Lecture #13: Lists, objects, and Arrows

Diagrams of Sequence Objects

- We’ve often depicted values as arrows to something. To illustrate
  \[ x = (1, 2, 3) \]

  \[ \begin{array}{c}
    \vdots \\
    \vdots \\
    1 \quad 2 \quad 3 \\
  \end{array} \]

- The value of \( x \) here is the arrow, not the box (object) at the end.

- Copying \( x \) copies the arrow, not the box. After \( y = x \):

  \[ \begin{array}{c}
    \vdots \\
    \vdots \\
    1 \quad 2 \quad 3 \\
  \end{array} \]

- 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 \( x_d \) and \( x_s \) as identical, and use either one:

  \[ \begin{array}{c}
    x_d \\
    \vdots \\
    1 \quad 2 \\
  \end{array} \quad \begin{array}{c}
    x_s \\
    \vdots \\
    1 \quad 2 \\
  \end{array} \]

- But if the boxes depicted (mutable) lists, we’d still have \( x_s == x_d \) (for now), but not necessarily in the future.

A List Problem

```python
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

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

---

Last modified: Fri Mar 2 00:39:17 2012

CS61A: Lecture #13

Last modified: Fri Mar 2 00:39:17 2012

CS61A: Lecture #13