



*Simple is better than complex.*

---

# GO: A VISÃO DE UM PYTHONISTA

---

*Minha opinião pessoal sobre Go e seu lugar diante das necessidades atuais de desenvolvimento.*



ThoughtWorks®

# LUCIANO RAMALHO

---

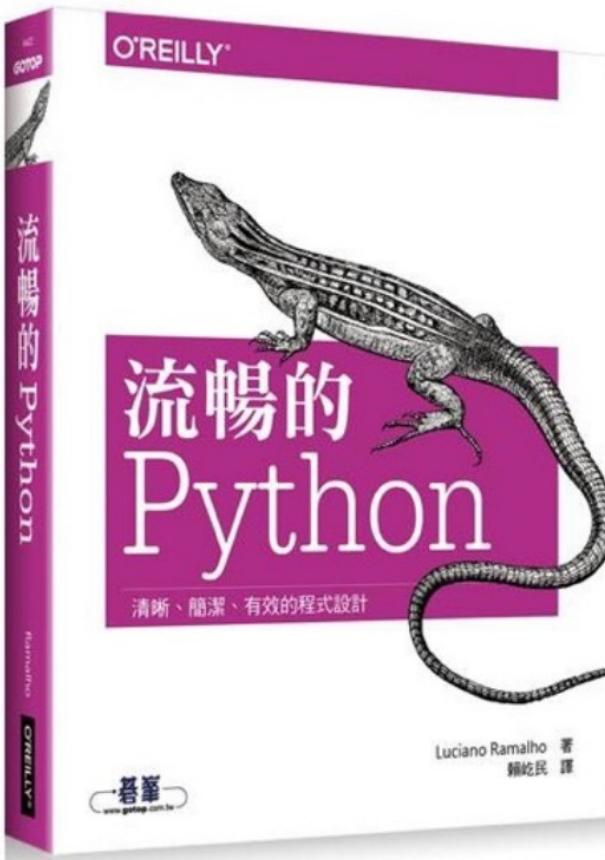
*Technical Principal*

---

@ramalhoorg  
[luciano.ramalho@thoughtworks.com](mailto:luciano.ramalho@thoughtworks.com)

# FLUENT PYTHON, MEU PRIMEIRO LIVRO

---



**Fluent Python** (O'Reilly, 2015)

**Python Fluente** (Novatec, 2015)

**Python к вершинам**

**мастерства\*** (DMK, 2015)

**流暢的 Python<sup>†</sup>** (Gotop, 2016)

also in **Polish, Korean...**

\* *Python. To the heights of excellence*

† *Smooth Python*

**ThoughtWorks®**

# **MINHA JORNADA**

---

*Codando desde 1979*

# JÁ CODEI PROFISSIONALMENTE EM...

---

- BASIC
- xBase
- **Turbo Pascal**
- C++
- Hypertalk e Lingo
- Visual Basic
- **Delphi**
- Perl
- JavaScript (Netscape Enterprise Server)
- PHP
- Java
- Python
- JavaScript (Node)
- Ruby

# JÁ ESTUDEI OU FIZ HACKS COM...

---

- TI-58/59, HP-25, HP-11c
- Assembly Z80, 6502, 8088
- Smalltalk
- Eiffel
- Tcl/Tk
- Scheme (um Lisp)
- C
- Elixir
- **Go**

# PERSPECTIVA ATUAL

---

- Desde 2013 ativamente buscando reforçar minhas habilidades em **outras linguagens**. (pausa para escrever *Fluent Python*)
- Contratado pela **ThoughtWorks**, uma empresa de devs **poliglotas**
- Estudando arquitetura de **sistemas distribuídos** para construir **plataformas**
- Foco atual: **Go, Elixir**
- Na fila: Erlang, Elm, Clojure, Rust

**ThoughtWorks®**

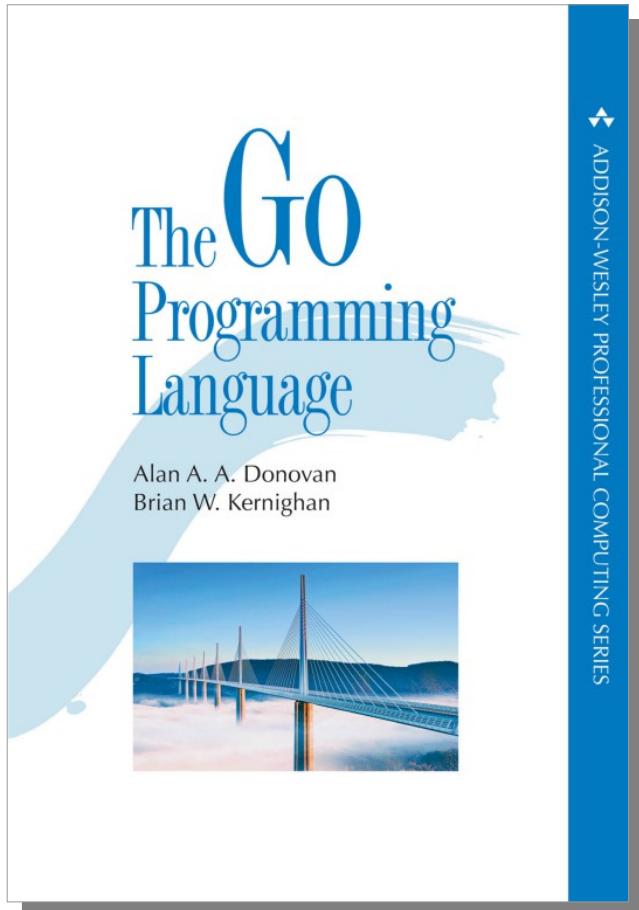
**GO**

---

*Pascal para o século 21*

# A “BÍBLIA”

---

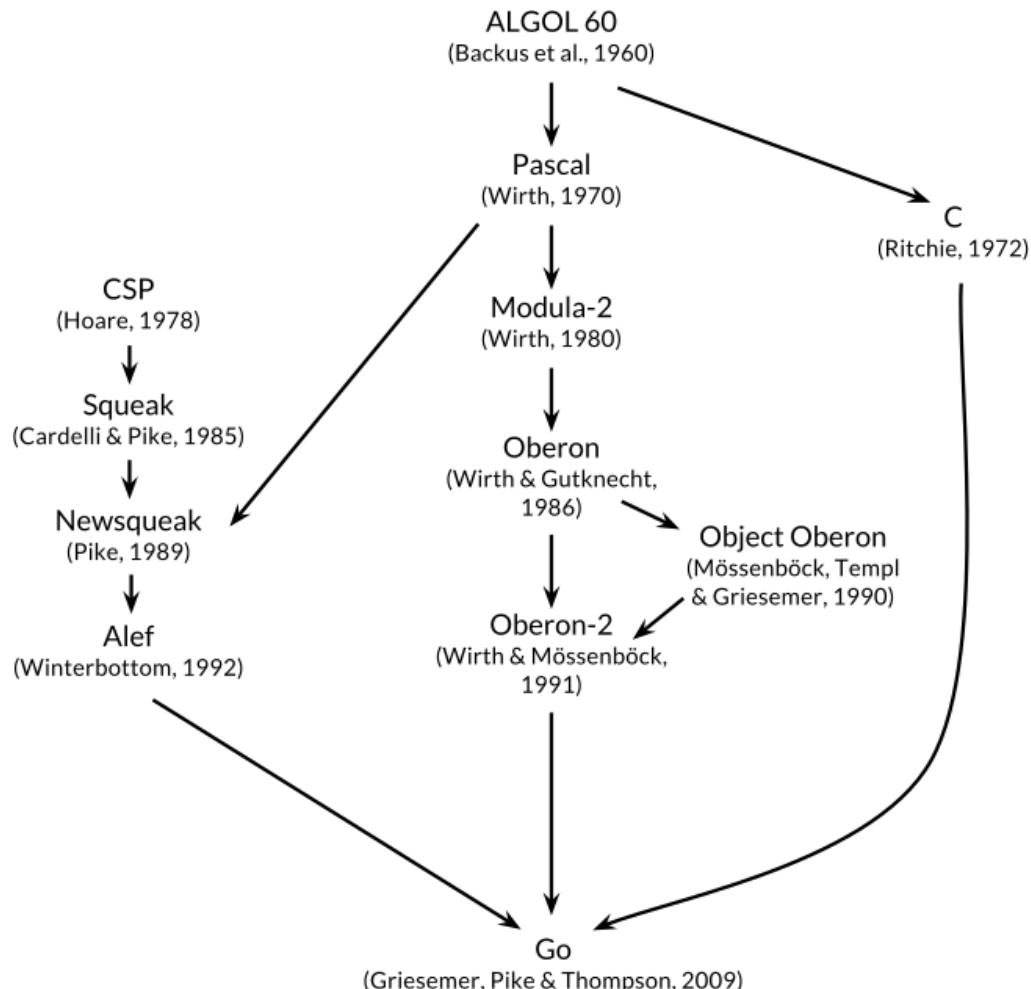


“GOPL”, de Alan  
Donovan e Brian  
Kernighan (o K de K&R)

Completo, preciso e com  
exemplos excelentes

No Brasil: *A Linguagem  
de Programação Go*  
(Novatec, 2017)

# ORIGENS DE GO\*



\* Figura do GOPL

# O SISTEMA DE TIPOS DE GO\*

---

Go tem um sistema de tipos suficiente para evitar a maioria dos erros por descuido que aflige programadores de linguagens dinâmicas, porém seu sistema é mais simples que os de linguagens tipadas comparáveis a ela. Essa abordagem, às vezes, pode resultar em regiões isoladas de programação “não tipada” em um framework mais amplo de tipos, e programadores que usam Go não chegam ao ponto de fazer o que programadores de C++ e Haskell fazem para expressar propriedades de segurança como provas baseadas em tipos. Na prática, porém, Go oferece aos programadores uma boa dose de proteção e de vantagens quanto ao desempenho em tempo de execução, resultantes de um sistema de tipagem relativamente forte sem o peso de um sistema complexo.

# PRINCIPAIS CARACTERÍSTICAS DE GO

---

- Compila direto para **binários nativos** otimizados: gera executáveis de alto desempenho que funcionam sem dependências externas
- Sistema de tipos simples: **structs, métodos** e **interfaces** (sem herança, sobrecarga etc.)
- Suporte a concorrência, paradigma CSP (Communicating Sequential Processes): **gorrotinas, canais** e instrução **select**
- Pilhas incluídas: **biblioteca** e **ferramentas**

# SINTAXE INTELIGENTE

---

- Única instrução de laço: **for** (com a variante **for/range**)
- Instrução **defer**
- Instruções **go** e **select**
- Indicação de público/privado pela letra inicial do identificador (Xxx é público)
- Notação expressiva para literais de tipos compostos diversos
- Sintaxe bastante regular (poucas surpresas) + **go fmt**

# PALAVRAS RESERVADAS EM LINGUAGENS

---

*Table 16-1. Number of keywords in programming languages*

Keywords	Language	Comment
5	Smalltalk-80	Famous for its minimalist syntax.
25	Go	The language, not the game.
32	C	That's ANSI C. C99 has 37 keywords, C11 has 44.
33	Python	Python 2.7 has 31 keywords; Python 1.5 had 28.
41	Ruby	Keywords may be used as identifiers (e.g., <code>class</code> is also a method name).
49	Java	As in C, the names of the primitive types ( <code>char</code> , <code>float</code> , etc.) are reserved.
60	JavaScript	Includes all keywords from Java 1.0, many of which are <b>unused</b> .
65	PHP	Since PHP 5.3, seven keywords were introduced, including <code>goto</code> , <code>trait</code> , and <code>yield</code> .
85	C++	According to <a href="#">cppreference.com</a> , C++11 added 10 keywords to the existing 75.
555	COBOL	I did not make this up. See this <a href="#">IBM ILE COBOL manual</a> .
∞	Scheme	Anyone can define new keywords.

# MAIS CARACTERÍSTICAS DE GO

---

- Tipos compostos embutidos:  
**array, slice, struct, map**
- Suporte a Unicode: tipo **string** é uma sequência de bytes, mas biblioteca padrão lida naturalmente com UTF-8 e o **for/range** itera por caracteres (**rune**) e não por bytes
- Biblioteca padrão excelente para sistemas distribuídos: já suporta HTTP/2!

# COISAS QUE EU SINTO FALTA

---

- Uma forma de reportar erros que preserve o *stack trace*
- Métodos para coleções nativas (ex. **.Contains**)
- Alguma solução para a necessidade de reimplementar tipos básicos como **set**
- Protocolo de iteração (interface **Iterable**) aberto para o usuário da linguagem criar coleções que funcionam com **for/range**

ThoughtWorks®

# SHOW ME THE CODE

*Comparando um exemplo em Python e Go*



# UMA COMPARAÇÃO

Python: 73 linhas  
Go: 138 linhas

## Código fonte:

<https://github.com/ramalho/go-pythonista>

# DEMO: BUSCA DE UNICODE POR NOME

---

```
LAlramalho:~ lramalho$ sinais umbrella
U+2602 伞 UMBRELLA
U+2614 ☔ UMBRELLA WITH RAIN DROPS
U+26F1 ☂ UMBRELLA ON GROUND
[U+1F302 🌂 CLOSED UMBRELLA
[U+1F3D6 🏖 BEACH WITH UMBRELLA
LAlramalho:~ lramalho$ sinais forty two
U+32B7 ㊲ CIRCLED NUMBER FORTY TWO
LAlramalho:~ lramalho$ sinais heart eyes
[U+1F60D 😍 SMILING FACE WITH HEART-SHAPED EYES
U+1F63B 😚 SMILING CAT FACE WITH HEART-SHAPED EYES
LAlramalho:~ lramalho$ sinais ellipsis
U+0EAF ፻ LAO ELLIPSIS
U+1801 ⠃ MONGOLIAN ELLIPSIS
U+2026 ... HORIZONTAL ELLIPSIS
U+22EE : VERTICAL ELLIPSIS
U+22EF ⁊ MIDLINE HORIZONTAL ELLIPSIS
U+22F0 ⁊ UP RIGHT DIAGONAL ELLIPSIS
U+22F1 ⁊ DOWN RIGHT DIAGONAL ELLIPSIS
U+FE19 ⁊ PRESENTATION FORM FOR VERTICAL HORIZONTAL ELLIPSIS
LAlramalho:~ lramalho$
```

# IMPORTS E CONSTANTES

sinais.py

```
1 #!/usr/bin/env python3
2
3 import os, sys
4 import urllib.request
5 import threading
6
7 # Dados ficam em http://www.unicode.org/Public/UNIDATA/Un
8 # mas unicode.org não é confiável, então esta URL alternativa
9 # http://turing.com.br/etc/UnicodeData.txt
10 URL_UCD = "http://turing.com.br/etc/UnicodeData.txt"
11
12
13 def analisar_linha(linha):
14     campos = linha.split(';')
15     código = int(campos[0], 16)
16     nome = campos[1]
17     palavras = set(nome.replace('-', ' ').split())
18     if campos[10]:
19         nome = '{} ({})'.format(nome, campos[10])
20         palavras.update(campos[10].replace('-', ' ').split())
21     return chr(código), nome, palavras
22
23
24 def listar(texto, consulta):
25     consulta = set(consulta.replace('-', ' ').split())
26     for linha in texto:
27         runa, nome, palavras_nome = analisar_linha(linha)
28         if consulta <= palavras_nome:
29             try:
30                 print('U+{:04X}\t{}\t{}'.format(ord(runa),
```

sinais.go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "io"
7     "log"
8     "net/http"
9     "os"
10    "os/user"
11    "strconv"
12    "strings"
13    "time"
14 )
15
16
17 // Dados ficam em http://www.unicode.org/Public/UNIDATA/Un
18 // mas unicode.org não é confiável, então esta URL alternativa
19 // http://turing.com.br/etc/UnicodeData.txt
20 const URLUCD = "http://turing.com.br/etc/UnicodeData.txt"
21
22 // AnalisarLinha devolve a runa, o nome e uma fatia de palavras que
23 // ocorrem no campo nome de uma linha do UnicodeData.txt
24 func AnalisarLinha(linha string) (rune, string, []string) {
25     campos := strings.Split(linha, ";")
26     código, _ := strconv.ParseInt(campos[0], 16, 32)
27     nome := campos[1]
28     palavras := separar(campos[1])
29     if campos[10] != "" { // se houver
30         nome += fmt.Sprintf(" (%s)", campos[10])
```

# FUNÇÃO PRINCIPAL

```
sinais.py                                     sinais.go
48 def baixar_UCD(url, caminho, feito):          116 arquivo, err := os.Create(caminho)
49     with urllib.request.urlopen(url) as resposta: 117 fecharSe(err)
50         octetos = resposta.read()                118 defer arquivo.Close()
51     with open(caminho, 'wb') as arquivo:           119 _, err = io.Copy(arquivo, resposta.Body)
52         arquivo.write(octetos)                   120 fecharSe(err)
53     feito.set()                                121 feito <- true
54
55
56 def abrir_UCD(caminho):                         122
57     try:                                         123 func abrirUCD(caminho string) (*os.File, error) {
58         ucd = open(caminho)                      124     ucd, err := os.Open(caminho)
59     except FileNotFoundError:                  125     if os.IsNotExist(err) {
60         print('%s não encontrado\nbaixando %s' % (caminho, 126         fmt.Printf("%s não encontrado\nbaixando %s\n", caminho,
61         feito = threading.Event()                 127         feito := make(chan bool)
62         threading.Thread(target=baixar_UCD, args=(URL_UCD, 128         go baixarUCD(URLUCD, caminho, feito)
63             progresso(feito)                   129         progresso(feito)
64             ucd = open(caminho)                 130         ucd, err = os.Open(caminho)
65         return ucd                           131     }
66
67
68 def main():                                       132     return ucd, err
69     with abrir_UCD(obter_caminho_UCD()) as ucd: 133 }
70         consulta = ' '.join(sys.argv[1:])        134
71         listar(ucd, consulta.upper())            135 func main() {
72
73
74 if __name__ == '__main__':                      136     ucd, err := abrirUCD(obterCaminhoUCD())
75     main()                                     137     if err != nil {
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
```

# FUNÇÃO: ABRIR UCD, BAIXAR SE NECESSÁRIO

```
sinais.py
48 def baixar_UCD(url, caminho, feito):
49     with urllib.request.urlopen(url) as resposta:
50         octetos = resposta.read()
51         with open(caminho, 'wb') as arquivo:
52             arquivo.write(octetos)
53         feito.set()
54
55
56 def abrir_UCD(caminho):
57     try:
58         ucd = open(caminho)
59     except FileNotFoundError:
60         print('%s não encontrado\nbaixando %s' % (caminho,
61         feito = threading.Event()
62         threading.Thread(target=baixar_UCD, args=(URL_UCD,
63         progresso(feito)
64         ucd = open(caminho)
65         return ucd
66
67
68 def main():
69     with abrir_UCD(obter_caminho_UCD()) as ucd:
70         consulta = ' '.join(sys.argv[1:])
71         listar(ucd, consulta.upper())
72
73
74 if __name__ == '__main__':
75     main()
76
77
sinais.go
111 func baixarUCD(url, caminho string, feito chan<- bool) {
112     resposta, err := http.Get(url)
113     fecharSe(err)
114     defer resposta.Body.Close()
115     arquivo, err := os.Create(caminho)
116     fecharSe(err)
117     defer arquivo.Close()
118     _, err = io.Copy(arquivo, resposta.Body)
119     fecharSe(err)
120     feito <- true
121 }
122
123 func abrirUCD(caminho string) (*os.File, error) {
124     ucd, err := os.Open(caminho)
125     if os.IsNotExist(err) {
126         fmt.Printf("%s não encontrado\nbaixando %s\n", caminho,
127         feito := make(chan bool)
128         go baixarUCD(URLUCD, caminho, feito)
129         progresso(feito)
130         ucd, err = os.Open(caminho)
131     }
132     return ucd, err
133 }
134
135 func main() {
136     ucd, err := abrirUCD(obterCaminhoUCD())
137     if err != nil {
138         log.Fatal(err.Error())
139     }
140     defer func() { ucd.Close() }()

```

# FUNÇÕES: BAIXAR EM THREAD/GORROTINA

```
sinais.py
35 def obter_caminho_UCD():
36     caminho_UCD = os.environ.get('UCD_PATH')
37     if caminho_UCD is None:
38         caminho_UCD = os.path.join(os.environ['HOME'], "Unicodedata")
39     return caminho_UCD
40
41
42 def progresso(feito):
43     while not feito.wait(.150):
44         print('.', end='', flush=True)
45     print()
46
47
48 def baixar_UCD(url, caminho, feito):
49     with urllib.request.urlopen(url) as resposta:
50         octetos = resposta.read()
51     with open(caminho, 'wb') as arquivo:
52         arquivo.write(octetos)
53     feito.set()
54
55
56 def abrir_UCD(caminho):
57     try:
58         ucd = open(caminho)
59     except FileNotFoundError:
60         print('%s não encontrado\nbaixando %s' % (caminho,
61         feito = threading.Event()
62         threading.Thread(target=baixar_UCD, args=(URL_UCD,
63         progresso(feito)
64         ucd = open(caminho)
```

```
sinais.go
88 func obterCaminhoUCD() string {
89     caminhoUCD := os.Getenv("UCD_PATH")
90     if caminhoUCD == "" {
91         usuário, err := user.Current()
92         fecharSe(err)
93         caminhoUCD = usuário.HomeDir + "/UnicodeData.txt"
94     }
95     return caminhoUCD
96 }
97
98 func progresso(feito <-chan bool) {
99     for {
100         select {
101             case <-feito:
102                 fmt.Println()
103                 return
104             default:
105                 fmt.Print(".")
106                 time.Sleep(150 * time.Millisecond)
107         }
108     }
109 }
110
111 func baixarUCD(url, caminho string, feito chan<- bool) {
112     resposta, err := http.Get(url)
113     fecharSe(err)
114     defer resposta.Body.Close()
115     arquivo, err := os.Create(caminho)
116     fecharSe(err)
117     defer arquivo.Close()
```

# FUNÇÃO: FILTRAR UCD PELA CONSULTA

sinais.py

```
22
23
24 def listar(texto, consulta):
25     consulta = set(consulta.replace('-', ' ').split())
26     for linha in texto:
27         runa, nome, palavras_nome = analisar_linha(linha)
28         if consulta <= palavras_nome:
29             try:
30                 print('U+{:04X}\t{}\t{}'.format(ord(runa),
31             except UnicodeEncodeError:
32                 print('U+{:04X}\t\uFFFD\t{}'.format(ord(ru
33
34
35 def obter_caminho_UCD():
36     caminho_UCD = os.environ.get('UCD_PATH')
37     if caminho_UCD is None:
38         caminho_UCD = os.path.join(os.environ['HOME'], "Ur
39     return caminho_UCD
40
41
42 def progresso(feito):
43     while not feito.wait(.150):
44         print('.', end='', flush=True)
45     print()
46
47
48 def baixar_UCD(url, caminho, feito):
49     with urllib.request.urlopen(url) as resposta:
50         octetos = resposta.read()
51     with open(caminho, 'wb') as arquivo:
```

sinais.go

```
56 // Listar exibe na saída padrão o código, a runa e o nome
57 // cujo nome contém as palavras da consulta.
58 func Listar(texto io.Reader, consulta string) {
59     termos := separar(consulta)
60     varredor := bufio.NewScanner(texto)
61     for varredor.Scan() {
62         linha := varredor.Text()
63         if strings.TrimSpace(linha) == "" {
64             continue
65         }
66         runa, nome, palavrasNome := AnalisarLinha(linha) // C
67         if contémTodos(palavrasNome, termos) {           // E
68             fmt.Printf("U+%04X\t[%1]c\t%s\n", runa, nome)
69         }
70     }
71 }
72
73 func contémTodos(fatia []string, procurados []string) bool {
74     for _, procurado := range procurados {
75         if !contém(fatia, procurado) {
76             return false
77         }
78     }
79     return true
80 }
81
82 func check(err error) {
83     if err != nil {
84         log.Fatal(err.Error())
85     }
```

# FUNÇÃO: ANALISAR UMA LINHA DA UCD

```
sinais.py
```

```
13 def analisar_linha(linha):
14     campos = linha.split(';')
15     código = int(campos[0], 16)
16     nome = campos[1]
17     palavras = set(nome.replace('-', ' ').split())
18     if campos[10]:
19         nome = '{} ({})'.format(nome, campos[10])
20         palavras.update(campos[10].replace('-', ' ').split())
21     return chr(código), nome, palavras
22
23
24 def listar(texto, consulta):
25     consulta = set(consulta.replace('-', ' ').split())
26     for linha in texto:
27         runa, nome, palavras_nome = analisar_linha(linha)
28         if consulta <= palavras_nome:
29             try:
30                 print('U+{:04X}\t{}\t{}'.format(ord(runa),
31             except UnicodeEncodeError:
32                 print('U+{:04X}\t\uFFFD\t{}'.format(ord(ru
33
34
35 def obter_caminho_UCD():
36     caminho_UCD = os.environ.get('UCD_PATH')
37     if caminho_UCD is None:
38         caminho_UCD = os.path.join(os.environ['HOME'], "Ur
39     return caminho_UCD
40
41
42 def progresso(feito):
```

```
sinais.go
```

```
24 func AnalisarLinha(linha string) (rune, string, []string)
25     campos := strings.Split(linha, ";")
26     código, _ := strconv.ParseInt(campos[0], 16, 32)
27     nome := campos[1]
28     palavras := separar(campos[1])
29     if campos[10] != "" { // ❶
30         nome += fmt.Sprintf(" (%s)", campos[10])
31         for _, palavra := range separar(campos[10]) { // ❷
32             if !contém(palavras, palavra) { // ❸
33                 palavras = append(palavras, palavra) // ❹
34             }
35         }
36     }
37     return rune(código), nome, palavras
38 }
39
40 func contém(fatia []string, procurado string) bool {
41     for _, item := range fatia {
42         if item == procurado {
43             return true // ❺
44         }
45     }
46     return false // ❻
47 }
48
49 func separar(s string) []string { // ❻
50     separador := func(c rune) bool { // ❽
51         return c == ' ' || c == '-'
52     }
53     return strings.FieldsFunc(s, separador) // ❾
54 }
```

# CONCLUSÃO

---

*A melhor forma de prever o futuro é construí-lo (Alan Kay)*

# PORQUE GO É UM SUCESSO

---

Uma linguagem que compila para **binários** em código de máquina de **alto desempenho**, mais **segura** e **fácil** de aprender que C/C++.

- Sintaxe legível, razoavelmente expressiva
- Compilação muito rápida
- Concorrência com CSP (canais+gorrotinas)
- Sistema de tipos simples
- Ferramental básico incluído na distribuição
- Biblioteca padrão: “pilhas incluídas”

# GO E O ZEN DO PYTHON

---

```
>>> import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

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

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

# GO E O ZEN DO PYTHON ✓

---

```
>>> import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit. ✓

Simple is better than complex. ✓

Complex is better than complicated. ✓

Flat is better than nested. ✓

Sparse is better than dense. ✓

Readability counts. ✓

Special cases aren't special enough to break the rules. ✓

Although practicality beats purity. ✓

Errors should never pass silently.

Unless explicitly silenced. ✓

In the face of ambiguity, refuse the temptation to guess. ✓

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

Although that way may not be obvious at first unless you're Dutch.

Now is better than never. ✓

Although never is often better than \*right\* now. ✓

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those! ✓

# O FUTURO DE GO

---

Go deve continuar sua trajetória de sucesso e será uma das 10+ do Tiobe Index em 2020\*.

- Crescimento movido por sistemas distribuídos e **estratégia de plataformas**
- Gerenciamento de dependências na próxima versão (**go dep**)
- Go para **Android**: além dos experimentos
- **Generics?!?**

\* Segundo minha bola de cristal no 1º semestre de 2017

# O FUTURO DE GO JÁ CHEGOU

TIOBE Index | TIOBE - The Software Quality Index

https://www.tiobe.com/tiobe-index/

## TIOBE Index for July 2017

**July Headline: Go language at all time high and in the top 10**

The Go programming language continues to rise. This month it is at an all time high and enters the top 10. This is an important landmark for the Go programming language, but it also makes you wonder what's next. Is Go really able to join the big stars in the programming language world and leave languages such as JavaScript and Python behind? We will see. The hipster programming languages Kotlin, Elixir and Hack didn't progress much this month. Kotlin lost 5 positions, Hack lost 6 positions and Elixir is still not in the top 50 losing also 5 positions.

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found [here](#).

Jul 2017	Jul 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.774%	-6.03%
2	2		C	7.321%	-4.92%
3	3		C++	5.576%	-0.73%
4	4		Python	3.543%	-0.62%
5	5		C#	3.518%	-0.40%
6	6		PHP	3.093%	-0.18%
7	8	▲	Visual Basic .NET	3.050%	+0.53%
8	7	▼	JavaScript	2.606%	-0.04%
9	12	▲	Delphi/Object Pascal	2.490%	+0.45%
10	55	▲	Go	2.363%	+2.20%
11	9	▼	Perl	2.334%	-0.09%
12	14	▲	Swift	2.253%	+0.29%

An orange arrow points to the Go entry in the table.

# MUITO GRATO

---

*Fale comigo:*

*Luciano Ramalho*

*Twitter: @ramalhoorg*

*luciano.ramalho@thoughtworks.com*

**Thought**Works®