going loopy

adventures in iteration with google go

@feyeleanor

the conditional loop

```
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])
  }
}
```

```
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
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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])
  }
}
```

```
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}
  i := 0
  for {
    Printf("%v: %v\n", i, s[i])
    i++
  }
}
```

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
package main
import . "fmt"

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

```
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

```
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)
  }
}
```

```
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)
  }
}
```

```
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)
    }
  }
}
```

```
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)
    }
  }
}
```

```
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)
    }
  }
}
```

```
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)
    }
  }
}
```

```
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", i, 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", i, 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", i, 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", i, 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", i, 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", i, 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", i, 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", i, 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", i, 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", i, s(i))
    }
  }
}
```

channels

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int, 16)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

```
package main
import . "fmt"

func main() {
    c := make(chan int)
    go func() {
        for _, v := range []int{0, 2, 4, 6, 8} {
            c <- v
        }
        close(c)
    }()
    Printf("elements: %v\n", print_channel(c))
}

func print_channel(c chan int) (i int) {
    for v := range c {
        Printf("%v: %v\n", i, v)
        i++
    }
    return
}
```

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] })
}

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

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

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

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

user-defined type

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

```
package main
import . "fmt"

type Iterable interface {
  Each(func(interface{}))
}

type IterableSlice []int
func (i IterableSlice) Each(f func(interface{})) {
  for _, v := range i {
    f(v)
  }
}

func main() {
  s := IterableSlice{ 0, 2, 4, 6, 8 }
  i := 0
  s.Each(func(v interface{}) {
    Printf("%v: %v\n", i, v)
    i++
  })
}
```

you now know go

#golang

http://golang.org/