# Going Loopy

## adventures in iteration with Google Go

Eleanor McHugh

## the conditional loop

```go
package main
import "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    fmt.Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    fmt.Printf("%v: %v\n", i, s[i])
  }
}
```

```
0: 0
1: 2
2: 4
3: 6
4: 8
```

```go
package main
import "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    fmt.Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    fmt.Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    fmt.Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
   Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i := 0; i < len(s); i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

## the infinite loop

```
package main
import . "fmt"

func main() {
  defer func() {
    recover()
  }()
  s := []int{0, 2, 4, 6, 8}
  for i := 0; ; i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```
package main
import . "fmt"

func main() {
  defer func() {
    recover()
  }()
  s := []int{0, 2, 4, 6, 8}
  for i := 0; ; i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  defer func() {
    recover()
  }()
  s := []int{0, 2, 4, 6, 8}
  for i := 0; ; i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```
package main
import . "fmt"

func main() {
  defer func() {
    recover()
  }()
  s := []int{0, 2, 4, 6, 8}
  for i := 0; ; i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

```go
package main
import . "fmt"

func main() {
  defer func() {
    recover()
  }()
  s := []int{0, 2, 4, 6, 8}
  for i := 0; ; i++ {
    Printf("%v: %v\n", i, s[i])
  }
}
```

## the range

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

## a functional interlude

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s []int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s []int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s []int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s []int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice(0, 2, 4, 6, 8)
}

func print_slice(s ...int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice(0, 2, 4, 6, 8)
}

func print_slice(s ...int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice(0, 2, 4, 6, 8)
}

func print_slice(s ...int) {
  for i, v := range s {
    Printf("%v: %v\n", i, v)
  }
}
```

## asserting type

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  for i, v := range s.([]int) {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  for i, v := range s.([]int) {
    Printf("%v: %v\n", i, v)
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  for i, v := range s.([]int) {
    Printf("%v: %v\n", i, v)
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  if s, ok := s.([]int); ok {
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  print_slice([]int{0, 2, 4, 6, 8})
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

## closures

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_slice(func(i int) int { return s[i] })
}

func print_slice(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  }
}
```

## upon reflection

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] * 2 })
}

func print_values(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] * 2 })
}

func print_values(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] * 2 })
}

func print_values(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```go
package main
import . "fmt"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] * 2 })
}

func print_values(s interface{}) {
  switch s := s.(type) {
  case func(int) int:
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s(i))
    }
  case []int:
    for i, v := range s {
      Printf("%v: %v\n", i, v)
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface{})
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i))
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface{})
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i))
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```go
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

```
package main
import . "fmt"
import . "reflect"

func main() {
  s := []int{0, 2, 4, 6, 8}
  print_values(s)
  print_values(func(i int) int { return s[i] })
}

func print_values(s interface{}) {
  switch s := ValueOf(s); s.Kind() {
  case Func:
    p := []Value{ ValueOf(i) }
    for i := 0; i < 5; i++ {
      Printf("%v: %v\n", s.Call(p)[0].Interface())
    }
  case Slice:
    for i := 0; i < s.Len(); i++ {
      Printf("%v: %v\n", i, s.Index(i).Interface())
    }
  }
}
```

## http://golang.org/

#golang