Homework 9: Reusing code & Regular expressions

L435/L555

Due Tuesday, December 6

1. Adapted from section 15.5 of *Think Python*, which gives the following suggested exercise (`http://greenteapress.com/thinkpython2/html/thinkpython2016.html`):

   > As an exercise, write a function named `move_rectangle` that takes a Rectangle and two numbers named `dx` and `dy`. It should change the location of the rectangle by adding `dx` to the `x` coordinate of `corner` and adding `dy` to the `y` coordinate of `corner`.

   The code for this is available, but I recommend trying to do this before moving on to your exercise. (Note that you will have to read, understand, and look at the code for everything in the chapter that comes before this.)

   Your task:

   (a) Set default width, height, and corner values for `Rectangle`

   (b) Change the way of setting these values to be a method-based way.

   (c) Create a method to calculate the area of a rectangle.

   (d) Create a `Square` class, which is a subclass of `Rectangle`. If necessary, change your methods from (b) to ensure that all sides of a square must be equal.

2. Extend the program from question #2 of In-class Exercise 8 and add a class for language courses, which is a subclass of the `course` class and has additional information about the language taught, the expected incoming level of proficiency, and the expected final level of proficiency. All of these types of information should be accessible and should have methods to change the values. Make sure that, if an object of this class is created, the requirements state that the classroom must be a multimedia room.

3. What does the following program do?

```
import re

x = 0
y = []
while x < 100:
    sent = input('gimme a sentence: ')
    if re.search('[A-Z]{2,5}$', sent):
        sent = re.sub('[0-9]+\.[0-9]+', '<PAT1>', sent)
        sent = re.sub(' [A-Z][a-z]+ [A-Z][a-z]+ [^A-Z]', '<PAT2>', sent)
        y += re.split('<PAT.>', sent)
    x += 1
print(y)
```

4. Write a program that:

   - takes a POS tagged text as input, where the POS tags are separated from the word by a '/';
   - using regular expressions, searches for all occurrences of noun phrases (NPs) with a determiner (DT), 1 or more adjectives (JJ, JJR, JJS) and a noun (NN, NNS);
   - prints out each NP, one per line, without the POS tags, i.e., text only.

   Use the file `newvm.pos` as testfile. (Note that the file has 50 sentences.)

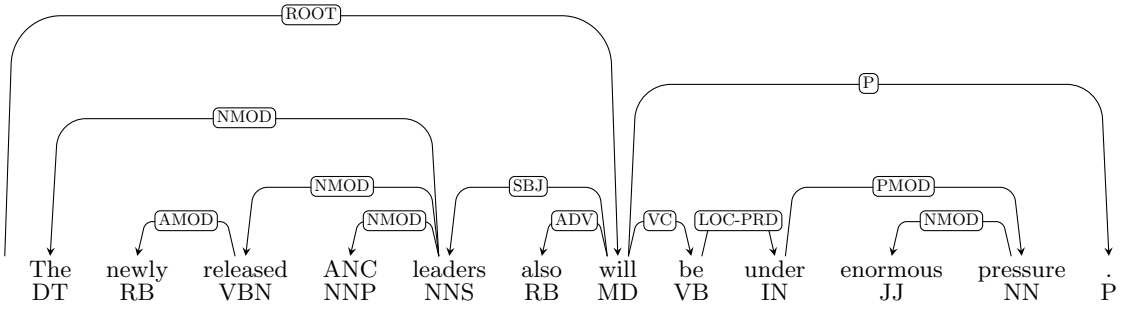
5. **Bonus:** Even if you don't do/finish this question for this assignment, I recommend working on it after the semester.

   Write a program that reads in a file that contains syntactically annotated sentences (dependencies). The program should provide a function that lists for each sentence all complex NPs, i.e. the noun and its modifiers (dependency label: `NMOD`). This means, you need to look for words that have the dependency label `NMOD`, and print them with their head. Sentences are separated by empty lines. Use the file `dep.txt` from Canvas as a test file.

   The dependency annotations have the following format:

```
1 The       _ DT  _ _ 5  NMOD    _ _
2 newly     _ RB  _ _ 3  AMOD    _ _
3 released  _ VBN _ _ 5  NMOD    _ _
4 ANC       _ NNP _ _ 5  NMOD    _ _
5 leaders   _ NNS _ _ 7  SBJ     _ _
6 also      _ RB  _ _ 7  ADV     _ _
7 will      _ MD  _ _ 0  ROOT    _ _
8 be        _ VB  _ _ 7  VC      _ _
9 under     _ IN  _ _ 8  LOC-PRD _ _
10 enormous _ JJ  _ _ 11 NMOD    _ _
11 pressure _ NN  _ _ 9  PMOD    _ _
12 .        _ .   _ _ 7  P       _ _
```

   The first column has an index for each word, the second column has the words, the fourth the POS tags, the seventh the index of the word's head, and the eighth the dependency label. A visualization might thus look like this:



   Taking all these points together, for this sentence the output might look like the following:

```
SENTENCE 1 NPs:
    The released ANC leaders
    enormous pressure
```