Exam Prep Section 4 Sols. - CS61A Spring 2018

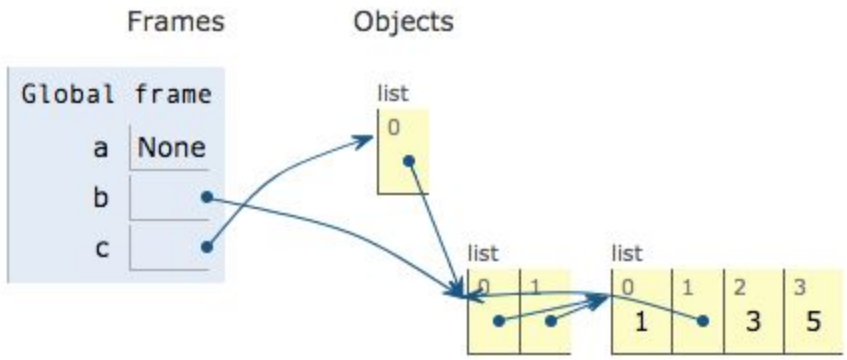
Worksheet 4: List Mutation, Dictionaries, and More Trees

Lots of Lists

Draw the environment diagram that results from executing the following code.
PythonTutor

```
a = [1, 2, 3, 4, 5]
a.pop(3)
b = a[:]
a[1] = b
b[0] = a[:]
b.pop()
b.remove(2)
c = [].append(b[1])
a.insert(b.pop(1), a[-3:4:3])
b.extend(b)
if b == b[:] and b[1][1][0] is b[0][1][1]:
    a, b, c = [c] + a[-4:4:2]
```

Solution



Trees About Equal

Fill in the function below so that it conforms to its docstring.

PythonTutor with solution and checks of the doctests

Solution

```
def about_equal(t1, t2):
    """Returns whether two trees are about equal.

    Two trees are about equal if and only if they contain the same
     labels the same number of times.

    >> x = tree(1, [tree(2), tree(2), tree(3)])
    >> y = tree(3, [tree(2), tree(1), tree(2)])
    >> about_equal(x, y)
    True
    >> z = tree(3, [tree(2), tree(1), tree(2), tree(3)])
    >> about_equal(x, z)
    False
    """
    def label_counts(t):

        if is_leaf(t):

            return {label(t): 1}

        Else:

            counts = dict()

            for b in branches(t) + [tree(label(t))]:

                for lab, count in label_counts(b).items():

                    if lab not in counts:

                        counts[lab] = 0

                    counts[lab] += count

            return counts

    return label_counts(t1) == label_counts(t2)
```

What color is it? (Sp15 Midterm 2 Q4b)

Implement decrypt which takes in a string `s` and a dictionary `d` that contains words as values and their secret codes as keys. It returns a list of all possible ways in which `s` can be decoded by splitting it into secret codes and separating the corresponding words by spaces.

Solution

```
def decrypt(s, d):
    """List all possible decoded strings of s.
    >>> codes = {
    ...     'alan': 'spooky',
    ...     'al': 'drink',
    ...     'antu': 'your',
    ...     'turing': 'ghosts',
    ...     'tur': 'scary',
    ...     'ing': 'skeletons',
    ...     'ring': 'ovaltine'
    ... }
    >>> decrypt('alanturing', codes)
    ['drink your ovaltine', 'spooky ghosts', 'spooky scary skeletons']
    """
    if s == '':

        return []

    messages = []

    if s in d:

        messages.append(d[s])

    for k in range(1, len(s)+1):

        first, suffix = s[:k], s[k:]

        if first in d:

            for rest in decrypt(suffix, d):

                messages.append(d[first] + ' ' + rest)

    return messages
```

Consistency is Key

Fill in the function below so that it conforms to its docstring.
[PythonTutor with solution and checks]

Solution

```
def ensure_consistency(fn):
      """Returns a function that calls fn on its argument, returns fn's
      return value, and returns None if fn's return value is different
      from any of its previous return values for those same argument.
      Also returns None if more than 20 calls are made.
      """
      n = 0
      z = {}
      def helper(x):
         nonlocal n
         n += 1
         if n > 20:
             return None
         val = fn(x)
         if x not in z:
             z[x] = [val]
         if z[x] == [val]:
             return val
         else:
             z[x] = None
             return None
    return helper
```