CS 61A
DISCUSSION 6

LINKED LISTS

Raymond Chan
Discussion 134
UC Berkeley Fall 16

AGENDA

- Announcements
- Linked Lists
- Quiz

ANNOUNCEMENTS

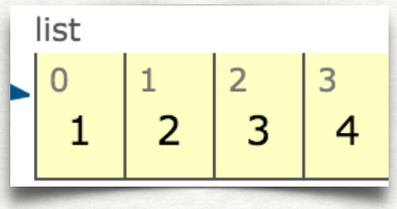
- Midterm 2 next Thursday 8-10pm!
  - Staff and HKN review sessions (check Piazza)
- Ants due Friday (submit early for extra point)
- No discussion next week before midterm. Will hold review session during section.
- Lab 7 due Friday

LINKED LISTS

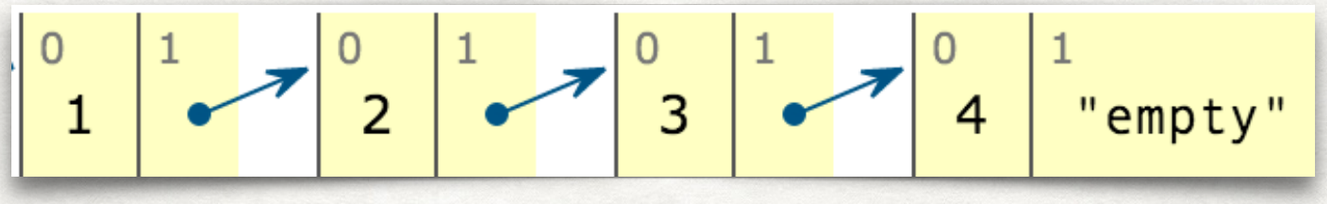
- A type of sequence that connects multiple *links*.
- Each link has *first* instance attribute and a *rest* attribute.
  - The last link has “empty” as the rest element.
- Think of connected chains with each chain containing information.

LINKED LISTS

Python List



Linked List



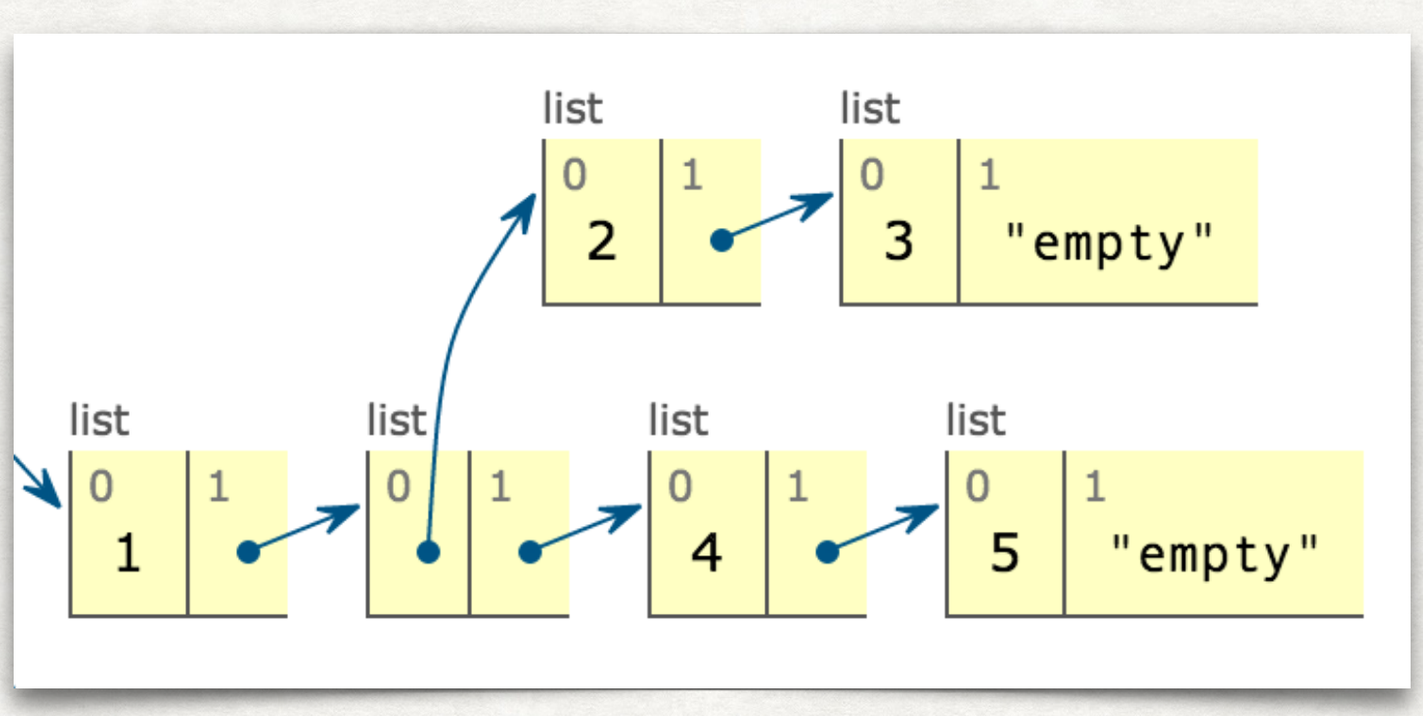
To form this linked list, use the constructor:

```
Link(1, Link(2, Link(3, Link(4, Link.empty))))
```

Linked lists are created from the back.

LINKED LISTS

- Linked lists can be **deep**.
- The first element can also be another linked list.

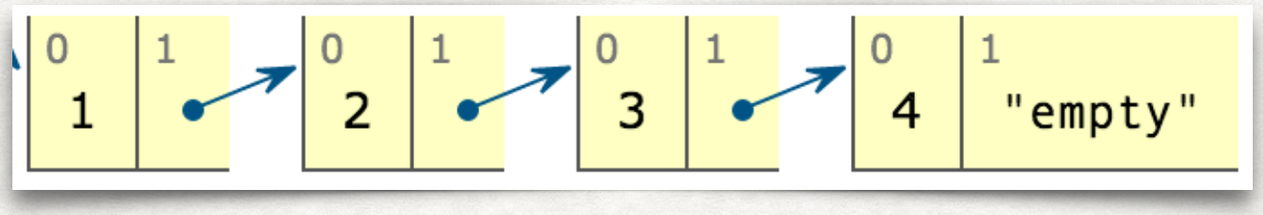


LINKED LISTS

- For each link box, you need to call the Link constructor.
  - Recursive data structure.
- Instance attributes **l.first** (**self.first**) obtains the first element and **l.rest** (**self.rest**) obtains the rest of the elements of the linked list **l**.
- **.rest** always returns another linked list.
- An empty linked list (**Link.empty**) is considered a linked list.

LINKED LISTS

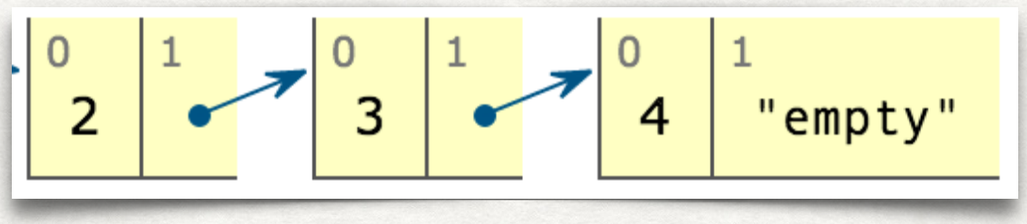
- It is very natural to use recursion for linked lists as we can split it up to **l.first** and **l.rest**



l.first

1

l.rest



LINKED LISTS

- When writing functions for linked lists, avoid using **len** and indexing. This won't be allowed on the midterm.
- If you have a recursive function and do the body access some link down the linked list, make sure each link is not empty.
  - i.e. using **link.rest.rest.first**
  - Make sure that **link**, **link.rest**, and **link.rest.rest** is not Link.empty *in that order*.

WWPP - LINKED LISTS

```
def f(l, n):
    s = '< '
    while n > 0 and l != Link.empty:
        s += str(l.first) + ' '
        l = l.rest
        n -= 1
    print(s + '>')

link = Link(1, Link(2, Link(3, Link(4, Link(5, Link(6))))))
linkA = link.rest
linkB = link.rest.rest
linkC = link.rest.rest.rest.rest

link.rest.rest, linkB.rest = linkB.rest, link.rest.rest
link.rest.rest.rest = linkC.rest.rest
linkC.rest.rest = linkC
link.rest = linkC.rest
linkC.rest = link

f(link, 5)
f(linkA, 5)
f(linkB, 5)
f(linkC, 5)
```

WWPP - LINKED LISTS

```
>>> f(link, 5)
< 1 6 5 1 6 >
>>> f(linkA, 5)
< 2 4 >
>>> f(linkB, 5)
< 3 3 3 3 3 >
>>> f(linkC, 5)
<5 1 6 5 1>
```

Function f prints out the first **n** elements of the linked list.

Draw a box and pointer diagram.