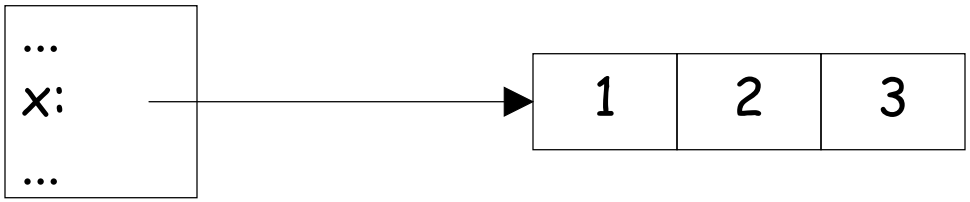
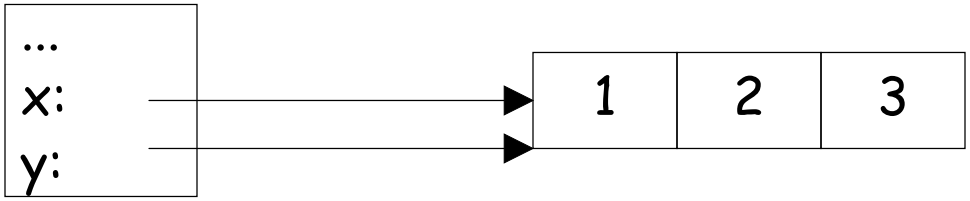
Lecture #13: Lists, objects, and Arrows

Diagrams of Sequence Objects

- We've often depicted values as arrows to something. To illustrate x = (1, 2, 3) you might see:



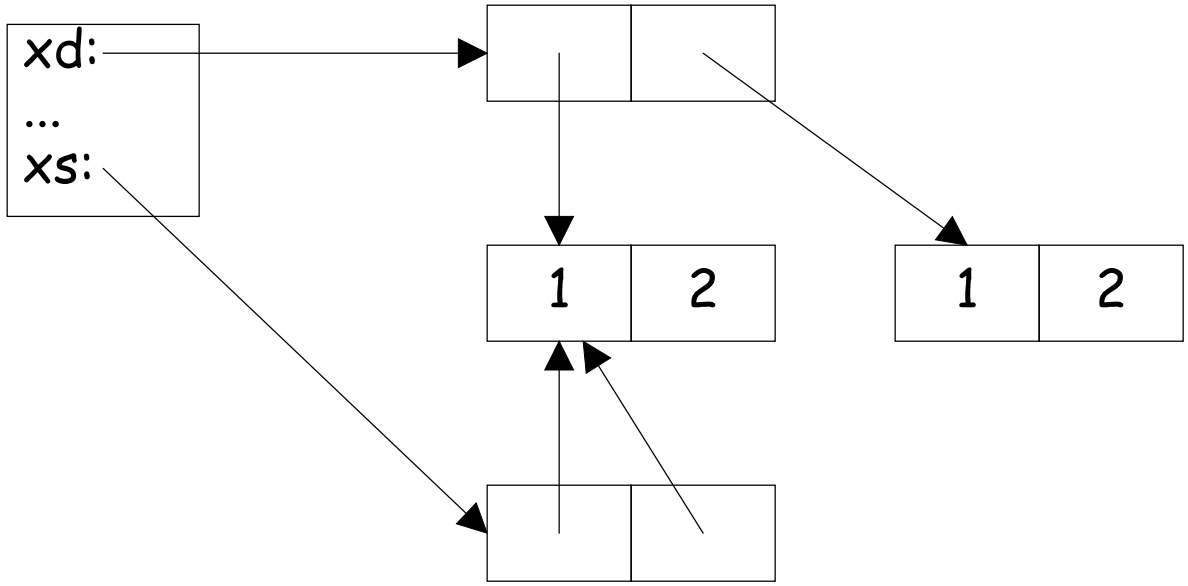
- The value of `x` here *is* the arrow, not the box (object) at the end.
- Copying `x` copies the arrow, not the box. After y = x:



- The is operator tests *equality of arrows* (or *object identity:* are they pointing at the same thing?), . . .
- While == is generally concerned with *equality of state* (are the arrows pointing at objects that contain equivalent things?)

Another Take on Tuples vs. Lists

- When dealing with tuples (or immutables in general), we can concern ourselves with equality alone.
- When dealing with lists (or mutable data in general), must consider object identity.
- For tuples, we can treat xd and xs as identical, and use either one:



- But if the boxes depicted (mutable) lists, we'd still have xs==xd (for now), but not necessarily in the future.

A List Problem

```
def partition(L, x):
    """Rearrange the elements of L so that all items < 'x' appear
    before all items >= 'x', and all are otherwise in their original
    order.  Modifies and returns L.
    >>> L = [0, 9, 6, 2, 5, 11, 1]
    >>> partition(L, 5)
    [0, 2, 1, 9, 6, 5, 11]
    >>> L
    [0, 2, 1, 9, 6, 5, 11]
    """
```

Another List Problem

```
def collapse_runs(L):
    """Remove the second and subsequent consecutive duplicates of
    values in L, modifying and returning L.
    >>> x = [1, 2, 1, 1, 1, 2, 0, 0]
    >>> collapse_runs(x)
    [1, 2, 1, 2, 0]
    >>> x
    [1, 2, 1, 2, 0]"""
```