Mutability

Announcements

Lab 5

Two Ways to Find Berries

```
def berry_finder(t):
    """Return whether tree t contains a node labeled 'berry'.

    >>> sproul = tree('roots',
    ...                [tree('branch1',
    ...                      [tree('leaf'),
    ...                       tree('berry')]),
    ...                 tree('branch2')])
    >>> berry_finder(sproul)
    True
    """
    if label(t) == 'berry':
        return True
    for b in branches(t):
        if berry_finder(b):
            return True
    return False
```

From Lecture 12:
sum, max, min, all, any
are built-in functions
that aggregate a sequence

```
def berry_finder(t):
    """Return whether tree t contains a node labeled 'berry'."""

    return label(t) == 'berry' or any([berry_finder(b) for b in branches(t)])
```

Using recursion to solve tree problems:

Write down a recursive call (usually on each branch), then ask...

1. What kind of value is returned from the call?
2. What does that value mean?
3. How is that value useful in implementing the function?

List Mutation

(Demo)

https://pythontutor.com/cp/composingprograms.html#code=s%20%3D%20%5B3,%205,%207,%209,%2011%5D%0Afor%20i%20in%20range%28len%28s%29%29%3A%0A%20%20%20%20s%5Bi%5D%20%3D%20s%5B-%28i%2B1%29%5D&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Building Lists Using Append

```
def sums(n, m):
    """Return lists that sum to n containing positive numbers up to m that
    have no adjacent repeats, for n > 0 and m > 0.

    >>> sums(5, 1)
    []
    >>> sums(5, 2)
    [[2, 1, 2]]
    >>> sums(5, 3)
    [[1, 3, 1], [2, 1, 2], [2, 3], [3, 2]]
    >>> sums(5, 5)
    [[1, 3, 1], [1, 4], [2, 1, 2], [2, 3], [3, 2], [4, 1], [5]]
    >>> sums(6, 3)
    [[1, 2, 1, 2], [1, 2, 3], [1, 3, 2], [2, 1, 2, 1], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
    """
    result = []
    for k in range(1,    min(m + 1, n)    ): # k is the first number of a list
        for rest in     sums(n-k, m)     :
            if rest[0] != k:
                result.append( [k] + rest )  # build a list out of k and rest
    if n <= m:
        result.append([n])
    return result
```

(Demo)

https://pythontutor.com/cp/composingprograms.html#code=result%20%3D%20%5B%5B1%5D,%20%5B2%5D,%20%5B3%5D%5D%0Afor%20k%20in%20range%281,%204%29%3A%0A%20%20%20%20for%20s%20in%20result%3A%0A%20%20%20%20%20%20%20%20if%20s%5B-1%5D%20!%3D%20k%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20s.append%28k%29%0A%20%20%20%20%20%20%20%20%20%20%20%20result.append%28s%29%0Aprint%28result%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Sameness and Change

- As long as we never modify objects, a compound object is just the totality of its pieces
- This view is no longer valid in the presence of change
- A compound data object has an "identity" in addition to the pieces of which it is composed
- A list is still "the same" list even if we change its contents
- Conversely, we could have two lists that happen to have the same contents, but are different

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a
[10, 20]
>>> b
[10, 20]
>>> a == b
True
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
>>> b
[10, 20]
>>> a == b
False
```

Identity Operators

Identity

`<exp0> is <exp1>`

evaluates to True if both <exp0> and <exp1> evaluate to the same object

Equality

`<exp0> == <exp1>`

evaluates to True if both <exp0> and <exp1> evaluate to equal values

Identical objects are always equal values

(Demo)

Mutation and Names

If multiple names refer to the same mutable object (directly or indirectly), then a change to that object is reflected in the value of all of these names.

What numbers are printed (and how many of them)?

```
s = [2, 7, [1, 8]]
t = s[2]
t.append([2])
e = s + t
t[2].append(8)
print(e)
```

https://pythontutor.com/cp/composingprograms.html#code=s%20%3D%20%5B2,%207,%20%5B1,%208%5D%5D%0At%20%3D%20s%5B2%5D%0At.append%28%5B2%5D%29%0Ae%20%3D%20s%20%2B%20t%0At%5B2%5D.append%288%29%0Aprint%28e%29%0A&cumulative=true&curInstr=6&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Tuples

(Demo)