Lecture 6, 26 August 2025

Madhavan Mukund

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

Programming and Data Structures with Python

Lists

Collective data types

List function

inplace

- `l.sort()`
- `l.reverse()`
- `l.append(v)` $\Rightarrow l = l + [v]$
- `l.extend(newl)` $\Rightarrow l = l + newl$

`sorted(l)`

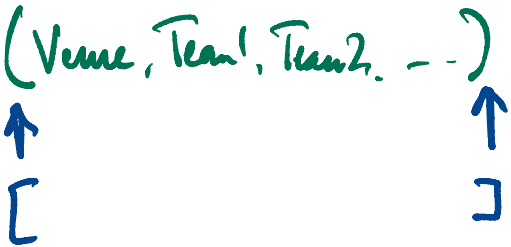
`x in l` — True if `x` is a value in `l`

Flexibility of lists

- Size
- Types of values

Tuples

Sequences, like lists



today = (26, 8, 2025)

date = today[0]

year = today[2]

datemon = today[0:2]

Cannot update a tuple

today[0] = 27 ✗ Not allowed

today = (27) + today[1:]

1 + (7)

(27,)

↑ to denote a singleton tuple

Multiple assignment

```
i = 0
j = 1
```

```
primelist, index = [], 2
```

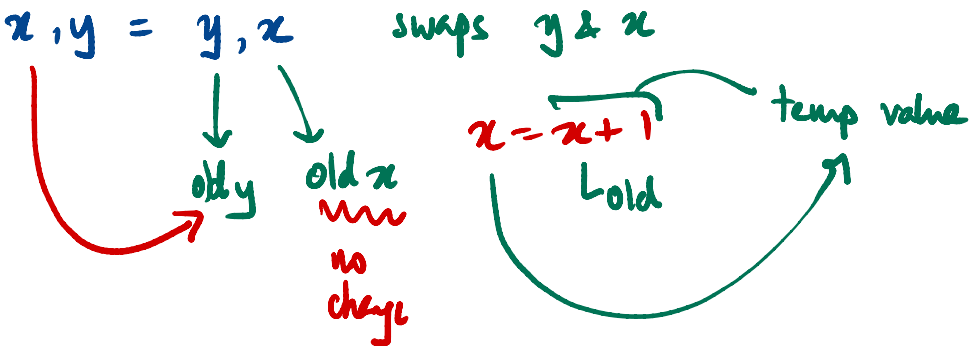
$(i, j) = (0, 1)$

$i, j = 0, 1$ also works

Initialization

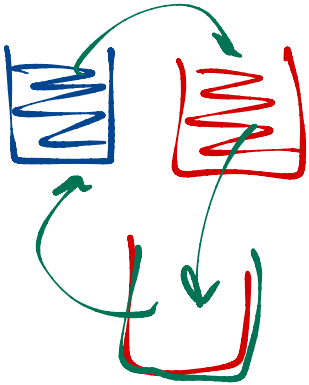
```
def nprimes(n):
    primelist = []
    index = 2
    while (len(primelist)<n):
        if isprime(index):
            primelist.append(.)
        index = index+1
```

Multiple assignment



$x = y$

$y = x$



```
temp = x
x = y
y = temp
```

vs. $x, y = y, x$

```
def swap(x,y):
    temp = x
    x = y
    y = temp
    return (y,x)
```

```
x = m
y = n
.
.
.
```

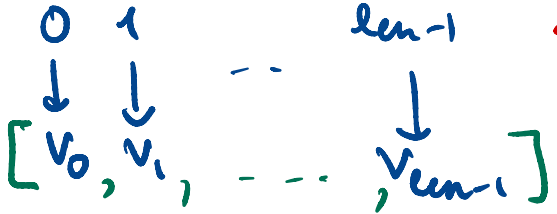
```
swap(m,n)
```

```
x,y = y,x
```

```
x,y = swap(x,y)
```

Dictionaries

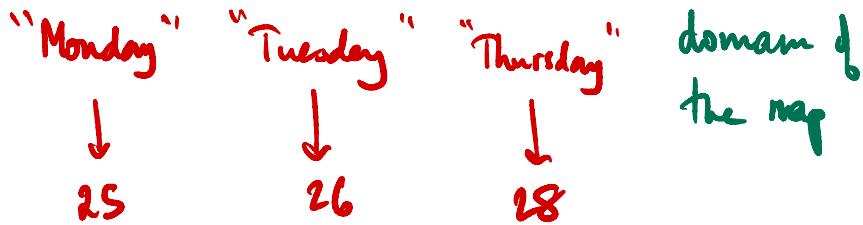
List, Tuple $f: [0, .., \text{length}-1] \rightarrow \text{Values}$



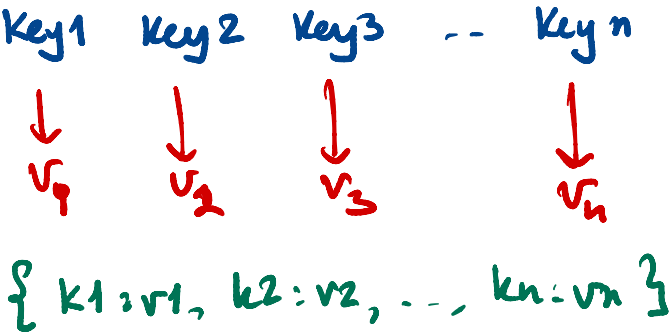
← no gaps

Dictionary :

Maps an arbitrary set to values — Keys = domain of the map



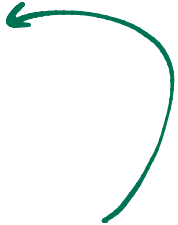
Dictionary



Empty dictionary

{ }

```
def nprimes(n):
    primelist = []
    while ...
        if ...
            primelist.append(v)
```



Initialize

Why is this legal?

d[k] → value of dict. d at key k

If k is not a key ⇒ KeyError

l = [10, 11, 12, 13]

l[4] = 14 ✗ IndexError

d = {0:10, 1:11, 2:12, 3:13}

d[4] = 14 ✓

Creates k:v on the fly

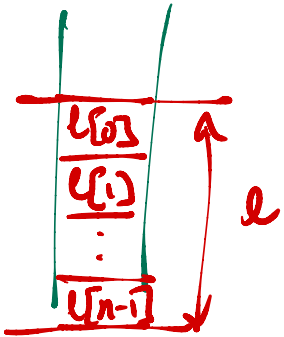
```
for k in d.keys():
```

```
for k in d:
```

Since Python 3.6, this sequence respects the order in which keys were added to d

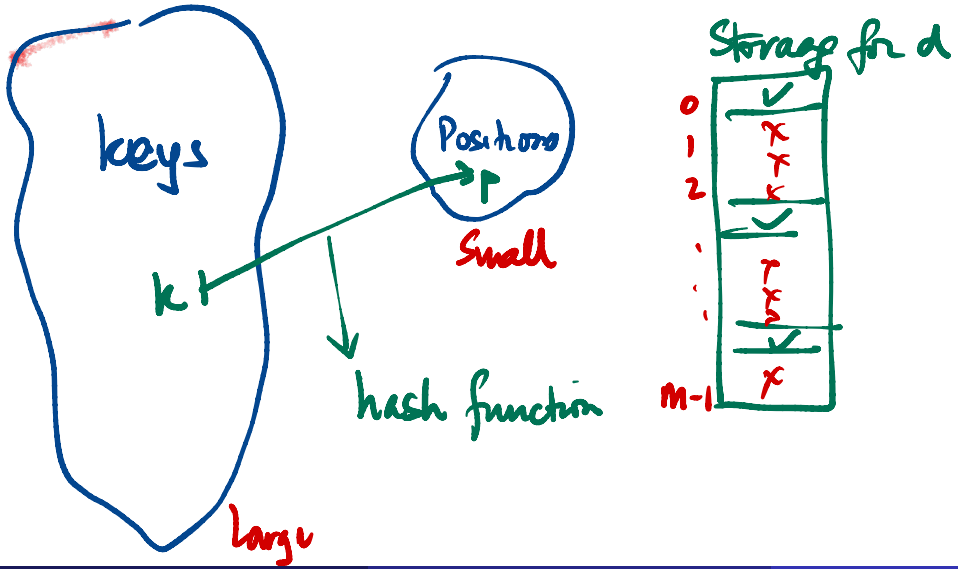
Why / how dictionaries work.

L = n elements



dictionary keys are arbitrary

Map keys to memory locations


keys
k
Positions
Small
hash function
Large
Storage for d
0
1
2
M-1

Cannot avoid $k \neq k'$, $h(k) = h(k')$ = "Collision"

"Good" hash fn → no collisions

Don't worry about this

$d[k] = v$

k in d

} constant time

Think about

Extracting unique elements from a list with repeated values