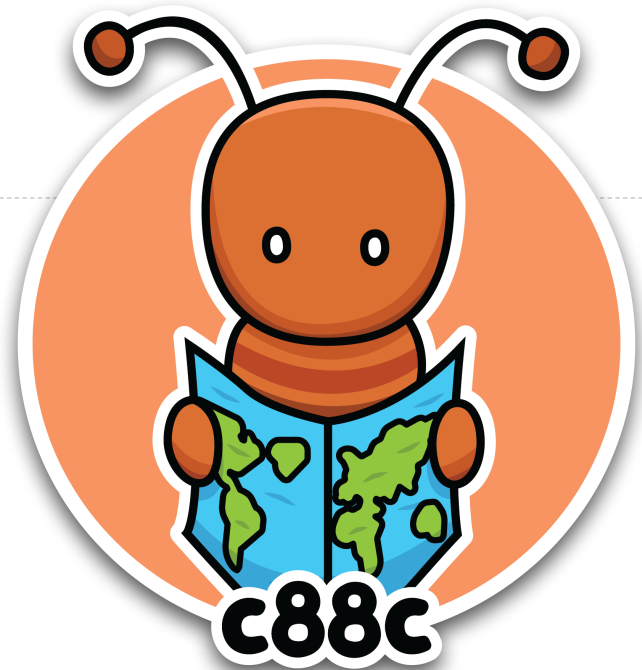


Conclusion



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 ~(˘▽˘)/~



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



John Teng 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

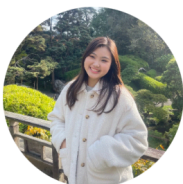
Hi! I am a sophomore studying Data Science and Cognitive Science. I love playing badminton and traveling! Excited to meet everyone!



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.

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.

`def smalls(t):` *Signature: Tree -> List of Trees*

"""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 = []` *Signature: Tree -> number*

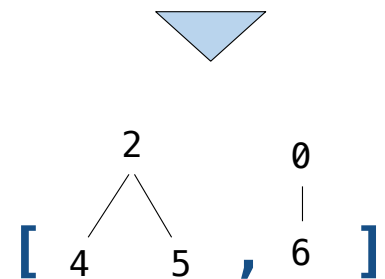
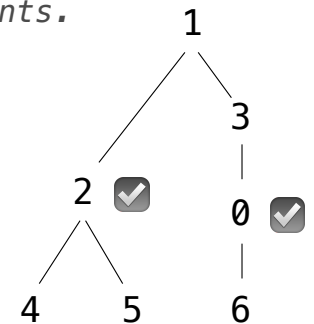
`def process(t):` *"Find smallest label in t & maybe add t to result"*

if t.is_leaf():
 return t.label
 else:

return min(...)

`process(t)`

`return 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.

`def smalls(t):` *Signature: Tree -> List of Trees*

"""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 = []` *Signature: Tree -> number*

`def process(t):` *"Find smallest label in t & maybe add t to result"*

if `t.is_leaf():`
 return `t.label`

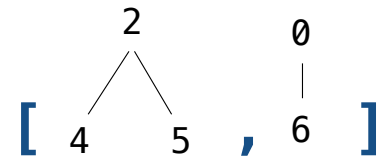
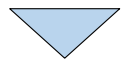
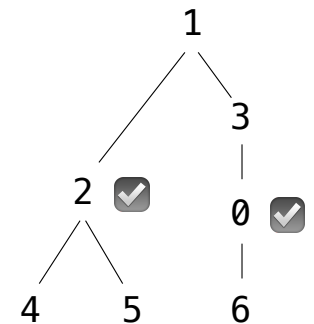
else:
 `smallest = min([process(b) for b in t.branches])`

if `t.label < smallest` *:*
 `result.append(t)`

return `min(smallest, t.label)`

`process(t)`

return `result`



Practice Question Fall 2021 Final #6

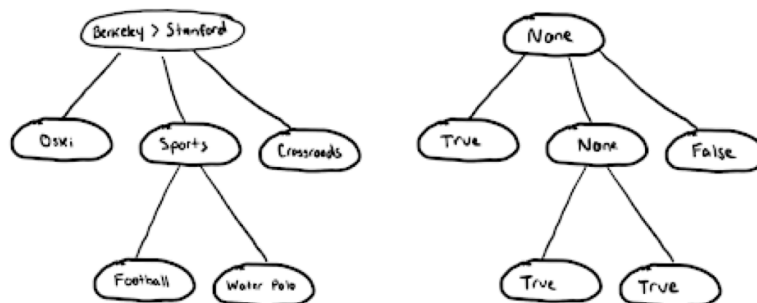
<https://c88c.org/sp25/assets/pdfs/exams/c88c/fa21-final.pdf>

6. (7.0 points) Objective Judge

A debate judge is in charge of declaring the winner of a two-sided debate. In order to be more objective, the judge starts by drawing out all the points (nodes) in the debate as a “concept tree.” A point’s children are sub-points that are related to that point.

Then the judge creates another tree of identical structure called the “winner’s tree” where non-leaf nodes are marked as `None` and leaf nodes are marked as `True` if the “for” side has won that point or `False` if the “against” side has won that point.

Complete the function `judge` which takes in a winner’s tree `t` corresponding to a debate and returns `True` if the “for” side is the winner and `False` if the “against” side is the winner. The judge decides the “for” side has won a given point if they have won the majority of points directly below that point. If there is a tie, the “against” side wins that point. Using these rules, the judge can start at the leaves and move up to determine who won the point that is the root node and therefore the debate.



Concept Tree (NOT used in function) | Winner's Tree (used in function)

Process

Read Question

Read Scaffold

Re-read Question

Assess structure

* base case

* type of problem

```
def judge(t):
    """
    >>> point1 = Tree(True)
    >>> point2 = Tree(None, [Tree(True), Tree(True)])
    >>> point3 = Tree(False)
    >>> point4 = Tree(None, [Tree(False), Tree(True), Tree(False)])
    >>> judge(Tree(None, [point1, point2, point3]))#debate1
    True
    >>> judge(Tree(None, [point1, point4, point3]))#debate2
    False
    """
    if -----:
        return t.value
    else:
        points_won = 0
        for b in t.branches:
            if -----:
                -----
        if -----:
            return True
        else:
            -----
```


-
- (a) (7.0 pt) Write the fully *completed* `judge` function below using the skeleton code provided. You may not add, change, or delete lines from the skeleton code.

```
def judge(t):
    if t.is_leaf():
        return t.value
    else:
        points_won = 0
        for b in t.branches:
            if judge(b):
                points_won += 1
        if 2 * points_won > len(t.branches):
            return True
        else:
            return False
```

Keep on Programming

That's all. Thanks!