CS61A: Discussion 3

Lists

What are Lists

- Sequential containers that can hold just about anything
  - Numbers
  - Strings
  - Lists
  - Functions
  - Etc……

How to navigate lists

- Lists are zero-indexed!
  - lst[0]
  - lst[-1]
- len(lst)
- Newlst = LstA + LstB

List Splicing

- Lst[start:stop:step]
  - Start - first index to look at (default: 0)
  - Stop - index where you should stop looking (exclusive!) - (default len(lst))
  - Step - everytime you look at something what index you should look at next (default: 1)

List Comprehensions

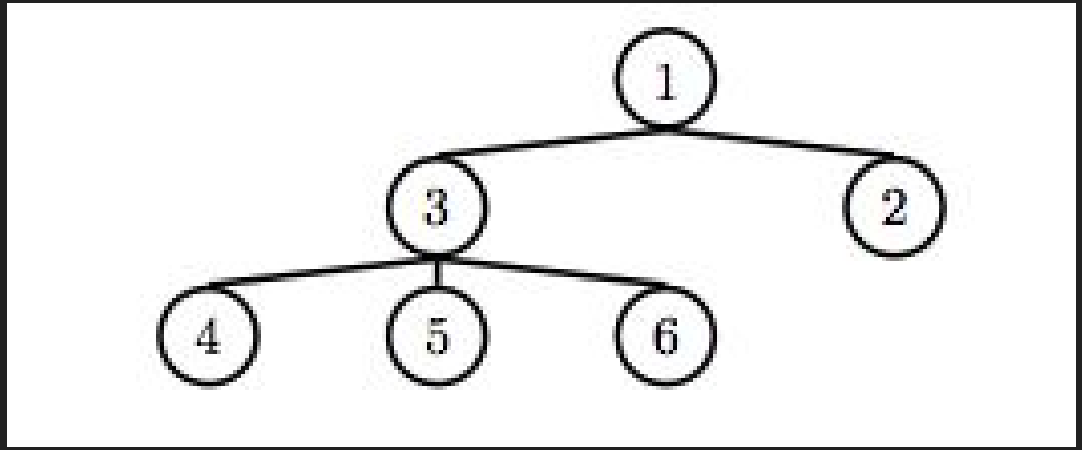
- [x for x in lst]
  - Copies the list in this case
- Note: x is just a temporary variable name. It will not exist/overwrite outside of the list comprehension. (no need to put in environment diagrams)
- Can be more complex with if, else, and body expressions
- Can have list comprehensions in list comprehensions!

Worksheet Time (pg 1-5)

Trees

- Abstract Data Structure
  - Not built-in
  - Created to help solve particular problems that fit the structure
  - Insides accessed ONLY through getters and setters

More Trees




```
# 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)
```

Important Things

- Do NOT break the abstraction barrier.
  - This mean, if we knew the tree was implemented with a list, we should not be doing any t[0][1], etc.
  - The data structure is abstract, meaning we should treat it in the way it was meant to be treated, not looking inside and messing around.
- Trees are naturally recursive structures (each child is another tree). Great for recursive problems!
  - On that note, knowing what a function should take in and what it should return are crucial to creating a working function (no TypeErrors please!)

Worksheet Time
(do as much as you can)