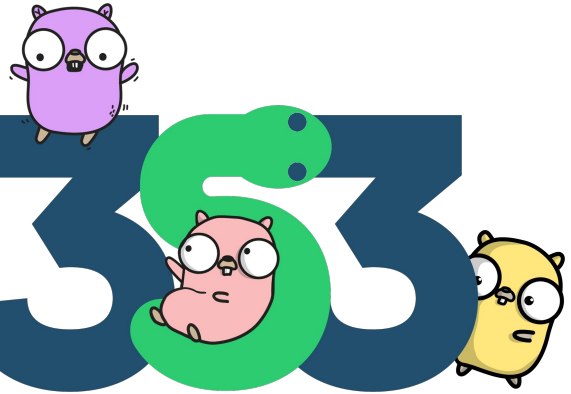
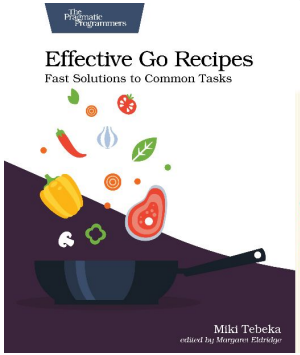
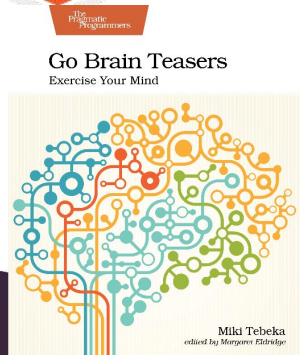
One Does Not Simply Iterate over Mordor

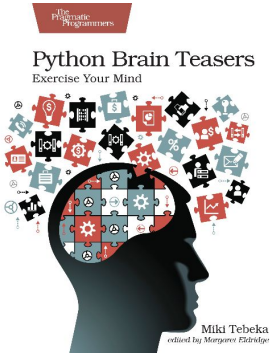
miki @tebeka

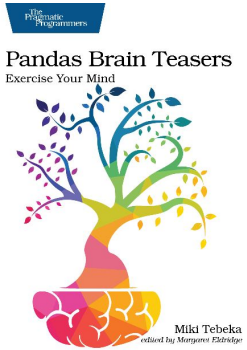
CEO,
CTO,
UFO











Range Over Function

Go 1.23, August 2024

Generators

Python 2.3, July 2003

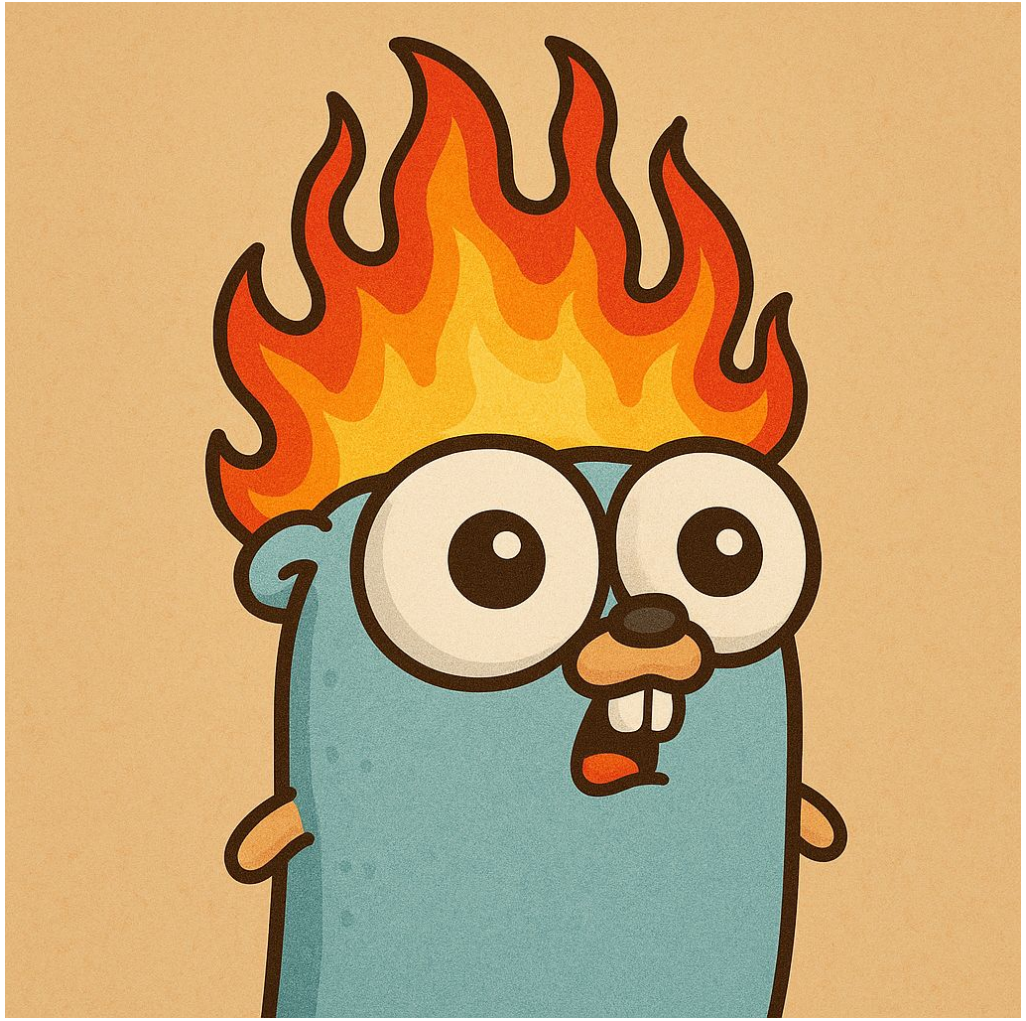
type Seq

```
type Seq[V any] func(yield func(V) bool)
```

type Seq2

```
type Seq2[K, V any] func(yield func(K, V) bool)
```

pkg.go.dev/iter



CODE

tick/tick.go

Range-Over Functions in Go
ArdanLab Blog

Load logs from compressed JSON file.

```
$ zcat logs.json.gz| wc
5350200 16045852 770589508
$ zcat logs.json.gz| head -1
{"host":"199.72.81.55","time":
"1995-07-01T00:00:01Z","reques
t":"GET /history/apollo/
HTTP/1.0","status":200,"bytes"
:6245}
```

CODE

log/log.go

CODE

slice/slice.go

CODE

bench/main.go

Map

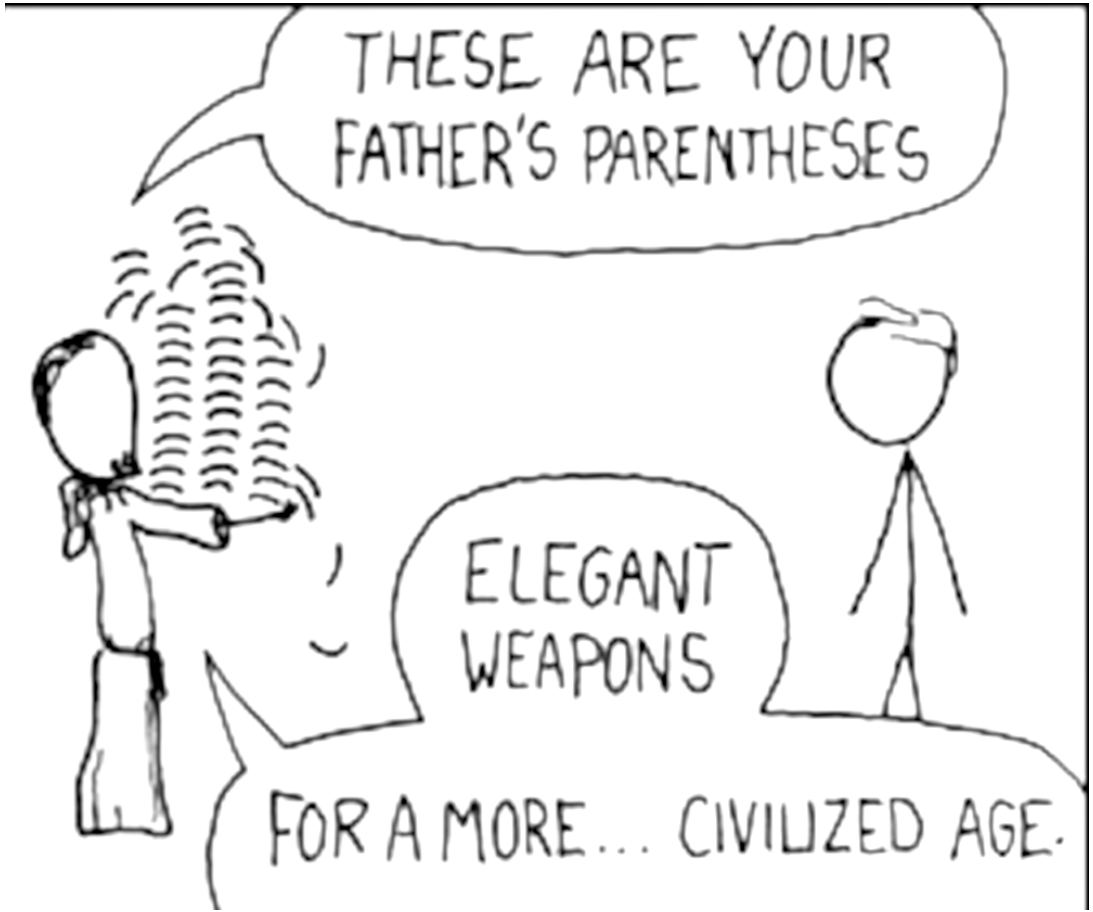
lower: ["Suite", "Up"] -> ["suite", "up"]

Filter

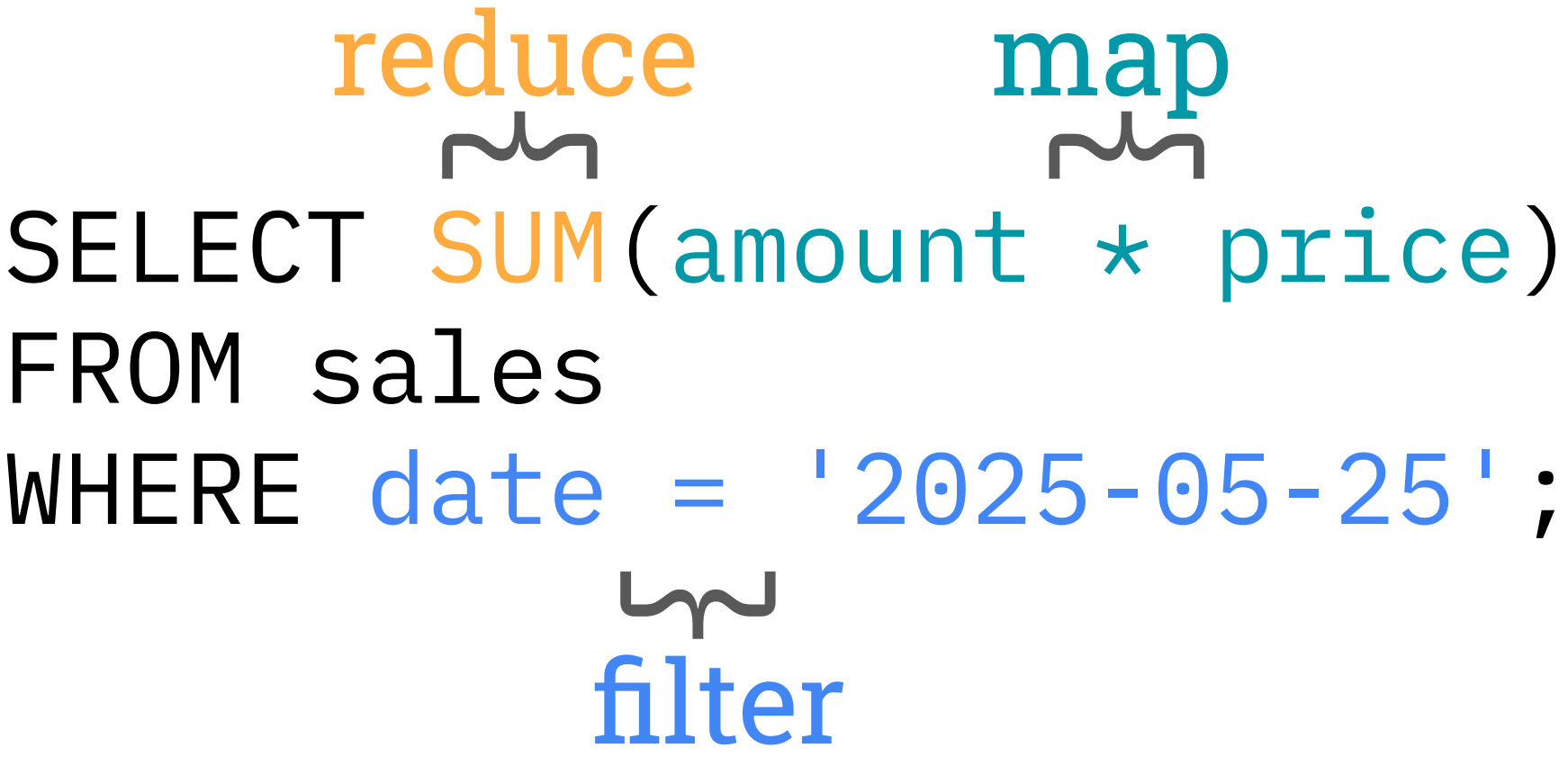
≥ 0: [2, -1, 3, 4] -> [2, 3, 4]

Reduce

sum: [1, 2, 3, 4] $\Rightarrow$ 10



https://xkcd.com/297/


reduce
map
SELECT SUM(amount * price)
FROM sales
WHERE date = '2025-05-25';
filter

CODE

lazy/lazy.go

CODE

push/push.go

CODE

pull/pull.go


QUIET

func Collect

```
func Collect[E any](seq iter.Seq[E]) []E
```

Collect collects values from seq into a new slice and returns it.

func Values

```
func Values[Slice ~[]E, E any](s Slice) iter.Seq[E]
```

Values returns an iterator that yields the slice elements in order.

pkg.go.dev/slices

func Values

```
func Values[Map ~map[K]V, K comparable, V any](m Map) iter.Seq[V]
```

Values returns an iterator over values in m. The iteration order is not specifie

func All

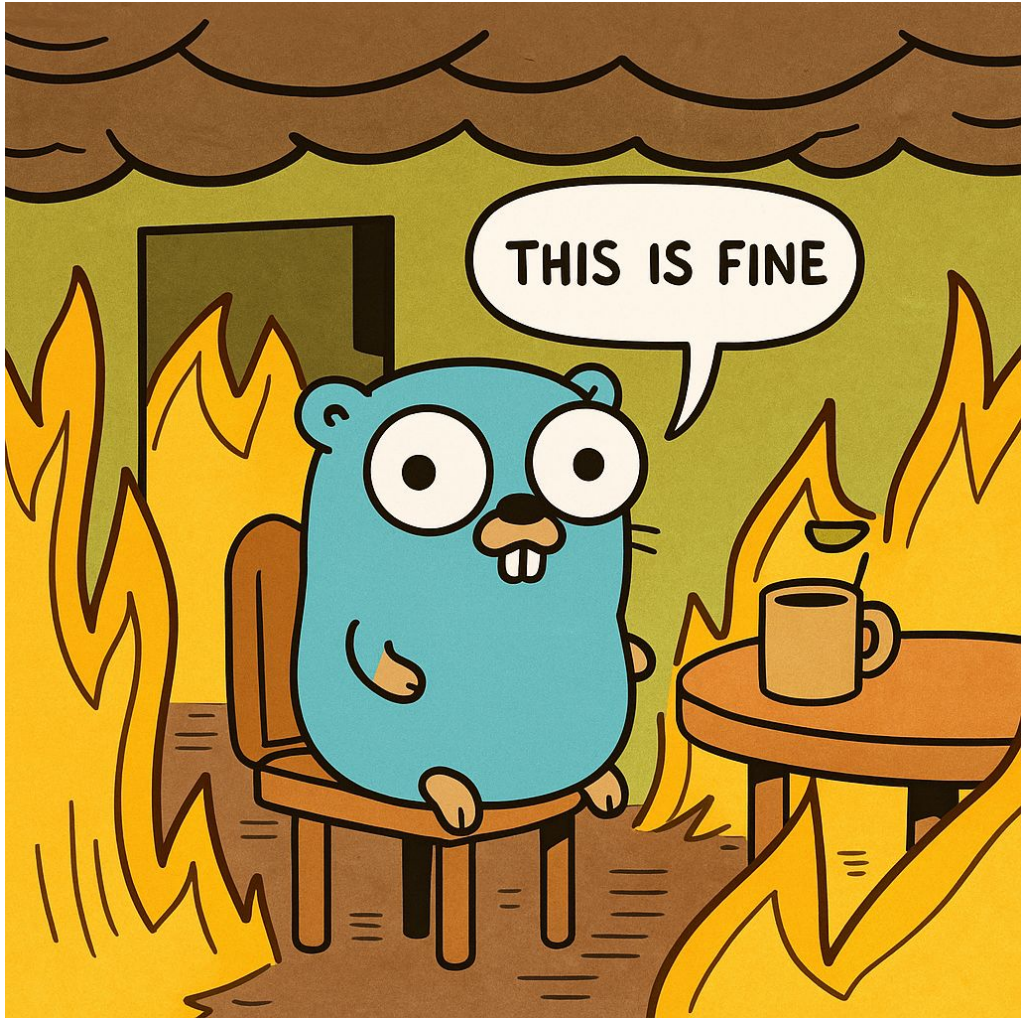
```
func All[Map ~map[K]V, K comparable, V any](m Map) iter.Seq2[K, V]
```

All returns an iterator over key-value pairs from m. The iteration order is not

pkg.go.dev/maps

Iterator	Arguments	Results	Example
accumulate()	p [,func]	p0, p0+p1, p0+p1+p2, ...	`accumulate([1,2,3,4,5]) → 1 3 6 10 15`
batched()	p, n	(p0, p1, ..., p_n-1), ...	`batched('ABCDEFG', n=3) → ABC DEF G`
chain()	p, q, ...	p0, p1, ... plast, q0, q1, ...	`chain('ABC', 'DEF') → A B C D E F`
chain.from_iterable()	iterable	p0, p1, ... plast, q0, q1, ...	`chain.from_iterable(['ABC', 'DEF']) → A B C D E F`
compress()	data, selectors	(d[0] if s[0]), (d[1] if s[1]), ...	`compress('ABCDEF', [1,0,1,0,1,1]) → A C E F`
dropwhile()	predicate, seq	seq[n], seq[n+1], start-ing when predicate fails	`dropwhile(lambda x: x<5, [1,4,6,3,8]) → 6 3 8`
filterfalse()	predicate, seq	elements of seq where predicate(elem) fails	`filterfalse(lambda x: x<5, [1,4,6,3,8]) → 6 8`
groupby()	iterable[, key]	sub-iterators grouped by value of key(v)	`groupby(['A','B','DEF'], len) → (1, A B) (3, DEF)`
islice()	seq, [start,] stop [, step]	elements from seq[start:stop:step]	`islice('ABCDEFG', 2, None) → C D E F G`

docs.python.org/3/library/itertools.html


THIS IS FINE

CODE

fibs/fibs.go

CODE

order/order.go

CODE

leak/leak.go

CODE

range/range.go

Questions?

github.com/tebeka/talks/tree/main/gcil-2025