Week 8 Small Groups

Fill in the Blanks - Efficiency

For the sake of this exercise let:

- D = {"hello": “world”, "goodbye": "midtermweek"}
- S = {“you”, “students”, “are”, “the”, “best}
- L = [1,5,1,1,2]

Operation	Complexity
if “hello” in D:	
D.get(“goodbye”, None)	
if “you” in S:	
for elem in S:	
for key in D:	
for elem in L:	
set(L)	
L.count(val)	
L.append(val)	
L.pop()	
L.pop(i) where i is somewhere inside L	
L[i:i+k]	
L[i]	

```
Great resource for extra revision!:
http://www.krivers.net/15112-s19/notes/notes-efficiency-builtin-runti
me-table.html
```

Polynomial class

Problem Statement

Write the class Polynomial along with the required methods so that the following test function works correctly. Do not hardcode any test cases.

```
class Polynomial(object):
    def __init__(self, coeffs):  # Complete this method...
        pass
    # ...and finish writing the rest of the class

def testPolynomialClass():
    print('Testing Polynomial class...', end='')
    f = Polynomial([2,3,1]) # 2x**2 + 3x + 1
    assert(f.evalAt(4) == 2*4**2 + 3*4 + 1) # returns f(4), which is 45
    assert(f.evalAt(5) == 2*5**2 + 3*5 + 1) # returns f(5), which is 66
    assert(f.getCoefficient(0) == 1) # get the x**0 coefficient
    assert(f.getCoefficient(1) == 3) # get the x**1 coefficient
    assert(f.getCoefficient(2) == 2) # get the x**2 coefficient
    assert(f.getCoefficient(33) == 0) # assume leading 0's...
    g = f.times(10) # g is a new polynomial, which is 10*f
                    # just multiply each coefficient in f by this value
                    # so g = 20x**2 + 30*x + 10
    assert(g.getCoefficient(0) == 10) # get the x**0 coefficient
    assert(g.getCoefficient(1) == 30) # get the x**1 coefficient
    assert(g.getCoefficient(2) == 20) # get the x**2 coefficient
    assert(g.getCoefficient(33) == 0) # assume leading 0's...
    assert(g.evalAt(4) == 20*4**2 + 30*4 + 10) # returns g(4), which is 450

    m = f.add(f, g) # m is a new polynomial, which is f + g
    assert(m.getCoefficient(0) == 11) # get the x**0 coefficient
    assert(m.getCoefficient(1) == 33) # get the x**1 coefficient
    assert(m.getCoefficient(2) == 22) # get the x**2 coefficient

    # The printing representation should be as follows:
    assert(str(f) == "2x^2 + 3x + 1")
    assert(str(g) == "20x^2 + 30x + 10")

    # There should also be a way to compare instances
    assert(f == f.times(1))

    print('Passed!')
testPolynomialClass()
```

busiestStudents

Problem Statement

Background: this problem uses rosters:

```
    rosters = {
        '15-112':{'deniz','winnie','kian','sedef'},
        '18-100':{'amy','claire','john','mark'},
        '21-127':{'anita','kruthi','zach'},
        '76-101':{'bob','john','margaret'},
    }
```

We see that rosters is a dictionary, where each key is the name of a course, and each value is a set of the students in that course.

With that in mind, write the function busiestStudents(rosters) that takes a dictionary of rosters such as the one above (but do not hardcode to that one!), and returns a set of the students who are taking the most courses. In the example above, claire and john are both taking 3 courses, which is the most, so busiestStudents(rosters) returns the set { 'claire', 'john' }.

Hint: it may be helpful if you first create a dictionary mapping each student name to a count of the number of courses that student is taking, but this is just a suggestion.

```
def busiestStudents(rosters):
    return 42

def testBusiestStudents():
    print('Testing busiestStudents()...', end='')
    rosters = {
        '15-112':{'amy','bob','claire','dan'},
        '18-100':{'amy','claire','john','mark'},
        '21-127':{'claire','john','zach'},
        '76-101':{'bob','john','margaret'},
    }
    assert(busiestStudents(rosters) == { 'claire', 'john' })
    print('Passed!')

testBusiestStudents()
```

Efficiency Tracing

Find the Big-O efficiency of the following function:

```
def f(L):
  total = count = 0
  for i in range(len(L)):
    v = L[i]
    M = L[i+1:]
    for i in range(3):
      print(i)
    if (v not in M):
      total += v
      count += 1
  return total/count
```

Solution

Code Tracing

[CT] Code Tracing 1

```
#Hint: Sets are mutable, so
#   what does u = t create?

def ct1(L):
    s = set()
    t = set()
    for i in range(len(L)):
        if L[i]==i: u = t
        else:       u = s
        u.add(L[i])
    return (s, t)
print(ct1([4, 1, 6, 3, 6]))
```

Solution