Golang :

<https://tutorialedge.net/golang/creating-restful-api-with-golang/>

To run go program :Go run filename

Save file name: main.go

Machine generated alternative text:
Questions! 
How do we run the 
code in our project? 
What does 'import 
"fmt'" mean? 
What does 
'package main' 
mean? 
What's that 'func' 
thing? 
How is the main.go 
file organized? 

Machine generated alternative text:
How do we run the 
code in our project? 
Go CLI 
go build 
go run 
go fmt 
go install 
go get 
go test 
Compiles a bumh of 
code files 
Compiles and executes one or 
two files 
Formats all the code in each file 
in the current directory 
Compiles and •installs• a 
Downloads the raw wurce code 
of someone else's package 
Runs any tests associated with 
the current project 

In **Go** language, the **main package** is a special **package** which is used with the programs that are executable and this **package** contains **main**() function. The **main**() function is a special type of function and it is the entry point of the executable programs. It does not take any argument nor return anything.

Machine generated alternative text:
What does 'package main' mean ? 
Exes,utable 
peckege . 
Reusable 
package 
Defines a package that can be 
compiled and then *executed*. 
Must have a func called 'main' 
Defines a package that can be 
used as a dependency (helper 
code) 
Defines a package that can be 
used as a dependency (helper 
code) 
package main 
"main" is special 
package calculator 
package uploader 

Fmt is short form of format .

"**fmt**" is a package that provides I/O functions like Println (you can import it with import "**fmt**" ).

Machine generated alternative text:
What does •import •trnt• mean? 
debug 
fmt 
math 
encoding 
main 
crypto 
io 

Machine generated alternative text:
What does 'import "fmt" mean? 
fmt 
Standard lib 
main 
calculator 
Reusable 
package 
uploader 
Reusable 
package 

Machine generated alternative text:
What's that 'func' thing? 
Tells go we're 
about to declare 
a function 
List of arguments 
Sets the name 
to pass the 
of the function 
function 
Function body. 
Calling the function 
runs this code 
func main() 

Machine generated alternative text:
How is the 
main.go file 
organized? 
package main 
import "fmt" 
func main() 
fmt.Println("hi there") 
Package 
Import Other 
packages that we 
Declare functions, tell 
Go to do things 

Machine generated alternative text:
Cards 
newDeck 
print 
shuffle 
deal 
saveToFiIe 
newDeckFromFile 
Create a list of playing 
Cards. Essentially an 
array of strings 
Log out the Contents of a 
deck Of cards 
Shuffles all the cards in a 
Create a hand' Of cards. 
Save a list Of cards to a file 
on the local machine 
Load a list Of cards from 
the local machine 

Machine generated alternative text:
var card string "Ace of Spades" 
We're about to 
create a new 
variable 
The name of the 
variable will be 
'card' 
Only a "string" will 
Assign the value 
ever be assigned 
"Ace of Spades" 
to this variable 
to this variable 

Machine generated alternative text:
Dynamic Types 
Javascript 
Ruby 
Python 
Static Types 
Java 
GO 

Machine generated alternative text:
Type 
bool 
string 
int 
float64 
Basic go Types 
Example 

Machine generated alternative text:
main 
package 
import "fmt" 
v func main() 
//var card string - 
"ace of spades" 
card 
card 
"ace of spades " 
" saurabh " 
fmt.Print1n(card) 

Variables can be initialized outside of a function, but cannot be assigned a variable.

**Function and return types**

Machine generated alternative text:
main 
package 
import "fmt" 
v func main() 
newCard() 
card 
fmt.Print1n(card) 
newCard() string 
v func 
return "five of diamond" 

Machine generated alternative text:
main 
package 
import "fmt" 
func main() 
- newCard() 
// card 
card int 
newCard() 
var 
fmt.Print1n(card) 
newCard() int 
func 
return 4 

============================================

Machine generated alternative text:
main 
package 
func main() 
printState() 

Machine generated alternative text:
state.go 
1 
2 
3 
4 
5 
6 
7 
main 
package 
import "fmt" 
printState() 
func 
fmt. Println( "California" ) 

Machine generated alternative text:
PS C: NgocodePXcards» go run main. go state. go 
California 

Machine generated alternative text:
Array 
Fixed length lis, 
of things 
Slice 
An array that can 
grow or shrink 

Machine generated alternative text:
Slices 
Every element in a slice must 
be of same type 
"Five of 
Diamonds" 
"Five of 
Spades" 
55525235 

Creating slice and iterating slice

NOTE: range is keyword which we used to iterate over every single element of slice

Machine generated alternative text:
index of this 
element in the 
for inde 
Current card 
we're iterating 
c 
ran 
-a 
Take the slice 
of 'cards' and 
loop ver it 
e cards 
fmt.Pri tlr, ar ) 
Run this one time 
for each card in the 
slice 

Machine generated alternative text:
8 
main.go main 
1 
2 
3 
4 
5 
6 
7 
8 
9 
11 
12 
13 
14 
15 
16 
17 
18 
main 
package 
import "fmt" 
func main() 
// creating slice 
of diamond", newCard()) 
cards 
// add new element to slice 
append(cards, 
"5 spades") 
cards 
fmt.Print1n(cards) 
// iterate over slice 
for i, card 
range cards 
fmt.Print1n(i, card) 
newCard() string 
func 
TERMINAL 
PROBLEMS OUTPUT DEBUG CONSOLE 
PS C: NgocodePXcards» go run main. go 
Lace of diamond spades 5 spades) 
e ace of diamond 
1 spades 
2 5 spades 
PS C: NgocodePXcards» 

Machine generated alternative text:
Cards 
newDeck 
rint 
sh 
dea 
saveToFile 
newDeckFromFile 
Create a list of playing 
cards. Essentially an 
array of strings 
Log out the contents of a 
deck of cards 
Shuffles all the cards in a 
deck 
Create a hand' of cards. 
Save a list of cards to a file 
on the local machine 
Load a list of cards from 
the local machine 

NOTE: go is not object oriented programming language

Machine generated alternative text:
Base Go Types 
wan to "extend" a base 
type and add some extra 
functiogality to it 
Tell GO we want to Create an 
array of strings and add a 
type deck 0 String bunch of functions specifically 
made to work with it 
Functions With A function with a receiver is like 
a "method" - a function that 
a 'receiver' belongs to an "instance" 

Machine generated alternative text:
main.go 
Code to create 
and manipulate 
a deck 
cards' folder 
dc 
Code that 
describes what 
a deck is and 
how it works 
deck_test.go 
Code to 
automatically 
test the deck 

Machine generated alternative text:
Base Go Types 
array 
map 
wan "extend" a base 
and add some extra 
functionality to it 
Tell GO we want to Create an 
array of strings and add a 
type deck bunch of functions specifically 
made to work with it 
Functions With A function with a receiver is like 
a "method" - a function that 
a 'receiver' belongs to an "instance" 

Note :

Deck type will give extra functionality to work with function

Receiver on a function

Machine generated alternative text:
unc d deck print() 
Any varia 
T type "deck" 
now gets access to the 
"print" method 

Receiver set a method on variable we create

Machine generated alternative text:
The actual copy of the deck 
we're working with is 
available in the function as 
a variable called d' 
Every variable of 
type deck' can call 
this function on itself 
func d 
) rint() 
for l, car 
range d 
fmt.Println(i, card) 

D is reference of variable of card deck type

Type of deck and method put together

Receiver use to call a method on deck type variable and it will get the reference of variable

To invoke a method on variable of deck type: receiver

Machine generated alternative text:
main.go main main 
1 
2 
3 
4 
5 
6 
7 
8 
9 
Le 
main 
package 
func main() 
// creating slice 
cards 
// add 
cards 
cards . 
of diamond", "spade") 
new element to slice 
append(cards, 
"5 spades") 
Print() 

Machine generated alternative text:
main 
package 
import "fmt" 
type deck astring 
(d deck) drint() 
func 
// iterate over slice 
for i, card 
range d 
fmt.Print1n(i, card) 

Machine generated alternative text:
main 
package 
import "fmt" 
type deck astring 
/ func main() 
// creating slice 
of diamond", "spade") 
cards 
//add new element to slice 
append(cards, 
"5 spades") 
cards 
cards . Print() 
(d deck) Print() 
/ func 
// iterate over slice 
for i, card 
range d 
fmt.Print1n(i, card) 

====================

Machine generated alternative text:
package main 
import "fmt" 
type book string 
func (b book) printTit1e() 
fmt. println(b) 
func main() 
var b book 
"Harry potter" 
b. printTit1e() 

Answer : harry potter

Machine generated alternative text:
1 
2 
3 
4 
5 
6 
7 
8 
9 
le 
11 
12 
13 
14 
15 
16 
main 
package 
import "fmt" 
type fun string 
(d fun) printName() 
func 
fmt.Print1n(d) 
func main() 
var test fun 
"saurabh" 
test . printName ( ) 
PROBLEMS OUTPUT DEBUG CONSOLE 
TERMINAL 
PS C: NgocodePXcards» go run state. go 
saurabh 
PS C: NgocodePXcards» 

Code to do

Machine generated alternative text:
Cards 
newDeck 
print 
shuffle 
deal 
saveToFiIe 
newDeckFromFiIe 
'Create and return a list of 
laying cards. Essentially 
'antrray of strings 
ag out the contents of a 
deck of cards 
all the cards in a 
de 
Create a hand' of cards. 
Save a list of cards to a file 
on the local machine 
Load a list of cards from 
the local machine 

Machine generated alternative text:
cards deckO 
cardSuits , "Hearts", "Diamonds") 
ardValues Elstririg(" 
"Two", "Three") 
cel 
for each suit in cardSuits 
for each value in cardValueso 
Add a new card of 'value of suk' to the 'cards' deck 

NOTE:

Whenever we have variable while iterating for loop and we don’t use it we have to replace with \_

Machine generated alternative text:
main.go main main 
main 
package 
1 
2 
3 v func main() 
4 
5 
6 
7 
8 
9 
cards 
— newDeck() 
cards . Print() 

Machine generated alternative text:
deck.go main 
string( " spades" , 
"diamond" , 
"heart") 
"three") 
"two 
range cardSuits 
1 
2 
3 
4 
5 
6 
7 
8 
9 
11 
12 
13 
14 
15 
16 
17 
18 
19 
21 
22 
main 
package 
import "fmt" 
Wpe deck astring 
newDeck() deck 
func 
— deck() 
cards 
cardSuits . 
cardVa1ues 
, suit 
for 
range cardVa1ues 
for 
value 
append(cards, value+"of"+suit) 
cards 
return cards 
(d deck) Print() 
func 
// iterate over slice 
for i, card 
range d 
fmt.Print1n(i, card) 
OUTPUT 
DEBUG CONSOLE 
TERMINAL 
PROBLEMS 
4 twoofdiamond 
threeofdiamond 
6 oneofheart 
7 twoofheart 
8 threeofheart 

Machine generated alternative text:
Ace Of Spades 
Two of sp 
Three Of Spades 
Aleal(3) 
Four Of Spades 
I card left in deck 
"hand" of 3 cards 

Splitting of slice

Machine generated alternative text:
fruits(0J 
fruits(3J 
Slices are zero-indexed 
1 
"apple" 
"orange" 
3 
2 

Machine generated alternative text:
1 
2 
3 
"banana" 
fruits(startlndexlncluding : upToNotIncIudingJ 
fruitstO:21 
"apple' 

Range syntax in go

Machine generated alternative text:
2 
3 
cardsLhandsize:J 
cardsL3:J 
3 
cardst:handsizel 
cardsL:3J 

Go support returning multiple value from one function

Machine generated alternative text:
main.go main main 
8 
main 
package 
1 
2 
3 v func main() 
4 
5 
6 
7 
8 
9 
le 
PROBLEMS 
— newDeck() 
cards 
hand, remainingCards . — deal (cards, 
hand . Print ( ) 
remainingCards.Print ) 
OUTPUT 
DEBUG CONSOLE 
TERMINAL 
PS C: NgocodePXcards» go run main. go deck. go 
e 
1 
e 
1 
2 
3 
4 
5 
6 
oneofspades 
twoofspades 
threeofspades 
oneofdiamond 
twoofdiamond 
threeofdiamond 
oneofheart 
twoofheart 
threeofheart 
PS C: NgocodePXcards» 

Machine generated alternative text:
main 
package 
import "fmt" 
type deck astring 
newDeck() deck 
func 
— deck() 
cards 
cardSuits . 
cardVa1ues 
, suit 
for 
string( " spades" , 
"diamond" , 
"heart") 
"three") 
"two" 
range cardSuits 
range cardVa1ues 
for 
value 
append(cards, value+"of"+suit) 
cards 
return cards 
(d deck) Print() 
func 
// iterate over slice 
for i, card 
range d 
fmt.Print1n(i, card) 
deal (d deck, handsize int) (deck, deck) 
func 
dt:handsize), dthandsize:) 
return 

Machine generated alternative text:
"Hi there!" 
string 
byte 
172 105 32 116 104 101 114 101 331 
slice 
asciitable.com 

**Deck type to string or byte of slice**

Machine generated alternative text:
func WriteFile 
func WriteFile(filename string, data 
by 
o 
FileMode) error 

Machine generated alternative text:
Ilbyte 
Type we 
want 
llHi there!ll 
Value we 
have 

Machine generated alternative text:
main.go main main 
- deal( 
1 
2 
3 
4 
5 
6 
7 
8 
9 
Le 
12 
14 
main 
package 
import "fmt" 
func main() 
newDeck() 
// cards 
// hand, remainingCards 
// hand. Print() 
/ / remainingCards . Print() 
greeting 
"Hi There" 
fmt. Println ( byte@greeting)) 
()BLEMS OUTPUT DEBUG CONSOLE TERMINAL 
C: NgocodePXcards» go run main. go 
72 les 32 84 194 lei 114 lei) 
C: NgocodePXcardO 

Machine generated alternative text:
we 
have 
we 
want 
deck 
qstring 
string 
IJbyte 

There are two reasons to **use** a pointer **receiver**. The first is so that the method can modify the value that its **receiver** points to. The second is to avoid copying the value on each method call. This can be more efficient if the **receiver** is a large struct, for example.

Machine generated alternative text:
main 
package 
import "fmt" 
func main() 
cards 
— newDeck() 
fmt.Println@cards.toString( 

Machine generated alternative text:
(d deck) toString() string 
v func 
trings . string(d), 
return 
EMS OUTPUT DEBUG CONSOLE TERMINAL 
1: go 
: NgocodePXcards» go 
: NgocodePXcards» go 
: NgocodePXcardO 
run main. go deck. go 
run main. go deck. go 
Fspades , twoofspades , threeofspades , oneofdiamond, twoofdiamond, threeofdiamond , oneofheart , twoofhea rt , th reeofhea rt 

Save to file:

Machine generated alternative text:
main 
package 
func main() 
cards 
— newDeck() 
cards.saveToFi1e@"my_cardsl") 

Machine generated alternative text:
(d deck) saveToFi1e(fi1ename string) error 
func 
•outil . WriteFi1e(fi1ename, e666) 
return 

**Reading file from hard drive**

Machine generated alternative text:
byteSIice 
err 
ioåtil. Read File(filename) 
Value of type err 
If nothing went 
wrong, it will have a 
value of 'nil' 

Machine generated alternative text:
X 8 deck.go 
main.go 
main.go main main 
main 
package 
1 
2 
3 v func main() 
4 
5 
6 
7 
8 
9 
newDeckFromFi1e( "my_cards") 
cards 
cards . Print() 

Machine generated alternative text:
newDeckFromFi1d(fi1ename string) deck 
func 
bs, err . : ioutil. ReadFi1e(fi1ename) 
if err nil 
fmt.Print1n( "error: ' 
os . Exit(l) 
- strings. Split(string(bs) 
s 
deck(s) 
return 

**Shuffling a deck**

Machine generated alternative text:
Ace of Spades 
Two of Spades 
Three of Spades 
Four of Spades 
onufflq 

Machine generated alternative text:
for each index, card in cards 
Generate a random number between 0 and len(cards) - 1 
Swap the current card and the card at cardsLrandomNumberJ 

Machine generated alternative text:
X 8 deck.go 
main.go 
main.go main main 
1 
2 
3 
4 
5 
6 
7 
8 
9 
main 
package 
func main() 
— newDeck() 
cards 
cards . shuffle 
Print() 
cards . 

Machine generated alternative text:
(d deck) 
shuffle() 
func 
— rand.NewSource(time .Now() . 
source 
New(source) 
rand. 
for i . 
range d 
newPosition 
r.lntn(len(d) 
1) 
dc i J, dtnewPosition) 
dCnewPositionJ, 

Machine generated alternative text:
deck.go 
newDeck 
saveToFile 
newDeckFromFile 
desk_test.go 
func TestNewDeck 
Code to make sure that a deck 
is created with x number of 
cards 
Code to make sure that the first 
card is an Ace of Spades 
Code to make sure that the last 
card is a Four of Clubs 
• func Te SaveToDeckandNewDeckFromFiIe 

Machine generated alternative text:
Card Struct Field 
Definition 
Card Struct 

Machine generated alternative text:
Tell go what fields the 
person struct ha 
firstName 
lastName 
person struct 
Create a new value 
of type person 
firstName 
lastName 
"Alex" 
' Anderson' 

Defining structs

Machine generated alternative text:
package main 
type person struct 
firstName string 
lastName string 
func main() 

Declaring structs

Machine generated alternative text:
import "fmt" 
type person struct 
firstName string 
lastName string 
func main() 
person(firstName: 
alex . 
fmt.PrintIn(aIex) 
lastName: 
"Anderson") 

Go default value

Machine generated alternative text:
Type 
string 
int 
Zero Value 

Another way to declare structs

Machine generated alternative text:
main.go main main 
1 
2 
3 
4 
5 
6 
7 
8 
9 
11 
12 
13 
14 
15 
main 
package 
import "fmt" 
type person struct 
firstname string 
string 
lastname 
func main() 
alex person 
var 
Println(alex) 
fmt. 
fmt. Printf "%+v" 

Machine generated alternative text:
main.go main main 
1 
2 
3 
4 
5 
6 
7 
8 
9 
11 
12 
13 
14 
15 
16 
17 
18 
main 
package 
import "fmt" 
type person struct 
firstname string 
string 
lastname 
func main() 
var alex person 
alex.firstname 
alex. lastname 
"saurabh" 
"kesarwani " 
fmt. Println(alex) 
fmt. Printf( "%+v", alex) 
OUTPUT 
DEBUG CONSOLE 
TERMINAL 
ROBLE-MS 
S C: NgocodePXstructs» go run main. go 
saurabh kesarwani) 
firstname:saurabh lastname:kesarwani) 
S C: NgocodePXstructs» 

Embedding structs:

Machine generated alternative text:
type person+ 
4:ontactInfc» 
contact 
type contactlnfo 

For defining structs we don’t use commas

Machine generated alternative text:
main.go main main 
8 
1 
2 
3 
4 
5 
6 
7 
8 
9 
11 
12 
13 
14 
15 
16 
17 
18 
19 
21 
22 
23 
24 
25 
26 
main 
package 
import "fmt" 
type contactlnfo struct 
email 
string 
zipCode int 
type person struct 
firstname string 
lastname 
contact 
string 
contactlnfo 
"saurabh" 
" kesarwani" , 
func main() 
jim 
person( 
firstname: 
lastname : 
contact: contactlnfo( 
email : 
"saurabh@gmail . com", 
1 : powershell 
zipcode: 12eee, 
jinOl 
PROBLEMS OUTPUT DEBUG CONSOLE 
TERMINAL 
ffirstname:saurabh lastname:kesarwani zipCode:12eee» 
PS C: NgocodePXstructO 

Machine generated alternative text:
main.go main main 
5 
6 
7 
8 
9 
12 
14 
18 
22 
23 
24 
25 
26 
28 
29 
type contactlnfo struct 
email 
string 
zipCode int 
type person struct 
firstname string 
string 
lastname 
contactlnfo 
func main() 
jim 
person 
firstname: 
"saurabh" 
" kesarwani" 
lastname : 
contact Info( 
contactlnfo: 
email : 
"saurabh@gmail . com", 
zi pCode : 
12eee, 
jim.print() 
(p person) print() 
func 
fmt.Printf( "%+v", p) 
1 : powershell 
OBLEMS OUTPUT DEBUG CONSOLE 
TERMINAL 
Firstname:saurabh lastname:kesarwani zipCode:12eee» 