Data Examples

Announcements

Midterm Scores out today/tomorrow
Regrade Requests Open Friday, Due 3/31

Context Setting: Where are we now?

Where are we now? Where are we going?

- For now – we’ve learned *most* of the basics of Python!
- There are plenty of Python we don’t see in CS88
- Apply OOP principles to explore new ideas
- We’re going to focus on storing / organizing data
- Lists, Tuples, and Dictionaries: Data Structures you already know!
- BUT: How do we build our own?
- We’ll build our own lists first, then talk about trees and other ways of organizing data
- Last few lectures: Switch to SQL

Why Learn “Data Structures”?

- OOP helps us organize our programs
- Data Structures help us organize our data!

  --> Can be implemented using OOP

- You already know lists and dictionaries!
- Enjoy this stuff? Take CS 61B!
- Find it challenging? Don’t worry! It’s a different way of thinking.

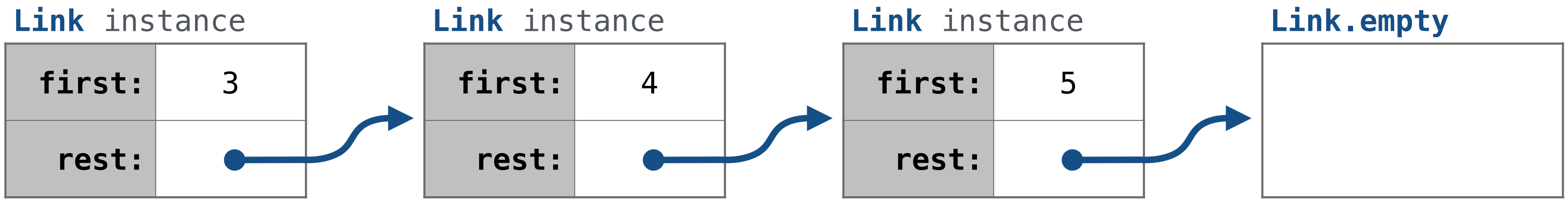
Aside: CGP Grey & Rock, Paper, Scissors [Watch Video #1]

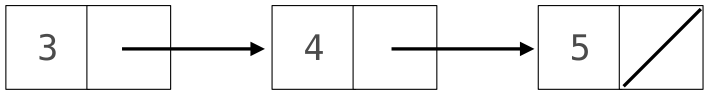
- How many rounds of Rock Paper Scissors is a 1 in 1,000,000,000 chance of winning?
- Each video leads to another set of videos.
- This is technically a tree, but we'll come back to that later.

Linked Lists Practice

Linked List Notation

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





Why use Linked Lists?

You might not have Python’s list() class

Linked lists are great for lots of cases:

- Modeling a Polynomial Equation
  each item is (coefficient, exponent, next_term)

- Items in a music Playlist
  each item is a (song, next_song) pair

Why?

easy to add/remove items: often want to remove the first item

- Model real-world relationships

- Anything that is a "chain" is a good option

Linked List Practice

Recursion and Iteration

Many linked list processing functions can be written both iteratively and recursively

Recursive approach:

- What recursive call do you make?
- What does this recursive call do/return?
- How is this result useful in solving the problem?

```
def length(s):
    """The number of elements in s.

    >>> length(Link(3, Link(4, Link(5))))
    3
    """

    if s is Link.empty:

        return 0

    else:

        return ____1 + length(s.rest)____
```

Iterative approach:

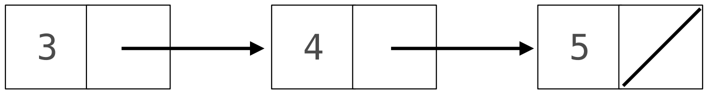
- Describe a process that solves the problem.
- Figure out what additional names you need to carry out this process.
- Implement the process using those names.

```
def length(s):
    """The number of elements in s.

    >>> length(Link(3, Link(4, Link(5))))
    3
    """

    k = __0__

    while __s is not Link.empty__:

        s, k = s.rest, __k + 1__

    return k
```

Constructing a Linked List

Build the rest of the linked list, then combine it with the first element.

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



```
def range_link(start, end):
    """Return a Link containing consecutive
    integers from start up to end.

    >>> range_link(3, 6)
    Link(3, Link(4, Link(5)))
    """

    if start >= end:

        return Link.empty

    else:

        return  Link(start, range_link(start + 1, end))
```

```
def range_link(start, end):
    """Return a Link containing consecutive
    integers from start to end.

    >>> range_link(3, 6)
    Link(3, Link(4, Link(5)))
    """

    s = Link.empty

    k =  end - 1

    while  k >= start :

        s = Link(k, s)
        k -= 1

    return s
```

Linked Lists Can Contain Anything:

What if we make an Album, which is a linked list of songs?

```
fearless = Song("Fearless", "Taylor Swift", 241)

fifteen = Song("Fifteen", "Taylor Swift", 294)

fearless_album = Link(fearless, Link(fifteen) … )
```

How can we…?

Calculate the total length of the album?

Find the longest song?

Finding the total Album Length

Given a linked list of Songs, which each has a length attribute, find the total length of the album

```
def total_album_length(album):
   if album == Link.empty:
        return 0
    else:
        return album.first.length + total_album_length(album.rest)
```

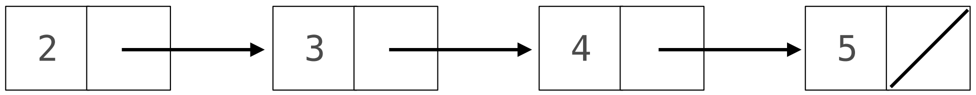
Finding the Longest Song

Given a linked list of Songs, which Song is the longest?

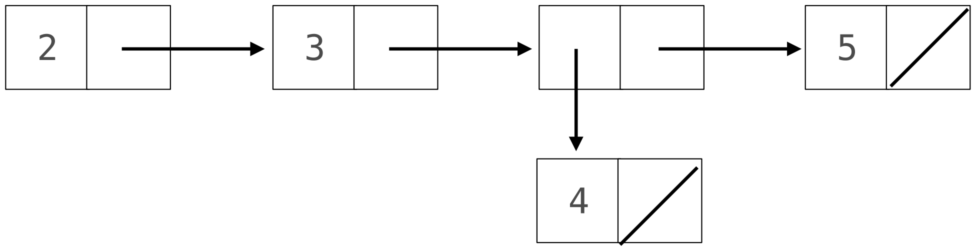
```
def total_album_length(album):
   if album == Link.empty:
        return None
   if album.rest == Link.empty:
        return album.first.length

   rest_longest = longest_song(album.rest)
   if album.first.length > rest_longest.length:
        return album.first
    else:
        return rest_longest
```

Nested Linked Lists

```
>>> s = Link(2, Link(3, Link(    4     , Link(5))))

>>> t = Link(2, Link(3, Link(  Link(4) , Link(5))))

>>> print(s)

<2 3 4 5>
```

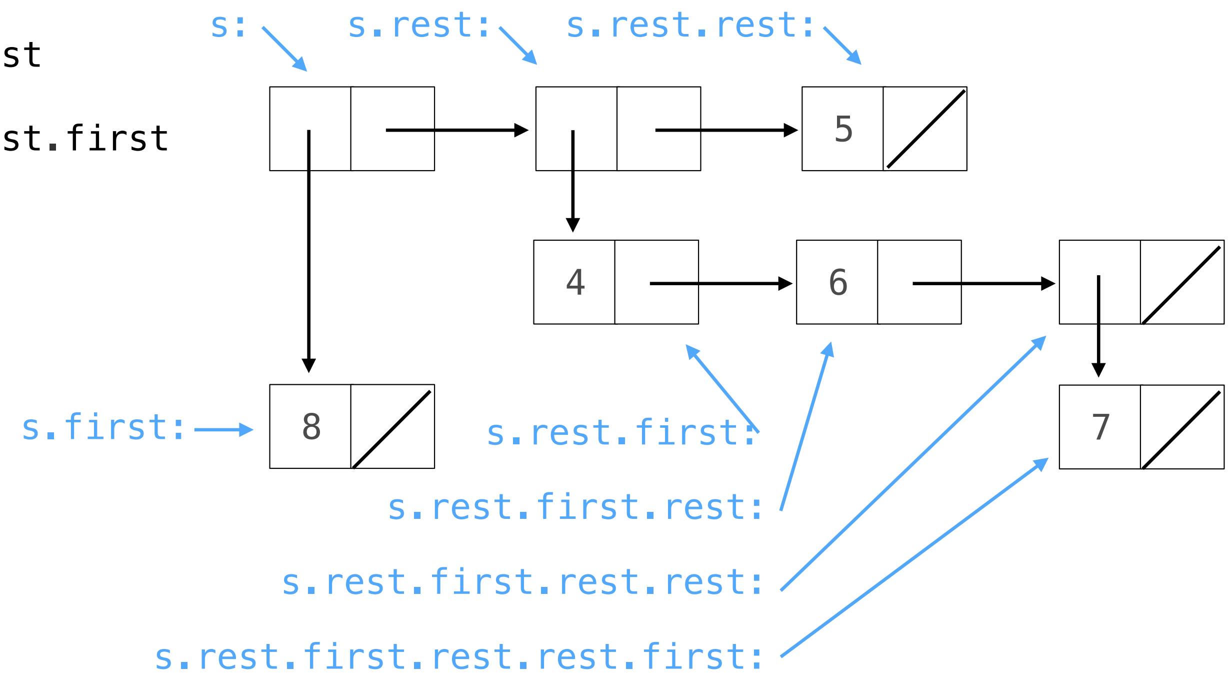


```
>>> print(t)

<2 3 <4> 5>
```



Nested Linked Lists

```
>>> s = Link(Link(8), Link(Link(4, Link(6, Link(Link(7)))), Link(5)))
>>> print(s)
<<8> <4 6 <7>> 5>
>>> s.first.first
8
>>> s.rest.first.rest.rest.first
Link(7)
>>> s.rest.first.rest.rest.first.first
7
```


s:
s.rest:
s.rest.rest:
5
4
6
s.first:
8
7
s.rest.first:
s.rest.first.rest:
s.rest.first.rest.rest:
s.rest.first.rest.rest.first: