Capitolo 1

Ready... set... Go!

Dopo questo capitolo saprete:

- che cosa distingue Go;
- visitare il Go Playground;
- visualizzare del testo sullo schermo;
- sperimentare con il testo in qualsiasi linguaggio naturale.

Go è il linguaggio di programmazione più attuale del *cloud computing*. Amazon, Apple, Canonical, Chevron, Disney, Facebook, General Electric, Google, Heroku, Microsoft, Twitch, Verizon e Walmart sono solo alcune delle grandi aziende che lo hanno adottato per progetti assolutamente seri (vedi `thenewstack.io/who-is-the-go-developer/` e `golang.org/wiki/GoUsers`). Molta dell'infrastruttura su cui si basa il Web sta passando a Go, a partire da società come CloudFlare, Cockroach Labs, DigitalOcean, Docker, InfluxData, Iron.io, Let's Encrypt, Light Code Labs, Red Hat CoreOS, SendGrid e organizzazioni come Cloud Native Computing Foundation.

Go opera in modo eccellente nei centri elaborazione dati, ma la sua adozione si estende ben oltre i luoghi di lavoro. Ron Evans e Adrian Zankich hanno creato Gobot (`gobot.io`), una libreria per il controllo di robot hardware. Alan Shreve ha creato lo strumento di sviluppo ngrok (`ngrok.com`), un progetto per l'apprendimento di Go, e poi lo ha trasformato in un'attività a tempo pieno.

La comunità formata da coloro che hanno adottato Go si fa chiamare "gopher", in onore della tenera mascotte di Go (Figura 1.1). La programmazione è,

In questo capitolo

in termini generali, un argomento complicato, ma grazie a Go e a questo libro, contiamo di far scoprire a tutti la gioia della programmazione.

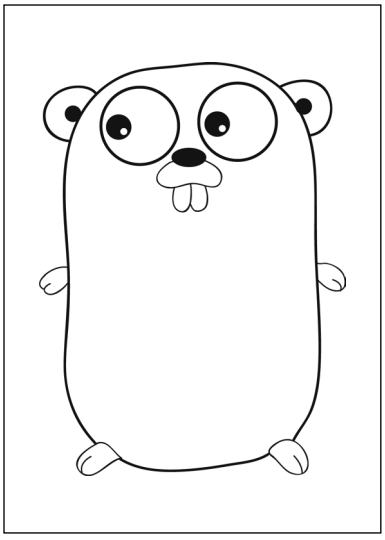


Figura 1.1 La talpa Gopher, mascotte di Go, disegnata da Renée French.

In questo capitolo inizieremo a sperimentare il funzionamento di un programma Go all'interno del browser web.

Considerazioni iniziali

Se dite a un assistente digitale, "Chiamami un taxi", siete sicuri che vi chiamerà davvero un taxi? Non penserà, magari, che d'ora in poi vogliate cambiare il vostro nome in "un taxi"? Il linguaggio naturale, l'italiano, è pieno di ambiguità.

La chiarezza è fondamentale nei linguaggi di programmazione. Se la grammatica o la sintassi del linguaggio lasciassero spazio ad ambiguità, il computer finirebbe per non capire che cosa gli state chiedendo di fare. A questo punto, che senso avrebbe scrivere un programma?

Go non è un linguaggio perfetto, ma punta sulla chiarezza più di qualsiasi altro linguaggio che abbiamo utilizzato. Come vedremo nel corso di questo capitolo, vi saranno alcune abbreviazioni da imparare e qualche termine gergale da scoprire. Non tutto sarà chiaro fin da subito, ma dategli un po' di tempo e Go vi aiuterà a fugare ogni ambiguità.

Che cos'è Go

Go è un linguaggio di programmazione *compilato*. Prima di eseguire un programma, Go utilizza un compilatore per tradurre il codice che avete scritto nei famosi "1" e "0" che le macchine sono in grado di capire. Poi compila il codice che avete scritto, per produrre

un file eseguibile, che potete decidere di lanciare o di distribuire ad altri. Nel corso di questo processo, il compilatore Go si occuperà di andare a caccia di tutti gli errori di ortografia o di sintassi che potete aver commesso.
Non tutti i linguaggi di programmazione impiegano questo approccio. Python, Ruby e molti altri linguaggi di programmazione utilizzano un *interprete*, il quale traduce un'istruzione del programma alla volta ed esegue il programma. Questo significa che al momento dell'esecuzione il codice potrebbe contenere errori (*bug*) che non sono ancora stati individuati.
D'altro canto, un interprete rende più veloce e interattivo il processo di scrittura del codice, rendendo pertanto il linguaggio più dinamico, accessibile e divertente. I linguaggi compilati, al contrario, hanno una certa reputazione di staticità e inflessibilità, come robot che i programmatori sono costretti ad accontentare; inoltre i compilatori sono noti per una certa lentezza. Ma le cose devono essere necessariamente così?

> Cercavamo di realizzare un linguaggio che avesse la sicurezza e le prestazioni dei linguaggi compilati in modo statico, come C++ e Java, ma la leggerezza e la facilità d'uso dei linguaggi interpretati, come Python.
>
> *Rob Pike*, Geek of the Week
> (vedi `mng.bz/jr8y`)

Go è stato realizzato privilegiando la gradevolezza nella scrittura del software. Grossi programmi possono essere compilati in pochi secondi con un unico comando. Il linguaggio omette tutti quegli elementi che possono causare ambiguità, incoraggiando una programmazione facilmente prevedibile e comprensibile. Inoltre Go rappresenta un'alternativa "leggera" alla rigida struttura imposta dai linguaggi "classici", come Java.

> Java omette molte funzionalità di utilizzo raro e di difficile comprensione del C++ le quali, per esperienza, danno più problemi che vantaggi.
>
> *James Gosling*, Java: an Overview

Ogni nuovo linguaggio di programmazione tende a raffinare le idee del passato. L'utilizzo efficiente della memoria in Go è più semplice e meno soggetto a errori rispetto ai linguaggi di un tempo; inoltre Go è in grado di sfruttare ogni *core* delle macchine multicore. Le sue storie di successo spesso citano una maggiore efficienza fra i motivi che hanno spinto molti a passare a Go. Iron.io è stato in grado di sostituire ben trenta server operanti in Ruby con soli due server gestiti da Go (vedi `mng.bz/Wevx` e `mng.bz/8yo2`). Bitly ha "rilevato notevoli e misurabili vantaggi prestazionali" nella riscrittura delle app Python in Go e, di conseguenza, ha sostituito le sue app C con nuove versioni in Go (vedi `mng.bz/EnYl`).
Go offre la piacevolezza e la facilità d'uso dei linguaggi interpretati, con l'aggiunta di una maggiore efficienza e affidabilità. Essendo un linguaggio compatto, dotato di pochi e semplici concetti, Go è relativamente facile da imparare. Sono tre le parole chiave che formano il motto di Go:

> Go è un linguaggio di programmazione open source che consente la produzione di software semplice, efficiente e affidabile, a qualsiasi livello di scala.
>
> *Go Brand Book*

SUGGERIMENTO

Quando cercate in Internet qualche argomento relativo a Go, utilizzate la parola chiave *golang*, ovvero "*Go language*". Il suffisso "*-lang*" (o il suffisso "*-linguaggio*" per le ricerche in italiano) è appropriato anche per altri linguaggi di programmazione dal nome breve, come Ruby, Rust e così via.

TEST

Troverete le soluzioni dei test alla fine di questo capitolo.

1. Quali sono i due vantaggi del compilatore Go?

Go Playground

Il modo più rapido per iniziare a utilizzare Go consiste nell'accedere alla pagina `play.golang.org`. Nel Go Playground (Figura 1.2) potete scrivere, eseguire e provare programmi Go senza dover installare nulla. Vi basterà fare clic sul pulsante *Run* e il playground compilerà ed eseguirà il vostro codice sui server Google e ne visualizzerà il risultato.

Figura 1.2 Go Playground.

Facendo clic sul pulsante *Share*, otterrete un link che rimanda al codice che avete appena scritto. Potete condividere questo link con gli amici o salvarlo in un segnalibro, per tornare a riutilizzarlo in un secondo tempo.

NOTA

Potete utilizzare Go Playground per ogni listato ed esercizio presente nel libro. In alternativa, se siete già abituati a utilizzare un editor di testi e la riga di comando, potete scaricare e installare Go sul vostro computer dall'indirizzo `golang.org/dl/`.

TEST

2. Che cosa fa il pulsante *Run* del Go Playground?

Package e funzioni

Quando aprite il Go Playground, compare il seguente codice, che rappresenta un ottimo punto di partenza.

Listato 1.1 Hello, playground: playground.go.

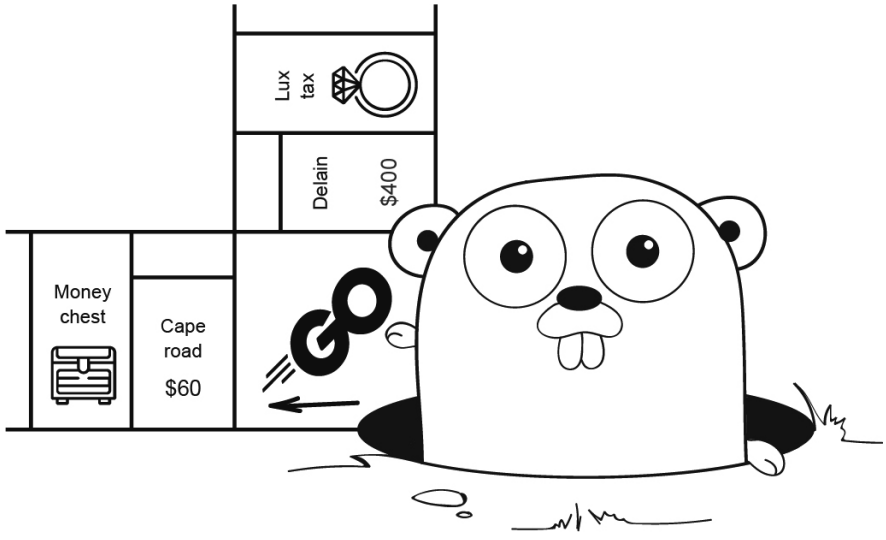
```
package main      ← Dichiara il pacchetto cui appartiene questo codice

import (
    "fmt"      ← Rende disponibile il package fmt (formattazione)
)

func main() {      ← Dichiara una funzione chiamata main
    fmt.Println("Hello, playground")      ← Presenta sullo schermo le parole "Hello, playground"
}
```

Pur breve, il listato precedente introduce già tre parole riservate: `package`, `import` e `func`. Ogni parola riservata ha uno scopo ben preciso.

La parola riservata `package` dichiara il package cui appartiene questo codice: in questo caso si tratta di un package chiamato `main`. Tutto il codice, in Go, è organizzato in *package*. Go fornisce una libreria standard comprendente package per la matematica, la compressione, la crittografia, la manipolazione delle immagini e molto altro ancora. Ogni singolo package corrisponde a un singolo ambito o argomento.

La riga successiva usa la parola riservata `import` per specificare i package che verranno utilizzati da questo codice. I package contengono un determinato numero di *funzioni*. Per esempio, il package `math` fornisce funzioni come `Sin`, `Cos`, `Tan` e `Sqrt` (radice quadrata). Il package `fmt` utilizzato qui fornisce funzioni per la *formattazione* dell'input e dell'output. La visualizzazione di testo sullo schermo è un'operazione molto frequente e quindi, per comodità, questo package è stato abbreviato il più possibile: `fmt`.

La parola riservata `func` dichiara una funzione, in questo caso una funzione chiamata `main`. Il corpo di ogni funzione deve essere racchiuso fra parentesi graffe `{ }`, che è il modo con il quale Go può sapere dove inizia e termina ogni funzione.



L'*identificatore* `main` ha un ruolo speciale. Quando si esegue un programma scritto in Go, l'esecuzione parte dalla funzione `main` sita nel package `main`. Senza `main`, il compilatore Go produrrà un errore, perché non avrà modo di capire dove deve iniziare il programma.
Per produrre sullo schermo una *riga* di testo, si usa la funzione `Println`. A `Println` si deve far precedere il prefisso `fmt` e un punto, poiché tale funzione si trova nel package `fmt`. Ogni volta che si usa una funzione tratta da un package importato, il nome della funzione deve essere preceduto dal nome del package e dal punto di separazione. In questo modo, quando leggete del codice scritto in Go, risulta immediatamente chiaro da quale package è stata tratta la funzione.
Provate a lanciare il programma in Go Playground e sullo schermo verranno visualizzate le parole *Hello, playground*. In pratica sullo schermo viene prodotto tutto ciò che è stato specificato fra i doppi apici. In italiano, l'assenza di una virgola può cambiare il significato di una frase. La punteggiatura è importante anche nei linguaggi di programmazione. Go conta su apici, parentesi e parentesi graffe per capire il significato del codice che avete scritto.

TEST

3. Dove inizia un programma Go?
4. Che cosa fornisce il package `fmt`?

L'uso delle parentesi graffe

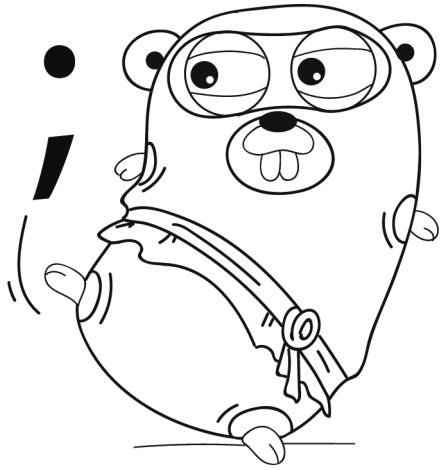
Go è piuttosto pignolo nel posizionamento delle parentesi graffe `{ }`. Nel Listato 1.1, la parentesi graffa aperta, `{`, si trova sulla stessa riga della parola riservata `func`, mentre la parentesi graffa chiusa, `}`, si trova su una riga a sé stante. Questo è l'unico stile di scrittura ammesso, non vi sono altre possibilità. Vedi `mng.bz/NdE2`.
Per capire perché Go sia così rigido, occorre risalire alla nascita di Go. A quei tempi, il codice era ricchissimo di punti e virgola. Si trovavano punti e virgola ovunque. Non c'era modo di sfuggire: i punti e virgola seguivano ogni singola istruzione come una paperetta segue mamma papera. Per esempio:

```
fmt.Println("Hello, fire hydrant");
```

Nel dicembre 2009, un gruppo di "Ninja gopher" ha deciso di espellere i punti e virgola dal linguaggio. Ehm... non esattamente. In realtà, il compilatore Go inserisce tutti questi ìpunti e virgola per voi, e tutto funziona perfettamente. Ma può funzionare perfettamente solo se adottate lo stile di *collocazione delle parentesi graffe* di cui abbiamo appena parlato. Se ponete una parentesi graffa su una riga a sé stante dopo la parola riservata `func`, il compilatore Go opporrà un errore di sintassi:

```
func main()     ← Corpo della funzione incompleto
{     ← Errore di sintassi: punto e virgola inatteso o
}       fine della riga prima della parentesi graffa aperta
```

Non è che il compilatore ce l'abbia con voi. Il fatto è che ha finito per inserire un punto e virgola nella posizione errata e per questo motivo è un po' confuso.



SUGGERIMENTO

Mentre procedete nella lettura di questo libro, provate a scrivere i listati a mano. Probabilmente commetterete degli errori di sintassi, se vi capiterà di scrivere male qualcosa, e questo è perfettamente normale. Essere in grado di leggere, comprendere e correggere gli errori è un'abilità importante e anche la perseveranza è una dote preziosa.

TEST

5. Dove deve essere collocata la parentesi graffa aperta, {, per evitare errori di sintassi?

Riepilogo

- Con Go Playground potete iniziare a usare Go senza installare nulla.
- Ogni programma Go è costituito da funzioni, contenute in package.
- Per mostrare del testo sullo schermo, utilizzate il package `fmt` fornito dalla libreria standard.
- La punteggiatura è importante, sia nei linguaggi di programmazione, sia nei linguaggi naturali.
- A questo punto avete utilizzato tre delle venticinque parole riservate di Go: `package`, `import` e `func`.

Vediamo se avete capito tutto…

Per svolgere il seguente esercizio, modificate il codice nel Go Playground e fate clic sul pulsante *Run* per vedere il risultato. Se avete problemi, aggiornate la pagina nel browser web e tornate al codice originario.

Esperimento: playground.go

- Cambiate il testo presentato sullo schermo, modificando ciò che si trova fra i doppi apici. Per esempio, provate a far scrivere al computer il vostro nome.
- Provate a scrivere due righe di testo, aggiungendo una seconda riga di codice nel corpo, `{ }`, della funzione `main`. Ecco un esempio:

```
fmt.Println("Hello, world")
fmt.Println("Hello, 世界")
```

- Go supporta l'uso dei caratteri di qualsiasi lingua. Potete produrre del testo in cinese, giapponese, russo o greco. Se non parlate queste lingue, potete usare Google Translate (`translate.google.com`) e copiare/incollare nel Go Playground il testo che verrà prodotto.

Usate il pulsante *Share* per ottenere un link al vostro programma e condividerlo con gli amici.
Confrontate la soluzione di questo esercizio con i risultati che trovate nell'Appendice.

Soluzioni dei test

1. I grossi programmi possono essere compilati in pochi secondi e il compilatore Go permette di individuare errori di ortografia e di sintassi prima di eseguire un programma.
2. Il pulsante *Run* compila e poi esegue il vostro codice sui server Google.
3. Un programma inizia dalla funzione `main`, all'interno del package `main`.
4. Il package `fmt` fornisce tutte le funzioni di formattazione dell'input e dell'output.
5. La parentesi graffa aperta deve trovarsi sulla stessa riga di `func` e non su una riga a sé stante. Questo è uno stile di scrittura obbligatorio.