Lecture 5, 22 August 2024

Madhavan Mukund

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

Programming and Data Structures with Python

So far

Variables, types (int, float, bool), values

Assignment $v = e$

Control flow

Conditional execution - if - elif - else

Loops for, while

Function definition

Collections

Lists, tuples, dictionaries

List [1, 3, 8, 22]

- sequence

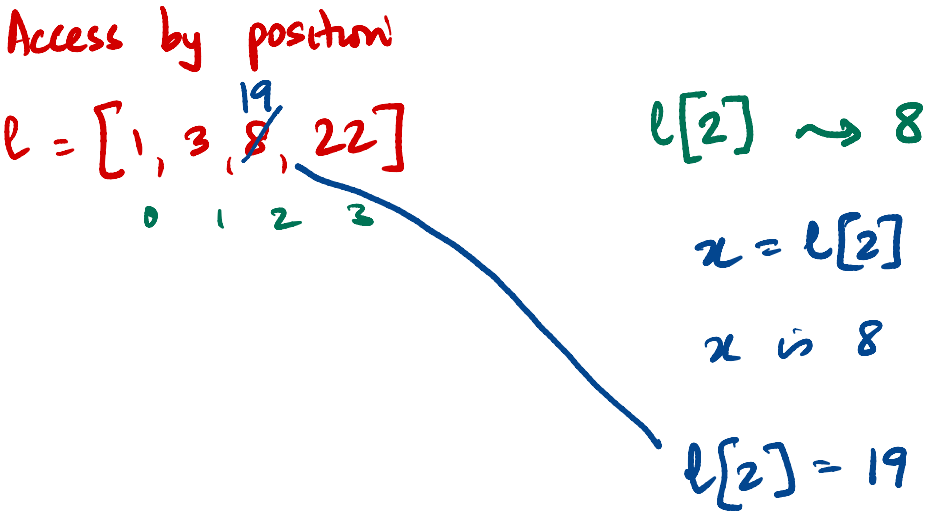
[1, 8, 3, 22]

1

positions

0 ... length − 1

Lists

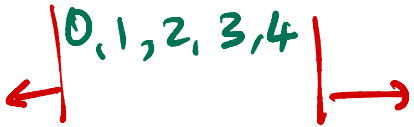


Lists

l = [1, 8, 17, 97, 200]

l[5] – IndexError

Cannot extend a list by



l[5] = 300 ✗

Lists

l = [1, 8, 17, 97, 200]

l[-1] should be IndexError

Last element l[len(l)-1]

len(l)

By convention l[-1] is l[len(l)-1]

l[-2] is l[len(l)-2]

⋮

l[-len(l)] is l[0]

Lists

plist = nprimes(100)

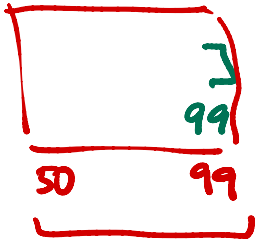
plist[-1] — largest of these = 100th prime

plist2 = primesupto(1000)

plist2[-1] — largest prime ≤ 1000

Lists

plist = nprimes(100) ⇝ [ ]



Want primes 51 to 100

plist[50:100]

"slice"

not included

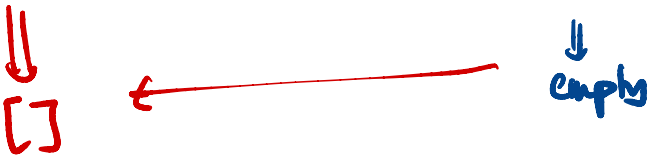
l[i:j] = [l[i], l[i+1], ..., l[j-1]]

list [i : j]

range (i, j)

list [50 : 49]

range (50, 49) ?



$l[4:len(l)+3]$ $l = [0, 1, 2, \ldots, 99]$

cannot ask for $l[len(l)+3]$

$l[4:len(l)]$ $\Rightarrow$ shortcut $l[4:]$

(remember range(n) 0..n-1

l[:7] → l[0] to l[6]

l[:] → all of l - full copy of l

range

range(n) 0, 1, 2 -- , n-1

range (i, j) i, i+1, --, j-1

Even numbers from 2 to 101

range(2, 101, 2)

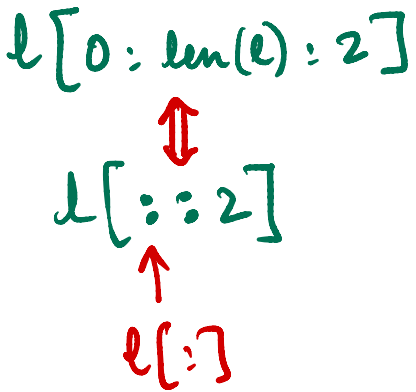
range (4, 23, 5)

4, 9, 14, 19 ~~24~~

Stop at last number that does not cross upper bound

Sequence 10, 9, 8, . . , −1

range(10, −2, −1)



l[0], l[2], l[4] ...

Reassign a slice

Positions change

```
l = list(range(20))
```

0 .. 19

```
l[7:10] = [27, 28, 29]
```

~ 0, .., 6, 27, 28, 29, 10 .. 19

```
l[7:10] = [27, 28, 29, 30]
```

0 .. 6, 27 .. 30, 10 .. 19

```
l[7:10] = []
```

0 .. 6, 10 .. 19

Useful facts

List concatenation : $l_1 + l_2$

```
l1 = [0,1,2]
l2 = [3,4,5]
l3 = l1+l2   - [0,1,2,3,4,5]
```

`l1 == l2` — same sequence of values

Useful fact

For any l, any j $\quad l = l[:j] + l[j:]$

Insert v at position j (i.e. between $l[j-1]$ & $l[j]$)

$$l = l[:j] + [v] + l[j:]$$

Delete l[j]

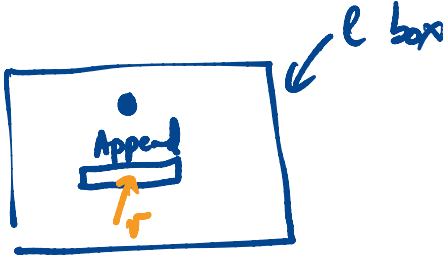
$$l = l[:j] + l[j+1:]$$

Append v to a list

```
l = l + [v]
```

```
l.append(v)
```

```
append(l,v)
```



l.reverse()

l.sort() → assuming sorting is sensible

Lists of uniform type

Python lists need not be uniform

[1, "CSE", True, 7.5]

l.insert(position, value)

l[:position] + [value] + l[position:]

Append a lot

l = l + newl

new list

l.extend(newl)

IN PLACE

`l.sort()` - updates `l`

Wanted `l` in sorted order, but without disturbing `l` itself

`lsort = sorted(l)`