61A Lecture 18

Announcements

Sequences

The Sequence Abstraction

red, orange, yellow, green, blue, indigo, violet.

0 , 1 , 2 , 3 , 4 , 5 , 6 .

There isn't just one sequence class or data abstraction (in Python or in general).

The sequence abstraction is a collection of behaviors:

Length. A sequence has a finite length.

Element selection. A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0.

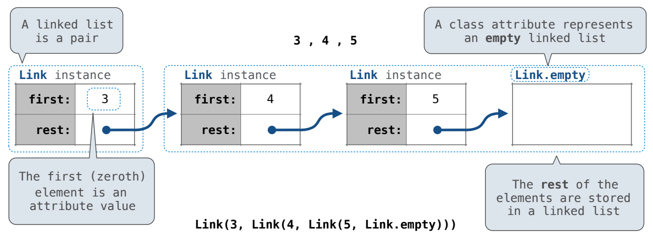
There is built-in syntax associated with this behavior, or we can use functions.

A list is a kind of built-in sequence

Linked Lists

Linked List Structure

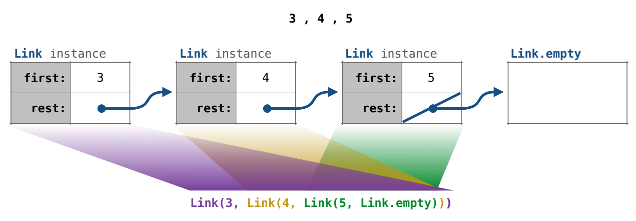
A linked list is either empty **or** a first value and the rest of the linked list


A linked list is a pair
3 , 4 , 5
A class attribute represents an **empty** linked list
Link instance
first: 3
rest:
Link instance
first: 4
rest:
Link instance
first: 5
rest:
Link.empty
The first (zeroth) element is an attribute value
The **rest** of the elements are stored in a linked list

Link(3, Link(4, Link(5, Link.empty)))

Linked List Structure

A linked list is either empty **or** a first value and the rest of the linked list



Link(3, Link(4, Link(5, Link.empty)))

Linked List Class

Linked list class: attributes are passed to __init__

```
class Link:
    empty = ()                                  Some zero-length sequence

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

help(isinstance): Return whether an object is an instance of a class or of a subclass thereof.

Link(3, Link(4, Link(5)))

(Demo)

Sequence Operations

Linked List Class

Linked lists are sequences

```
class Link:

    empty = ()

    def __init__(self, first, rest=empty):
        assert ...
        self.first = first
        self.rest = rest

    def __getitem__(self, i):
        if i == 0:
            return self.first
        else:
            return self.rest[i-1]

    def __len__(self):
        return 1 + len(self.rest)
```

Calls this method

This element selection syntax

Recursive call to __len__

More special method names:

`__getitem__`	Element selection []
`__len__`	Built-in len function

Methods can be recursive too!

(Demo)

Property Methods

Property Methods

Often, we want the value of instance attributes to stay in sync

For example, what if we wanted a Ratio to keep its proportion when its numerator changes

```
>>> s = Link(3, Link(4, Link(5)))
>>> s.second
4
>>> s.second = 6
>>> s.second
6
>>> s
Link(3, Link(6, Link(5)))
```

No method calls!

The @property decorator on a method designates that it will be called whenever it is looked up on an instance

A @<attribute>.setter decorator on a method designates that it will be called whenever that attribute is assigned. <attribute> must be an existing property method.

(Demo)

Linked List Processing

`[<map exp> for <name> in <iter exp> if <filter exp>]`

(Demo)