Golang для начинающих

Клестов Дима,
Golang-разработчик ООО «Инносети»

https://t.me/MaJloe_3Jlo
https://github.com/MaJloe3Jlo

Результаты занятия 6

Все молодцы!

План занятия

На прошлом занятии:
1) Пакеты в Go
2) Документирование кода
3) Обработка ошибок
4) Домашняя работа

План занятия на сегодня:
1) Тестирование
2) Стандартная библиотека
3) Дальнейшие шаги
4) Домашняя работа

Тестирование

- Код без тестов — это плохой код!

- package testing

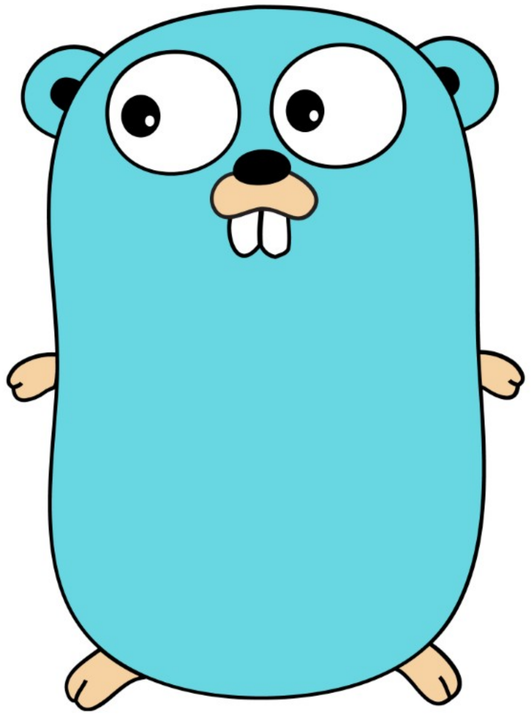
  Пример теста для функции Plus из 6го урока:

```
func TestPlus(t *testing.T) {
    value := Plus(1, 2)
    if value != 3 {
        t.Errorf("Expected 3, got %d", value)
    }
}
```

- Обычно код теста размещается в отдельном файле, рядом с тестируемым кодом и имеет постфикс _test.go
  Пример: plus_test.go

- Запуск тестов: go test (или got если настроены алиасы)

- Запуск тестов с отображением покрытия кода ими:
  go test -cover



Бенчмарк тестирование

- Бенчмарк — тестирование производительности

  Пример бенчмарк-теста для функции Plus из 6го урока:

```
func BenchmarkPlus(b *testing.B) {
    for n := 0; n <= 10; n++ {
        Plus(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    }
}
```

- Запуск:
  go test -bench <имя тестируемой функции>
  go test -bench Plus

- Go test -bench Plus -benchmem - позволит тестировать потребление памяти и количество аллокаций памяти

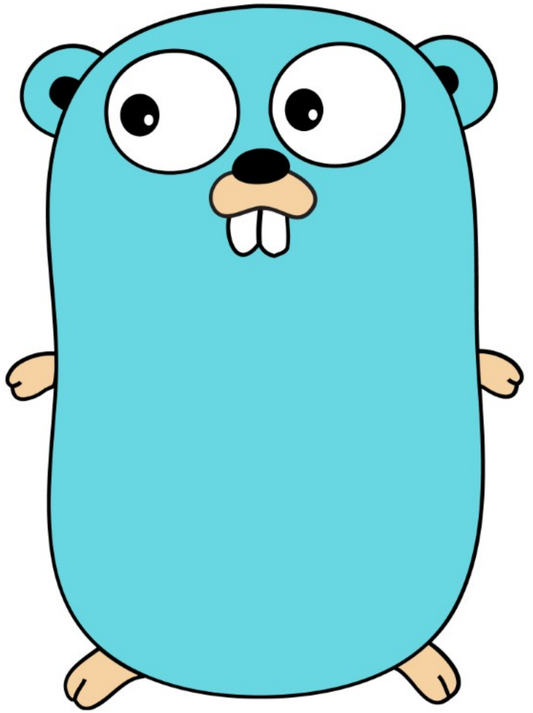
Примеры на практике

Стандартная библиотека

- Вместо того, чтобы каждый раз писать всё с нуля, реальный мир программирования требует от нас умения взаимодействовать с уже сущ библиотеками.

- Строки (package strings)

```
strings.Contains("test", "es") // true
strings.Count("test", "t") // 2
strings.HasPrefix("test", "te")  // true
strings.HasSuffix("test", "st") // true
strings.Index("test", "e") // 1
strings.Join([]string{"a","b"}, "-") // "a-b"
strings.Repeat("a", 5) // "aaaaa"
strings.Replace("aaaa", "a", "b", 2)  // "bb
strings.Split("a-b-c-d-e", "-") //
[]string{"a","b","c","d","e"}
strings.ToLower("TEST") // "test"
strings.ToUpper("test") // "TEST"
```



Стандартная библиотека

- package io

- Пакет io состоит из нескольких функций, но в основном, это интерфейсы.

  Два основных интерфейса — это Reader и Writer:
- Reader (чтение методом Read).
- Writer (запись методом Write).

- Многие функции принимают в качестве аргумента Reader или Writer.
  io Содержит функцию Copy, которая копирует данные из Reader во Writer:

  func Copy(dst Writer, src Reader) (written int64, err error)

- Buffer из пакета bytes:

  var buf bytes.Buffer
  buf.Write([]byte("test"))

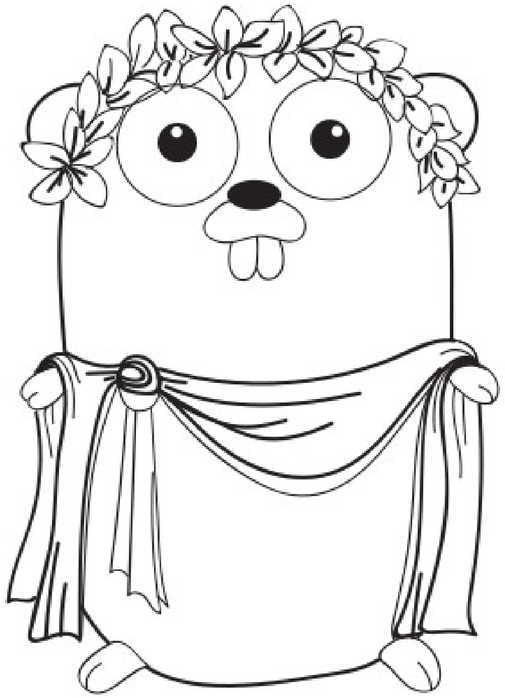
- Buffer не требует инициализации и поддерживает интерфейсы Reader и Writer.
- buf.Bytes() - конвертация в byte.

Стандартная библиотека

- Файлы и папки

- Для открытия файла Go использует функцию Open из пакета os. Вот пример того, как прочитать файл и вывести его содержимое в консоль:

```
 file, err := os.Open("test.txt")
if err != nil {
    // здесь перехватывается ошибка
    return
}
defer file.Close() // закрытие файла после выполнения всех операций над ним
// получить размер файла
stat, err := file.Stat()
if err != nil {
    return
}
// чтение файла
bs := make([]byte, stat.Size())
_, err = file.Read(bs)
if err != nil {
    return
}
str := string(bs)
fmt.Println(str)
}
```



Стандартная библиотека

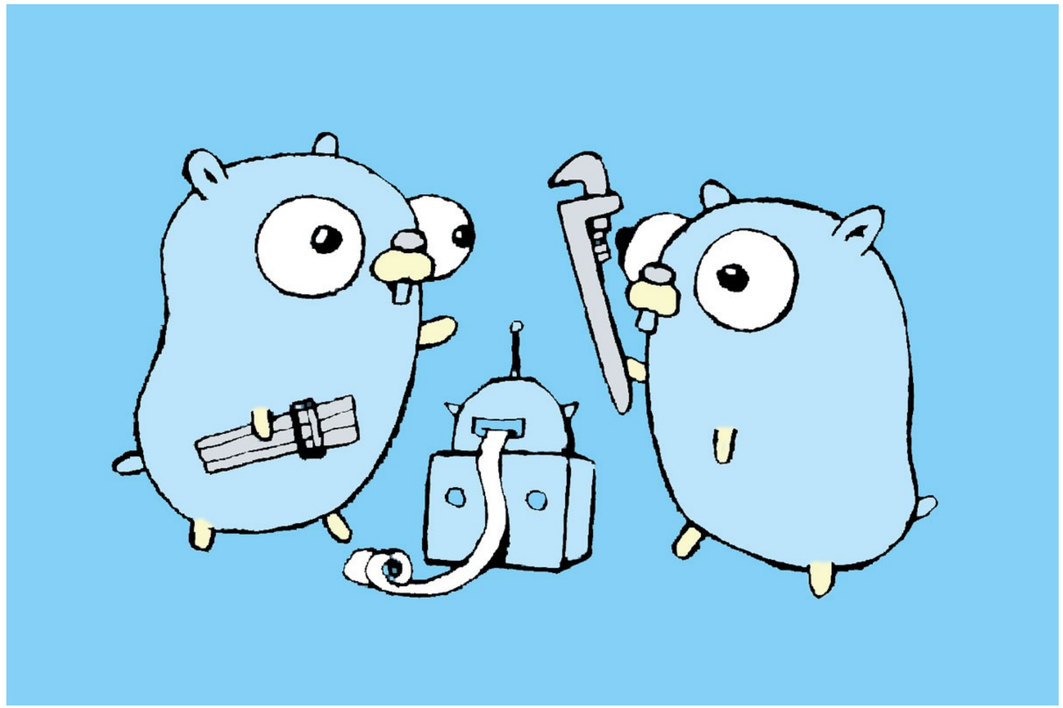
- Чтение с помощью пакета ioutil:

```
bs, err := ioutil.ReadFile("test.txt")
    if err != nil {
        return
    }
    str := string(bs)
    fmt.Println(str)
```

- Создание с помощью пакета os:

```
file, err := os.Create("test.txt")
    if err != nil {
        // здесь перехватывается ошибка
        return
    }
    defer file.Close()

    file.WriteString("test")
```



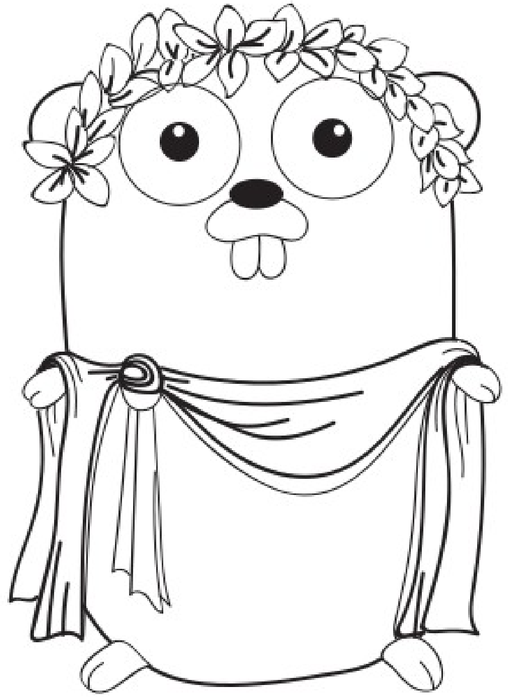
Стандартная библиотека

- Чтобы рекурсивно обойти каталоги (прочитать содержимое текущего и всех вложенных каталогов). Это делается просто с помощью функции Walk, предоставляемой пакетом path/filepath:

```
package main

import (
    "fmt"
    "os"
    "path/filepath"
)

func main() {
    filepath.Walk(".", func(path string, info os.FileInfo, err error) error {
        fmt.Println(path)
        return nil
    })
}
```



Стандартная библиотека

- Хэши и криптография
- Хэш-функции в Go подразделяются на две категории: криптографические и некриптографические.

- Некриптографические функции можно найти в пакете hash, который включает такие алгоритмы как adler32, crc32, crc64 и fnv. Вот пример использования crc32:

```
package main

import (
    "fmt"
    "hash/crc32"
)

func main() {
    h := crc32.NewIEEE()
    h.Write([]byte("test"))
    v := h.Sum32()
    fmt.Println(v)
} // Вывод: 3632233996
```

Стандартная библиотека

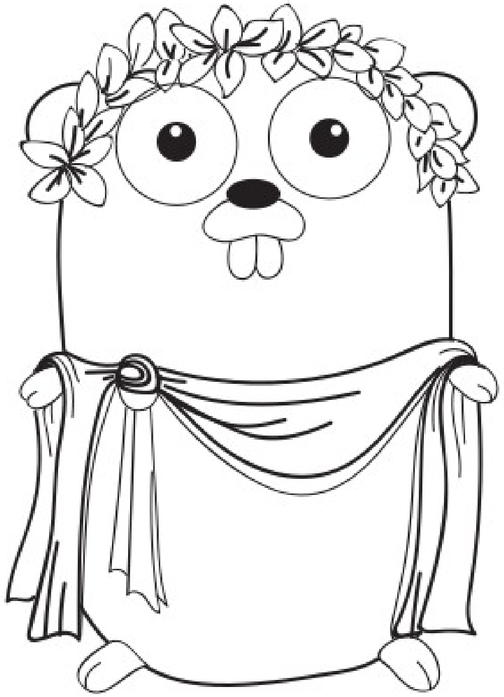
- Криптографические хэш-функции аналогичны некриптографическим.
- Особенность: их сложно обратить вспять.
- Хэши часто используются в системах безопасности.

Пример использования SHA-1:

```
package main

import (
    "fmt"
    "crypto/sha1"
)

func main() {
    h := sha1.New()
    h.Write([]byte("t"))
    bs := h.Sum([]byte{})
    fmt.Println(string(bs))
}
// Вывод: xXSu
```



Стандартная библиотека

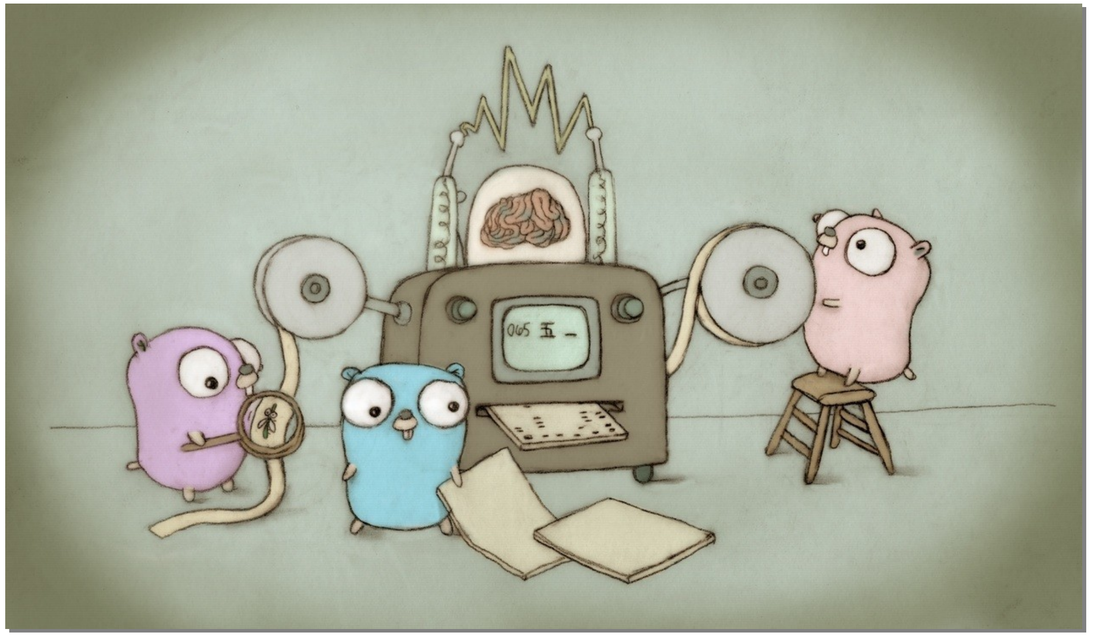
HTTP-сервер. Пример из 1 занятия.

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func greetings(w http.ResponseWriter, r *http.Request) {
    name := r.URL.Query().Get("name")
    age := r.URL.Query().Get("age")
    fmt.Fprint(w, "Hello, my name is ", name)
    fmt.Fprint(w, ", and age is ", age)
}

func main() {
    http.HandleFunc("/", greetings)
    log.Println("http://localhost:8000")
    http.ListenAndServe(":8000", nil)
}
```



Стандартная библиотека

Получение аргументов из командной строки

При вызове команды в консоли, есть возможность передать ей определенные аргументы. Мы видели это на примере вызова команды go:

go run myfile.go

run и myfile.go являются аргументами. Мы так же можем передать команде флаги:

go run -v myfile.go — здесь флаги передаются команде go run
go run myfile.go -check — здесь флаги передаются уже программе myfile.go

Пакет flag позволяет анализировать аргументы и флаги, переданные нашей программе. Вот пример программы, которая генерирует число от 0 до 6. Но мы можем изменить максимальное значение, передав программе флаг -max=100.

```
func main() {
    // Определение флагов
    maxp := flag.Int("max", 6, "the max value")
    // Парсинг
    flag.Parse()
    // Генерация числа от 0 до max
    fmt.Println(rand.Intn(*maxp))
}
```

Любые дополнительные не-флаговые аргументы могут быть получены с помощью flag.Args(), которая вернет []string.

И многое многое другое…
Ваши вопросы — пишите в чат :)

Дальнейшие шаги в Go

- Учитесь у мастеров
- Делайте что-нибудь
- Работайте в команде

Полезные материалы:

1)https://golang.org/doc/effective_go.html
2)https://www.coursera.org/learn/golang-webservices-1/home/welcome
3)https://golangs.org/
4)https://www.gopl.io/
5)https://habr.com/ru/company/mailru/blog/314804/#22
6)https://habr.com/ru/post/421411/
7)https://astaxie.gitbooks.io/build-web-application-with-golang/en/
8)http://tinystruggles.com/2015/10/21/golang-concurrency.html
9)https://github.com/Alikhll/golang-developer-roadmap
10)https://gobyexample.com

Курс окончен!

Интересные ТЗ от компаний yandex, wildberries и g-core:

Благодарю за внимание!!!