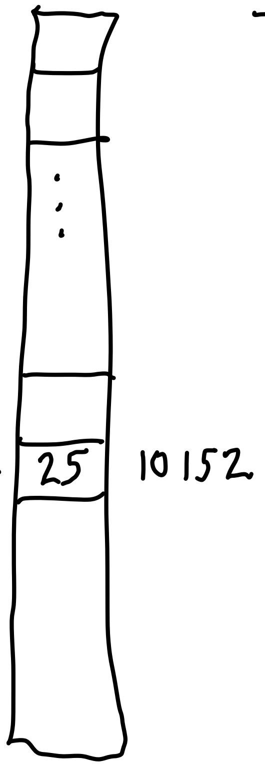
Why hash tables are fast
(dictionaries)

On insertion

h("salmon") = 10152

some function that returns an arbitrary number that is very different for similar strings

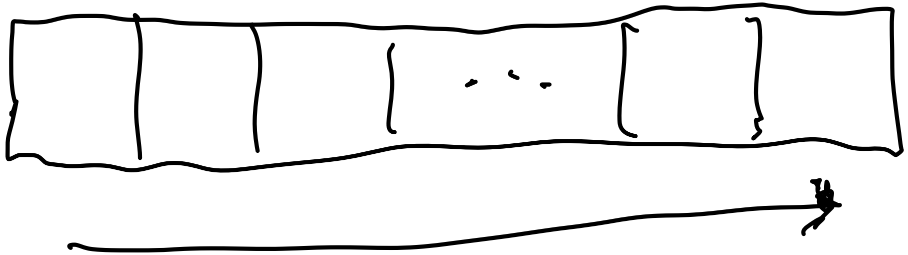


The Benefit:
we can retrieve values in time that depends only on the hash function, not size of data

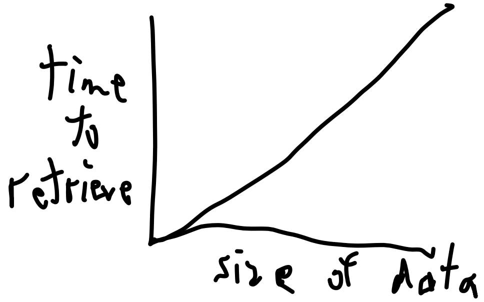
On lookup
h("salmon") = 10152

Contrast with lists

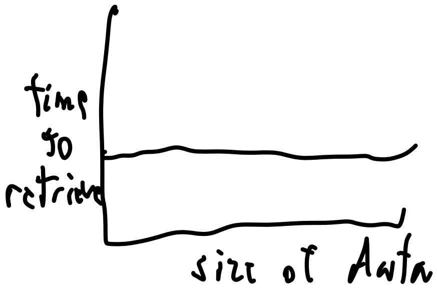
[("salmon", 25), ... 10,000 more items, ("mac and cheese", 18)]



linear time in the worst case

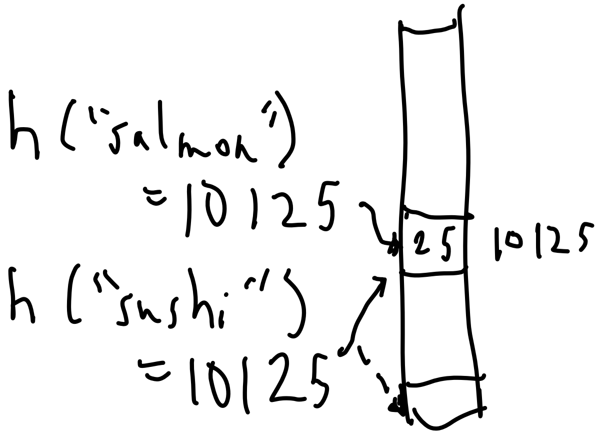


vs hash



What about <u>collisions</u>?

If an item would go to the same place as an item already in the array, Python uses "open addressing": it skips ahead some spaces and checks whether that space is free



In the worst case we keep colliding and keep looking for a new spot until we try the whole table.