Lecture 9, 12 September 2024

Madhavan Mukund

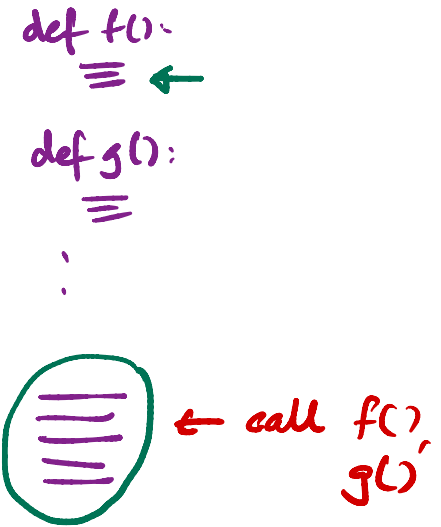
https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Scope of a variable

Local & global

Scope of a variable
- Where is it visible?



Scope

Variables defined inside a function are local to that function

```
def matsum(m):
    ans = 0
    for i in m:
        ans = ans + rowsum(i)
    return(ans)
```

```
def rowsum(r):
    rsum = 0
    for i in r
        rsum = rsum + i
    return(rsum)
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

[[1,2,3], [4,5,6], [7,8,9]]

Scope

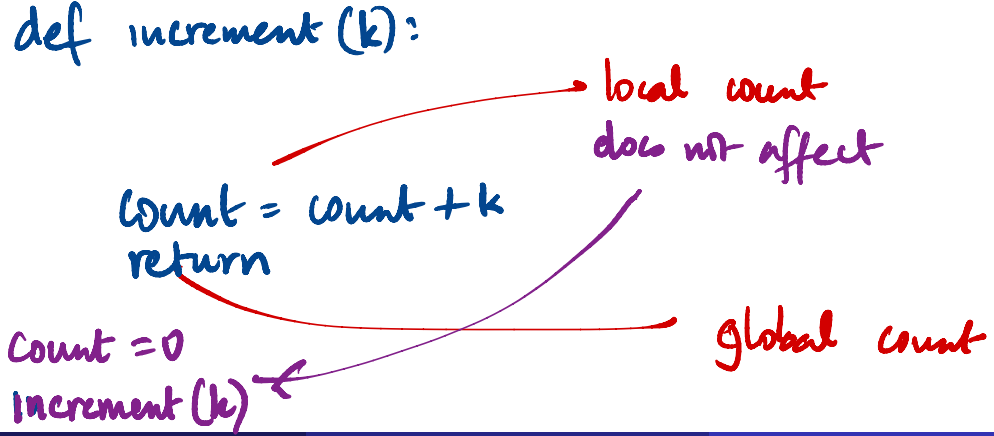
What if function refers to value not defined locally?

```
def f():
    y = x+22
    print(y)
    return

x = 17
f()
```

```
def f():
    y = x+22
    print(y)
    x = 33
    return

x = 17
f()
```

local

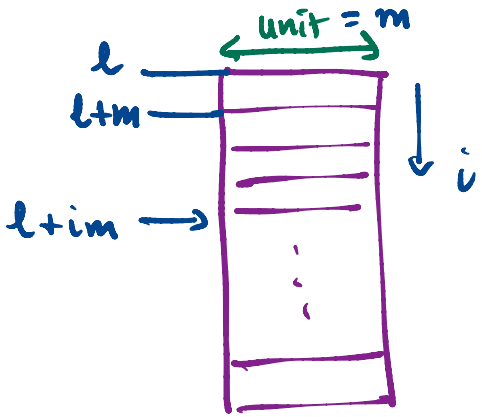
Scope


def increment(k):
local count
does not affect
count = count + k
return
global count
count = 0
increment(k)

```
def increment(k):
    global count
    count = count + k
    return

count = 0
increment(k)
```

Arrays, Lists, Dictionaries

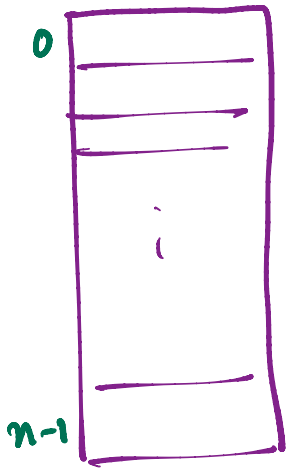
Array Contiguous sequence of equal size memory units



"Random access"

Each A[i] takes same amount of time to access

Arrays



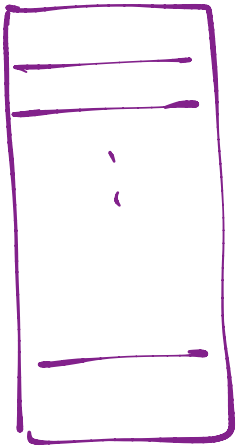
Suppose array has integers sorted in ascending order

Search if a given K exists in the array

Check A[mid] $mid = \frac{0+n}{2}$

Binary Search — Random access

Array



Growing & shrinking is expensive

l.insert(p,x)

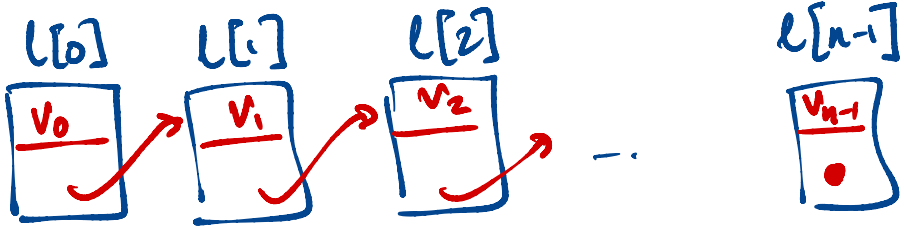
l[p] = x

All positions p, p+1, --

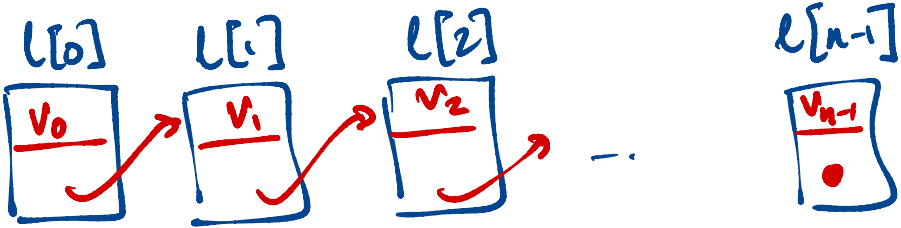
shift to p+1, p+2, --

List

Flexible sequence - train



"Linked" list

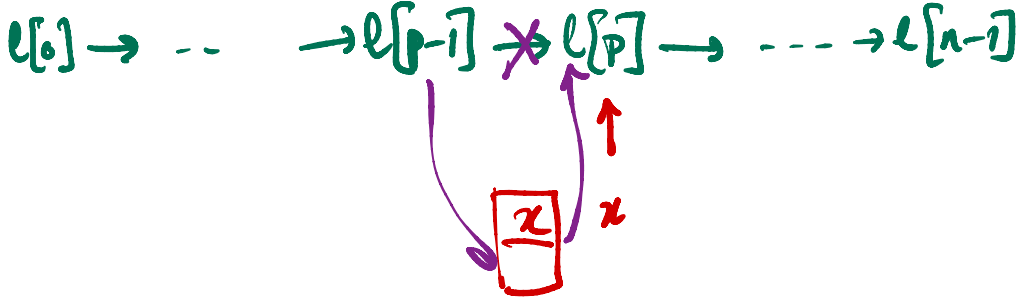


To access l[mid], traverse l[0] → l[1] ... → l[mid]

Not random access

Shrink or expand at l[p] (assume we are already at l[p])

l.insert(p, x)



Shrink

Remove L[p]

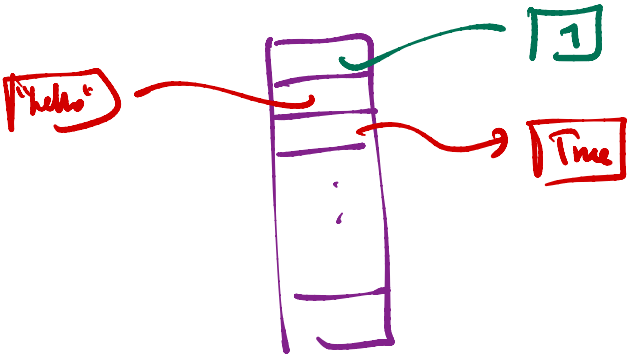
What are Python "lists"?

`l[p]=x` – suggests random access

`l.insert(8,22)`

Python lists are "flexible" arrays

- Uniform size of each element



Initially allocate a "large" array

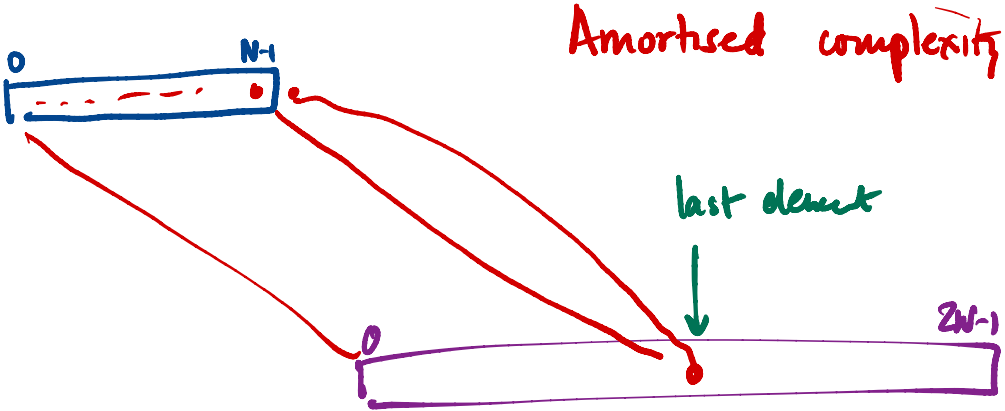
Start with a block of N contiguous locations

Upto N elements, no problem

Element N+1?

Allocate a new array of size 2N (double)

Copy


Amortised complexity
0
N-1
last element
0
2N-1