homework-3

September 30, 2015

1 Introduction to Python

2 Homework #3

3 Due Tuesday, Oct 6, 11:59pm in Courseworks

4 Academic Honesty

- The computer science department has strict polices. Check the department web page for details.
- Do not look at anybody else's source code. Do not show anybody your source, or leave your source where somebody could see it. You MUST write your own code.
- For this class, feel free to discuss issues with other people, but suggest waiting an hour or two after a discussion, before writing your code.
- Cases of non original source will be refered to the Judical Committee.

5 Problem 1 - Substitution Cipher

- you will implement a simple substitution cipher encode/decode of the lower case letters, a-z
- encoding works by mapping the character set into a permutation of itself.
- decoding undoes the mapping

for example, if the map looks like

```
a <-> f
b <-> m
c <-> y
...
```

- to encode 'bac', go from left to right, and get 'mfy'
- to decode 'mfy', go from right to left, and get back 'bac'
- to implement, define classes that act as 'keys', with 'encode' and 'decode' methods
- pickle will save/load classes and their attributes

```
In [6]: # string module has lowercase letters
        import string

        alphabet = string.ascii_lowercase
        alphabet
```

```
Out[6]: 'abcdefghijklmnopqrstuvwxyz'
```

```
In [7]: # Simple cipher

        class swapaz:
            ''' swap 'a' and 'z' '''
            def encode(self, s):
                s = s.replace('a', 'A').replace('z', 'Z').replace('Z','a').replace('A', 'z')
                return(s)

            def decode(self, s):
                return(self.encode(s))
```

```
In [10]: s = swapaz()
         e = s.encode('larry')
         print(e)
         s.decode(e)
```

```
lzrry
```

```
Out[10]: 'larry'
```

```
In [11]: # a cipher that does not permute the alphabet is no good

         class bad:
             # substitution cipher must always return
             # same # of chars
             def encode(self, s):
                 return('x')

             def decode(self, s):
                 return('xy')

         class bad2:
             # substitution cipher must map all characters 1 to 1
             # here 'a' and 'b' both map to 'b'
             def encode(self, s):
                 return(''.join([ 'b' if e == 'a' else e for e in s]))
             def decode(self, s):
                 return(s)
```

6 Problem 1a - Define a predicate, 'goodperm'

- value is True for cipher that generates a valid permutation, False otherwise

```
In [16]: [goodperm(c) for c in [swapaz(), bad(), bad2()]]
```

```
Out[16]: [True, False, False]
```

7 Problem 1b - Define a function, saveKey(path, key)

- key is a key object
- path is a filename where the key object should be saved by using pickeling.
- normally saveKey should run silently and return None, but if it determines the key is bad, it should raise an 'Exception'
- the 'Exception' constructor takes a string arg, which is the message.

- remember you need a “binary” file stream when using pickling.

```
In [22]: # a valid key obj
         import pickle

         keypath = '/tmp/key.pickle'
         saveKey(keypath, swapaz())
```

8 Problem 2a

- write encode and decode functions that take:
- a path to a pickled key object
- a string to process

```
In [26]: e = encode(keypath, "larry")
         print(e)
         decode(keypath, e)
```

```
lzrry
```

```
Out[26]: 'larry'
```

9 Problem 2b

- write a key class that takes an integer, and makes a permutation by just cyclically rotating the character set
- any number of ways to implement this
- examples should make this clear

```
In [29]: # rotate 3 chars to the right
         e3 = encrot(3)
         e3.encode(alphabet)
```

```
Out[29]: 'xyzabcdefghijklmnopqrstuvw'
```

```
In [30]: # rotate 5 chars to the left
         em3 = encrot(-5)
         em3.encode(alphabet)
```

```
Out[30]: 'fghijklmnopqrstuvwxyzabcde'
```

10 Problem 3

- MIT has the complete works of Shakespeare in a simple html format
- You will do a simple analysis of Hamlet by reading the html file, one line at a time(usual iteration scheme) and doing pattern matching
- The goal is to return a list of the linecnt, total number of ‘speeches’(look at the file format), and a dict showing the number of ‘speeches’ each character gives
- Your program should read directly from the url given below, but you may want to download a copy to examine the structure of the file.
- There are any number ways to do this:
  - use string ‘find’ method
  - use regular expressions
  - use the ‘beautiful soup’ module

- you might find 'defaultdict' convenient
- python is very popular in 'digital humanities'
- here's a short sample of the file

```
<A NAME=speech25><b>HORATIO</b></a>
<blockquote>
<A NAME=1.1.37>Tush, tush, 'twill not appear.</A><br>
</blockquote>

<A NAME=speech26><b>BERNARDO</b></a>
<blockquote>
<A NAME=1.1.38>Sit down awhile;</A><br>
<A NAME=1.1.39>And let us once again assail your ears,</A><br>
<A NAME=1.1.40>That are so fortified against our story</A><br>
<A NAME=1.1.41>What we have two nights seen.</A><br>
</blockquote>

<A NAME=speech27><b>HORATIO</b></a>
<blockquote>
<A NAME=1.1.42>Well, sit we down,</A><br>
<A NAME=1.1.43>And let us hear Bernardo speak of this.</A><br>
</blockquote>

<A NAME=speech28><b>BERNARDO</b></a>
<blockquote>
<A NAME=1.1.44>Last night of all,</A><br>
<A NAME=1.1.45>When yond same star that's westward from the pole</A><br>
<A NAME=1.1.46>Had made his course to illume that part of heaven</A><br>
<A NAME=1.1.47>Where now it burns, Marcellus and myself,</A><br>
<A NAME=1.1.48>The bell then beating one,--</A><br>
<p><i>Enter Ghost</i></p>
</blockquote>

<A NAME=speech29><b>MARCELLUS</b></a>
<blockquote>
<A NAME=1.1.49>Peace, break thee off; look, where it comes again!</A><br>
</blockquote>

<A NAME=speech30><b>BERNARDO</b></a>
<blockquote>
<A NAME=1.1.50>In the same figure, like the king that's dead.</A><br>
</blockquote>
```

```
In [45]: import collections
         import re
         import urllib

         # use this url for the hamlet text, don't hit MIT
         url =
         'https://courseworks.columbia.edu/access/content/group/COMSW3101_002_2015_3/week3/hamlet.html'

         # note the lines returned will be of type 'byte'. you can see this by the
         # "b'" prefix. to get a string, the binary array must be decoded into unicode
```

```
        with urllib.request.urlopen(url) as ef:
            for bin in ef:
                print(bin)
                s = bin.decode('utf-8')
                print(s)
                break
```

```
b'<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"\n'
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
```

In [48]:
```
# when i first ran the program, i noticed the file had a few bugs
# oddly, ROSENCRANTZ and GUILDENSTERN each have a trailing colon once.


actors(url)
```

Out[48]:
```
[8881,
 1150,
 defaultdict(<class 'int'>, {'MARCELLUS': 36, 'FRANCISCO': 8, 'Captain': 7, 'First Clown': 33,
```

In [57]:
```
# could fix the ':' problem by tweaking the regex
actors2(url)
```

Out[57]:
```
[8881,
 1150,
 defaultdict(<class 'int'>, {'MARCELLUS': 36, 'FRANCISCO': 8, 'Captain': 7, 'First Clown': 33,
```

11 Problem 4

- implement a class 'Interval', that does 'interval arithmetic'
- an interval consists of a min and max value. use attribute names 'imin', 'imax' to avoid confusion with 'min' and 'max' functions
- let 'i' and 'i2' be intervals
- `i + i2 represents a new interval, where the new imin and imax is the min and max of (x + x2), where i.imin <= x <= i.imax and i2.imin <= x2 <= i2.imax`
- `i * i2 represents a new interval, where the new imin and imax is the min and max of (x * x2), where i.imin <= x <= i.imax and i2.imin <= x2 <= i2.imax`
- adding intervals is easy
- multiplying intervals - think for a second
- should be able to add or multiply by a scalar(an integer) on the right
- let i be an Interval, s a scalar(integer)
  - `i + s is the same as i + Interval(s, s)`
  - `i * s is the same as i * Interval(s, s)`
- an interval should print as `Interval<imin, imax>`
- use only integers, no floats

In [79]:
```
i = Interval(-1,6)
i2 = Interval(5, 13)
i3 = Interval(10,10)

[i + i2, i * i2, i + 10, i * 10, i + i3, i * i3]
```

```
Out[79]: [Interval<4, 19>,
          Interval<-13, 78>,
          Interval<9, 16>,
          Interval<-10, 60>,
          Interval<9, 16>,
          Interval<-10, 60>]
```

12 Problem 5 - Polynomials

- in class, we discussed two different ways to represent a polynomial
  - polylist, a 'dense' represenation, that hold the coefficients in a list
  - polydict, a 'sparse' representation, that holds (exponent, coefficent) pairs in a dict
- add a method, 'topolydict()' to class 'polylist', that converts the polylist into a polydict
- add a method, 'topolylist()' to class 'polydict', that converts the polydict into a polylist
- note that polylist->polydict will always work, but polydict->polylist can fail, because a polylist cannot represent negative exponents. in this case, raise a ValueError
- just to tell them apart, polylist prints with a leading '+'

```
In [71]: # prefab polylists

         pl1 = polylist([1,2,3])
```

```
In [74]: # prefab polydicts

         pd1 = polydict({2:3, 1:2, 0:1})
```

```
In [78]: [pl1, pl1.topolydict(), type(pl1.topolydict())]
```

```
Out[78]: [+ 3*X**2 + 2*X + 1, 3*X**2 + 2*X + 1, __main__.polydict]
```

```
In [77]: [pd1, pd1.topolylist(), type(pd1.topolylist())]
```

```
Out[77]: [3*X**2 + 2*X + 1, + 3*X**2 + 2*X + 1, __main__.polylist]
```

pdn = polydict({-2:3})
pdn.topolylist() would throw: ValueError: Negative exponent: -2