Go for Pythonistas

PyCon PL

17 October 2014

Rodolfo Carvalho
Python Lead Developer, Base Lab

大家好！

Go is very boring

Go

"Go is more about software engineering than programming language research."

- "It must work at scale"
- "It must be familiar"
- "It must be modern"

(Rob Pike)

Why?

The Importance of Programming Literacy

PyCon 2007
Robert M. Lefkowitz, r0ml

the art of

communicating through symbols

ideas about reality

syntax

repeating

expand

express

Entenderam?

Do you understand?

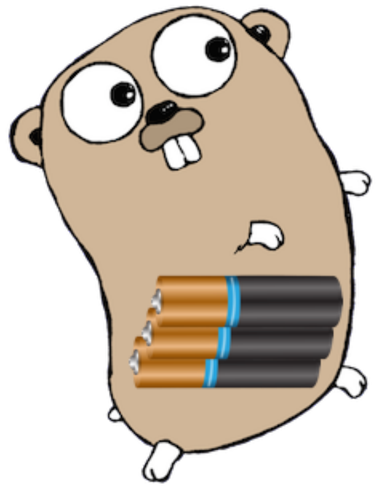
明白了吗?

Czy rozumiecie?

Go is fun and addictive

Like Python

- Batteries included
- Fits in your brain
- Community



Let's Go

List popular repos from GitHub

```go
func main() {
    // Search GitHub repositories
    query, language := "python", "go"
    url := `https://api.github.com/search/repositories?q=` + query +
        `+language:` + language + `&sort=stars&order=desc&per_page=3`
    resp, err := http.Get(url)
    checkErr(err)
    defer resp.Body.Close()

    // Decode JSON response
    dec := json.NewDecoder(resp.Body)
    var sr GitHubSearchResponse
    err = dec.Decode(&sr)
    checkErr(err)

    // Print results
    sr.print()
}
```

Run

Code with us @ Base stand

BASE

Go

Like C

- Statically typed
- Fast
- Easy to distribute

Nothing really new, but

- Great tools
- Concurrency primitives

Let's Go again

```go
package main

import "fmt"

func main() {
    fmt.Println("Cześć! 你好! Olá!")
}
```

Run

Packages

```
package main
```

```
package json
```

```
package amqp
```

- *"Namespaces are one honking great idea -- let's do more of those!"*
- package != module != file

Imports

```
import "fmt"
import "net/http"
import "encoding/json"
```

```
import (
    "fmt"
    "net/http"
    "encoding/json"
)
```

`go get`

```
import "github.com/rhcarvalho/basecrm"
```

Exports

- Public = Upper case

```
http.Get(...)
json.NewDecoder(...)

dec := json.NewDecoder(resp.Body)
dec.Decode()
```

- private = lower case

```
checkErr(...)
sr.print()

// cannot access from outside
http.defaultUserAgent      // private const
http.parseRequestLine(...) // private func
```

No more

```
class Foo:
    def __i_am_very_private(self):
        obj._please_dont_call_me(hack, it)
```

Basic types

```
bool

string

int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr

byte // alias for uint8

rune // alias for int32
     // represents a Unicode code point

float32 float64

complex64 complex128

array slice map struct interface
```

Functions

```
package main

import "fmt"

func max(a, b int) int {
    if a > b {
        return a
    }
    return b
}

func main() {
    fmt.Println(max(6, 9))
    fmt.Println(max(8, 1))
}
```

Run

Type definitions

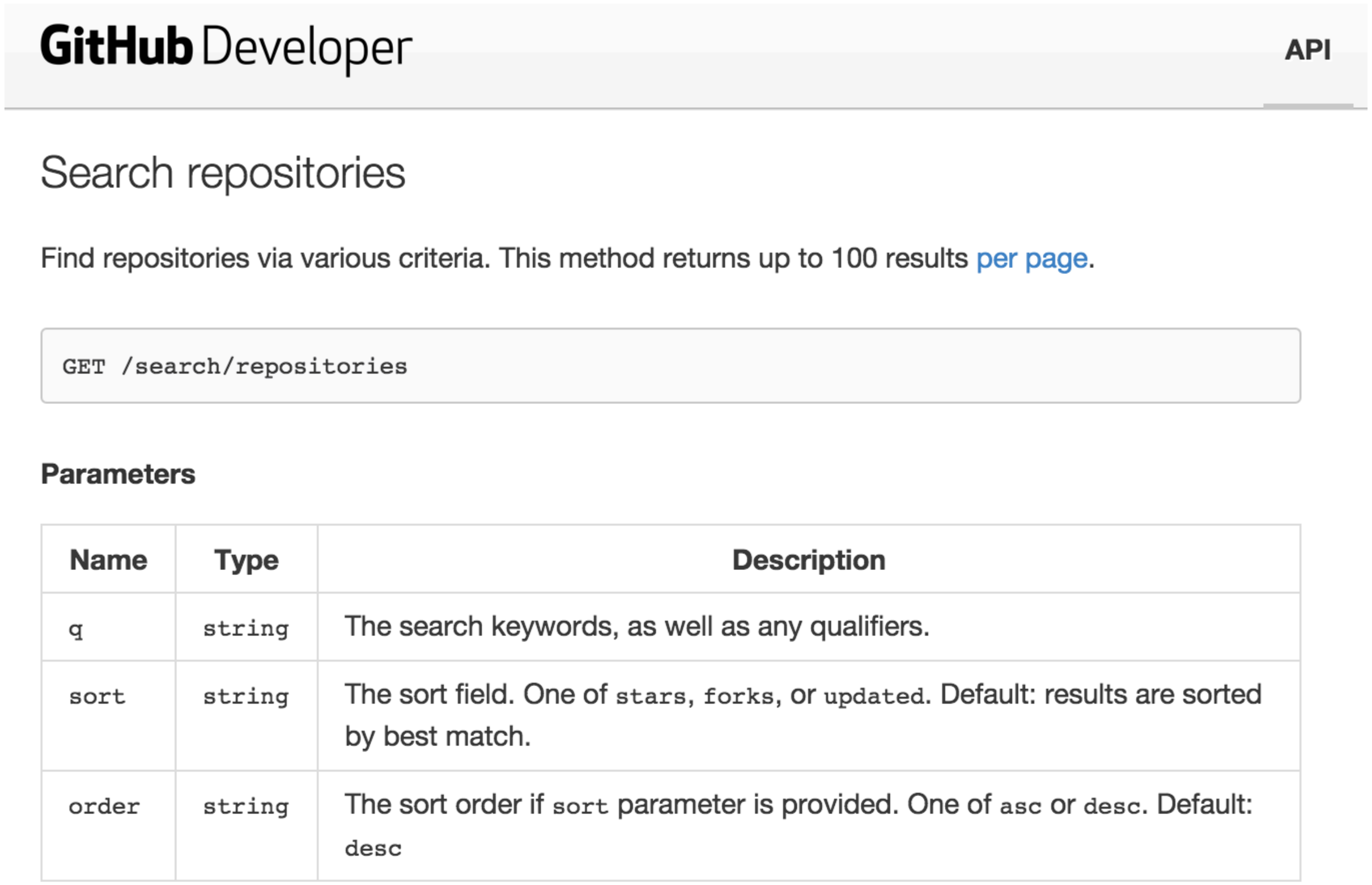
```go
type GitHubSearchResponse struct {
    Items []*Repo
}

type Repo struct {
    Fullname string `json:"full_name"`
    Stars    int    `json:"stargazers_count"`
}
```

Multiple assignment

```
    query, language := "python", "go"
```

GitHub API


GitHub Developer

API

Search repositories

Find repositories via various criteria. This method returns up to 100 results per page.

```
GET /search/repositories
```

Parameters

Name	Type	Description
`q`	`string`	The search keywords, as well as any qualifiers.
`sort`	`string`	The sort field. One of `stars`, `forks`, or `updated`. Default: results are sorted by best match.
`order`	`string`	The sort order if `sort` parameter is provided. One of `asc` or `desc`. Default: `desc`

HTTP request

```go
    query, language := "python", "go"
    url := `https://api.github.com/search/repositories?q=` + query +
        `+language:` + language + `&sort=stars&order=desc&per_page=3`
    resp, err := http.Get(url)
    checkErr(err)
    defer resp.Body.Close()
```

Error handling

```go
func checkErr(err error) {
    if err != nil {
        log.Fatal("ERROR:", err)
        // fmt.Print(...)
        // os.Exit(1)
    }
}
```

Request in Python

- *"Explicit is better than implicit."*

```
import requests
resp = requests.get("https://api.github.com/search/repositories?q=...")
resp.json()
```

If it can break, it will break

```
>>> import requests
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named requests
```

If it can break, it will break

```
>>> resp = requests.get("https://api.github.com/search/repositories?q=...")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  ...
  ...
requests.exceptions.ConnectionError: HTTPSConnectionPool(host='api.github.com', port=443):
Max retries exceeded with url: /search/repositories?q=...
(Caused by <class 'socket.gaierror'>: [Errno 8] nodename nor servname provided, or not known)
```

If it can break, it will break

```
>>> resp.json()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  ...
  ...
ValueError: No JSON object could be decoded
```

Pattern #1: htfe

- Every programmer produces errors, no exceptions :-)
- Handle errors **explicitly**
- Return ASAP
- Use multiple return values
- Pass errors around

Pattern #2: defer

```
    resp, err := http.Get(url)
    checkErr(err)
    defer resp.Body.Close()
```

- Context managers

```
f = open("hello.txt")
try:
    for line in f:
        print line,
finally:
    f.close()
```

==

```
with open('some.file') as f:
for line in f:
    print line,
```

Not really context managers

- No exceptions

```
package main

import "fmt"

func main() {
    fmt.Println("A")
    defer func() {
        fmt.Println("Deferred 1")
    }()
    fmt.Println("B")
    defer func() {
        fmt.Println("Deferred 2")
    }()
}
```

Run

Decoding JSON

```
Status: 200 OK
X-RateLimit-Limit: 20
X-RateLimit-Remaining: 19
```

```
{
  "total_count": 40,
  "incomplete_results": false,
  "items": [
    {
      "id": 3081286,
      "name": "Tetris",
      "full_name": "dtrupenn/Tetris",
      "owner": {
        "login": "dtrupenn",
        "id": 872147,
        ...
      },
      "description": "A C implementation of Tetris using Pennsim through LC4",
      "url": "https://api.github.com/repos/dtrupenn/Tetris",
      "created_at": "2012-01-01T00:31:50Z",
      "updated_at": "2013-01-05T17:58:47Z",
      "pushed_at": "2012-01-01T00:37:02Z",
      "stargazers_count": 1,
      "watchers_count": 1,
      "language": "Assembly",
      ...
    }
  ]
}
```

Decoding JSON

```
    dec := json.NewDecoder(resp.Body)
    var sr GitHubSearchResponse
    err = dec.Decode(&sr)
    checkErr(err)
```

Github Search meets Python

```python
import urllib2, json

query, language = "python", "go"

url = ("https://api.github.com/search/repositories?q={}"
       "+language:{}&sort=stars&order=desc&per_page=3").format(query, language)
resp = urllib2.urlopen(url)
data = json.load(resp)

print(" #  {:<20} {}".format("Repo URL", "Stars"))
print("-" * 30)
for i, r in enumerate(data["items"], 1):
    print("{:02d}. {:<20} {:d}".format(i, r["full_name"], r["stargazers_count"]))
```

Run

Use requests

```
import requests

query, language = "python", "go"

url = ("https://api.github.com/search/repositories?q={}"
       "+language:{}&sort=stars&order=desc&per_page=3").format(query, language)
resp = requests.get(url)
data = resp.json()

print(" #  {:<20} {}".format("Repo URL", "Stars"))
print("-" * 30)
for i, r in enumerate(data["items"], 1):
    print("{:02d}. {:<20} {:d}".format(i, r["full_name"], r["stargazers_count"]))
```

Run

- *"There should be one-- and preferably only one --obvious way to do it."*

Refactoring

```go
func main() {
    sr, err := searchPopularByLanguage("python", "go")
    checkErr(err)
    sr.print(os.Stdout)
}
```

Run

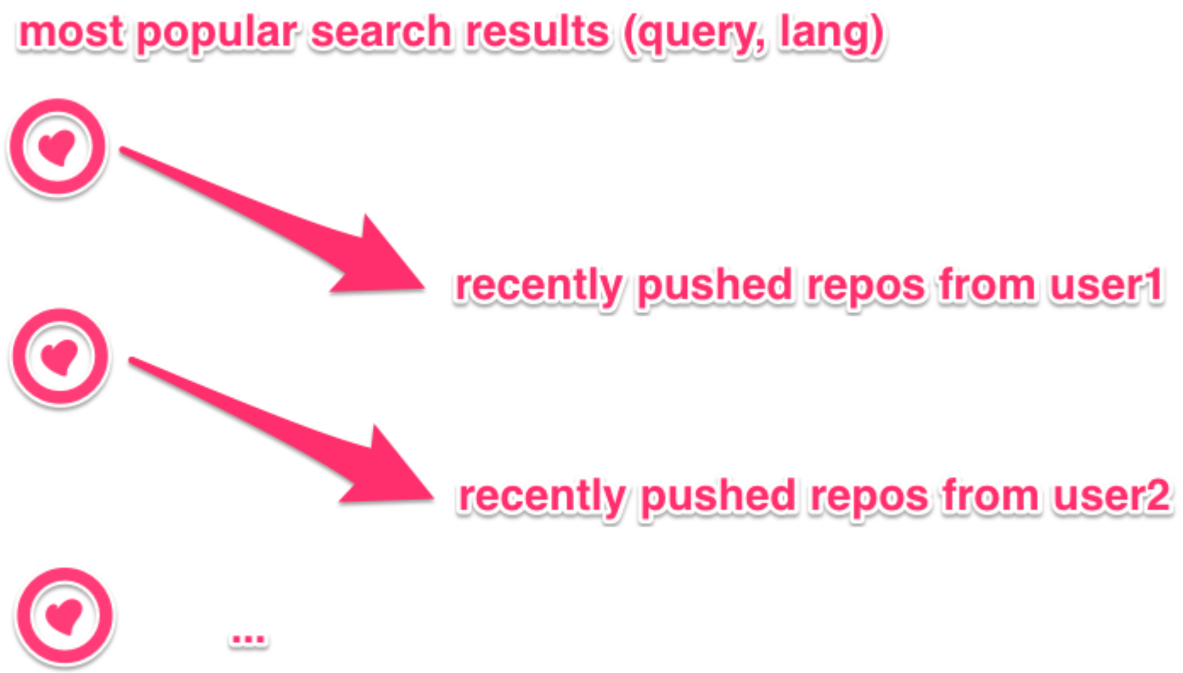
Refactoring

```
func searchPopularByLanguage(query, language string) (*GitHubSearchResponse, error) {
    queryString := `q=` + query + `+language:` + language + `&sort=stars&order=desc&per_page=3`
    return searchRepo(queryString)
}

func searchRepo(queryString string) (*GitHubSearchResponse, error) {
    var sr *GitHubSearchResponse
    url := `https://api.github.com/search/repositories?` + queryString
    resp, err := http.Get(url)
    if err != nil {
        return nil, err
    }
    defer resp.Body.Close()
    dec := json.NewDecoder(resp.Body)
    err = dec.Decode(&sr)
    if err != nil {
        return nil, err
    }
    return sr, nil
}
```

Run

Let's do more


most popular search results (query, lang)
recently pushed repos from user1
recently pushed repos from user2
...

Loop

```
    for i := 0; i < 100; i++ {
        // like C
    }
```

```
    for i, el := range []int{11, 7, 87} {
        // like Python:
        // for i, el in enumerate([11, 7, 87]):
        fmt.Println(i, el)
    }
```

Run

Or

```
    for condition {
        // while loop
    }
```

```
    for {
        // infinite loop
    }
```

Changes

```
    Repo struct {
        Fullname string    `json:"full_name"`
        Stars    int       `json:"stargazers_count"`
        PushedAt time.Time `json:"pushed_at"`
        Owner    struct {
            Login       string
            RecentRepos []*Repo
```

```
func main() {
    sr, err := searchPopularByLanguage("python", "go")
    checkErr(err)
    for _, r := range sr.Items {
        recent, err := recentUserRepos(r.Owner.Login)
        checkErr(err)
        r.Owner.RecentRepos = recent.Items
    }
    printResponse(os.Stdout, sr)
}
```

Run

Changes

```go
func recentUserRepos(user string) (*GitHubSearchResponse, error) {
    queryString := `q=user:` + user + `&sort=updated&per_page=3`
    return searchRepo(queryString)
}
```

```go
func printResponse(w io.Writer, sr *GitHubSearchResponse) {
    now := time.Now()
    width := 30
    fmt.Fprintln(w, " #  Repo URL")
    fmt.Fprintln(w, strings.Repeat("-", width+35))
    for i, r := range sr.Items {
        fmt.Fprintf(w, "%02d. %-*s stars: %5d  pushed: %3.0fd ago\n",
            i+1, width, r.Fullname, r.Stars, daysSince(now, r.PushedAt))
        for _, rr := range r.Owner.RecentRepos {
            fmt.Fprintf(w, "    %-*s        %5d          %3.0fd ago\n",
                width, rr.Fullname, rr.Stars, daysSince(now, rr.PushedAt))
        }
        fmt.Fprintf(w, "    %s\n", strings.Repeat("-", width+31))
    }
}
```

Run

Concurrency

Goroutines

```go
package main

import "fmt"

func main() {
    go func() {
        fmt.Println("Will you see me?")
    }()
}
```

Run

Goroutines

```
package main

import "fmt"
import "time"

func main() {
    go func() {
        fmt.Println("Will you see me?")
    }()
    time.Sleep(1 * time.Second)
}
```

Run

Channels

```go
package main

import "fmt"

func main() {
    done := make(chan bool)
    go func() {
        fmt.Println("Will you see me?")
        done <- true
    }()
    <-done
}
```

Run

- No " result = **yield from** func(args) "

Channels

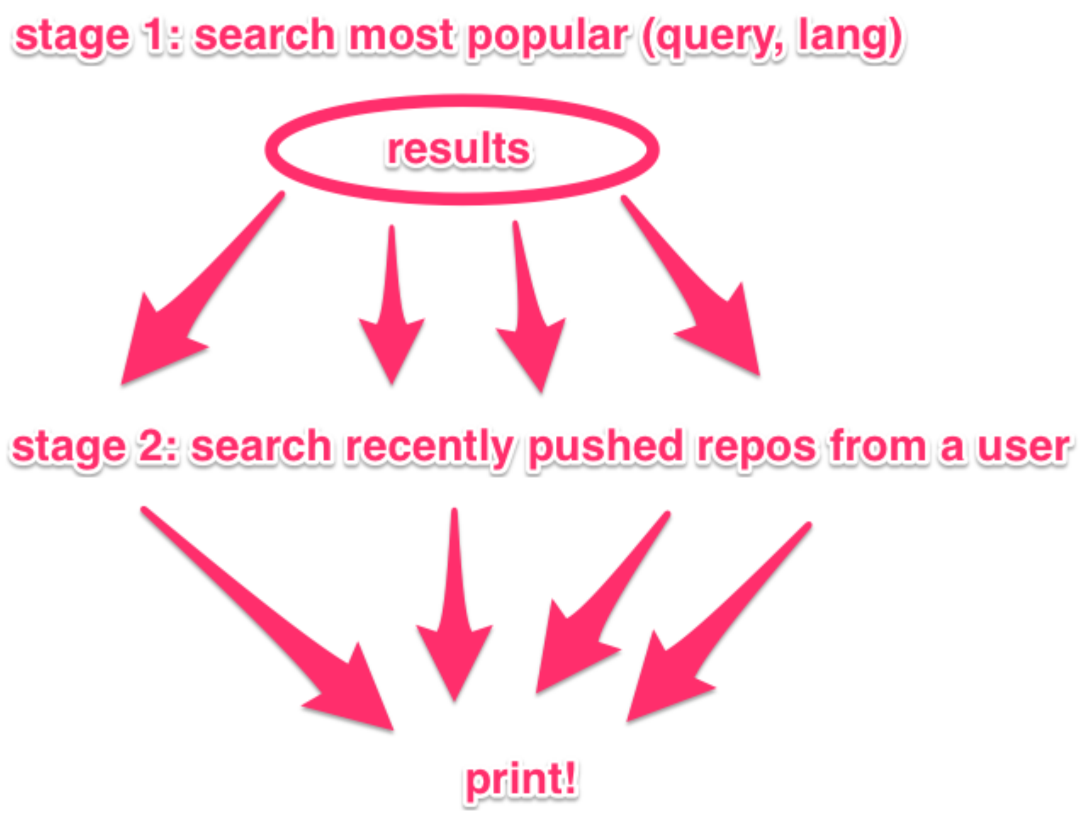
```
package main

import "fmt"

func main() {
    done := make(chan struct{})
    go func() {
        fmt.Println("Will you see me?")
        done <- struct{}{}
    }()
    <-done
}
```

Run

Pipeline


stage 1: search most popular (query, lang)
results
stage 2: search recently pushed repos from a user
print!

Pipeline

```go
func stage1(in chan *GitHubSearchQuery, out chan *GitHubSearchResponse) {
    for q := range in {
        sr, err := searchPopularByLanguage(q.Query, q.Language)
        checkErr(err)
        out <- sr
    }
    close(out)
}
```

```go
func stage2(in chan *GitHubSearchResponse, out chan *GitHubSearchResponse) {
    for sr := range in {
        for _, r := range sr.Items {
            recent, err := recentUserRepos(r.Owner.Login)
            checkErr(err)
            r.Owner.RecentRepos = recent.Items
        }
        out <- sr
    }
    close(out)
}
```

Pipeline

```
type GitHubSearchQuery struct {
    Query, Language string
}
```

```
func main() {
    in := make(chan *GitHubSearchQuery)
    out1 := make(chan *GitHubSearchResponse)
    out2 := make(chan *GitHubSearchResponse)

    go stage1(in, out1)
    go stage2(out1, out2)

    in <- &GitHubSearchQuery{"python", "go"}
    in <- &GitHubSearchQuery{"sql", "go"}
    close(in)

    for sr := range out2 {
        printResponse(os.Stdout, sr)
    }
}
```

Run

More

Interfaces

- Static typed duck typing!
- No more `__magic_method__`

Docstrings :)

Testing

Struct embedding

Mutexes

Methods

Zero values

Cannot compile with unused imports

Call to action: learn languages!

What are your questions?

Thank you

Rodolfo Carvalho
Python Lead Developer, Base Lab
rodolfo@getbase.com (mailto:rodolfo@getbase.com)
http://rodolfocarvalho.net (http://rodolfocarvalho.net)
http://py.getbase.com (http://py.getbase.com)