MA3: Tree Generators

Tom Smedsaas

This lecture discusses
generators for tree structures

UPPSALA UNIVERSITET

The LinkedList generator

We noticed the the easiest way to to write the __iter__ function in the LinkedList class was to do it as an generator:

```
    def __iter__(self):
        current = self.first
        while current:
            result = current.data
            yield result
            current = current.succ
```

With this construction we could iterate over a list wit the code:

```
ll = LinkedList()
...
for n in ll:
    do_something(n)
```

The for statement will use the generator to access the elements in the list.

UPPSALA UNIVERSITET

Tree generator

The situation in the `BST` class is more complicated since we have no easy way to say what the next element is.

However, if we write a generator in the `Node` class, we can iterate over the nodes in the left and right subtree by using the generators in the two root nodes there.

UPPSALA UNIVERSITET

BST-classes

```
class BST:

    class Node:
        def __init__(self, key,
                     left = None,
                     right = None):
            self.key = key
            self.left = left
            self.right = right


    def __init__(self, root = None):
        self.root = root

        . . . .
```

UPPSALA UNIVERSITET

Generator for BST

```
class BST:
    class Node:
        def __init__(. . .): . . .

        def __iter__(self):
            if self.left:
                for key in self.left:
                    yield key
            yield self.key
            if self.right:
                for key in self.right
                   yield key

    def __init__(. . .): . . .

    def __iter__(self):
        if self.root:
            for key in self.root:
                yield key
```

UPPSALA UNIVERSITET

Using `yield from`

The code can be a little simplified by using the `yield from` construction:

```
        def __iter__(self):                 # In the Node class
            if self.left:
                yield from self.left:
            yield self.key
            if self.right:
                yield from self.right


    def __iter__(self):                     # In the BST class
        if self.root:
            yield from self.root:
```

Note how easily the generator can be changed to do other traversals!

UPPSALA UNIVERSITET

The end