Do I need to switch to Go(lang)

Max Tepkeev
20 July 2016
Bilbao, Spain

About me

Max Tepkeev
Russia, Moscow

- python-redmine
- architect
- instructions

https://www.github.com/maxtepkeev

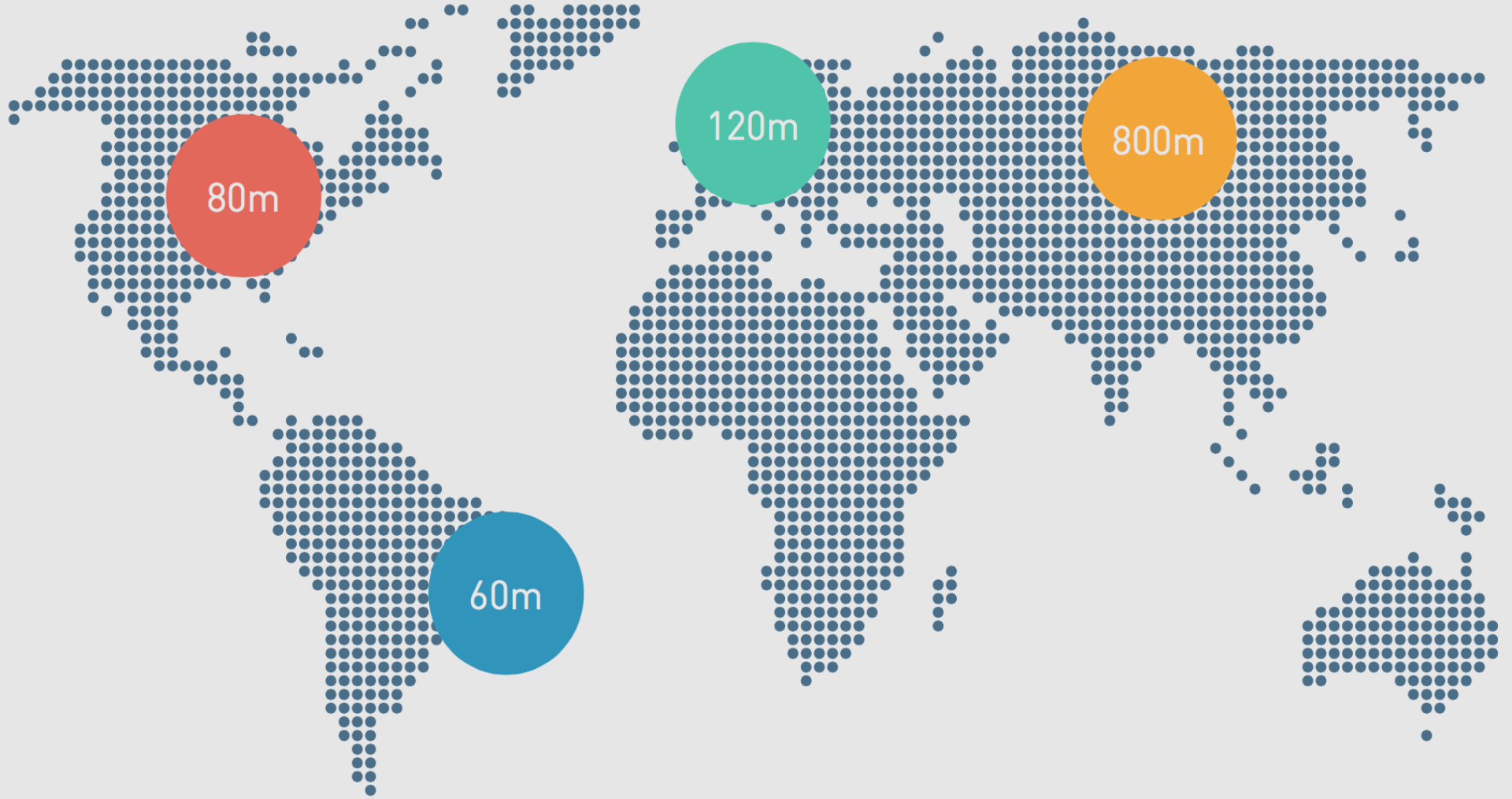
About us

Aidata – online and offline user data collection and analysis

1 000 000 000
unique users per month

http://www.aidata.me

About us



Why switch from Python

- Speed
- Concurrency
- GIL
- No ”true” binaries
- Dynamic types

Language requirements

- Modern
- Fast
- Easy
- Compiled
- Statically typed
- Support main platforms

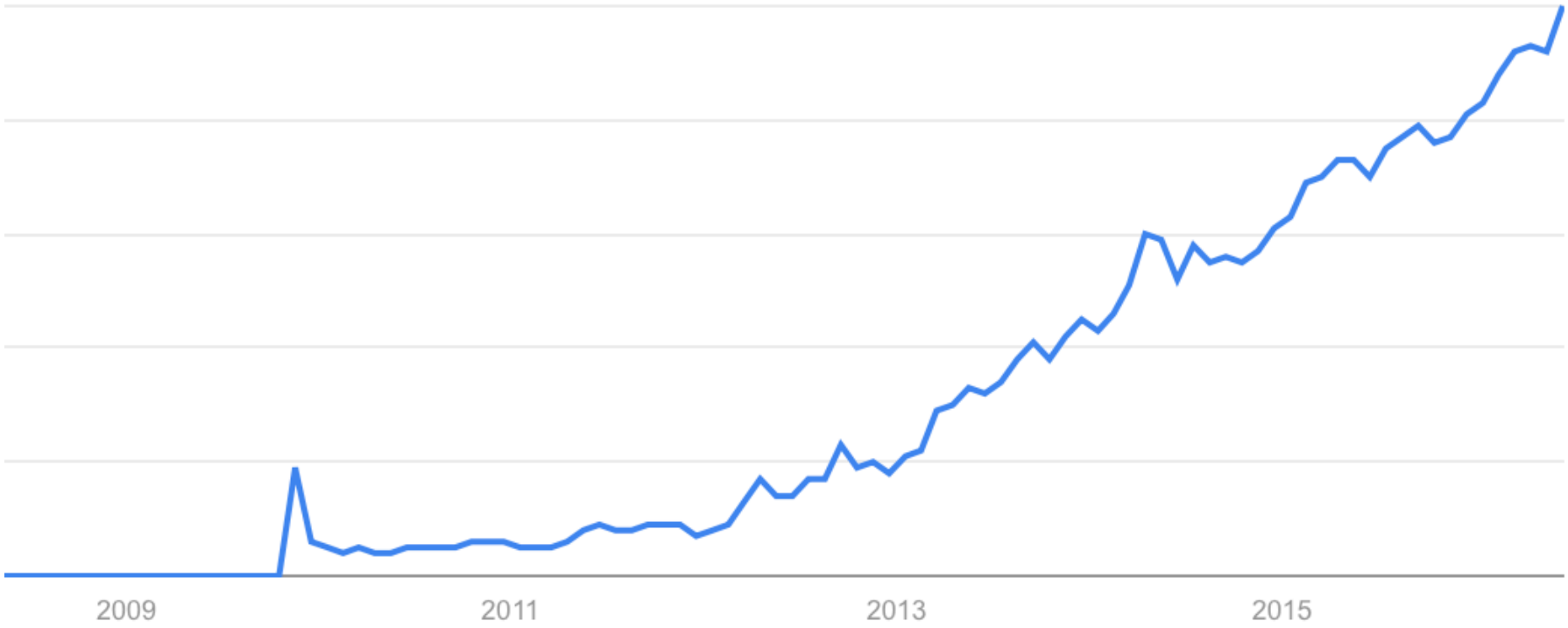
Why Go

StackOverflow questions by tag*:

- Go: 16795 (1113)
- Rust: 4226 (429)
- D: 2025 (34)
- Nim: 136 (8)

* as of 19 July 2016

Popularity



Go

- Created in Google
- Open-sourced in November 2009
- Designed by smart people:
  - Robert Griesemer
  - Rob Pike
  - Ken Thompson

https://talks.golang.org/2012/splash.article

Who uses Go

- Google
- Dropbox
- Facebook
- IBM
- Intel
- SpaceX
- Uber
- VMware
- Yahoo
- Twitter
- Heroku
- DigitalOcean

https://github.com/golang/go/wiki/GoUsers

Project layout

```
$GOPATH/
    bin/
      hello                 # command executable
      outyet                # command executable
    pkg/
      linux_amd64/
        github.com/golang/example/
          stringutil.a      # package object
    src/
      github.com/golang/example/
        .git/                 # Git repository metadata
        hello/
          hello.go          # command source
        outyet/
          main.go           # command source
        stringutil/
          reverse.go        # package source
```

Package management

```
$ go get github.com/golang/example
```

```
src/
   github.com/golang/example/
      .git/
      hello/
         hello.go
      outyet/
         main.go
      stringutil/
         reverse.go
```

Package management

- vendoring
- gopkg.in
- getgb.io

github.com/tools/godep
github.com/gpmgo/gopm
github.com/pote/gpm
github.com/nitrous-io/goop
github.com/alouche/rodent
github.com/jingweno/nut
github.com/niemeyer/gopkg
github.com/mjibson/party
github.com/kardianos/vendor
github.com/kisielk/vendorize
github.com/mattn/gom

github.com/dkulchenko/bunch
github.com/skelterjohn/wgo
github.com/Masterminds/glide
github.com/robfig/glock
bitbucket.org/vegansk/gobs
launchpad.net/godeps
github.com/d2fn/gopack
github.com/laher/gopin
github.com/LyricalSecurity/gigo
github.com/VividCortex/johnny-deps

Commands

- go fmt
- go test
- go fix
- go run
- go run -race
- go vet

Hello World

$GOPATH/src/github.com/user/hello/hello.go:

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, world.")
}
```

```
$ go install github.com/user/hello
$ $GOPATH/bin/hello
Hello, world.
```

Comparison: Imports

Python	Go
`import os`	`import "os"`

`import os`
`import gzip`
	`import (`
	`  "os"`
	`  "compress/gzip"`
	`)`

`import os as os1`	`import os1 "os"`

Comparison: Import Names

Python

```
import os

os.getcwd()
```

Go

```
import "os"

os.Getwd()
```

Comparison: Types

Python

```
b  = True
s  = "hey"
si = 1
ui = -1
f  = 1.0
c  = 3j
l  = [1, 2]
m  = {1:1}
```

Go

```
var (
    b  bool       = true
    s  string     = "hey"
    si int        = -1
    ui uint       = 1
    f  float32    = 1.0
    c  complex64  = 3i
    l  []int      = []int{1, 2}
    m  map[int]int = map[int]int{1: 1}
)
```

Comparison: Variables

Python

```
i = 1
j = "hey”


i, j = 1, "hey"
```

Go

```
var (
   i = 1
   j = "hey"
)

i := 1
j := "hey”

i, j := 1, "hey"
```

Comparison: 1st Class Functions

Python

```
def add(x, y):
    return x + y

add = lambda x, y: x + y
```

Go

```
func add(x, y int) int {
    return x + y
}

add := func (x, y int) int {
    return x + y
}
```

Comparison: Optional args

Python

```
def sum(*nums):
    result = 0
    for num in nums:
        result += num
    return result


sum(1, 2, 3)
sum(*[1, 2, 3])
```

Go

```
func sum(nums ...int) {
    result := 0
    for _, num := range nums {
        result += num
    }
    return result
}
sum(1, 2, 3)
sum([]int{1, 2, 3}...)
```

Comparison: Optional kwargs

Python

```
def join(foo=None, bar=None):
    if foo is None:
        foo = 'foo'
    if bar is None:
        bar = 'bar'
    return foo + bar



join()
join(bar='foo')
```

Go

```
type Kwargs struct {
    foo, bar string
}

func join(kw Kwargs) string {
    if kw.foo == "" {kw.foo = "foo"}
    if kw.bar == "" {kw.bar = "bar"}
    return kw.foo + kw.bar
}

join(Kwargs{})
join(Kwargs{bar: "foo"})
```

Comparison: Loops

Python

```
seq = [1, 2, 3]
for x in seq:
    print x
```

Go

```
seq := []int{1, 2, 3}
for _, x := range seq {
    fmt.Println(x)
}
```

Comparison: Loops

Python

```
seq = [1, 2, 3]
num, count = 0, len(seq)

while num < count:
    print seq[num]
    num += 1
```

Go

```
seq := []int{1, 2, 3}


for i := 0; i < len(seq); i++ {
    fmt.Println(seq[i])
}
```

Comparison: Loops

Python

```
while True:
    print "Python rocks"
```

Go

```
for {
    fmt.Println("Go rocks")
}
```

Comparison: Comprehensions

Python

```
word = "hello"
chars = [char for char in word]
print chars  # ['h', 'e', 'l', 'l', 'o']
```

Go

```
word := "hello"
chars := []string{}
for _, char := range word {
    chars = append(chars, string(char))
}
fmt.Println(chars) // [h e l l o]
```

Comparison: Conditionals

Python

```
x = 1
if x > 0:
    print x
else:
    print "Sorry"
```

Go

```
if x := 1; x > 0 {
    fmt.Println(x)
} else {
    fmt.Println("Sorry")
}
```

Comparison: Conditionals

Python

```
x = 1
if x > 0:
    print "Positive"
elif x < 0:
    print "Negative"
else:
    print "Zero"
```

Go

```
switch x := 1; {
case x < 0:
    fmt.Println("Negative")
case x > 0:
    fmt.Println("Positive")
default:
    fmt.Println("Zero")
}
```

Comparison: Slicing

Python

```
seq = [1, 2, 3, 4, 5]
print seq[:2]    # [1, 2]
print seq[3:]    # [4, 5]
print seq[2:3]   # [3]
```

Go

```
seq := []int{1, 2, 3, 4, 5}
fmt.Print(seq[:2])    // [1 2]
fmt.Print(seq[3:])    // [4 5]
fmt.Print(seq[2:3]) // [3]
```

Comparison: Error Handling

Python

```
try:
    conn = db.Connect()
except ConnectionError:
    print "Can't connect"
```

Go

```
conn, err := db.Connect()
if err != nil {
    fmt.Print("Can't connect")
}
```

panic() / recover()

Comparison: Classes

Python

```python
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return "{}: {}".format(
            self.name, self.age)


p = Person("Batman", 45)
print p  # Batman: 45
```

Go

```go
type Person struct {
    Name string
    Age  int
}

func (p Person) String() string {
    return fmt.Sprintf(
        "%s: %d", p.Name, p.Age)
}

p := Person{"Batman", 45}
fmt.Println(p) // Batman: 45
```

Comparison: Constructors

Python

```python
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age


p = Person("Batman", 45)
```

Go

```go
type Person struct {
    Name string
    Age  int
}

func NewPerson(n string, a int) *Person {
    return &Person{Name: n, Age: a}
}

p := NewPerson("Batman", 45)
```

Comparison: Inheritance

Python

```
class Person(object):
    def __init__(self, name):
        self.name = name
    def eat(self):
        print "Eating"

class Doctor(Person):
    def heal(self):
        print "Healing"


d = Doctor("Gregory House")
d.eat()   # "Eating"
d.heal()  # "Healing"
```

Go

```
type Person struct{ Name string }
type Doctor struct{ Person }

func (p Person) Eat() {
    fmt.Println("Eating")
}

func (d Doctor) Heal() {
    fmt.Println("Healing")
}

d := Doctor{Person{"Gregory House"}}
d.Eat()   // Eating
d.Heal() // Healing
```

Comparison: Cleanup

Python

```python
def read_file():
    result = None
    fpath = "/tmp/file"
    with open(fpath) as f:
        result = f.read()
    # file is closed here
    return result
```

Go

```go
func readFile() (result string) {
    fpath := "/tmp/file"
    f, err := os.Open(fpath)
    if err != nil {
        panic(err)
    }
    defer f.Close()
    // reading file here
    return result
}
```

Comparison: Concurrent progs

Python

```
urls = ['http://python.org', 'http://golang.org']

async def fetch(session, url):
    async with session.get(url) as response:
        return await response.read()

async def fetch_all(session, urls, loop):
    tasks = [fetch(session, url) for url in urls]
    return await asyncio.gather(*tasks)

loop = asyncio.get_event_loop()

with aiohttp.ClientSession(loop=loop) as session:
    responses = loop.run_until_complete(
        fetch_all(session, urls, loop))

print(responses)
```

Go

```
urls := []string{"http://python.org",
                 "http://golang.org"}

responses := make(chan string)

for _, url := range urls {
    go func(url string) {
        resp, _ := http.Get(url)
        defer resp.Body.Close()
        body, _ := ioutil.ReadAll(resp.Body)
        responses <- string(body)
    }(url)
}

for response := range responses {
    fmt.Println(response)
}
```

More Go features

- make() / new()
- Arrays
- Pointers
- Interfaces
- Type assertions
- Type switches
- Buffered channels / select statement
- Unsafe package
- Cross compilation

Python features Go lacks

- list / dict comprehensions
- generators
- decorators
- exceptions
- metaclasses
- descriptors
- __magic__ methods
- set / tuple

Conclusion

Python

+ Good standard library
+ 3rd Party libs for everything
+ Less code
+ Syntactic sugar
+ Has Soul

- Interpreted
- Dynamically typed
- Concurrency

Go

+ Good standard library
+ 3rd Party libs for almost everything
+ Compiled
+ Statically typed
+ Built-in concurrency

- Verbose
- No syntactic sugar
- No Soul

Go Useful Links

1. https://tour.golang.org/
2. http://golang.org/doc/effective_go.html
3. https://play.golang.org/
4. http://golangweekly.com/
5. http://devs.cloudimmunity.com/gotchas-and-common-mistakes-in-go-golang/

Questions

slides: http://slideshare.net/maxtepkeev

github: https://github.com/maxtepkeev
email: tepkeev@gmail.com
skype: max.tepkeev
company: http://www.aidata.me