Lecture 6, 27 August 2024

Madhavan Mukund

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

Programming and Data Structures with Python

Collections

Lists - positions 0 .. n-1
-1 .. -n

slices i:j:k

l.append()
l.extend()
l.insert()
l.reverse()

l.sort()
vs
sorted(l)

Lists

v in l — Iteration

```
def element(l,v):
    for x in l:
        if l == v:
            return(True)
    return(False)
```

Worst case

$v \notin l$

Lists

```
def uniq(l):  # return unique elements in l
    newl = []
    for x in l:  # newl all unique values before x
        if not (x in newl):
            newl.append(x)
    return(newl)
```

Tuples

$(v_1, v_2, \ldots, v_k)$

even (v)

Typically not uniform

Iterate over a tuple

for x in t:

Membership

x in t

Positional indexing, slicing

print(..) prints output

Tuples

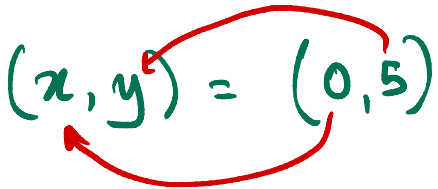
Cannot update a tuple

```
t[2] = 'RR'    - error
```

Convert tuples to sequences

```
list(t)
```

Tuples



$x = 0$
$y = 5$

```
def nprimes(n):
    primelist = []
    p = 2
    .
    .
    .
```

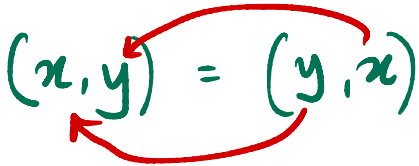
(primelist, p) = ([], 2)

Tuples

```
i=0
j=0
while i<n:
    :
    while j<m:
        :
```

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

Tuples




```
t = y
y = x
x = t
```

Swap values

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

works

Strings

City = "Chennai"

'Chennai'

7 vs "7"

statement = 'He said "This!"'

Even ''' or both ' & " '''

Strings

Strings are also sequences

s = "hello" t = "there"

s[i] is "e" No individual character type

s[1:4] is "ell"

s+t ⇝ "hellothere"

Strings

Cannot update a string in place — like tuples

for x is l:

list(t)

int(math.sqrt(n))

str(n) → " "

print(v)
displays str(v)

Strings

int("77") ⇝ 77

int("hello") ✗

Dictionaries

Lists, tuples, strings - sequences

$f: 0 \rightarrow v_0$
$1 \rightarrow v_1$
$\vdots$
$n-1 \rightarrow v_{n-1}$

$d: \{k_1 \rightarrow v_1,$
$k_2 \rightarrow v_2,$
$\vdots$
$k_m \rightarrow v_m\}$

Dictionaries

d = { k1 : v1 , k2 : v2 , .. , km : vm }

key value

d.values()

d.keys() — sequence of keys

d means d.keys()

Dictionaries

Frequency count (letters in a string)

```
def frequency(s):
    d = {}
    for c in s:
        if c in d.keys(): # if c in d
            d[c] = d[c]+1
        else:
            d[c] = 1      # create a new key c
```

'hello'

{'h': 1, 'e': 1, 'l': 2, 'o': 1}

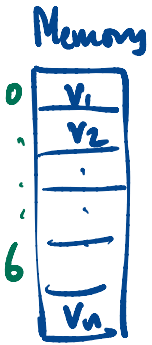
Dictionaries

Hash tables

List

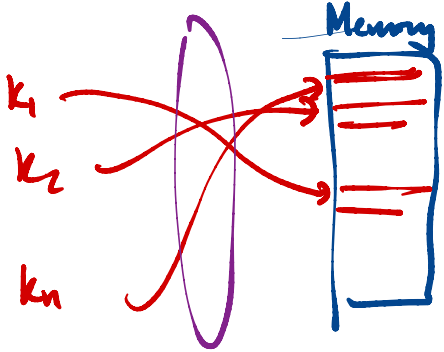
$l = [v_1, v_2, \ldots, v_n]$

$l[6]$



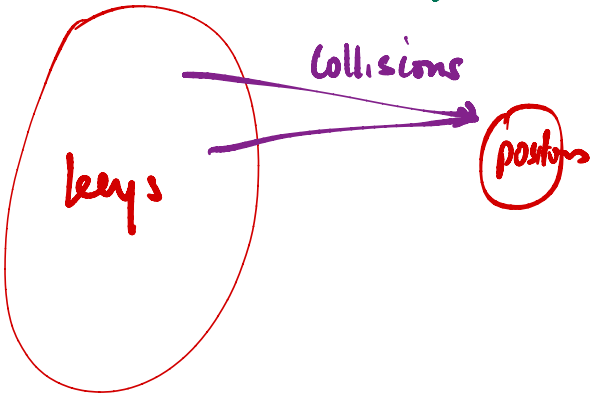
Dictionary

key ⇝ position



Dictionaries

Map : keys → positions



Good function that avoids / minimizes collisions

Hash function