Linked Lists

Announcements

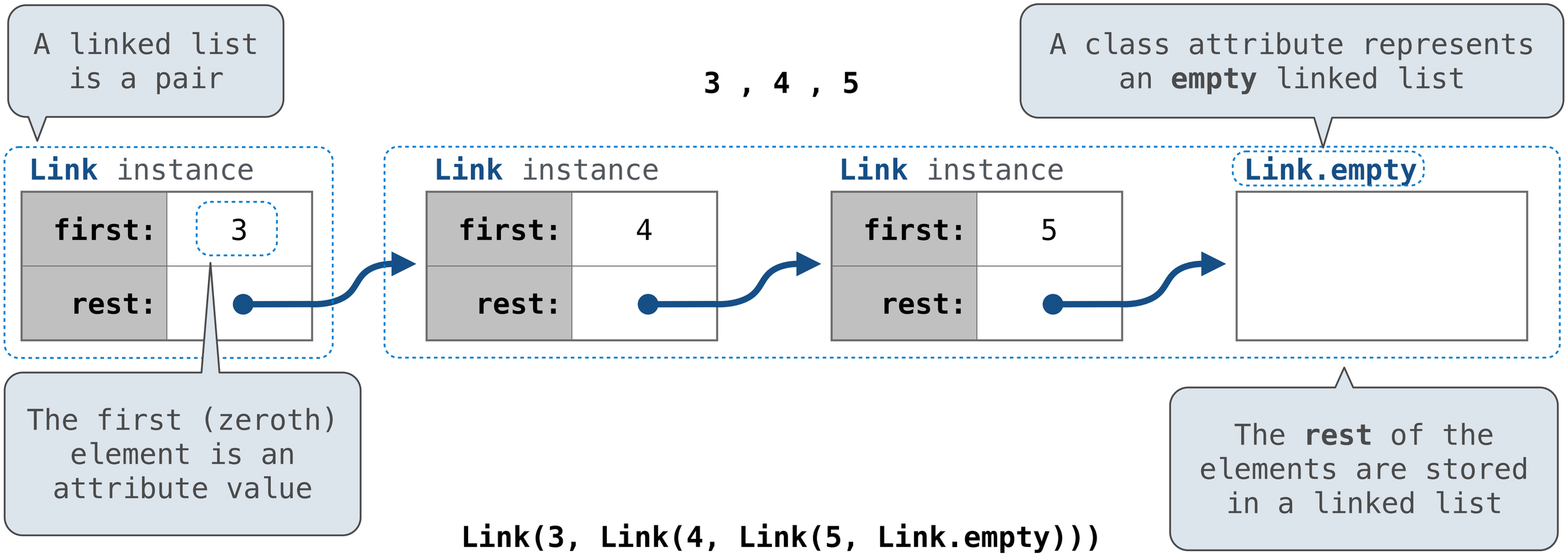
Inheritance Practice (Summer 2018 Final Q1)

https://pythontutor.com/cp/composingprograms.html#code=class%20Animal%3A%0A%20%20%20%20def%20__init__%28self,
%20parent%29%3A%0A%20%20%20%20%20%20%20%20self.parent%20%3D%20parent%0A%20%20%20%20%20%20%20%20is_alive%20%3D%20False%0A%0Aclass%20Cat%28Animal%29%3A%0A%20%20%20%20is_alive%20%3D%20True%0A%20%20%20%20def%20meow%28self%29%3A%0A%20%20%20%20%20%20%20%20def%20meower%28self%29%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20if%20self.is_alive%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20return%20'meower'%0A%20%20%20%20%20%20%20%20%20%20%20%20return%20'I%20am%20a%20ghost!'%0A%20%20%20%20%20%20%20%20%20%20%20%20print%28'meow'%29%0A%20%20%20%20%20%20%20%20%20%20%20%20return%20meower%0A%0Aclass%20CalicoCat%28Cat%29%3A%0A%0A%20%20%20%20def%20meow%28self%29%3A%0A%20%20%20%20%20%20%20%20print%28'purr'%29%0A%20%20%20%20%20%20%20%20return%20Cat.meow%28self%29%0A%0Arachel%20%3D%20Cat%28None%29%0Aaaron%20%3D%20CalicoCat%28rachel%29%0Aamy%20%3D%20Cat%28None%29%0A%0Aprint%28CalicoCat.meow%28rachel%29%29%0Aprint%28Animal.meow%28aaron%29%29%0ACat.meow%20%3D%20rachel.meow%28%29%0Aprint%28amy.meow%28%29%29%0A&cumulative=false&mode=edit&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Linked Lists

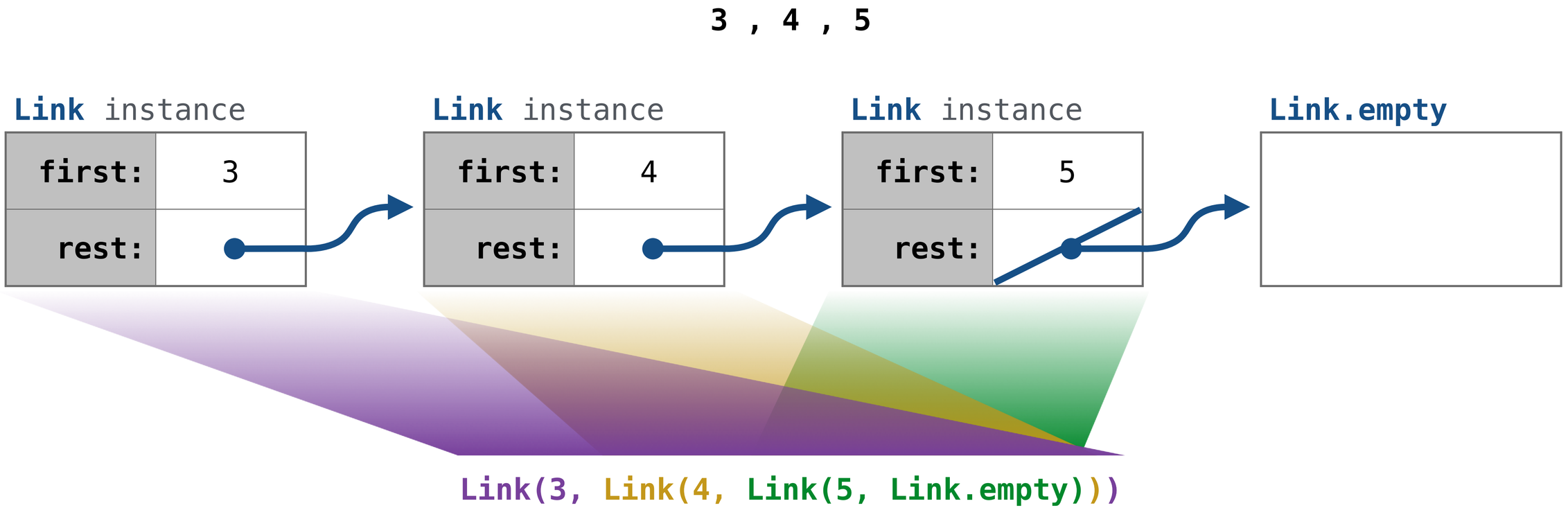
Linked List Structure

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



Linked List Structure

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



Linked List Class

Linked list class: attributes are passed to `__init__`

```
class Link:

    empty = ()

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

Some zero-length sequence

Returns whether 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              )))
```

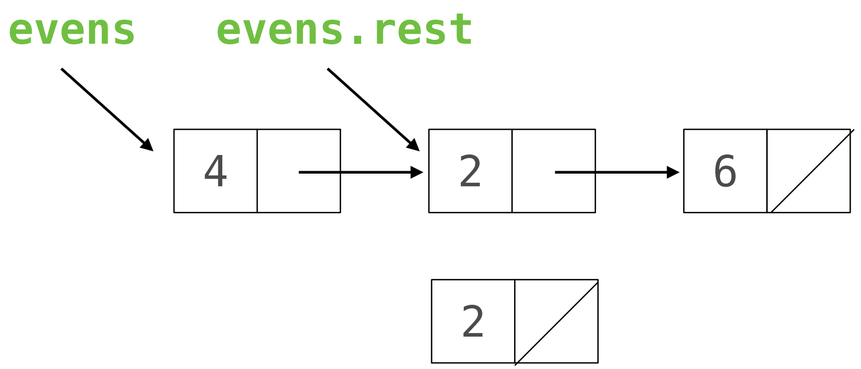
(Demo)

Linked List Practice

Slicing a Linked List

Normal slice notation (such as s[1:3]) doesn't work if s is a linked list.

```
def slice_link(s, i, j):
    """Return a linked list containing elements from i:j.

    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0
    if  j == 0  or s is Link.empty:
        return Link.empty
    elif  i == 0 :
        return Link(s.first,  slice_link(s.rest, i, j-1) )
    else:
        return slice_link(s.rest,  i-1  ,  j-1  )
```



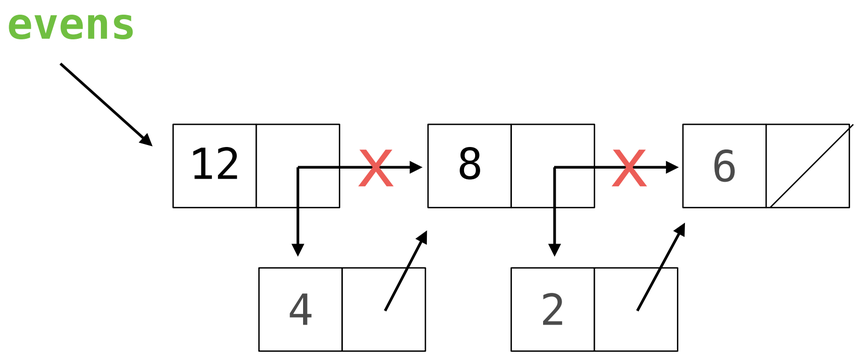
slice_link(evens, 1, 2) returns

slice_link(evens.rest, 0, 1) links 2 to

slice_link(evens.rest.rest, 0, 0) returns Link.empty

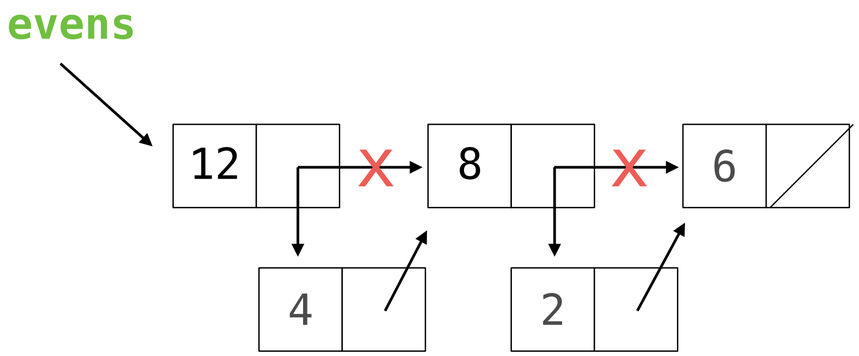
Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.

    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    Index out of range
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = ____________________
        s.first = _______________
        s.rest = second
    else:
        insert_link(s.rest, x, i-1)
```



Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.

    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    Index out of range
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    else:
        insert_link(s.rest, x, i-1)
```



Spring 2023 Midterm 2 Question 3(b)

Definition. A *prefix sum* of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6)))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if ____suffix is not Link.empty____:
            ____f(suffix.rest, total + suffix.first)____
    ____f(s.rest, s.first)____
```