GO cheatsheet

- https://www.programming-idioms.org/cheatsheet/Go

1 Print Hello World
Print a literal string on standard output

```
fmt.Println("Hello World")
```

2 Print Hello 10 times
Loop to execute some code a constant number of times

```
for i := 0; i < 10; i++ {
        fmt.Println("Hello")
}
```

3 Create a procedure
Like a function which doesn't return any value, thus has only side effects (e.g. Print to standard output)

```
func finish(name string) {
  fmt.Println("My job here is done. Good bye " + name)
}
```

4 Create a function which returns the square of an integer

```
func square(x int) int {
  return x*x
}
```

5 Create a 2D Point data structure
Declare a container type for two floating-point numbers ***x*** and ***y***

```
type Point struct {
    x, y float64
}
```

6 Iterate over list values
Do something with each item ***x*** of an array-like collection ***items***, regardless indexes.

```
for _, x := range items {
    doSomething( x )
}
```

http://golang.org/ref/spec

Go heľpovník

- **package & import**

```
package id                          // prvý riadok v module
import "path"
import ( "path" "path" ... )        // import viacerých
```

- **const, type, var, func**

```
const id ,... type = value, ...// deklarácia konštánt s
                                   explicitným typom
const id ,... = value ,...          // deklarácia konštánt s
                                   implicitným typom
type id different_type              // typové synonymum
var id ,... type = value ,...       // deklarácia premenných
                                   s explicitným typom
var id ,... = value ,...            // deklarácia premenných
                                   s implicitným typom
const|type|var ( spec; ... )       // viacnásobná deklarácia
```

http://golang.org/ref/spec

Go heľpovník

- **if-then[-else]**

```
if [statement;] condition { block } [else if-or-block]
```

- **statement**

```
expression                          // výraz, napr. 5*6
function call                       // exp(2.71)
target ,... = expression ,...       // parallelné priradenie
target op= expression               // a += 5
target ++, target --                // !!! nie je to výraz
id ,... := expression ,...          // skrátená deklarácia
```

- **for cyklus**

```
for { block }                       // for ;; {}, resp. while (true) {}
for condition { block }             // while
for init; condition; post { block } // ako v Jave
for index, value = range expression { block }
                                    // cyklus cez pole, slice, kanál, ...
```

http://golang.org/ref/spec

Go heľpovník

- **arrays**

```
type id [ length ] type                  // typ poľa s konštantou dĺžkou
type id [ length ] [ length ] type       // matica 2, 3-dimen.
len( array )                             // veľkosť poľa
expression [ expression ]                // indexovanie od 0..len(array)-
new([ length ] type )                    // vytvorenie pola s dĺžkou length
[ length ] type { expression ,... }      // konštanta typu pole s dĺžkou
[ ] type { expression ,... }             // konštanta typu pole bez dĺžky
```

- **slices**

```
[] type                                  // slice type (without length)
make([] type, length)                    // construction, slice and array
make([] type, length, capacity)
len( slice ), cap( slice )               // length and capacity
expression [low : high]                  // construction from array or slice
expression [low :]
expression [ expression ]                // element, index from 0
```

http://golang.org/ref/spec

Go heľpovník

- **function**

```
func id ( parameters ) { body }          // procedúra
func id ( parameters ) result { body }   // fcia s výstupným typom
func id ( parameters ) ( results ) { body }  // viac výstupov
func ( parameters ) ( results ) { body }     // anonymná funkcia
func ( types ) ( types ) // type          // funkčný typ
id ,... type                             // parametre, výstupné premenné
id ... Type                              // neurčený počet argumentov
```

- **methods**

```
func (id type) id ( parameters ) ( results ) { body }
func (id *type) id ( parameters ) { body }   // definícia
func ( type ,... ) ( results )               // typ
expression . id ( arguments )                // volanie metódy
type . id                                    // metóda ako funkcia
(* type ). id
```

http://golang.org/ref/spec

Go heľpovník

- **switch**

```
switch statement; expression {
    case expression ,...: statement; ...
    default: statement; ...
}
switch statement; id := expression .(type) {
    case type ,...: statement; ...
    default: statement; ...
}
```

- **maps**

```
var id map [ type1 ] type2    // deklarácia zobrazenia type1 -> type2
make(map[type1 ] type2 )      // vytvorenie zobrazenia type1 -> type2
make(map[type1] type2, initial-size)
map [ key ] = value           // pridanie (key -> value)
map [ key ] = dummy, false    // zmazanie key
map [ key ]                   // vyhľadanie value pre key
value, ok = map[ key ]        // ok je true alebo false
```

http://golang.org/ref/spec

Go heľpovník

- **struct**

```
type id struct {                                    // deklarácia
    id ,... type // named field
}
```

- **new-make-nil**

```
new( type )                                 // alokuje štruktúru a vráti *type
make([] type, capacity )                    // pre slice, alokuje []type
make([] type, length, capacity )
make(chan type )                            // pre kanál, alokuje chan type
make(chan type, capacity )
make(map[ type ] type )                     // pre map, alokuje map[type]type
make(map[ type ] type, initial-capacity )
```