func main() {
      Defer fmt.Println(“world”)
      Fmt.Println(“hello”)
}
```

Unused Imports?

- Compiler will refuse to compile.
- Encourages good practices

Everything Bundled Together

- Package Manager
  - go get package name
- Testing framework
  - go test
- Code formatter
  - go fmt

Concurrency Example

```
func main()
	messages := make(chan string)

	go func() { messages <- “Hello World” }()

	msg := <- messages
	fmt.Println(msg)
}
```

Concurrency

- Communicating Sequential Processes (CSP)
- Goroutines: “lightweight threats” that allow functions to run concurrently
- Channels: Used to pass messages between goroutines

Recommended Resources Cont. - Testing

- A Tour of Go
- Go by Example
- Common data structures in Go
- Regular Expressions and In-Place Slice Manipulation in Go
- Traversing Directories Recursively and Sorting Objects by Attribute Value in Go

Recommended Resources Cont.

- https://medium.com/rungo/unit-testing-made-easy-in-go-25077669318
- https://medium.com/rate-engineering/go-test-your-code-an-introduction-to-effective-testing-in-go-6e4f66f2c259

Questions?