Conclusion

Announcements

CONGRATULATIONS!!

Is Your Brain Full Yet?

- Data: values, literals, operations,
- Functions
- Variables
- List, Tuples, Dictionaries
- Function Definition Statement
- Conditional Statement
- Iteration: list comp, for, while
- Lambda function expr.
- Higher Order Functions

- Higher order function patterns
  - Map, Filter, Reduce
- Recursion
- Abstract Data Types
- Mutation
- Class & Inheritance
- Exceptions
- Iterators & Generators
- SQL / Declarative Programming

Course Staff

Edwin Vargas Navarro he/him/his

@ DSP jedwin321@berkeley.edu

So what can I say except I like Data Science ¯\(ツ)/¯

Ethan Yoo he/him/his

@ DSP ethanyoo7912@berkeley.edu

Fourth-year CS major. The only reason why I'm a math double is so I can avoid taking CS70, but then I dropped math ;-;.

Isabelle Ng she/her/hers

@ DSP isabelle.ng@berkeley.edu

Hi there! My name is Isabelle and this is my 3rd semester being a TA for 61A/C88C. In my free time I like making music and hanging out with friends :) Excited to meet you all

Rebecca Dang she/her/hers

@ DSP rdang@berkeley.edu

Hey there, I'm a 4th year EECS major and I'm super excited to teach C88C! Happy to chat about this course, classes and clubs at Berkeley, professional development, guitar, books, movies, TV, music, and more :D

ඞ John Teng ඞ he/him/his

@ DSP johnteng9@berkeley.edu

Hi, I'm John and I'm a fourth year CS major from Pennsylvania.

Mira Wagner she/her/hers

mirawagner@berkeley.edu

Hi! I am a sophomore planning to major in data science/statistics and linguistics. I love reading, especially mysteries, swimming and baking! Excited for this semester :)

Ramya Chitturi she/her/hers

ramya.chitturi@berkeley.edu

Hi! I'm Ramya, a senior majoring in CS and linguistics. I enjoy sci-fi/fantasy books, crosswords, rock music, museums, and more! Excited to get to know you this semester :)

Alicia Wang she/her/hers

awwang629@berkeley.edu

Hi! I am a sophomore studying Data Science and Cognitive Science. I love playing badminton and traveling! Excited to meet everyone!

Dhruv Syngol he/him/his

dhruvsyngol@berkeley.edu

Hey everyone, I'm a sophomore studying Data Science and Economics, originally from the Chicago Suburbs! I love to watch sports (Go Bears!), play pickleball, explore cafes and restaurants, and go on hikes! Super excited for this semester!

Grace Baek she/her/hers

gracebaek@berkeley.edu

Hi! I'm Grace, a junior majoring in Computer Science and Economics. In my free time, I like to bake, try going to new cafes, and watch kdramas :) Super excited to meet everyone!

Grace Xie she/her/hers

gracexie@berkeley.edu

Hello! My name is Grace. I'm a third-year majoring in MCB and Data Science :0 I love reading sci-fi and baking in my free time.

Lia Fernandez-Grinshpun she/her/hers

liafg@berkeley.edu

Hi! I'm Lia, a fourth year Business & Data Science major. In my free time, I love to listen to podcasts, run/weightlift, hike, nerd out over credit cards, and cafe hop. So excited to meet all of you :)

Reema Rafifar she/her/hers

reemarafifar@berkeley.edu

Hi everyone! I'm Reema, a second-year majoring in Neuroscience. I absolutely love movies so come talk to me about your favorite films! I can't wait to get through C88C with you!

Designing Functions

Approaching The Exam

- Skim the topics (~1 min)
- Handle the "easy"(est) questions first
- Read the whole question first!
- Read the text
- Read the doctests!
- What techniques might be applicable?
  - Pattern matching is OK
- Draft a solution on scratch paper!
- Write yourself notes

How to Design Programs

From Problem Analysis to Data Definitions
Identify the information that must be represented and how it is represented in the chosen programming language. Formulate data definitions and illustrate them with examples.

Signature, Purpose Statement, Header
State what kind of data the desired function consumes and produces. Formulate a concise answer to the question *what* the function computes. Define a stub that lives up to the signature.

Functional Examples
Work through examples that illustrate the function's purpose.

Function Template
Translate the data definitions into an outline of the function.

Function Definition
Fill in the gaps in the function template. Exploit the purpose statement and the examples.

Testing
Articulate the examples as tests and ensure that the function passes all. Doing so discovers mistakes. Tests also supplement examples in that they help others read and understand the definition when the need arises—and it will arise for any serious program.

https://htdp.org/2018-01-06/Book/

Tree Processing

Tree-Structured Data

```
class Tree:
    def __init__(self, label, branches=[]):
        self.label = label
        self.branches = list(branches)

    def is_leaf(self):
        return not self.branches
```

A tree can contains other trees:

```
[5, [6, 7], 8, [[9], 10]]

(+ 5 (- 6 7) 8 (* (- 9) 10))

(S
  (NP (JJ Short) (NNS cuts))
  (VP (VBP make)
      (NP (JJ long) (NNS delays)))
  (. .))

<ul>
  <li>Midterm <b>1</b></li>
  <li>Midterm <b>2</b></li>
</ul>
```

Tree processing often involves recursive calls on subtrees

Designing a Function

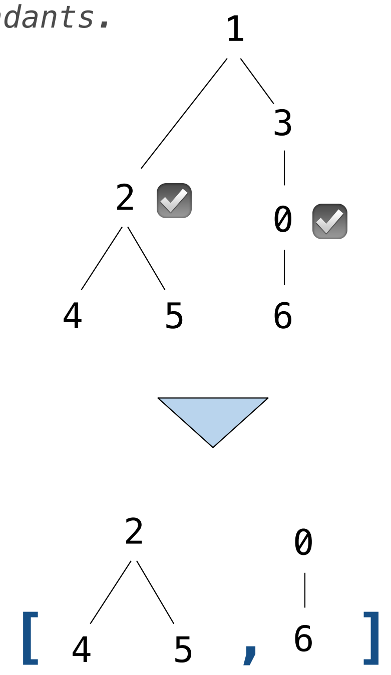
Implement **smalls**, which takes a Tree instance t containing integer labels. It returns the non-leaf nodes in t whose labels are smaller than any labels of their descendant nodes.

Signature: Tree -> List of Trees

```
def smalls(t):
    """Return a list of the non-leaf nodes in t that are smaller than all their descendants.

    >>> a = Tree(1, [Tree(2, [Tree(4), Tree(5)]), Tree(3, [Tree(0, [Tree(6)])])])
    >>> sorted([t.label for t in smalls(a)])
    [0, 2]

    """
    result = []
    def process(t):
        if t.is_leaf():
            return t.label
        else:



            return min(...)
    process(t)
    return result
```

Signature: Tree -> number

"Find smallest label in t & maybe add t to result"



Designing a Function

Implement **smalls**, which takes a Tree instance t containing integer labels. It returns the non-leaf nodes in t whose labels are smaller than any labels of their descendant nodes.

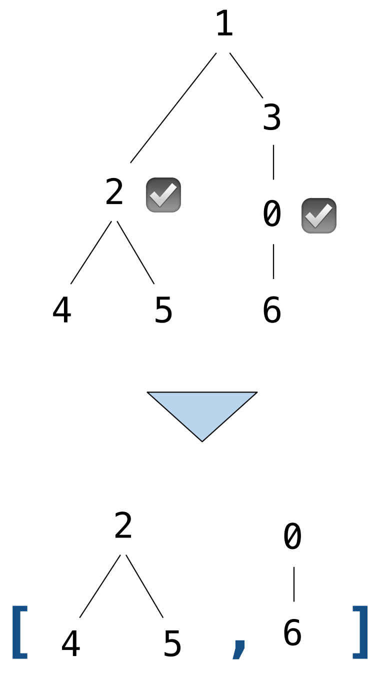
Signature: Tree -> List of Trees

```
def smalls(t):
    """Return a list of the non-leaf nodes in t that are smaller than all descendants.

    >>> a = Tree(1, [Tree(2, [Tree(4), Tree(5)]), Tree(3, [Tree(0, [Tree(6)])])])
    >>> sorted([t.label for t in smalls(a)])
    [0, 2]

    """
    result = []
    def process(t):
        if t.is_leaf():
            return ___________t.label___________
        else:
            smallest = _min([process(b) for b in t.branches])_
            if ________t.label < smallest________:
               _________result.append( t )________
            return min(smallest, t.label)
    process(t)
    return result
```

Signature: Tree -> number

"Find smallest label in t & maybe add t to result"

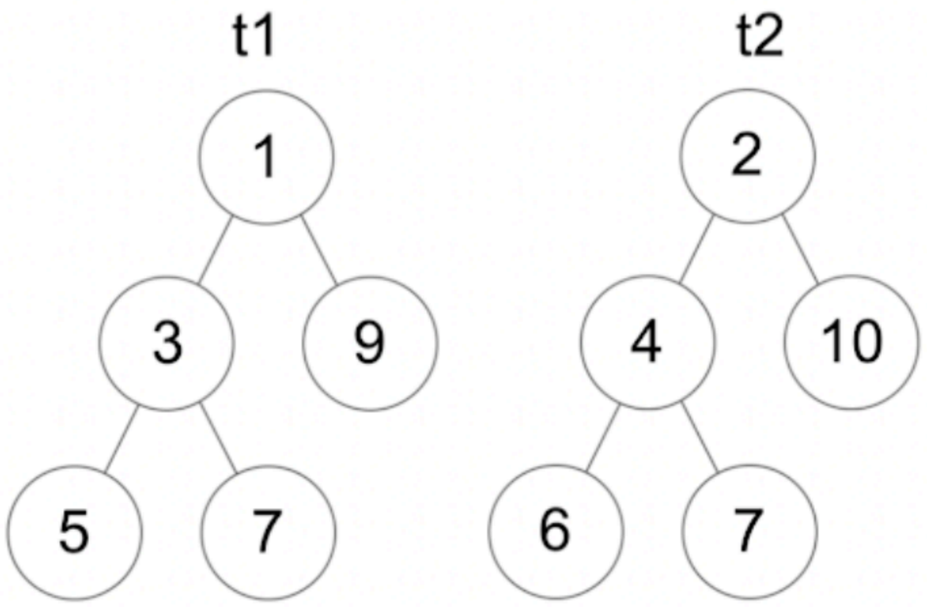
smallest label in a branch of t



Fall 2021 Q9 – Tree Farm

9. (12.0 points) Tree Farm

You've decided to get into the tree growing business! All the trees you grow have the same structure as each other but may have different values. You want to detect the nodes that are in the same position in two given trees but have different values. Write a function that takes in two trees, `t1` and `t2`, with the same structure and yields the mismatching node values as a tuple.



Fall 2021 Q9 – Tree Farm

```
def tree_mismatches(t1, t2):
    """
    >>> t1 = Tree(1, [Tree(3, [Tree(5), Tree(7)]), Tree(9)])
    >>> t2 = Tree(2, [Tree(4, [Tree(6), Tree(7)]), Tree(10)])
    >>> a = tree_mismatches(t1, t2)
    >>> next(a)
    (1, 2)
    >>> next(a)
    (3, 4)
    >>> next(a)
    (5, 6)
    >>> next(a)
    (9, 10)
    >>> next(a)
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    StopIteration
    """
    if __________________:
        ________________________
    n = __________________________
    for i in range(n):
        branch_mismatches = _______________________
        for _________________________:
            _________________________
```

(a) (6.0 pt) Complete the skeleton code. You may not add, change, or delete lines from the skeleton code.

```
def tree_mismatches(t1, t2):
    if t1.value != t2.value:
        yield t1.value, t2.value
    n = len(t1.branches)
    for i in range(n):
        branch_mismatches = tree_mismatches(t1.branches[i], t2.branches[i])
        for val in branch_mismatches:
            yield val
```

Keep on Programming

That's all. Thanks!