CS Match is starting for registration

- necessary for registering in all CS courses except 111 and 201
- match alg tries to give each student 1 course
- reserves a spot for you during regular registration

Your match reservation lasts until the end of your priority registration time

---

Friday: CoK - everything since last one (no hashing)

I am leaving town after class on Wed through weekend/midterm break

What's hashing for?

ADT
- stack
- queue
- (lunch lender)
- list

- map/dictionary
- set

Map ≈ key ⇒ value

---

e.g. id ⇒ name

34987 → Becca
12345 → Dave
⋮

mydict[34987]
= 'Becca'
(Python)

Map ADT

- put(key, value)
- get(key)
- remove(key)

Set: same thing, but only keys

(if you like [but I don't], think of a Set as a Map where all the values are "true"

Set ADT

- add(key)
  - contains(key): Boolean
  - remove(key)

(Can think of a set like

34987 → true
12345 → true

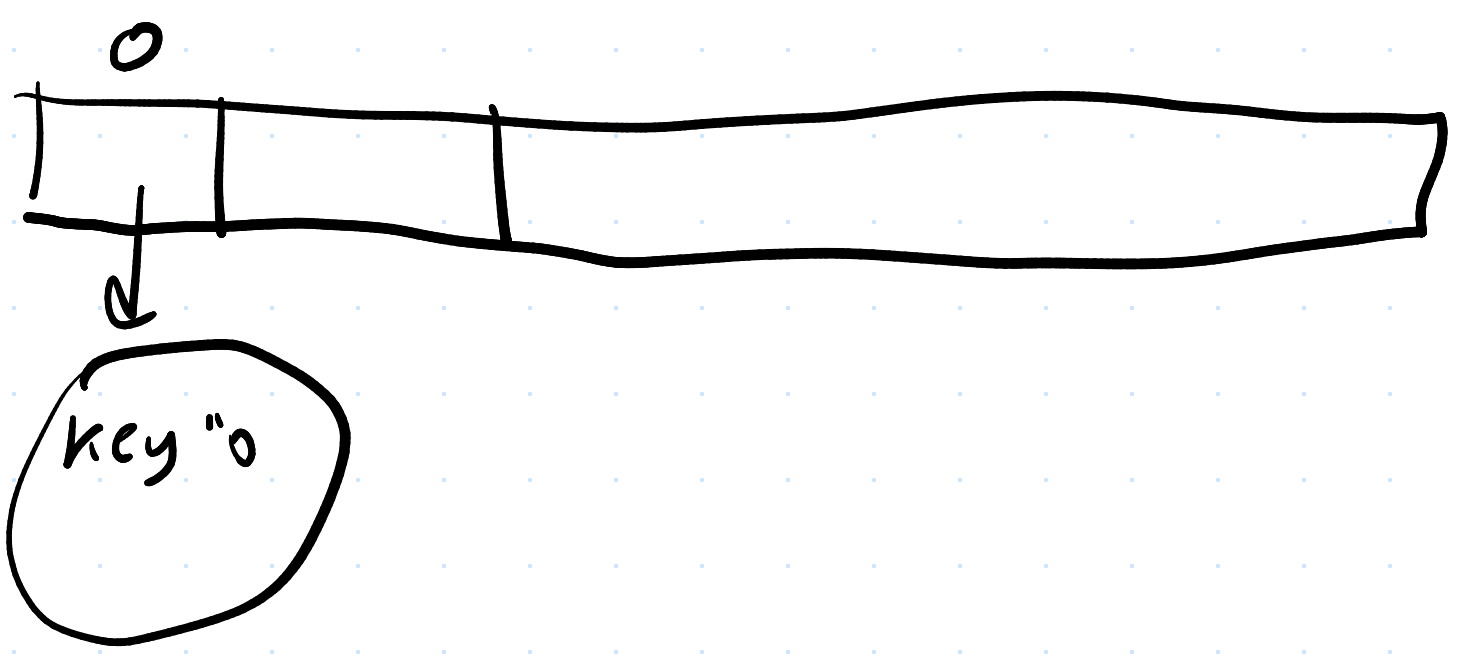
Implementing a set is the same as a map, you just ignore the values.

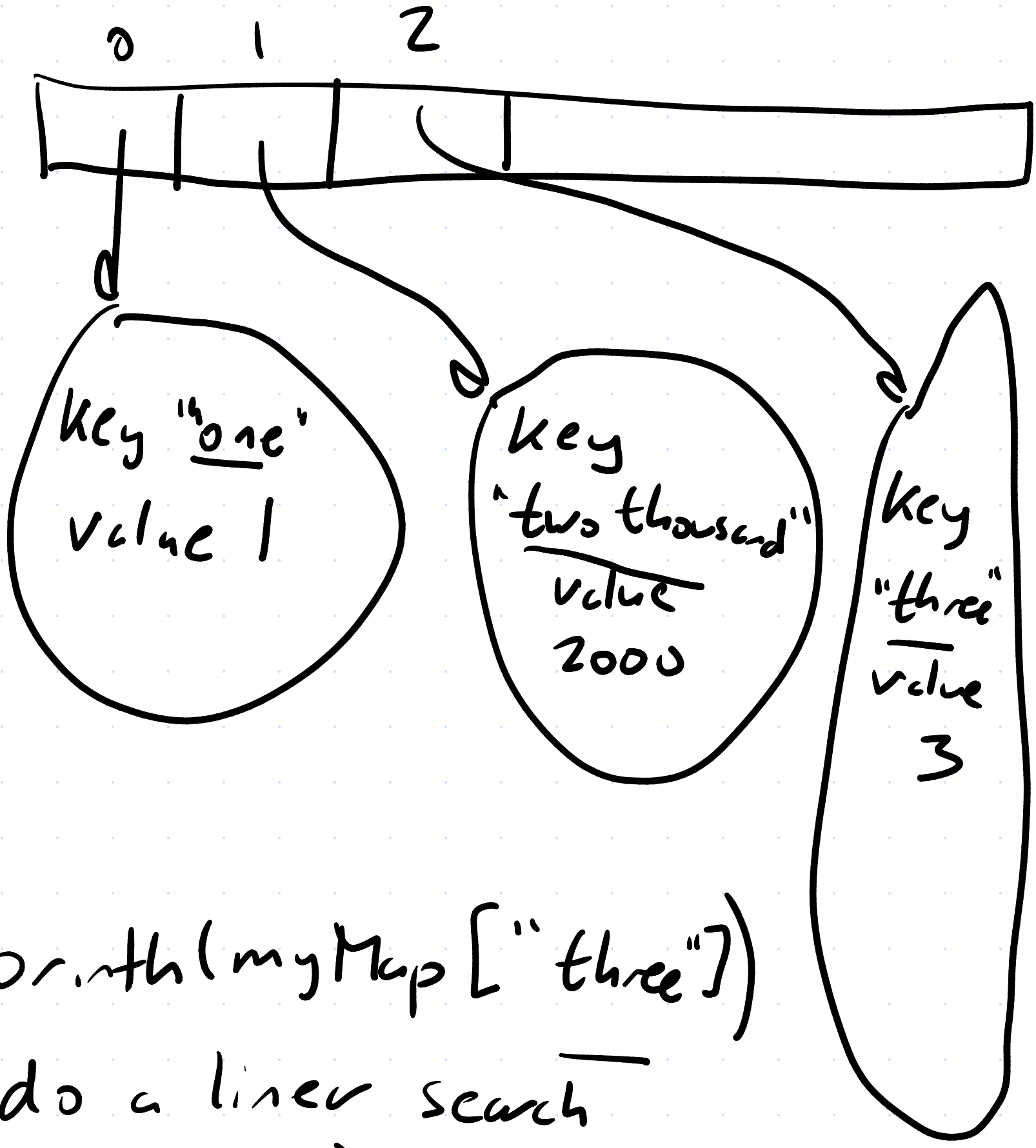
What we are doing is discussing how dictionaries/maps in Python / Java/Kotlin are actually implemented, inside

---

How would you build it?
- how do you make it fast?

Naive approach (not a great idea) - just put it in a list





print(myMap["three"])

do a linear search

O(n) to find anything

Why not keep data sorted by key?

- resort every time you insert
  BAD
- find spot, slide everything down
  O(n)

Hash tables: O(1) *(come back)
constat-time lookup

Why are lists fast?

```
val thing = listOf(— -- )
println(thing[2000])
```

In a computer that is designed like any of our computers are

- accessing any memory location by its address is constant-time lookup (takes a fixed amount of time)

For a list

```
thing[2000]
```

↑

we know where in memory the list starts, so we just add 2000 * size of each entry

But for a dictionary

myDict["dave"] = "pizza"

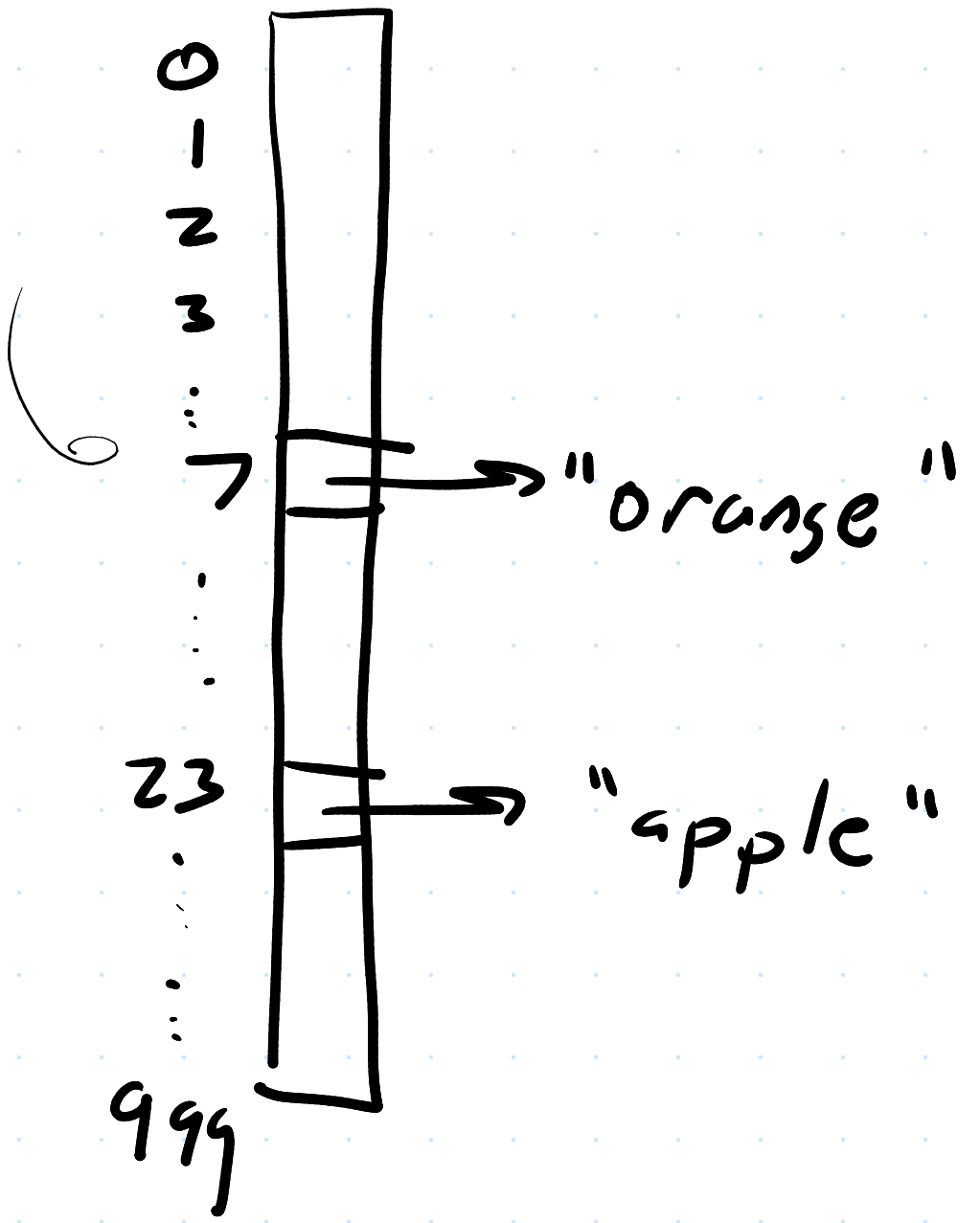
- there is no memory location "dave" so how do you jump to somewhere that doesn't exist?
- You could store them all in a list and search sequentially but that's slow.

Idea: suppose to keep it simple all keys range 0 ↔ 999

myDict[23] = "apple"

myDict[7] = "orange"

Make a list of size 1000



Lookup

mydict[7]

↳ "orange"

instant

(constant-time)

lookup

because it

is the

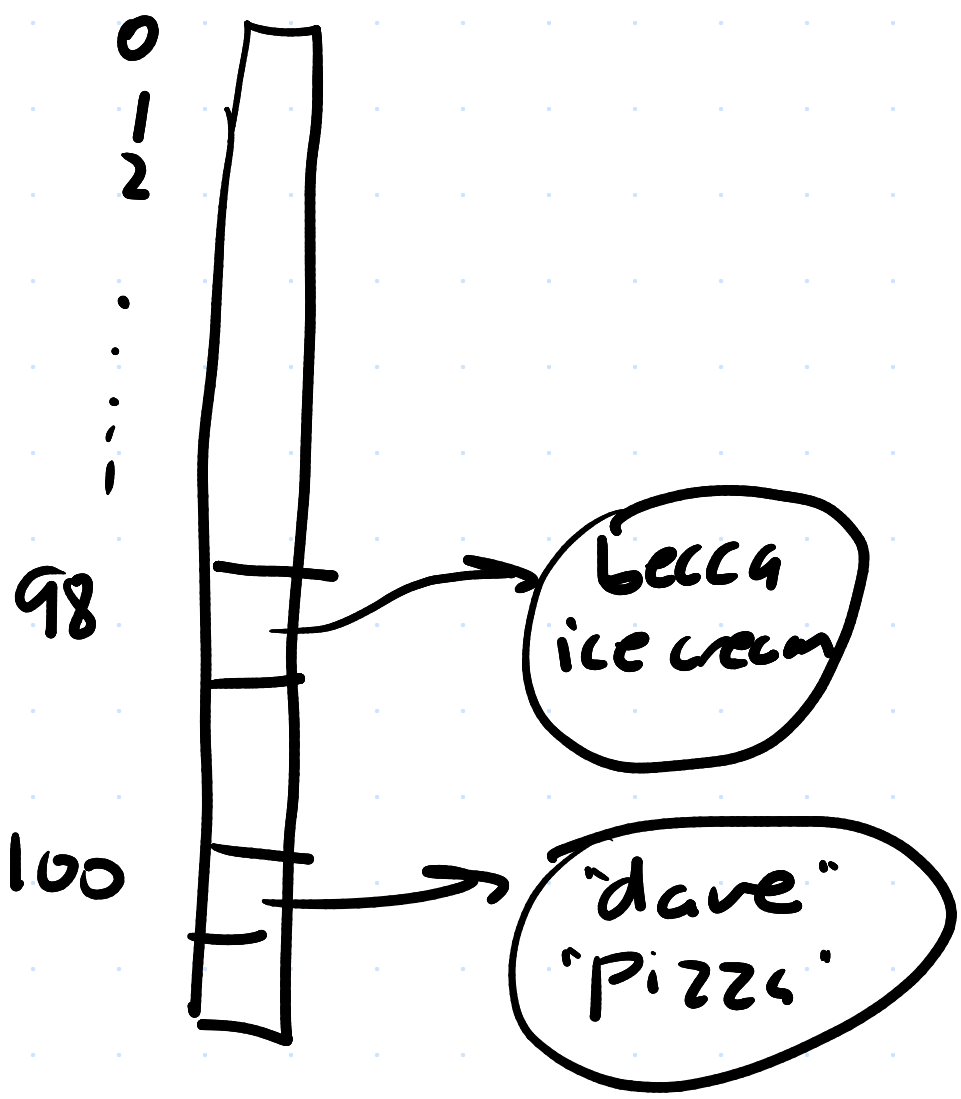
index of

a list

For things that can't ints,
turn them into ints!

myDict["dave"] = "pizza"

myDict["becca"] = "ice cream"



(but in values:

a → 97

b → 98

h("dave") = 100

h("becca") = 98

Hash function: converts key to an integer

Let's pick one! Idea: numeric value (Unicode value) of first letter

To find something late:

printh(myDict["dave"])

h("dave") = 100

↑

Unicode value of first letter

pizza!

Done! Fast!

Glorious!