연습문제 12장

6

In [1]:
```
class Counter(object):
    def __init__(self, start=0):
        self.value = start
        self.step = 1
    def incr(self):
        self.value += self.step
    def __repr__(self):
        return str(self.value)
    __str__ = __repr__
```

In [2]:
```
c = Counter()
c.incr()
c
```

Out[2]: 1

In [3]:
```
class Counter(object):
    def __init__(self, start=0):
        self.value = start
        self.step = 1
    def incr(self):
        self.value += self.step
    def __repr__(self):
        return str(self.value)
    __str__ = __repr__
    def __call__(self):
        self.incr()
        return self.value
```

In [4]:
```
c = Counter()
c()
```

Out[4]: 1

In [5]:
```
c()
```

Out[5]: 2

__coerce__는 초기 파이썬에 사용되던 기능으로 파이썬 2.7 및 파이썬 3에서는 삭제되었다. 이 기능을 이해하려고 노력할 필요는 없어보인다.

7

In [6]:
```
class Set(object):
    def __init__(self, data):
        self.data = self._remove_duplicates(data)

    def _remove_duplicates(self, data):
        L = []
        for ele in data:
            if ele not in L:
                L.append(ele)
        return L


    def __and__(self, s2):
        L = []
        for e1 in self.data:
            if e1 in s2:
                L.append(e1)
        return Set(L)

    def __or__(self, s2):
        L = self.data[:]
        for e1 in s2:
            if e1 not in L:
                L.append(e1)
        return Set(L)

    def __sub__(self, s2):
        L = []
        for ele in self.data:
            if ele not in s2:
                L.append(ele)
        return Set(L)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, i):
        return self.data[i]

    def __setitem__(self, i, value):
        self.data[i] = value
        self.data = self._remove_duplicates(self.data)

    def __delitem__(self, i):
        del self.data[i]

    def __contains__(self, value):
        return value in self.data

    def __nonzero__(self):
        return bool(self.value)

    def __repr__(self):
```

```
        return repr(self.data)
```

In [7]:
```
s1 = Set([1,2,3,4])
s2 = Set([3,4,5,6])
s1 & s2
```

Out[7]: [3, 4]

In [8]:
```
s1 | s2
```

Out[8]: [1, 2, 3, 4, 5, 6]

In [9]:
```
s1 - s2
```

Out[9]: [1, 2]

8

In [10]:
```
class BNode:
    def __init__(self, value=None, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right
    def __repr__(self):
        return '%s (\n%s\n%s)' % (self.value, self.left, self.righ
t)
```

In [11]:
```
root = BNode('root')
root.left = BNode('left')
root.right = BNode('right')
root.left.left = BNode('left-left')
root.left.right = BNode('left-right')

print root
```

```
root (
left (
left-left (
None
None)
left-right (
None
None))
right (
None
None))
```

In [12]:
```
class BNode:
    def __init__(self, value=None, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right
        self.level = 0
        self.indent = '    '
    def __repr__(self):
        if self.left:
            self.left.level = self.level + 1
            left_repr = repr(self.left)
        else:
            left_repr = self.indent * (self.level+1) + 'None'
        if self.right:
            self.right.level = self.level + 1
            right_repr = repr(self.right)
        else:
            right_repr = '    ' * (self.level+1) + 'None'
        return '%s%s (\n%s\n%s)' % (self.indent*self.level, self.v
alue, left_repr, right_repr)
```

In [13]:
```
root = BNode('root')
root.left = BNode('left')
root.right = BNode('right')
root.left.left = BNode('left-left')
root.left.right = BNode('left-right')

print root
```

```
root (
    left (
        left-left (
            None
            None)
        left-right (
            None
            None))
    right (
        None
        None))
```

In [14]:
```

```