CS61A Discussion #03

TA: Hyun Jae Moon (hyunjaemoon@berkeley.edu)

Trees & Sequences

Announcements

- LAB 04 Due Tonight
- Maps Due 2/22 Thursday

Post-Midterm Thoughts / Attendance

- Attendance: tinyurl.com/moondisc03 (Secret Word: Valentine)
- Please fill out the survey!

Agenda

We will go over:

- Lists, Slicing, List Comprehension, Trees
- 1.1, 1.2, 2.1, 3.1, 3.2, 3.3

Lists

```
>>> a = [1, 2, 3]          #Lists are ordered collection of values

>>> b = [abs, “1”, [2]]    #Lists can contain various types

>>> a += b                 #’+’ will append a list at the end.

>>> a

[1, 2, 3, abs, “1”, [2]]
```

List Indexing

```
>>> a = [1, 2, 3]
>>> len(a)              #Gives you the length of a list
3
>>> a[0] == 1           #List index starts from 0
True
>>> a[-2]               #equivalent of saying “a[len(a)-2]”
2
>>> a[3]
IndexError: list index out of range
```

‘in’ statement

Solve 1.1

```
>>> a = [1, 2, [3]]
>>> 2 in a              #Checks if 2 is included in the list.
True
>>> 3 in a              #’in’ does not check nested values.
False
>>> [3] in a            # a[2] == [3]
True
```

List Slicing

- We can use *slice* to access more than one element of a list at a time.
- `lst[<start index> : <end index> : <step size>]`

	Start Index	End Index	Step Size
Default (left empty)	0	len(lst)	1
-1	-1 OR len(lst) - 1	-1 OR len(lst) - 1	Flips it backwards

Confusing Parts on List Slicing

```
>>> a = [1, 2, 3, 4]
>>> a[:]              #equivalent of "a[0:-1]", which is copying the whole list.
[1, 2, 3, 4]
>>> a[:2]             #copy up to (but not including) the end index
[1, 2]
>>> a[::-2]           #flip the list first, then step size by 2
[4, 2]
>>> a[1:-2]           #equivalent of "a[1:2]"
[2]
>>> a[3:-2]           #equivalent of "a[3:2]", but start > end.
[]
```

Solve 1.2

List Comprehension

- Return a new list of elements, using some rule.
- `[<expr> for <var> in <sequence> if <filter_expr>]`

List Comprehension

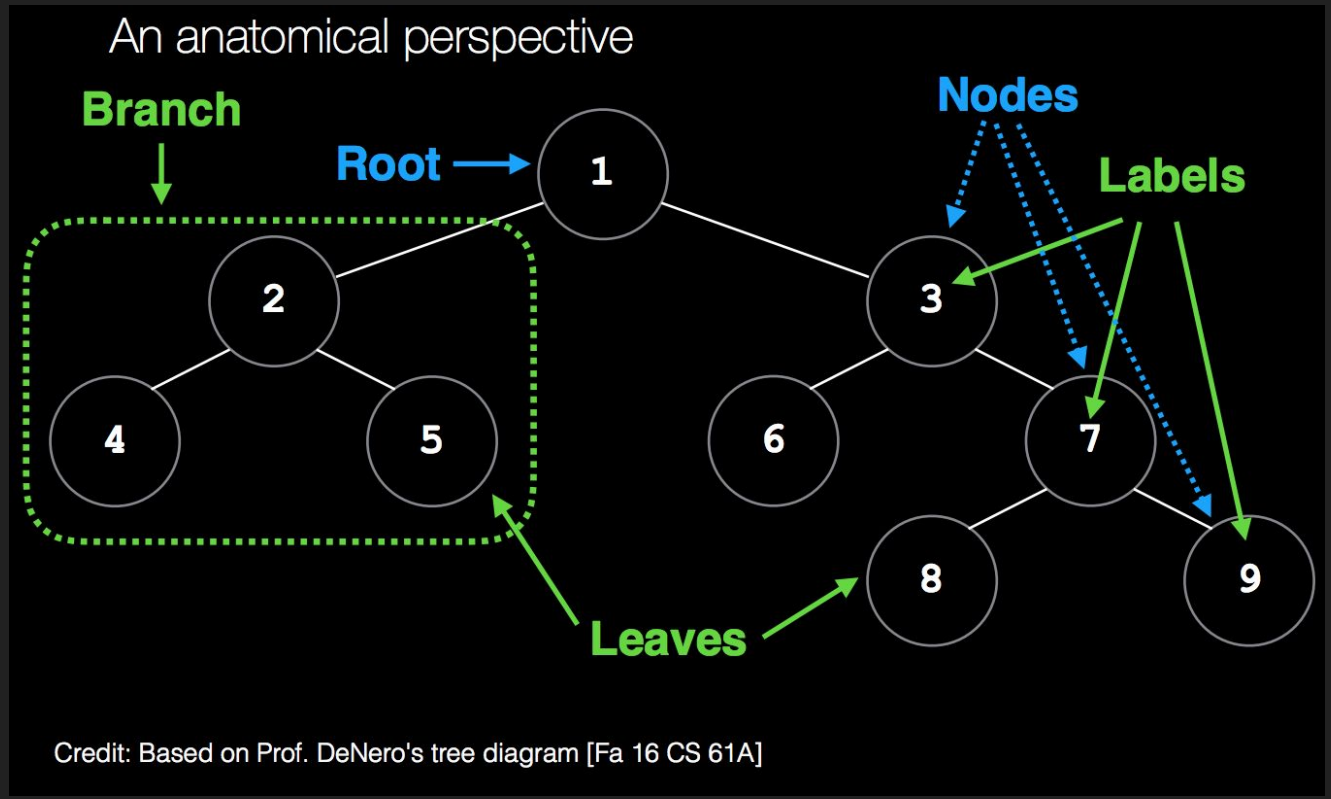
```
>>> [i for i in [1, 2, 3, 4]]                    #getting each element in sequence
[1, 2, 3, 4]
>>> [i for i in [1, 2, 3, 4] if i % 2 == 0]     #extracting even numbers
[2, 4]
>>> [i+1 for i in [1, 2, 3, 4] if i % 2 == 0] #adding one to each extracted numbers
[3, 5]
```

List Comprehension - Harder

```
>>> [[i for i in [1]]]                          #nested the whole list comprehension
[[1]]
>>> [[i for i in [x]] for x in [1]]             #same as above, but with more complexity
[[1]]
>>> [[i for i in [x]] for x in [1, 2, 3]]       #added more elements in the sequence
[[1], [2], [3]]
```

Solve 2.1

Trees


An anatomical perspective
Branch
Root
Nodes
Labels
1
2
3
4
5
6
7
8
9
Leaves
Credit: Based on Prof. DeNero's tree diagram [Fa 16 CS 61A]

Trees Implementation

```
#Constructor
def tree(label, branches=[]):
    for branch in branches:
        assert is_tree(branch)
    return [label] + list(branches)

#Selectors
def label(tree):
    return tree[0]

def branches(tree):
    return tree[1:]
```

```
#For convenience
def is_leaf(tree):
    return not branches(tree)
```

LIVE CODING!!!

Solve 3.1, 3.2, 3.3
We'll go over one by one!