Generators

Announcements

Tree Practice

Spring 2023 Midterm 2 Question 4(a)

Implement exclude, which takes a tree t and a value x. It returns a tree containing the root node of t as well as each non-root node of t with a label not equal to x. The parent of a node in the result is its nearest ancestor node that is not excluded.

```
def exclude(t, x):
    """Return a tree with the non-root nodes of tree t labeled anything but x.

    >>> t = tree(1, [tree(2, [tree(2), tree(3), tree(4)]), tree(5, [tree(1)])])
    >>> exclude(t, 2)
    [1, [3], [4], [5, [1]]]
    >>> exclude(t, 1)  # The root node cannot be excluded
    [1, [2, [2], [3], [4]], [5]]
    """
    filtered_branches = map(lambda y: ____________, branches(t))
    bs = []
    for b in filtered_branches:
        if ____________:
            bs.________(________)
        else:
            bs.append(b)
    return tree(label(t), bs)
```

exclude(y, x)

In Spring 2023, 20% of students got this right

label(b) == x

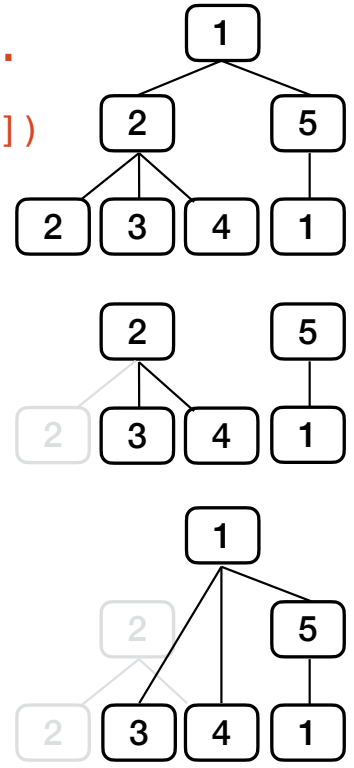
37% of students got this right

30% got it right; 1 of 4 options

extend

branches(b)

24% got it right



Min Practice

Match the description to the code

```
w = {...}  # a dict with unique keys and values

m = {v: k for k, v in w.items()}

Which expression evaluates to?
```

1. The key that has the smallest value in w
2. The value that has the smallest key in w
3. The smallest absolute difference between a key and its value

```
min(w.keys(), key=lambda k: w[k])
min(w.keys(), key=lambda k: m[k])
min(w.values(), key=lambda v: w[v])
min(w.values(), key=lambda v: m[v])
min(w.keys(), key=lambda k: abs(k - w[k]))
min(w.keys(), key=lambda k: abs(k - m[k]))
min(map(lambda k: abs(k - w[k]), w.keys()))
min(map(lambda k: abs(k - m[k]), w.keys()))
```

All

Discussion Question

all(s) iterates through s until a false value is found (or the end is reached).

What's printed when evaluating:

```
x = all(map(print, range(-3, 3)))
```

Why?

- print(-3) returns None after displaying -3
- None is a false value
- all([None, ...]) is False for any ...
- The map iterator never needs to advances beyond -3

Generators

Generators and Generator Functions

```
>>> def plus_minus(x):
...     yield x
...     yield -x

>>> t = plus_minus(3)
>>> next(t)
3
>>> next(t)
-3
>>> t
<generator object plus_minus ...>
```

A *generator function* is a function that **yield**s values instead of **return**ing them

A normal function **return**s once; a *generator function* can **yield** multiple times

A *generator* is an iterator created automatically by calling a *generator function*

When a *generator function* is called, it returns a *generator* that iterates over its yields

(Demo)

Spring 2023 Midterm 2 Question 5(b) Revisited

Definition. When parking vehicles in a row, a motorcycle takes up 1 parking spot and a car takes up 2 adjacent parking spots. A string of length n can represent n adjacent parking spots using % for a motorcycle, <> for a car, and . for an empty spot.

For example: '.%%.<><>' (Thanks to the Berkeley Math Circle for introducing this question.)

Implement **park**, a **generator function** that yields all the ways, represented as strings, that vehicles can be parked in n adjacent parking spots for positive integer n.

```
def park(n):
    """Yield the ways to park cars and motorcycles in n adjacent spots.

    >>> sorted(park(1))
    ['%', '.']
    >>> sorted(park(2))
    ['%%', '%.', '.%', '..', '<>']
    >>> len(list(park(4)))  # some examples: '<><>', '.%%.', '%<>%', '%.<>'
    29
    """
```