Lecture 10, 10 September 2026

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Lists & dictionaries

Collections

List - indexed sequence of values

positions $0 .. n-1 \longrightarrow v_0 .. v_{n-1}$

Dictionary - values are indexed by keys

keys $k_0 .. k_{n-1} \mapsto v_0 .. v_{n-1}$

$k_i \to h(k_i) \longrightarrow v_i$ collisions

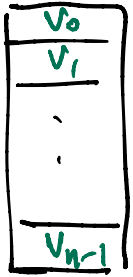
Lists

How is a list stored?

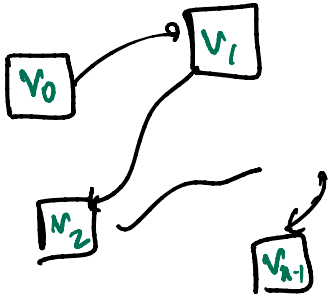
Sequence of values

- Arrays
- "List"

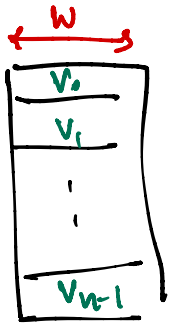
Array



List



Arrays



Storage is "contiguous" – No gaps

Fixed storage "width" for values

W

A[i] A[0] + i * w

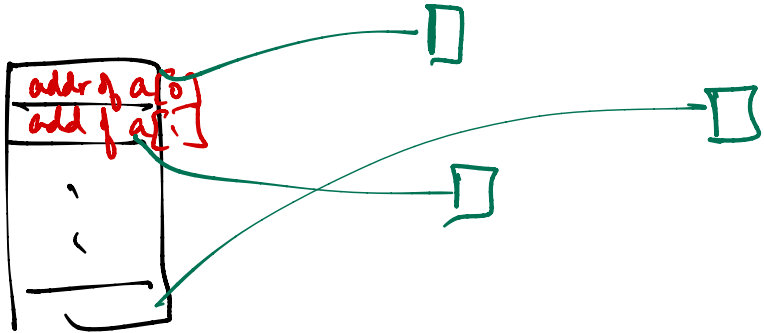
Random Access

Access A[i] in "uniform" time independent of i

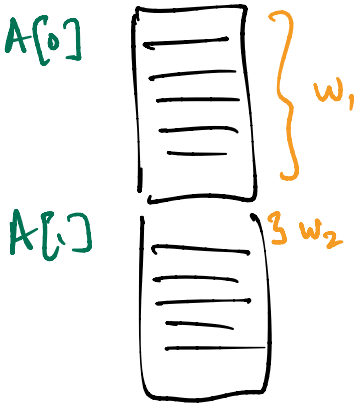
Arrays

Fixed width? In Python?

Arrays are normally homogeneous



Arrays



A[i][j]

i * w_1 + j * w_2

Arrays

\+ Random Access

\- Rigidness / inflexibility

- Insertion / deletion requires shifting a block — also slice update

`l.insert(j,v)`

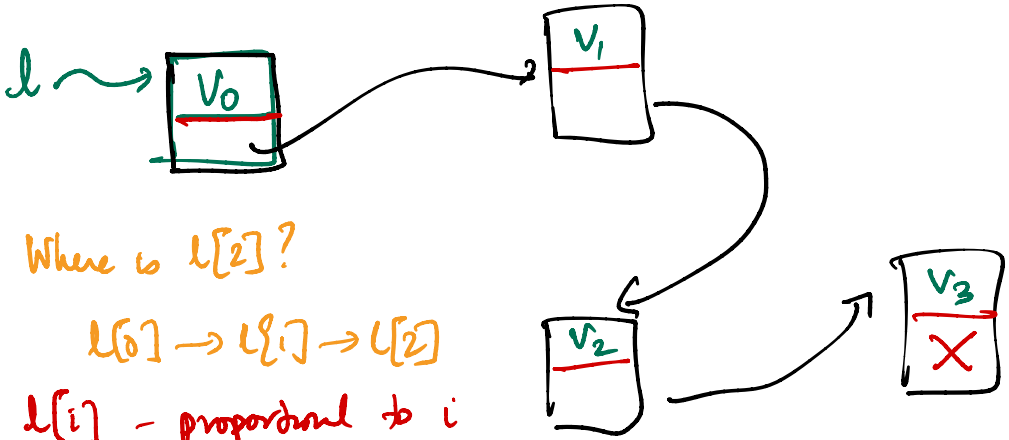
`l.insert(0,v)`

Size is fixed — normally fix type & size

Arrays

- Accessing `l[i]` — constant time
- Insert/delete — proportional to length

"List"



Where is l[2]?

l[0] → l[1] → l[2]

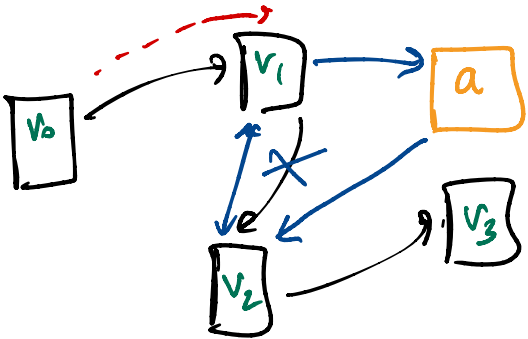
l[i] - proportional to i

List

Lose random access

Gain flexibility

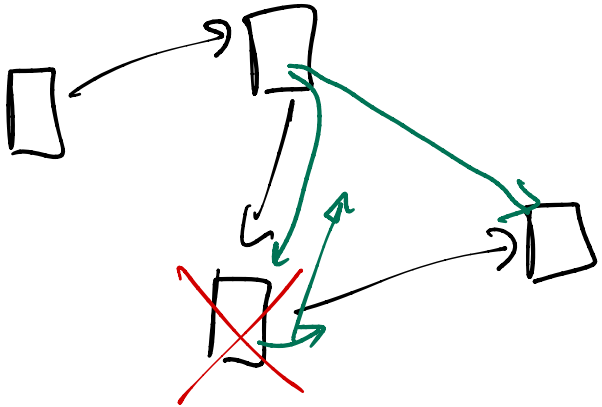
L.insert(2, a)



"Plumbing"

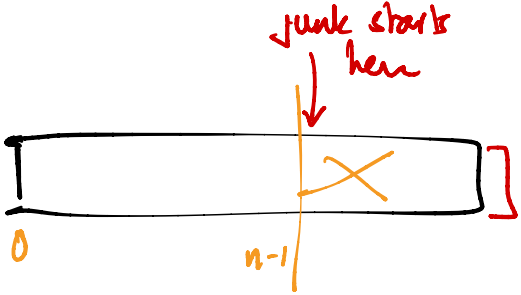
Lists

delete l[2]



What does Python do?

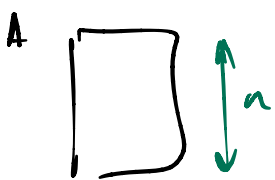
Array



Run out of space?

Double space & copy

Doubling?



"Amortized" analysis

Time proportional to n

Apportion this cost to each earlier append