CS 61A SU26

Lecture 16: Linked Lists

July 20, 2026

Rebecca Dang

Join pollev.com/rebeccadang831 (or scan QR code) on your phone or laptop

We'll begin at Berkeley time (10 minutes after), as per tradition!

Potpourri

5 min

- Announcements
- Computing in the News

Announcements

- **Midterm regrade requests will be resolved by end of day Tue 7/21**
- Sign up **for exam support meetings** for advice on studying (happening this week!)
- Reminder: Due to the Soda → Gateway building move, **some discussion sections will be in Grimes or McLaughlin starting this week**
- Reminder: Quiz 3 logistics → Using the coding question autograder for any other questions on the quiz is considered **academic misconduct**.
- **Quiz 4** reservations are out, please sign up ASAP

Computing in the News

- Netflix Paid $587 Million for Ben Affleck's AI Company (The Hollywood Reporter)
  - About 300 Netflix Titles Used Generative AI This Year, Company Reveals (Variety)
    - "Netflix told shareholders in a letter that the technology's usage expands across every level of a program's production process, from its concept and pre-visualization to post-production and release."
- 4,500 Google Employees Demand Job Security Amid Big Tech Layoffs (KQED)
- Databricks secures funding round at $188 billion valuation (Reuters) → Series M??

Linked Lists

15 min

- Motivation
- Definitions
- `Link` class

Motivation

- Python has a very convenient built-in lists data structure that is useful for many cases, **but not all of them**.
- Python lists are implemented as "dynamic arrays" which means that **inserting into a random location of the list** (especially to the front of the list) **is memory-inefficient**
  - More on this if you take CS 61B!
- Enter: Linked Lists
- Demo: https://github.com/phrdang/cs61a-su26-lec16
  - Idea/code credit: Pamela Fox, former CS 61A Lecturer (FA21)
  - Python list: 0.68 seconds
  - **Linked list: 0.00095 seconds**

Definitions (1 of 3)

A **linked list** is an ordered data structure that is either:

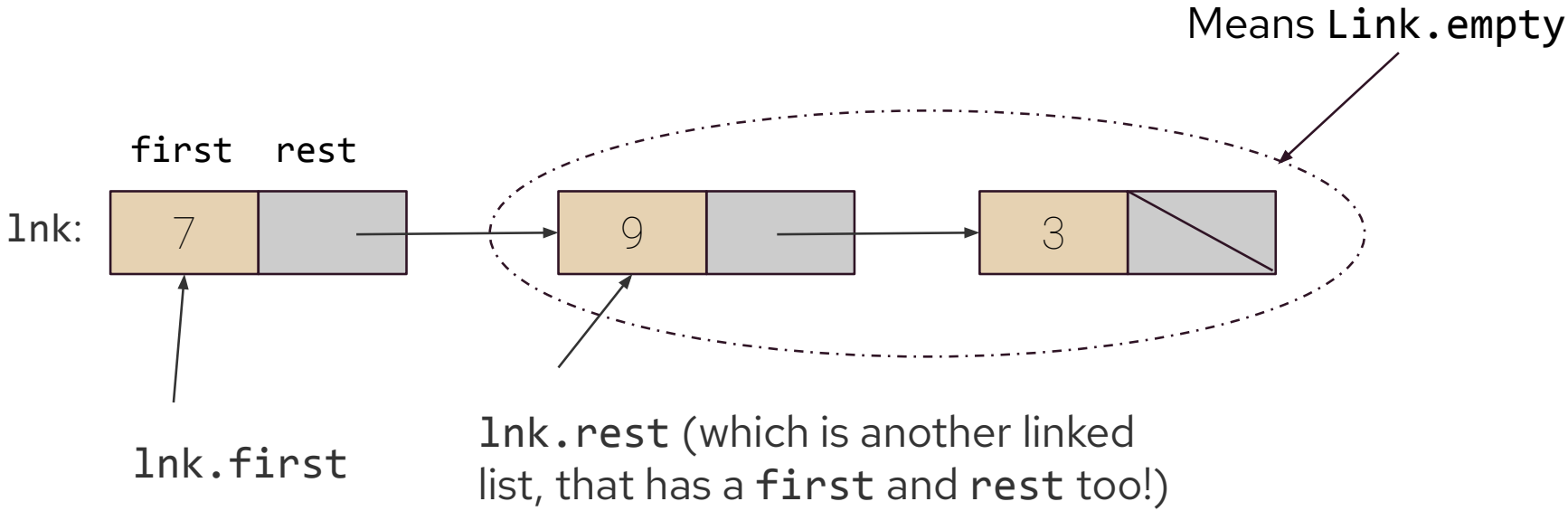
1. **Empty,** or
2. Contains a **first** value and the **rest** of the linked list

Metaphor: Links in a chain

In CS 61A, we represent a linked list using our own `Link` class.

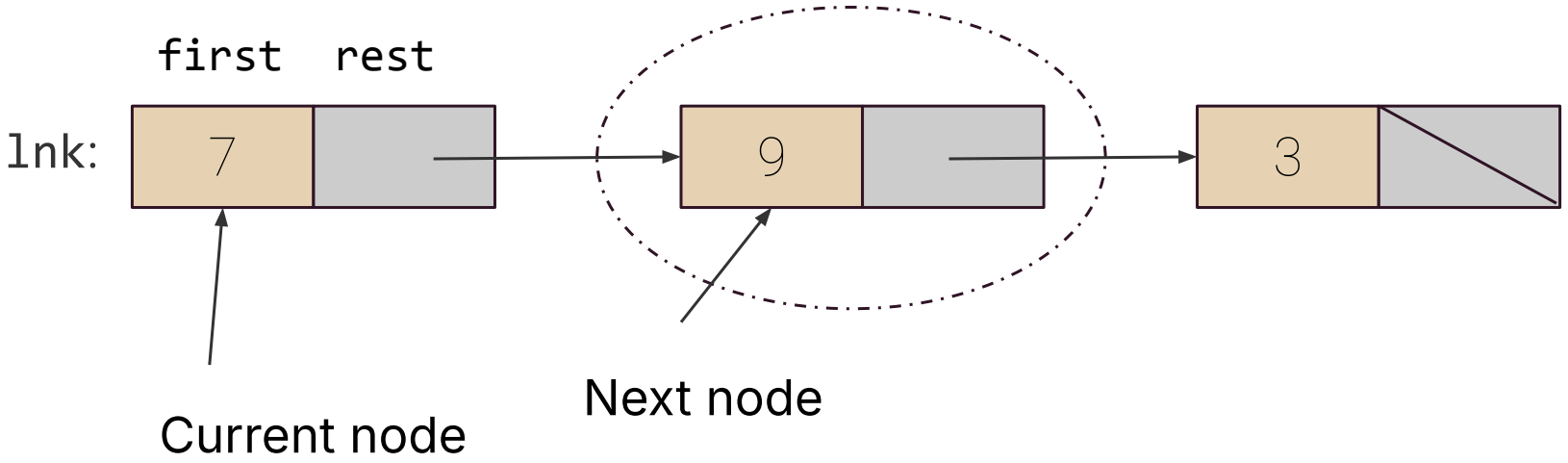
Definitions (2 of 3)

Like trees, linked lists are recursive data structures which can be thought of both **recursively** or relatively. Suppose we want to store 7, 9, and 3:



Definitions (3 of 3)

Like trees, linked lists are recursive data structures which can be thought of both recursively or **relatively**. Suppose we want to store 7, 9, and 3:



Link class (1 of 4)

```
class Link:
    empty = ()

    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

Link class (2 of 4)

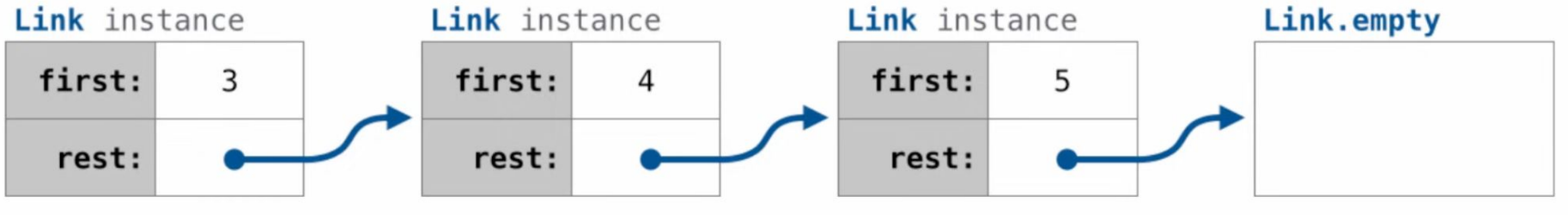
```
class Link:
    # see previous slide

    def __repr__(self):
        if self.rest is not Link.empty:
            rest_repr = ', ' + repr(self.rest)
        else:
            rest_repr = ''
        return 'Link(' + repr(self.first) + rest_repr + ')'

    def __str__(self):
        string = '('
        while self.rest is not Link.empty:
            string += str(self.first) + ' '
            self = self.rest
        return string + str(self.first) + ')'
```

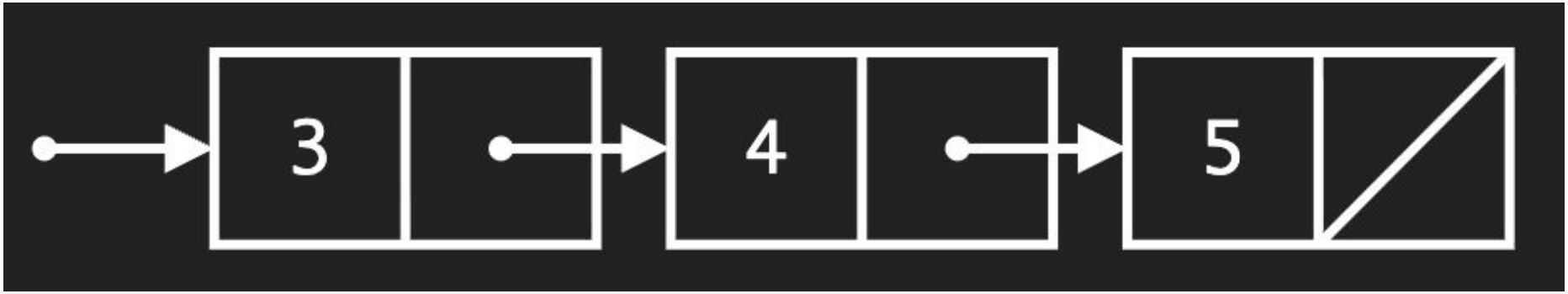
Link class (3 of 4)

```
>>> s = Link(1)
>>> s.first
1
>>> s.rest is Link.empty
True
>>> s = Link(3, Link(4, Link(5)))
>>> s
Link(3, Link(4, Link(5)))
>>> print(s)
(3 4 5)
```

Link class (4 of 4)

```
s = Link(3, Link(4, Link(5)))
```





WWPD

10 min

(For all exercises, assume the `Link` class has already been defined)

- WWPD #1
- WWPD #2
- WWPD #3

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

WWPD #1: What would be displayed in the Python interpreter when you execute print(Link(Link(5), 6))?

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

WWPD #2: What would Python display?

When poll is active respond at **PollEv.com/rebeccadang831**

Activate this Poll with the Poll Everywhere Live app.

If video conferencing, you must be sharing your full screen to share the activated poll.

WWPD #3: What would Python display?

Code Examples

10 min

- Length
- Double

Length (1 of 2)

```
def length(lnk: Link) -> int:
    """Returns the length of a linked list"""
    if lnk is Link.empty:
        return 0
    return 1 + length(lnk.rest)
```

Length (2 of 2)

```
def length(lnk: Link) -> int:
    """Returns the length of a linked list"""
    curr = lnk
    total = 0
    while curr is not Link.empty:
        total += 1
        curr = curr.rest
    return total
```

Double (1 of 2)

```
def double(lnk: Link):
    """Doubles a linked list in place (e.g. only mutates it)
    >>> s = Link(2, Link(3, Link(4)))
    >>> double(s)
    >>> s
    Link(4, Link(6, Link(8)))
    """
    if lnk is not Link.empty:
        lnk.first *= 2
        double(lnk.rest)
```

Double (2 of 2)

```
def double(lnk: Link) -> Link:
    """
    Returns a new linked list which is
    the result of doubling the input
    >>> s = Link(2, Link(3, Link(4)))
    >>> double(s)
    Link(4, Link(6, Link(8)))
    >>> s
    Link(2, Link(3, Link(4)))
    """
    if lnk is Link.empty:
        return Link.empty
    else:
        return Link(lnk.first * 2, double(lnk.rest))
```

5 min break

Practice

(reminder to self to use high contrast theme)

20 min

- Filter Link
- Insert

Filter Link

- Implement `filter_link`
- Download starter code `16.py` from the course website under Lecture 16
- Run doctests with `python3 -m doctest 16.py`
- 5 min: Try implementing it yourself
- 5 min: Discuss with someone next to you and compare your implementations
  - What worked?
  - What didn't?
  - Why?

Insert

- Implement `insert`
  - Hint: Remember, you do not need to handle the case where `index` is 0
  - Challenge: Implement it both iteratively and recursively
- Download starter code `16.py` from the course website under Lecture 16
- Run doctests with `python3 -m doctest 16.py`
- 5 min: Try implementing it yourself
- 5 min: Discuss with someone next to you and compare your implementations
  - What worked?
  - What didn't?
  - Why?