

CSE 399-005 Python Programming, Spring 2006

Homework 3

April 10, 2006

Instructions

1. This assignment is due on Monday April 17 at 11:59 PM.
2. You may discuss this HW *in a high level* with another student, but you should write your partner's name in `debrief.txt`. High-level discussions include algorithmic design (but **not** how to implement them), input/output specifications, language-independent concepts like regular expressions and command-line arguments, etc.
3. Other instructions follow HW 2.

Problems

problem 1 + problem 2 + pythonic-style = 70 + 20 + 10 = 100 points.

Problem 1 - Small World Phenomena (70 points)¹

Main program: `shortest.py`

Background. There is a well established theory of *small world phenomena* in the field of social networks (cf. CSE 112, Networked Life, taught by Prof. Kearns). The observation is that there tend to be relatively short paths of links connecting people or webpages. In this problem, you are to find the *shortest paths* between two given URLs.

Specification. The command line arguments are as follows:

```
./shortest.py [-h] [--help] [-d max] URL1 URL2
```

where `max` is the maximum number of steps allowed, with the default value 3 in case `-d` option is not specified. If there is no path within `max` links, print

```
unreachable within max link(s).
```

¹this problem is inspired by Mark Dreze.

Otherwise first print the number of links in the shortest path and then print the sequence of webpages from URL1 to URL2. Each webpage (including the source and the destination) have two lines, the second of which being its title (see examples for output format). If either `-h` or `--help` switch is present, the program will just print a “usage” for first-time users and exit. Your program should handle command-line arguments correctly: in case the user supplies incorrect options, it should also print the “usage” and exit. Please refer to the slides for a typical example. The skeleton program is available online for your convenience:

```
http://www.seas.upenn.edu/~cse39905/hw/hw3/shortest.py
```

Example 1. Output of this example is available online as `example1.out`.

```
bash $ ./shortest.py http://www.cis.upenn.edu/~lhuang3/ http://www.seas.upenn.edu/cets
shortest distance = 2
0 http://www.cis.upenn.edu/~lhuang3/
  Liang Huang's homepage
1 http://www.cis.upenn.edu
  Computer & Information Science
2 http://www.seas.upenn.edu/cets/
  Penn Engineering > CETS
```

Example 2.

```
bash $ ./shortest.py -d 1 http://www.cis.upenn.edu/~lhuang3/ http://www.seas.upenn.edu
unreachable within 1 link(s).
```

Caveats/Hints.

1. You just need to consider webpages under domain `upenn.edu`.
2. The source and destination URLs both start with `http://` and are both working webpages under `upenn.edu`.
3. Hyperlinks usually look like `link name`.
4. The title of a webpage is inside `<title>...</title>`.
5. Comments are *not* considered as part of the webpage. In Example 2, for instance, there is a commented hyperlink to SEAS from my homepage:

```
<!-- <a href="http://www.seas.upenn.edu">I don't have this link!</a> -->
which does not count as a legitimate link.
```

6. You need to translate some HTML escaped sequences back into normal characters. For example, in link 3 of Example 1, the raw title retrieved from the HTML is:

```
Penn Engineering &gt; CETS
```

where sequence `>` means “>” in HTML, because “>” itself has been reserved for a special meaning as the end of a tag (e.g., `<title>`). This is also true for link 2. You may want to consult the web for other such tags.

7. Some hyperlinks might be broken and your program should handle this correctly (it should not crash when following a broken link).
8. Some hyperlinks use *relative URLs*. For example, in the CIS homepage, there is a link:

```
<a href="./departmental/about.shtml" class="largeWHITE">
```

which links to

```
http://www.cis.upenn.edu/./departmental/about.shtml
```

You can always concatenate the current URL with relative URLs to get the absolute address.

9. Be careful about the slash / at the end of a URL. It is equivalent without this last slash. See Example 1 for details.
10. In this problem, we use the python interpreter specified by `/usr/bin/env`. Please make sure the following line is in your `.bashrc` file:

```
export PATH=/usr/local/python/2.4.2/bin/:$PATH
```

so that your default `python` in the Unix/Linux command-line points to version 2.4.2.

Algorithms. You can use either depth-first search or breadth-first search but we recommend the former (with recursion). Memoization is good but should be used carefully. You are also recommended to implement branch-and-bound with depth-first search.

Problem 2 - Rational Numbers Library (20 points)

Main program: `rational.py`

You are to write an object-oriented library of *rational numbers*, represented by the pair of integers (nominator and denominator). The skeleton code is as follows:

```
class Rational:
    "rational numbers"

    def __init__(self, nom, denom):
        self.nom = nom
        self.denom = denom
        self.reduce()

    def __str__(self):
        return str(self.nom) + "/" + str(self.denom)
```

where `reduce()` is a function that reduces the rational number to lowest terms (using the `gcd` program in HW 1).

You are to implement the following operators (`p` and `q` represent rational objects and `i` represents an integer):

operation	function	note
<code>p * q</code>	<code>__mul__</code>	
<code>p + q</code>	<code>__add__</code>	
<code>p - q</code>	<code>__sub__</code>	
<code>p / q</code>	<code>__div__</code>	
<code>i * p</code>	<code>__rmul__</code>	
<code>i + p</code>	<code>__radd__</code>	
<code>- p</code>	<code>__neg__</code>	
<code>~ p</code>	<code>__invert__</code>	(1/p)
<code>p ** i</code>	<code>__pow__</code>	
<code>p < q, p > q, p == q</code>	<code>__cmp__</code>	
<code>p += q</code>	<code>__iadd__</code>	
<code>p -= q</code>	<code>__isub__</code>	
<code>p *= q</code>	<code>__imul__</code>	
<code>p /= q</code>	<code>__idiv__</code>	

Caveats.

1. For all operations, the resulting rational number should already be in its lowest terms.
2. The resulting denominator can *not* be negative (e.g., $2/-3$ should be written as $-2/3$).
3. In case the denominator is 0, you'd raise an error like the following:

```
raise "Denominator is zero!"
```

Examples.

```
>>> from rational import *
>>> Rational(1,2) - Rational(2,1)
-3/2
>>> Rational(1,2) * Rational(2,3)
1/3
>>> print Rational(1,2) / ~Rational(2,-4)
-1/4
>>> p = Rational(1,2)
>>> p *= Rational(2,5)
>>> print p
1/5
>>> 5 + (-Rational(1,2))
9/2
>>> Rational(2,4) ** 10
1/1024
```

Debriefing

Please answer these questions in `debrief.txt` and submit it along with the programs. There will be 5 points off if you didn't submit this part. If you realized that you forgot this part after the deadline, you can just email your `debrief.txt` to TA Bill (`kandylas@cis`).

1. How many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Are the lectures too fast, too slow, or just in the right pace?
4. Any other comments?
5. If you discussed this HW *in a high-level* with somebody else, who is your partner?