



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Data Structures: Linked Lists



Announcements

- November Holidays!
 - Please check the calendar.
 - Nov 11, Veteran's Day: No C88C / Other Classes
 - Thanksgiving Week, Nov 23-25:
 - » Weds / Thur / Fri: Berkeley is off
 - » Tues 11/22: No Lecture
 - » No C88C labs
 - » OH Monday / Tues: TBD.



Why “Data Structures”? (Next Few lectures)

- Data Structures
 - OOP helps us organize our *programs*
 - Data Structures help us organize our data!
 - You already know lists and dictionaries!
 - We’ll see a new one today
- Enjoy this stuff? Take 61B!
- Find it challenging? Don’t worry! It’s a different way of thinking.



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Linked Lists



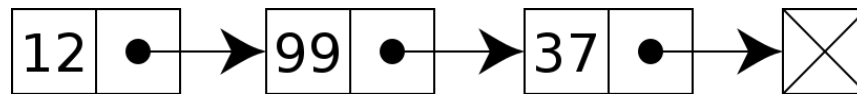
Data Structures

- A data structure is a way to organize or group a bunch of independent pieces of data.
 - Lists (arrays)
 - Dictionaries
 - Tuples
- A class, on its own, is **not** necessarily a data structure, it represents a new data type.
 - a "car" or a "person" is an instance of that data type.
 - Lists, Dicts, etc are also data types; their goal is to organize other data.
- These are common patterns that can be used to solve a wide variety of problems.
- Sometimes we're giving structure to make it easier as a programmer, sometimes we're trying to be fast or efficient.



Linked Lists

- A Recursive List, sometimes called a "rlist"
- Linked lists contain other linked lists
- A series of items with two pieces:
 - A value, usually called "first"
 - A "pointer" to the rest of the items in the list.



- We'll use a very small Python class "Link" to model this.
- `Link(12, Link(99, Link(37, Link.empty)))`



The Link Class

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        self.first = first
        self.rest = rest
```

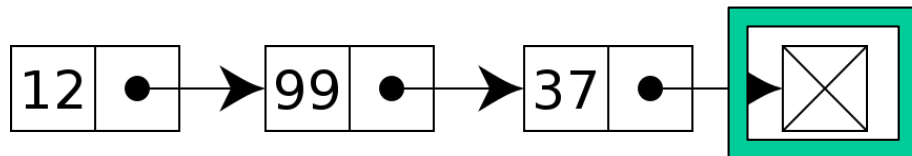
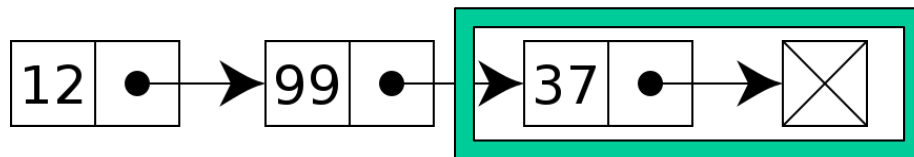
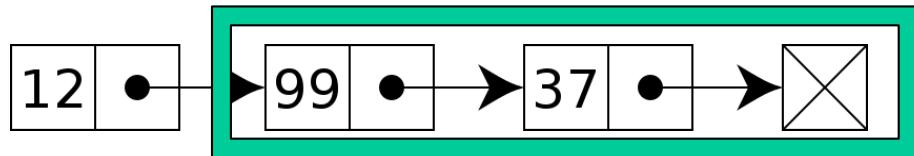
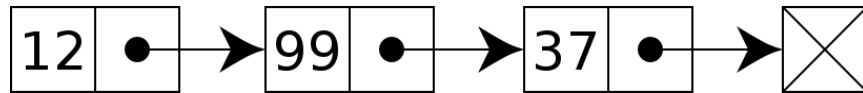
That's all we need!

- We can add a `__repr__` method, `length`, etc.
- Use an empty tuple for clarity / easier than `None`.
 - `()` has lots of useful methods defined, like `len()`



Recursion Is Implicit

`self.rest`





Iterating Over All Items in Linked List

```
def print_link(link):  
    if not link:  
        return  
    print(link.first)  
    print_link(link.rest)
```

- Base Case: No more items
- Do Action
- Recurse on the rest of the list

```
def print_link(link):  
    if not link:  
        return  
    item = link  
    while item:  
        print(item.first)  
        item = item.rest
```

- Handle the empty list
- Keep track of current item
- Update item to be the next in sequence.

Demo – See the Notebook





What's Needed For a Linked List?

- `self.first`
- `self.rest`
- Nothing else is *necessary*
- `__repr__`, `__len__` methods are all useful shortcuts and useful recursion practice.



Why are linked lists useful?

- Honestly, a list() is easier *most* of the time
 - Python handles all the hard details!
 - When data gets large, there are lots of edge cases.
- In terms of *efficiency*: Linked list make it fast to move items around, inserts and deletes.
 - But they are slower to finding any single item.
- In Ants Project: You'll see a list of `Place` objects which are linked together via an entrance and an exit.



Uses for a Linked List

- Modeling a Polynomial Equation
 - each item is (coefficient, exponent, next_term)
- Items in a music playlist
 - each item is a (song, next_song) pair
 - easy to add/remove items
 - » Specifically: often want to remove the first item
- Model real-world relationships
 - Anything that is a “chain” is a good option
 - Next week: We’ll extend this idea to “trees”



Efficiency of Linked Lists vs Lists

- Linked Lists generally use less memory.
- Linked Lists:
 - Once you've found an item, inserting / removing is easy, $O(1)$
 - Finding anything other than the first/last item is $O(n)$
- "Regular" Lists:
 - Inserting / Removing items, other than the last is $O(n)$ – due to internal copying
 - Finding any random item is $O(1)$.
- What if you need to iterate over all items in order?
 - $O(n)$ in both cases