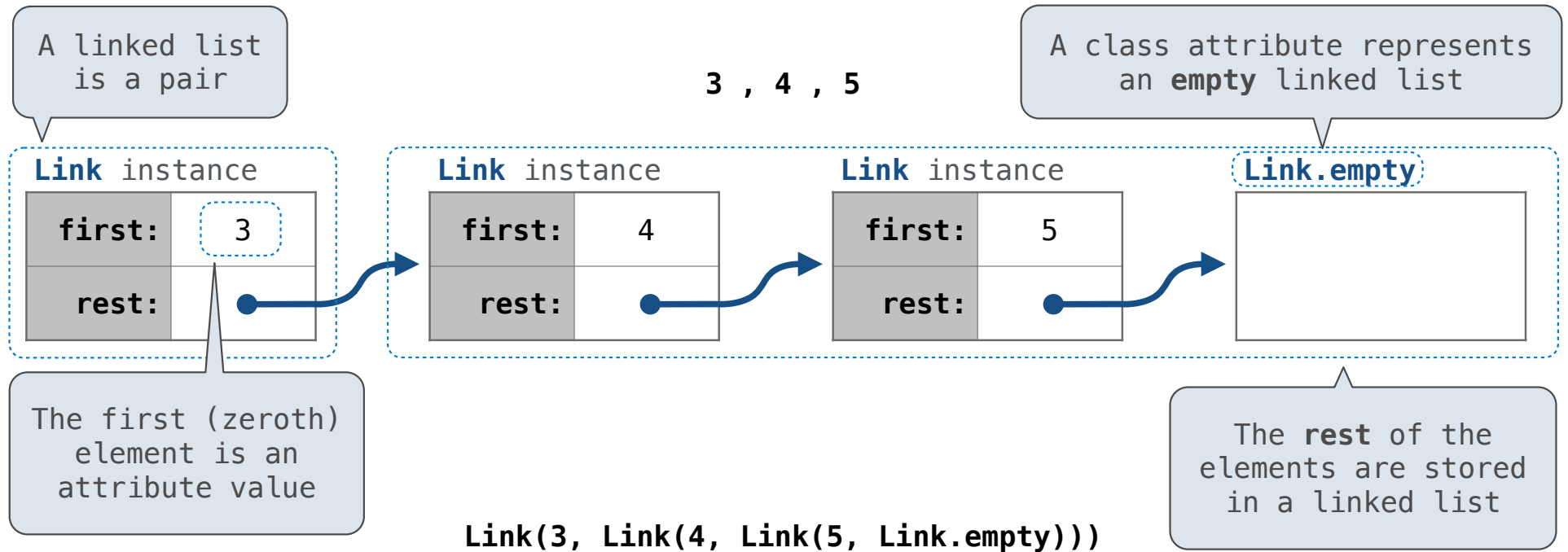# Composition

Announcements

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

A class attribute represents an **empty** linked list

3 , 4 , 5

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

3 , 4 , 5

Link(3, Link(4, Link(5, Link.empty)))
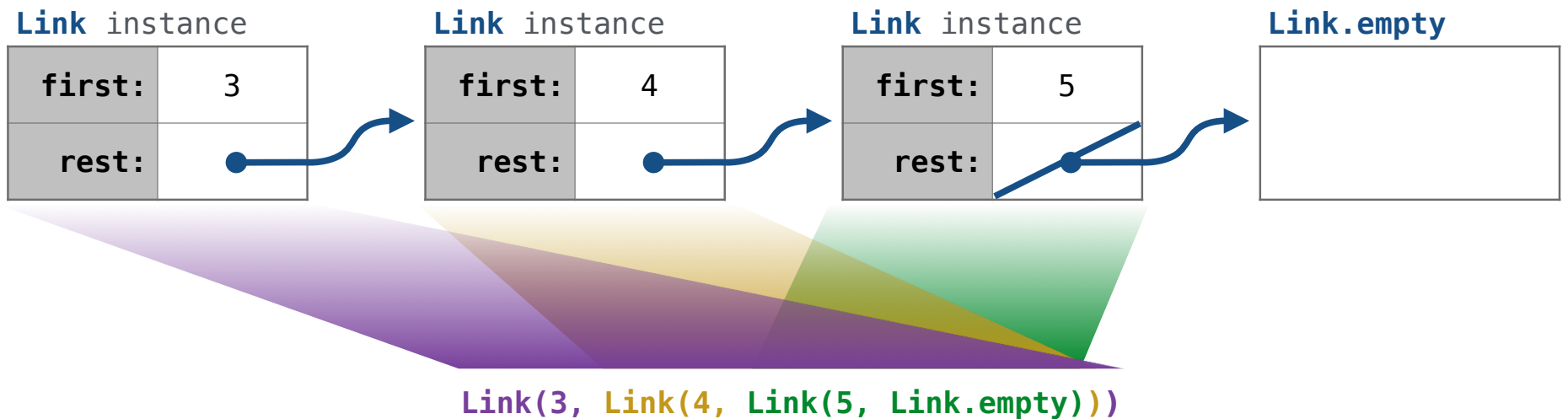
Linked List Class

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
        self.rest = rest
                              Returns whether
                              rest is a Link
```

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.

```python
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 )
```

evens   **evens.rest**

| 4 | → | 2 | → | 6 |/|

| 2 |/|
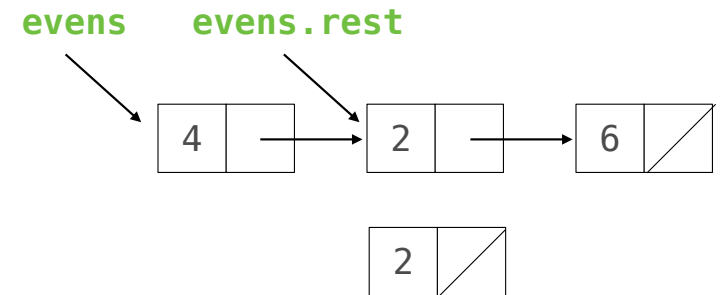
slice_link(evens, 1, 2) returns
slice_link(evens.rest, 0, 1) links 2 to
slice_link(evens.rest.rest, 0, 0) returns Link.empty

Inserting into a Linked List

```python
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)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    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
    elif  i == 1 and s.rest is Link.empty :
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```
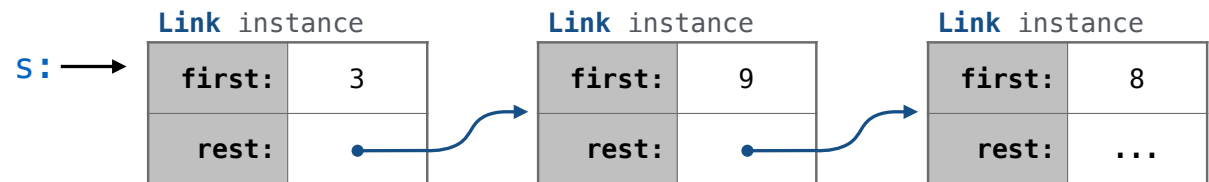
evens
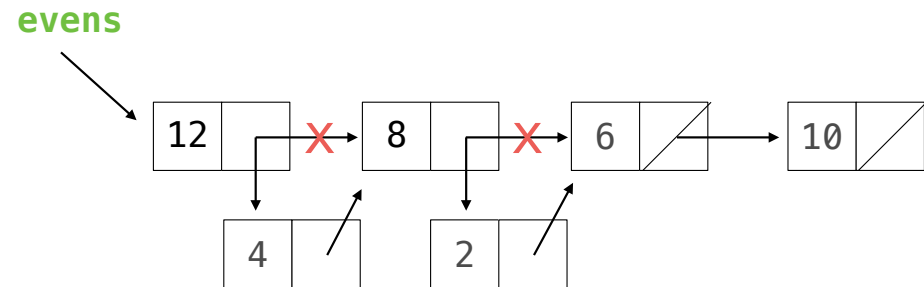
Definition. A *prefix sum* of a sequence of numbers is the sum of the first n elements for some positive length n.

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.

```python
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)
```

Link instance

s: →

first:	3
rest:	●

Link instance

first:	9
rest:	●

Link instance

first:	8
rest:	...

suffix: