



GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

Go+ 演进之路

姜 智@七牛云




大纲

- Go+ 发展历程
- Go+ 特性
- Go+ 实现解析

Go+ 意外走红

Hacker News 100 @newsyc100 · Jun 19

GoPlus The Go+ language for data science github.com/qiniu/goplus (bit.ly/3fx3ISi)



qiniu/goplus
GoPlus - The Go+ language for data science.
Contribute to qiniu/goplus development by creatin...
github.com

2 1

Hacker News @HNTweets · Jun 19

GoPlus – The Go+ language for data science: github.com/qiniu/goplus
Comments: news.ycombinator.com/item?id=235500...



qiniu/goplus
GoPlus - The Go+ language for data science.
Contribute to qiniu/goplus development by creatin...
github.com

1 3

news.ycombinator.com

实时音视频云简介... TED_网易公开课 公众号运营 教育 在七牛 rtc 音视频行业 IM S

Hacker News new | past | comments | ask | show | jobs | submit


1. ▲ **Friendly Feudalism: The Tibet Myth (2003)** (swans.com)
24 points by greencore 1 hour ago | hide | 2 comments
2. ▲ **Breakthrough in inverse Laplace transform procedures** (inverselaplace.org)
73 points by pabo 3 hours ago | hide | 17 comments
3. ▲ **The Return of the 90s Web** (mxb.dev)
524 points by mxbck 12 hours ago | hide | 225 comments
4. ▲ **The Silicon Oligarchy** (themargins.substack.com)
39 points by wheresvic3 1 hour ago | hide | 11 comments
5. ▲ **Practical Python Programming** (github.com)
256 points by signa11 11 hours ago | hide | 47 comments
6. ▲ **Hash: A free, online platform for modeling** (joelonsoftware.com)
69 points by tobr 6 hours ago | hide | 17 comments
7. ▲ **On Cultures That Build** (scholars-stage.blogspot.com)
114 points by barry-cotter 9 hours ago | hide | 67 comments
8. ▲ **Mozilla VPN** (blog.mozilla.org)
1057 points by caution 16 hours ago | hide | 436 comments
9. ▲ **Caffeine: Livecoding environment for web browsers, Node.js, and WebAssembly** (caffeine.js.org)
47 points by memexy 7 hours ago | hide | 9 comments
10. ▲ **Procedural Lake Village** (anastasiaopara.com)
108 points by memexy 10 hours ago | hide | 30 comments
11. ▲ **"Secret" History of Silicon Valley (2008)** (steveblank.com)
142 points by cushychicken 10 hours ago | hide | 27 comments
12. ▲ **Berlin Gold Hat** (wikipedia.org)
32 points by benbreen 6 hours ago | hide | 21 comments
13. ▲ **Amazon's Enforcement Failures Leave Open a Back Door to Banned Goods** (themarkup.org)
9 points by Sumitmic 5 hours ago | hide | discuss
14. ▲ **Zombie Satellites Return from the Graveyard** (ieee.org)
45 points by ashleyyang 9 hours ago | hide | 16 comments
15. ▲ **GoPlus – The Go+ language for data science** (github.com)
107 points by dx034 12 hours ago | hide | 36 comments
16. ▲ **Bessemer Ventures Anti-Portfolio** (bvp.com)
27 points by jkuria 9 hours ago | hide | discuss
17. ▲ **Bees Can Sense the Electric Fields of Flowers (2013)** (nationalgeographic.com)
75 points by EndXA 11 hours ago | hide | 17 comments

Go+ 意外走红


github.com/trending/go?since=weekly

Explore Topics **Trending** Collections Events GitHub Sponsors [Get email updates](#)

Repositories Developers Spoken Language: Any Language: Go Date range: This week

 **qiniu / goplus** [★ Unstar](#)


GoPlus - The Go+ language for data science

Go ☆ 1,696 🍴 165 Built by  ☆ 650 stars this week


github.com/trending/go?since=daily

Explore Topics **Trending** Collections Events GitHub Sponsors [Get email updates](#)

Repositories Developers Spoken Language: Any Language: Go Date range: Today

 **qiniu / goplus** [★ Unstar](#)

GoPlus - The Go+ language for data science

Go ☆ 1,696 🍴 165 Built by  ☆ 401 stars today

GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

Worldwide Go+



🍛 I love curry 🍛

mattn
mattn

♡ Sponsor

Unfollow

Long-time Golang user&contributor,
Google Dev Expert for Go, and author of
many Go tools, Vim plugin author.
Windows hacker C#/Java/C/C++

📍 Osaka, Japan

✉ mattn.jp@gmail.com

🌐 <https://mattn.kaoriya.net/>

🐦 @mattn_jp

Overview Repositories 1.6k Projects 0 Packages 1 Stars 1.8k Followers 6.2k Following 1.7k

Popular repositories

emmet-vim

emmet for vim: <http://emmet.io/>

● Vim script ☆ 5.1k 🍴 377

go-sqlite3

sqlite3 driver for go using database/sql

● C ☆ 4.1k 🍴 742

go-gtk

Go binding for GTK

● Go ☆ 1.6k 🍴 222

vim-gist

Vim plugin for Gist

● Vim script ☆ 1.5k 🍴 136

goreman

foreman clone written in go language

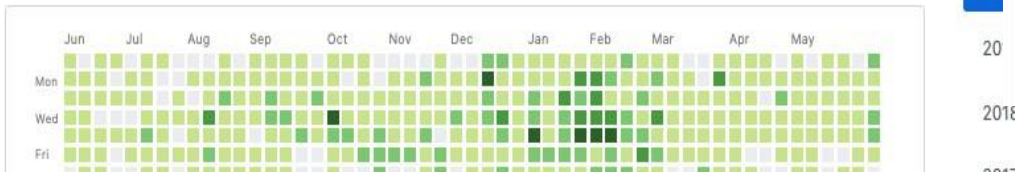
● Go ☆ 1.4k 🍴 123

gom

Go Manager - bundle for go

● Go ☆ 1.4k 🍴 103

3,211 contributions in the last year



mattn

@mattn_jp

データサイエンスに向けた Go 言語の拡張言語。ほぼ Go 言語のまま型を書かなくて良いスクリプト言語。
[#golang](#) / "GitHub - qiniu/goplus: GoPlus - The Go+ language for data science"



qiniu/goplus

GoPlus - The Go+ language for data science. Contribute to qiniu/goplus development by creating an account on GitHub.

github.com


下午2:43 · 2020年6月9日 · はてなブックマーク

GOPHER CHINA 2020

中国 上海 / 2020-11-21-22

Worldwide Go+

github.com/mewmew



Robin Eklind
mewmew


Follow

Sweden

PRO

Block or report user

Organizations



2,670 contributions in the last year

Overview Repositories

- decomp/decomp**
Components of a decompressor
Go 311 25
- minds-eye**
A futuristic dystopian movie
2

gonum
Consistent, composable, and comprehensible scientific code
23 repositories 8 members



Nate Fischer
nfischer

Unfollow

@google engineer working on Chromium and Android WebView. @shelljs lead dev

@google
Mountain View, California
<https://nfischer.github.io/>

Block or report user

Organizations

GO

Overview Repositories 47 Projects 0 Stars 264

Pinned

shelljs/shelljs
Portable Unix shell commands for Node.js
JavaScript 11.1k 641

shelljs-transpiler
Easily transpile Bash to ShellJS
HTML 45 6

rainbows-lang
A prototype of the rainbows programming language
JavaScript 9

145 contributions in the last year



Worldwide Go+



Yoshi Yamaguchi 
@ymotongpoo

Developer Advocate of @Google Cloud for Observability & Go; Google Cloud Operations and OpenTelemetry. An enthusiastic Gopher. Tweets are on my own.

📍 日本 東京 📅 Joined April 2007

654 Following 8,378 Followers

Followed by ドッグ and mattn

Tweets Tweets & replies Media Likes

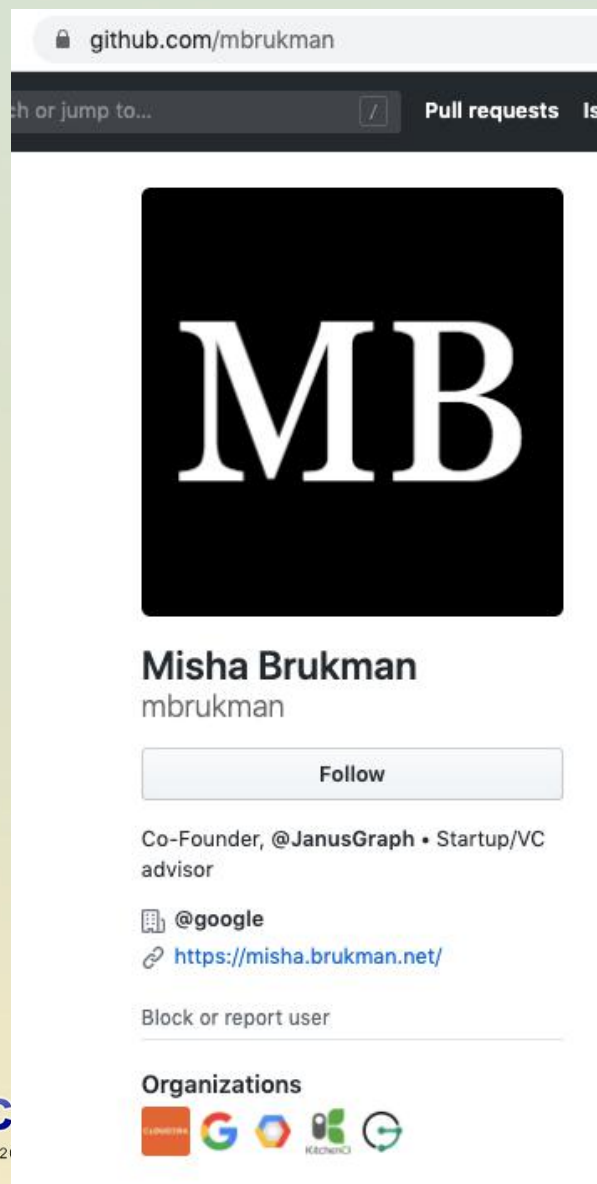
Yoshi Yamaguchi  @ymotongpoo · 47m

I have heard this programming language name as some sort of joke but it became reality. 📺 qiniu/goplus: GoPlus - The Go+ language for data science




qiniu/goplus
GoPlus - The Go+ language for data science.
Contribute to qiniu/goplus development by creatin...
github.com

🗨️ 🔄 4 ❤️ 3 📤



github.com/mbrukman


Search or jump to... Pull requests



Misha Brukman
mbrukman


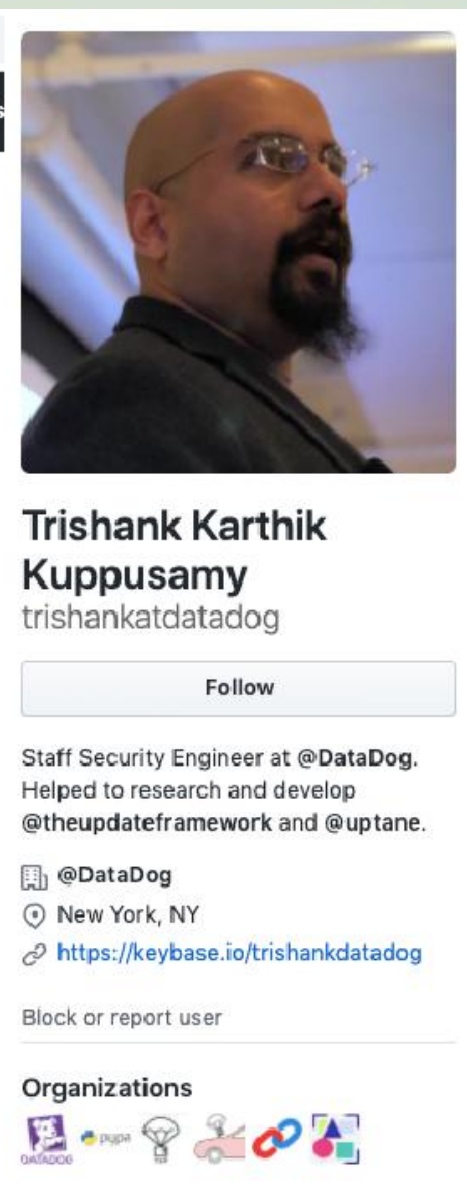

Follow

Co-Founder, @JanusGraph • Startup/VC advisor

 @google
<https://misha.brukman.net/>

Block or report user


Organizations



Trishank Karthik Kuppusamy
trishankatdatadog


Follow

Staff Security Engineer at @DataDog. Helped to research and develop @theupdateframework and @uptane.

 @DataDog
📍 New York, NY
<https://keybase.io/trishankdatadog>


Block or report user

Organizations



Worldwide Go+

mp to... Pull requests Is



TJ Holowaychuk
tj

[Sponsor](#)

[Follow](#)

Founder of Apex • <https://apex.sh/blog/> • <https://twitter.com/tjholowaychuk> • <https://instagram.com/tjholowaychuk> • <http://tjholowaychuk.com>

[Apex](#)
[London, UK](#)
tj@apex.sh
<https://apex.sh>

[GitHub Sponsor](#)
[PRO](#)

[Block or report user](#)

github.com/expressjs/express/graphs/contributors

expressjs / **express**

Used by - 6.9m Watch - 1.8k Star 49k Fork 8.2k

<> Code Issues 89 Pull requests 60 Actions Wiki Security 0 Insights

Pulse

Contributors

Community

Commits

Code frequency

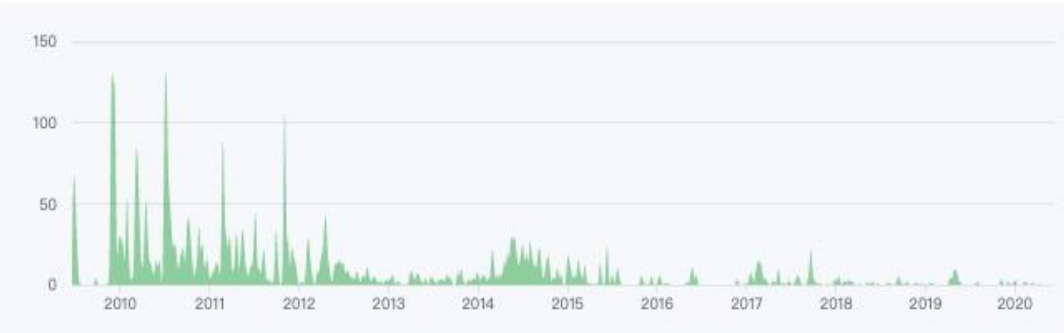
Dependency graph

Network

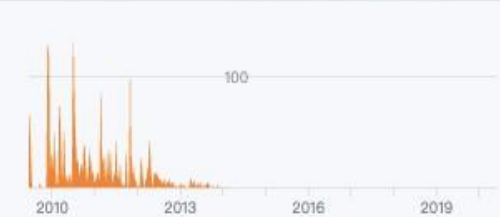
Forks

Jun 21, 2009 – Jun 13, 2020 Contributions: Commits

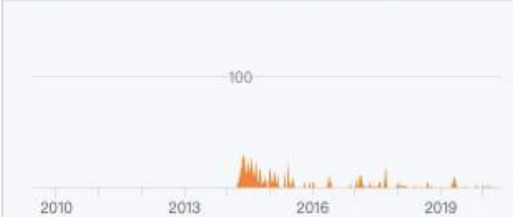
Contributions to master, excluding merge commits



#1 **tj** 3,527 commits 99,315 ++ 88,420 --



#2 **dougwilson** 987 commits 16,206 ++ 5,810 --



Why Go+

- 正面：为什么这么多牛人会关注 Go+？
- 反面：好像很难搞的样子，为什么要干？
- 反面：Python 这么强，怎么打得过？

Data Science 的发展

- 从前

- Limited Domains (有限领域): 比如 BI (Business Intelligence)
- Limited Data (有限数据规模): 比如 Excel、Matlab

- 未来

- Full Domains (全领域): 智能应用 (Intelligent Application)
 - 典型代表: 抖音、快手
- Big Data (大规模数据)
- Any Where (随处): 云 (Cloud)、智能手机 (Small Phone)、嵌入式设备 (IoT)

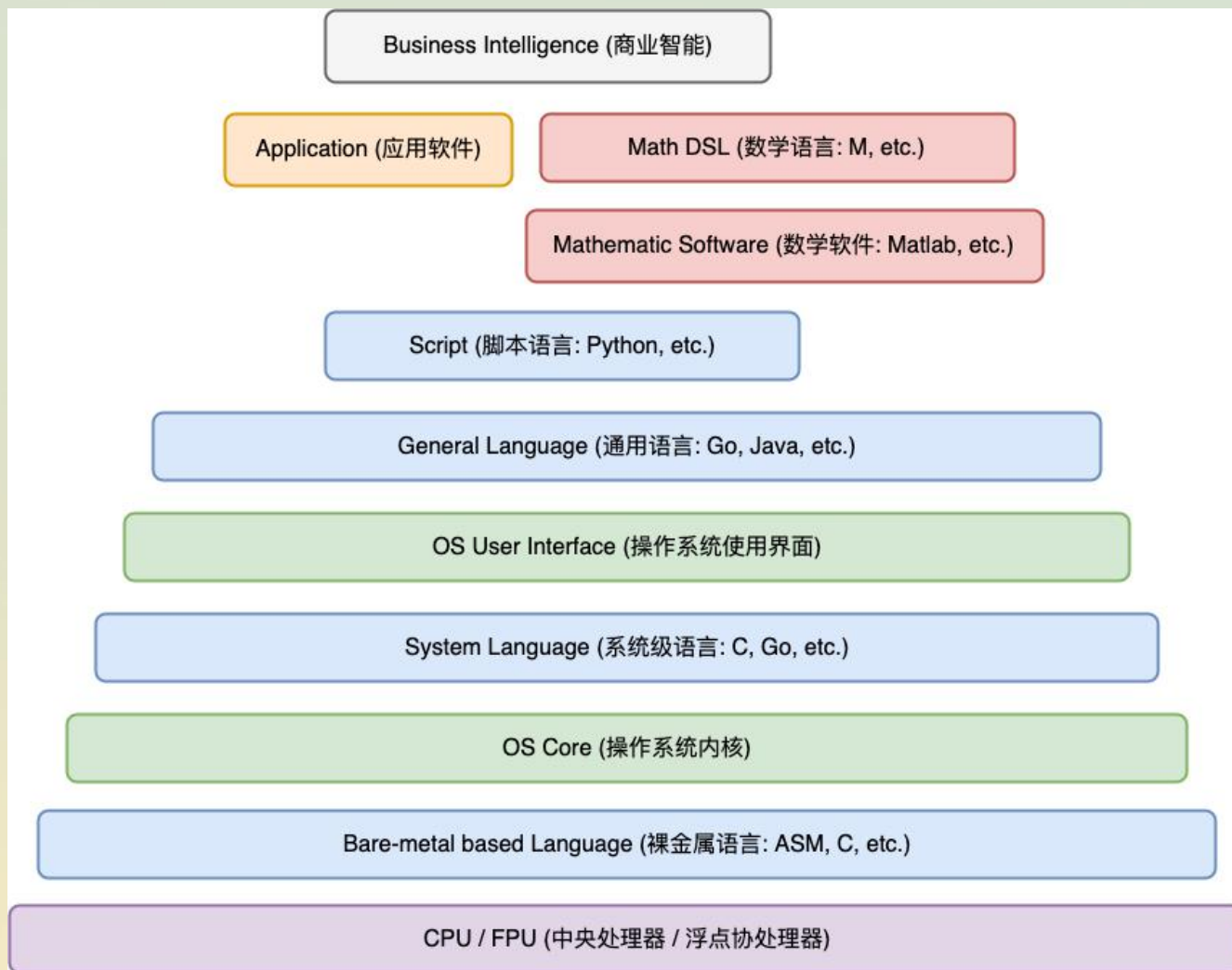
信息科技发展的自然结果

- 应用极大化地丰富（充分竞争）
 - 差异化竞争让应用越来越“聪明”— Intelligent Application
- 数字化信息（数据）极大化地产生

这就是 DT 时代

- IT => DT
- 数据地位的变化
 - Businesses Intelligence => Intelligent Application
 - 数据是副产品 => 数据是原材料（石油），无处不在，深植于业务流
- Data Science 的地位变化
 - 这意味着，Data Science 将基础设施化
 - 数学软件（Application）=> 基础设施（Infrastructure）

数据科学领域 技术栈



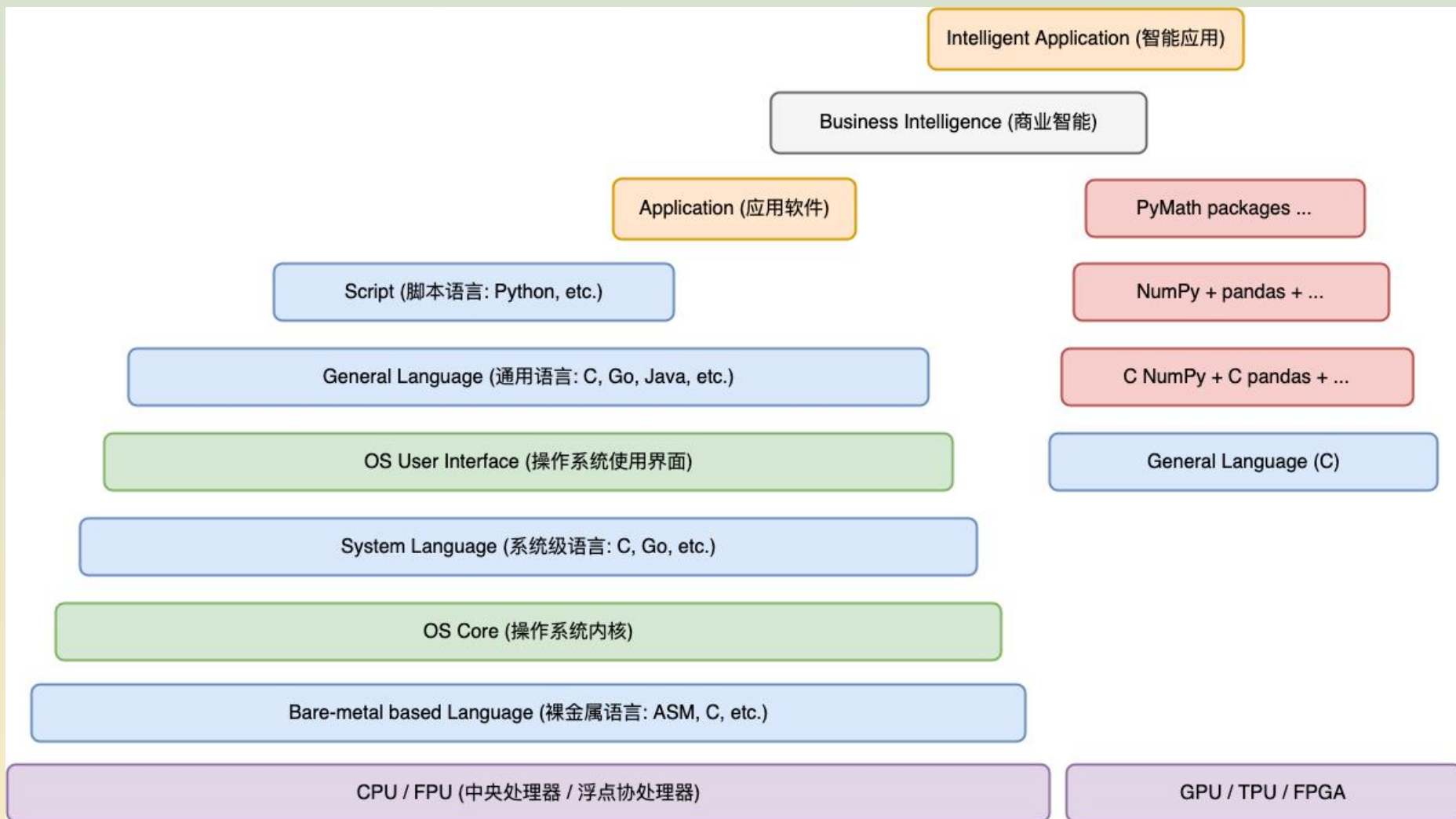
变化难度

- 经典的汉诺塔问题（假设只改变一个依赖）
 - $f(n) = 2 * f(n-1) + 1$
 - $f(1) = 1$
- 改变技术依赖栈是极其困难的

数据科学的汉诺塔第一层迁移

- 数学软件 Python 化
 - 数学软件的平民化（与脚本语言结合）
- 汉诺塔定则：只能一层层迁移
 - 技术栈迁移的步子迈得越大，越不可完成！

技术依赖栈



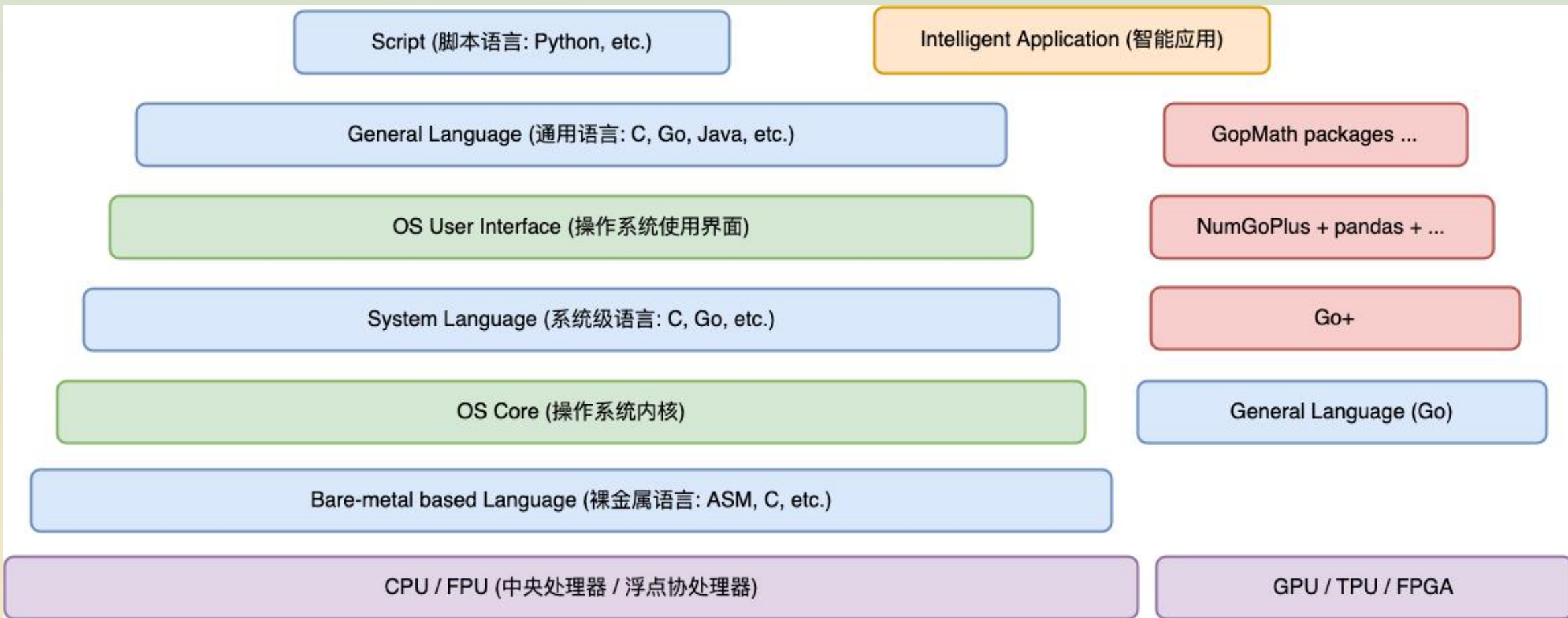
Python 不会是 Data Science 的终局

- 因为，Python 成不了基础设施（Infrastructure）。
- Data Science 本质上是算力革命，是计算密集型的业务。
- Data Science 进一步下沉，终局会是什么？

Go+ for Data Science

- 终局是：数学软件与通用语言的融合
 - 完成 Data Science 基础设施化
- 所以 Go+ 来了！

汉诺塔的再一次迁移



大纲

- Go+ 发展历程
- Go+ 特性
- Go+ 实现解析

Go+ 特性

- 静态语言
- 与Go完全兼容
- 语法简洁(for data science)

Go+ 与 Go的互操作

- Go+ 将支持所有Go的feature(已经支持 基本语法, 流程控制、结构体方法、defer, goroutine, channel等)
- 所有Go package均可以被Go+ import
- 所有Go+的package均可以被convert成Go package, 并被Go来import

Go+ 双引擎

- Bytecode backend
- Go code generation

Go+ 示例

For example, the following is legal Go+ source code:

```
a := [1, 2, 3.4]
println(a)
```

How do we do this in the Go language?

```
package main

func main() {
    a := []float64{1, 2, 3.4}
    println(a)
}
```


运行方式

- Bytecode backend
- Go code generation

```
$ gop go var_and_op.gop
```

```
$ cat var_and_op.gop
#!/usr/bin/env gop run
x := 123.1 - 3i
y, z := "Hello, ", 123
println(y+"complex:", x+1, "int:", z)

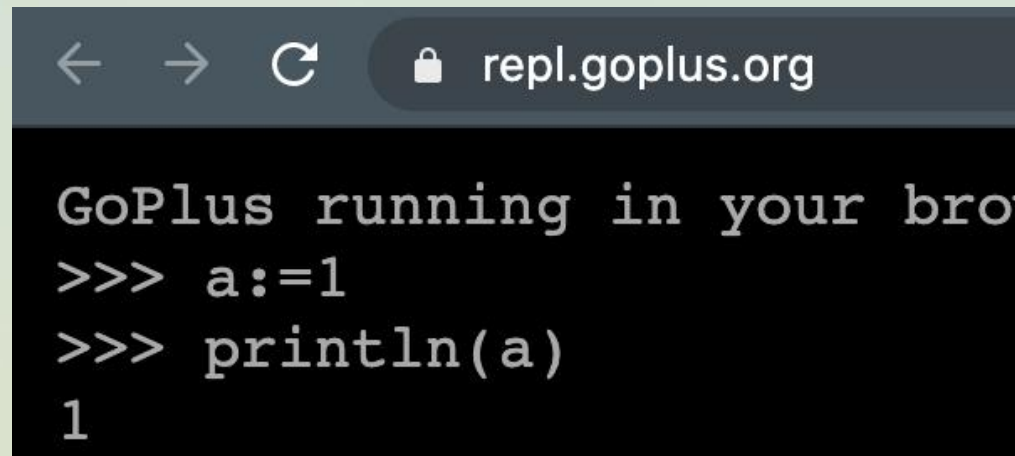
# [redacted] in ~/goplus/tutorial/02-Var-
$ gop run var_and_op.gop
Hello, complex: (124.1-3i) int: 123

# [redacted] in ~/goplus/tutorial/02-Var-
$ ./var_and_op.gop
Hello, complex: (124.1-3i) int: 123
```

Go+ 相关

- Repl
- Go+ playground
- Go+ vscode plugin

```
$ gop repl
Welcome to Go+ console!
>>> a:=1
>>> println(a)
1
```



```
repl.goplus.org
GoPlus running in your bro
>>> a:=1
>>> println(a)
1
```

```
$ gop
Gop is a tool for managing Go+ source code.
```

Usage:

```
gop <command> [arguments]
```

The commands are:

run	Run a Go+ program
go	Convert Go+ packages into Go packages
fmt	Format Go+ packages
export	Export Go packages for Go+ programs
repl	Play Go+ in console



```
play.goplus.org
The Go+ Playground Run Format Share Hello, Go+ Star
1 println("Hello, Go+")
2
3 println(1r << 129)
4 println(1/3r + 2/7r*2)
5
6 arr := [1, 3, 5, 7, 11, 13, 17, 19]
7 println(arr)
8 println([x*x for x <- arr, x > 3])
9
10 m := {"Hi": 1, "Go+": 2}
11 println(m)
12 println({v: k for k, v <- m})
13 println([k for k, _ <- m])
14 println([v for v <- m])
15
```

Go+ 特有语法

- 有理数

```
a := 1r << 65 // bigint, large than int64
b := 4/5r      // bigrat
c := b - 1/3r + 3 * 1/2r
println(a, b, c)
```

- Map

```
x := {"Hello": 1, "xsw": 3.4} // map[string]float64
y := {"Hello": 1, "xsw": "Go+"} // map[string]interface{}
z := {"Hello": 1, "xsw": 3} // map[string]int
empty := {} // map[string]interface{}
```

- Slice

```
x := [1, 3.4] // []float64
y := [1] // []int
z := [1+2i, "xsw"] // []interface{}
a := [1, 3.4, 3+4i] // []complex128
b := [5+6i] // []complex128
c := ["xsw", 3] // []interface{}
empty := [] // []interface{}
```

Go+ 特有语法

- List/map comprehension

```
a := [x*x for x <- [1, 3, 5, 7, 11]]
b := [x*x for x <- [1, 3, 5, 7, 11], x > 3]
c := [i+v for i, v <- [1, 3, 5, 7, 11], i%2 == 1]
d := [k+", "+s for k, s <- {"Hello": "xsw", "Hi": "Go+"}]

arr := [1, 2, 3, 4, 5, 6]
e := [[a, b] for a <- arr, a < b for b <- arr, b > 2]

x := {x: i for i, x <- [1, 3, 5, 7, 11]}
y := {x: i for i, x <- [1, 3, 5, 7, 11], i%2 == 1}
z := {v: k for k, v <- {1: "Hello", 3: "Hi", 5: "xsw", 7: "Go+"}, k > 3}
```

- For range

```
sum := 0
for x <- [1, 3, 5, 7, 11, 13, 17], x > 3 {
    sum += x
}
```

Go+ 特有语法

Error handling

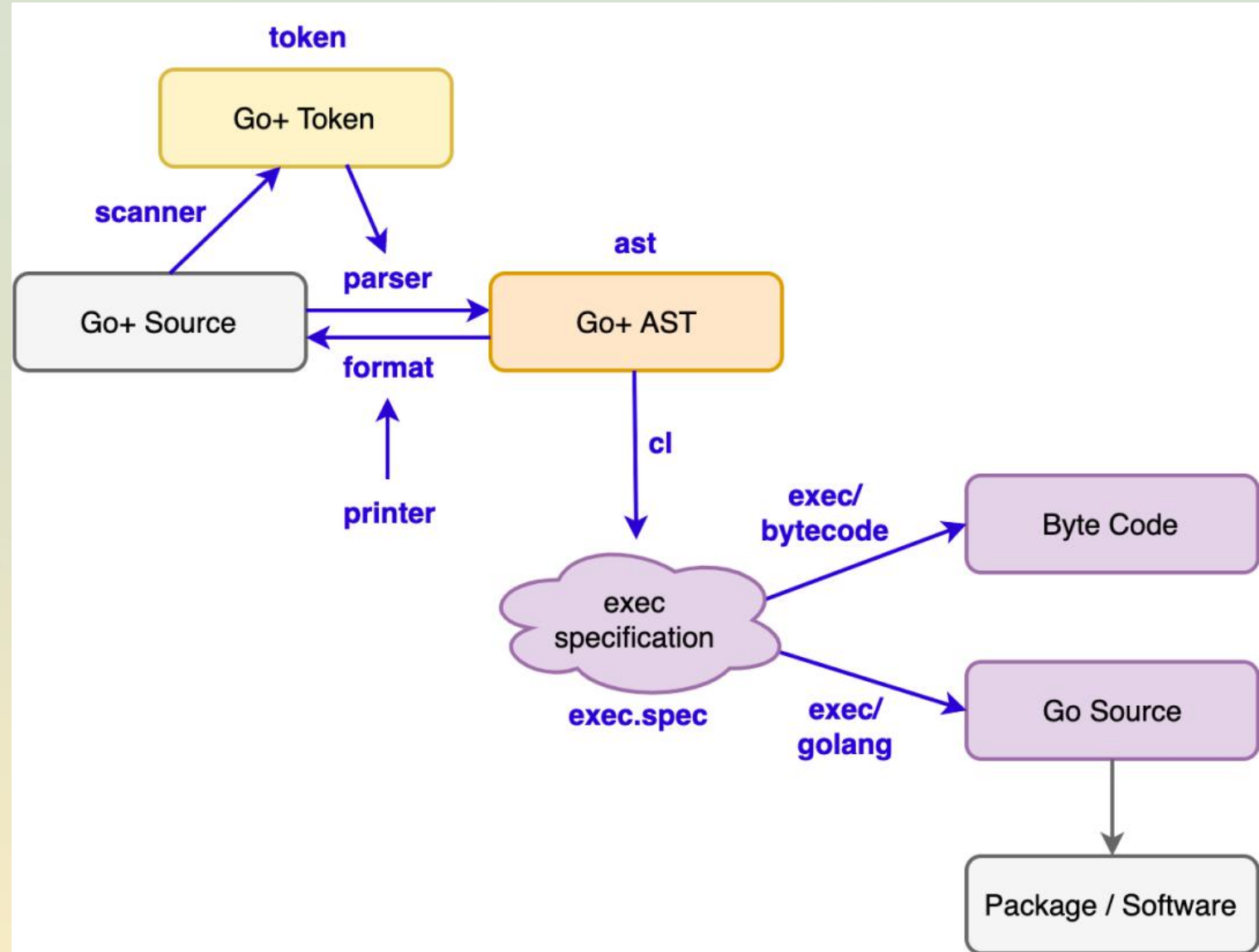
```
expr! // panic if err  
expr? // return if err  
expr?:defval // use defval if err
```

```
import (  
    "strconv"  
)  
  
func add(x, y string) (int, error) {  
    return strconv.Atoi(x)? + strconv.Atoi(y)?, nil  
}  
  
func addSafe(x, y string) int {  
    return strconv.Atoi(x)?:0 + strconv.Atoi(y?):0  
}  
  
println(`add("100", "23"):`, add("100", "23")!)  
// add("100", "23"): 123  
  
println(`addSafe("10", "abc"):`, addSafe("10", "abc"))  
// addSafe("10", "abc"): 10
```

大纲

- Go+ 发展历程
- Go+ 特性
- Go+ 实现解析

Go+ 运行逻辑



Go+ 目录结构

- github.com/goplus/gop/token
- github.com/goplus/gop/scanner
- github.com/goplus/gop/parser
- github.com/goplus/gop/ast
- github.com/goplus/gop/format
- github.com/goplus/gop/printer
- github.com/goplus/gop/cl
- github.com/goplus/gop/exec.spec
- github.com/goplus/gop/exec/bytecode
- github.com/goplus/gop/exec/golang

Go+ Token

- 词法分析：将Go+ 原文件转换成Token序列
- 主要实现：
github.com/goplus/gop/scanner
- Go+ 独有token：
 - Error handling: ?
 - 有理数: r

```
1 | y, z := "Hello, ", 123
2 | println("string:", y, "int:", z)
```

Token

文件:行号:列号	Token名	Token面值
hello.go:2:1	IDENT	"y"
hello.go:2:2	,	""
hello.go:2:4	IDENT	"z"
hello.go:2:6	:=	""
hello.go:2:9	STRING	"\"Hello, \""
hello.go:2:18	,	""
hello.go:2:20	INT	"123"
hello.go:2:23	;	"\n"
hello.go:3:1	IDENT	"println"
hello.go:3:8	(""
hello.go:3:9	STRING	"\"string:\""
hello.go:3:18	,	""
hello.go:3:20	IDENT	"y"
hello.go:3:21	,	""
hello.go:3:23	STRING	"\"int:\""
hello.go:3:29	,	""
hello.go:3:31	IDENT	"z"
hello.go:3:32)	""
hello.go:3:33	;	"\n"

Go+ AST

- 语法分析: 将Token序列转换为抽象语法树(AST)
- 主要实现:
github.com/goplus/gop/parser
- Go+和Go的抽象语法树的区别?

```
1 | y, z := "Hello, ", 123
2 | println("string:", y, "int:", z)
```

```
24 . . . Body: *ast.BlockStmt {
25 . . . . Lbrace: 1:26
26 . . . . List: []ast.Stmt (len = 2) {
27 . . . . . 0: *ast.AssignStmt {
28 . . . . . . Lhs: []ast.Expr (len = 2) {
29 . . . . . . . 0: *ast.Ident {
30 . . . . . . . . NamePos: 1:27
31 . . . . . . . . Name: "y"
32 . . . . . . . . Obj: *ast.Object {
33 . . . . . . . . . Kind: var
34 . . . . . . . . . Name: "y"
35 . . . . . . . . . Decl: *(obj @ 27)
36 . . . . . . . . }
37 . . . . . . . }
38 . . . . . . . 1: *ast.Ident {
39 . . . . . . . . NamePos: 1:30
40 . . . . . . . . Name: "z"
41 . . . . . . . . Obj: *ast.Object {
42 . . . . . . . . . Kind: var
43 . . . . . . . . . Name: "z"
44 . . . . . . . . . Decl: *(obj @ 27)
45 . . . . . . . . }
46 . . . . . . . }
47 . . . . . . }
48 . . . . . TokPos: 1:32
49 . . . . . Tok: :=
50 . . . . . Rhs: []ast.Expr (len = 2) {
51 . . . . . . 0: *ast.BasicLit {
52 . . . . . . . ValuePos: 1:35
53 . . . . . . . Kind: STRING
54 . . . . . . . Value: "\"Hello, \""
55 . . . . . . }
56 . . . . . . 1: *ast.BasicLit {
57 . . . . . . . ValuePos: 1:46
58 . . . . . . . Kind: INT
```

Go+ AST

- Go+ 独有语法
- 需要Go+的AST实现

```
1 | x := [1, 3.4]
2 | println(x)
```

```
24 . . . Body: *ast.BlockStmt {
25 . . . . Lbrace: 1:26
26 . . . . List: []ast.Stmt (len = 2) {
27 . . . . . 0: *ast.AssignStmt {
28 . . . . . . Lhs: []ast.Expr (len = 1) {
29 . . . . . . . 0: *ast.Ident {
30 . . . . . . . . NamePos: 1:27
31 . . . . . . . . Name: "x"
32 . . . . . . . . Obj: *ast.Object {
33 . . . . . . . . . Kind: var
34 . . . . . . . . . Name: "x"
35 . . . . . . . . . Decl: *(obj @ 27)
36 . . . . . . . . }
37 . . . . . . . . }
38 . . . . . . . . }
39 . . . . . TokPos: 1:29
40 . . . . . Tok: :=
41 . . . . . Rhs: []ast.Expr (len = 1) {
42 . . . . . . 0: *ast.SliceLit {
43 . . . . . . . Lbrack: 1:32
44 . . . . . . . Elts: []ast.Expr (len = 2) {
45 . . . . . . . . 0: *ast.BasicLit {
46 . . . . . . . . . ValuePos: 1:33
47 . . . . . . . . . Kind: INT
48 . . . . . . . . . Value: "1"
49 . . . . . . . . . }
50 . . . . . . . . 1: *ast.BasicLit {
51 . . . . . . . . . ValuePos: 1:36
52 . . . . . . . . . Kind: FLOAT
53 . . . . . . . . . Value: "3.4"
54 . . . . . . . . . }
55 . . . . . . . . }
56 . . . . . . . Rbrack: 1:39
57 . . . . . . . Incomplete: false
58 . . . . . . }
59 . . . . . }
```

Go+ AST

- Go+ SliceLit
- 抽象语法树实现

```
1 | x := [1, 3.4]
2 | println(x)
```

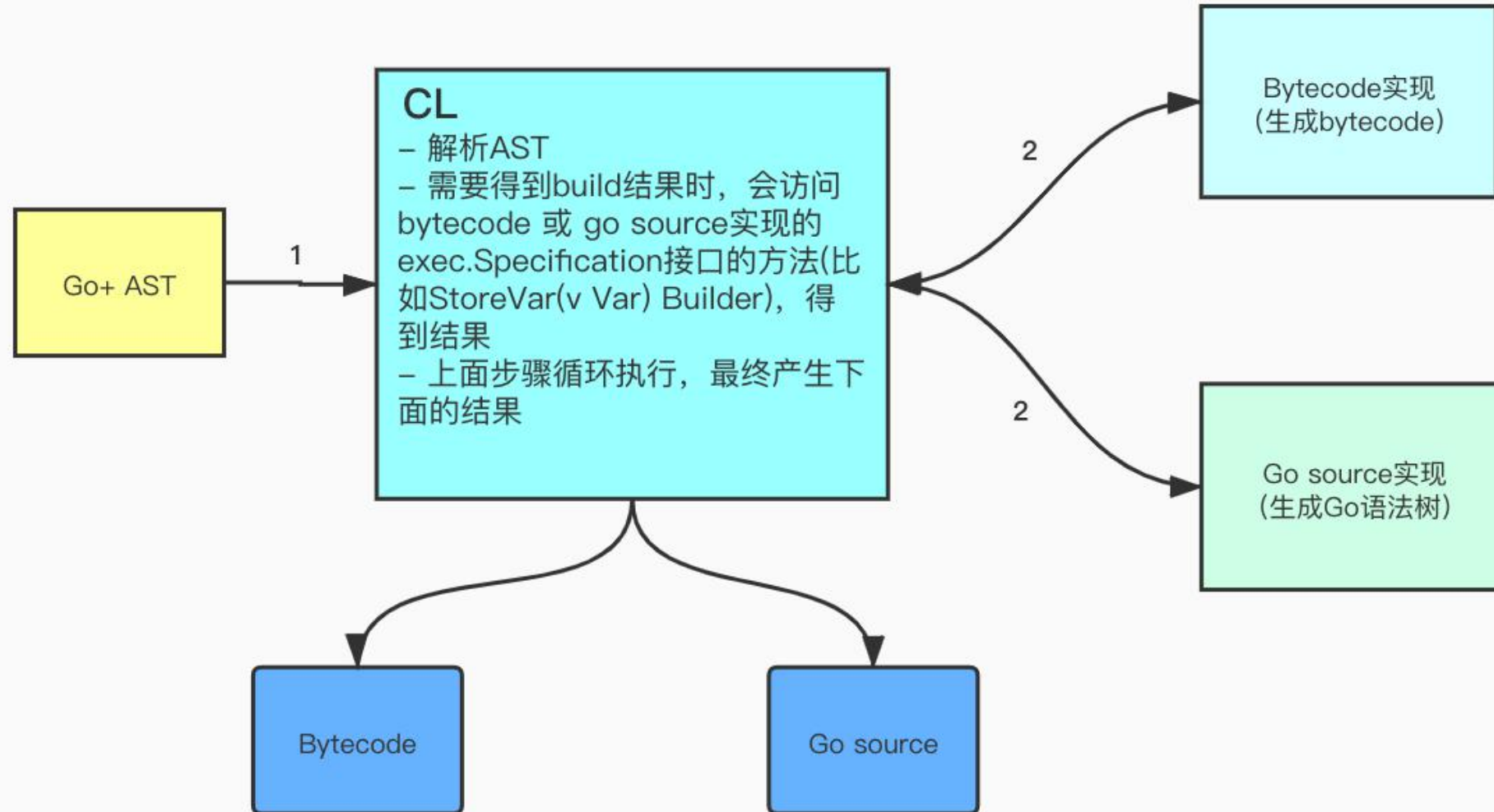
```
func (p *parser) parseArrayType() ast.Expr {
    669 669
    if p.trace {
        670 670
        defer un(trace(p, "ArrayType"))
        671 671
    }
    672 672
    lbrack := p.expect(token.LBRACK)
    673 673
    p.exprLev++
    674 674
    var len ast.Expr
    675 675
    // always permit ellipsis for more fault-tolerant parsing
    676 676
    if p.tok == token.ELLIPSIS {
        677 677
        len = &ast.Ellipsis{Ellipsis: p.pos}
        678 678
        p.next()
        679 679
    } else if p.tok != token.RBRACK {
        680 680
        len = p.parseRHS()
        681 681
    }
    682 682
    p.exprLev--
    683 683
    p.expect(token.RBRACK)
    684 684
    elt := p.parseType()
    685 685
    return &ast.ArrayType{Lbrack: lbrack, Len: len, Elt: elt}
    686 686
}

func (p *parser) makeIdentList(list []ast.Expr) []ast.Ident {
    687 687
    688 688
    689 689
    690 690
    691 691
    692 692
    693 693
    694 694
    695 695
    696 696
    697 697
    698 698
    699 699
    700 700
    701 701
    702 702
    703 703
    704 704
    705 705
    706 706
    707 707
    708 708
    709 709
    710 710
    711 711
    712 712
    713 713
    714 714
    715 715
    716 716
    717 717
    718 718
    719 719
    720 720
    721 721
    722 722
    723 723
    724 724
    725 725
    726 726
    727 727
    728 728
    729 729
    730 730
    731 731
    732 732
    733 733
    734 734
    735 735
    736 736
    737 737
    738 738
    739 739
    740 740
    741 741
    742 742
    743 743
    744 744
    745 745
    746 746
    747 747
    748 748
    749 749
    750 750
    751 751
    752 752
    753 753
    754 754
    755 755
    756 756
    757 757
    758 758
    759 759
    760 760
    761 761
    762 762
    763 763
    764 764
    765 765
    766 766
    767 767
    768 768
    769 769
    770 770
    771 771
    772 772
    773 773
    774 774
    775 775
    776 776
    777 777
    778 778
    779 779
    780 780
    781 781
    782 782
    783 783
    784 784
    785 785
    786 786
    787 787
    788 788
    789 789
    790 790
    791 791
    792 792
    793 793
    794 794
    795 795
    796 796
    797 797
    798 798
    799 799
    800 800
    801 801
    802 802
    803 803
    804 804
    805 805
    806 806
    807 807
    808 808
    809 809
    810 810
    811 811
    812 812
    813 813
    814 814
    815 815
    816 816
    817 817
    818 818
    819 819
    820 820
    821 821
    822 822
    823 823
    824 824
    825 825
    826 826
    827 827
    828 828
    829 829
    830 830
    831 831
    832 832
    833 833
    834 834
    835 835
    836 836
    837 837
    838 838
    839 839
    840 840
    841 841
    842 842
    843 843
    844 844
    845 845
    846 846
    847 847
    848 848
    849 849
    850 850
    851 851
    852 852
    853 853
    854 854
    855 855
    856 856
    857 857
    858 858
    859 859
    860 860
    861 861
    862 862
    863 863
    864 864
    865 865
    866 866
    867 867
    868 868
    869 869
    870 870
    871 871
    872 872
    873 873
    874 874
    875 875
    876 876
    877 877
    878 878
    879 879
    880 880
    881 881
    882 882
    883 883
    884 884
    885 885
    886 886
    887 887
    888 888
    889 889
    890 890
    891 891
    892 892
    893 893
    894 894
    895 895
    896 896
    897 897
    898 898
    899 899
    900 900
    901 901
    902 902
    903 903
    904 904
    905 905
    906 906
    907 907
    908 908
    909 909
    910 910
    911 911
    912 912
    913 913
    914 914
    915 915
    916 916
    917 917
    918 918
    919 919
    920 920
    921 921
    922 922
    923 923
    924 924
    925 925
    926 926
    927 927
    928 928
    929 929
    930 930
    931 931
    932 932
    933 933
    934 934
    935 935
    936 936
    937 937
    938 938
    939 939
    940 940
    941 941
    942 942
    943 943
    944 944
    945 945
    946 946
    947 947
    948 948
    949 949
    950 950
    951 951
    952 952
    953 953
    954 954
    955 955
    956 956
    957 957
    958 958
    959 959
    960 960
    961 961
    962 962
    963 963
    964 964
    965 965
    966 966
    967 967
    968 968
    969 969
    970 970
    971 971
    972 972
    973 973
    974 974
    975 975
    976 976
    977 977
    978 978
    979 979
    980 980
    981 981
    982 982
    983 983
    984 984
    985 985
    986 986
    987 987
    988 988
    989 989
    990 990
    991 991
    992 992
    993 993
    994 994
    995 995
    996 996
    997 997
    998 998
    999 999
    1000 1000
}

func (p *parser) parseArrayTypeOrSliceLit(allowSliceLit bool) (expr ast.Expr, isSliceLit bool) {
    669 669
    if p.trace {
        670 670
        defer un(trace(p, "ArrayType"))
        671 671
    }
    672 672
    lbrack := p.expect(token.LBRACK)
    673 673
    p.exprLev++
    674 674
    var len ast.Expr
    675 675
    // always permit ellipsis for more fault-tolerant parsing
    676 676
    if p.tok == token.ELLIPSIS {
        677 677
        len = &ast.Ellipsis{Ellipsis: p.pos}
        678 678
        p.next()
        679 679
    } else if p.tok != token.RBRACK {
        680 680
        len = p.parseRHS()
        681 681
    }
    682 682
    if allowSliceLit && p.tok == token.COMMA { // [a, b, c, d ...]
        683 683
        elts := p.parseSliceLit(lbrack, len)
        684 684
        p.exprLev--
        685 685
        return elts, true
        686 686
    }
    687 687
    rbrack := p.expect(token.RBRACK)
    688 688
    var elt ast.Expr
    689 689
    if allowSliceLit {
        690 690
        elt = p.tryType()
        691 691
        if elt == nil { // [a]
            692 692
            return newSliceLit(lbrack, rbrack, len), true
            693 693
        }
        694 694
    } else {
        695 695
        elt = p.parseType()
        696 696
    }
    697 697
    if p.trace {
        698 698
        log.Debug("parseArrayType:", len, "elt:", elt)
        699 699
    }
    700 700
}
```

Go+ 编译和双擎

- Go+双擎:
 - Bytecode & Go Source
- Executing Specification, 定义双擎输出编译结果的接口, 双擎独立实现各自的编译输出行为。



Go+ ByteCode

- 编译
针对Go+ AST编译生成bytecode
编译代码：
 - 发现是ast.AssignStmt
 - 拿2个右值ast.Rhs, 分别压栈
 - 轮询左值ast.Lhs, 将右值出栈, 校验
 - 生成byte code

```
1 y, z := "Hello, ", 123
2 println("string:", y, "int:", z)
```

bytecode

右移26bit为操作指令, 低位记录数据偏移等

```
11000000000000000000000000000000000001
110000000000000000000000000000000000001111011
1000000000000000000000000000000000000000
1000000000000000000000000000000000000000000000000001
110000000000000000000000000000000000000000000000000100
111100000000000000000000000000000000000000000000000001
11000000000000000000000000000000000000000000000000000101
1111000000000000000000000000000000000000000000000000000
1000000000100000000000000000000000000000000000000000011
1100011111111111111111111111111111111111111111111111111
```

Go+ ByteCode

- 执行阶段，读取bytecode和数据进行操作

```
1 | y, z := "Hello, ", 123
2 | println("string:", y, "int:", z)
```

bytecode执行过程

bytecode	操作指令	数据栈
11000000000000000000000000000001	pushConstR	[]
11000000000000000000000001111011	pushInt	[Hello,]
10000000000000000000000000000000	storeVar	[Hello, 123]
10000000000000000000000000000001	storeVar	[Hello,]
11000000000000000000000000000100	pushConstR	[]
11110000000000000000000000000001	loadVar	[string:]
11000000000000000000000000000101	pushConstR	[string: Hello,]
11110000000000000000000000000000	loadVar	[string: Hello, int:]
1000000001000000000000000000011	callGoFuncv	[string: Hello, int: 123]

Go+ 生成Go Source

- 编译阶段，将Go+ AST转换为Go AST
 - 轮询Go+ AST语句，将表达式入栈
 - 当需要生成一个Go AST的语句时，出栈数据进行组装
- 运行阶段，从Go AST转换为Go source

Go+ 下一步

- 当前
 - Go+的语法
 - 支持Go的feature
- 未来，专注数据科学领域
 - GoTorch
 - Numgoplus
 - GopMath
 - Pandas
 - ...
- 把冷板凳捂热，让Go+助力数据科学



GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

Thanks

<https://github.com/goplus/gop>

