Homework 6: Regular Expressions

Due: 03.12.2021 at 11:59 pm ET

The length of this homework is inversely proportional to your knowledge in writing regular expressions, both for finding matches and for doing substitutions.

Background

Please refresh your memory of regular expressions using the class notes. You may also find the Python documentation on regular expressions useful.

A few helpful reminders:

Testing for Patterns

When you use `re.search` to find a regular expression match, it returns a `Match` object if the pattern exists in the string (we will see more about objects later in the semester). If *there is no match*, then `re.search` (and `re.match` , `re.findall` and `re.findall`) will return `None` , which you can test for:

```
p = re.compile('pattern')
if (p.search(s)) :
   # This branch will execute if the pattern is found
else :
   # this branch will execute if the pattern is *not* found
```

Substituting with functions

A "normal" use of `re.sub` is to substitute one string for another (remember that you can use the *groups* that you match in a pattern as part of your string substitution):

```
s = "loooool"
p = re.compile('(l)o+(l)')
p.sub(r'\1o\2', s) #replace "loooool" with "lol"
```

You can also call a method instead of providing a replacement string. This method will be called with the `Match` object corresponding to the matched string, and should return a string:

```
def replFun(m) :
   return m.group(2).upper()
s = "loooool"
```

```
p = re.compile('(l)(o+)(l)')
p.sub(r'\1'+replFun(m) +r'\3', s) #replace "loooool" with "lOOOOOl"
```

Instructions

0) Set up your repository

Click the link on Piazza to set up your repository for HW 6, then clone it.

The repository should contain two files:

1. `problems.py`, the file in which you will fill in the functions for the problems. This also contains test code you can use to test your solutions.
2. This README.

Problem 1: Regular expression matches

Fill in the function `problem1`. This function should return `True` if the input string *is a valid phone number* and `False` if not. We define a valid phone number as follows:

First, it contains an *optional* area code (3 digits) followed by 7 digits.

Second, there could be one of several possible formats for the phone number, where the `X` s are digits:

```
(XXX) XXX-XXXX
XXX-XXX-XXXX
XXX-XXXX
```

ANY other format should not count as a valid phone number. Spaces before or after an otherwise number is considered invalid.

Remember that `(`, `)`, `-` and `.` are special characters for regular expressions. To search for those characters, you need to precede them with a backslash: `\(` `\)`, `\-`, `\.`.

Because we are looking for the entire string to be a phone number, you can either use `^` and `$` to force a match to be at the beginning and end of a string, or you can use `fullmatch` instead of `match` or `search`.

Problem 2: Groups

Consider a regular expression that identifies street addresses, with the format:

1. **One or more digits**, followed by a space.

2. **One or more words**, each **starting with a capital letter** and then **followed by zero or more lowercase letters**. This will be followed by a space.
3. A road type, **one of "Rd.", "Dr.", "Ave." or "St."**

So the following are valid street names:

```
465 Northwestern Ave.
```

```
201 South First St.
```

```
22 What A Wonderful Ave.
```

```
123 Mayb3 Y0u 222 Did not th1nk 333 This Through Rd.
```

Assume that we will only test with valid street names. There will only be 'one valid street name' in a test case. However, there may be other words preceeding or following the valid street name. Please note the last test case and strictly adhere to the specifications in the address format mentioned above.

We consider street names to be valid if they satify the above mentioned rubric in problem 2. Hint: Using |w or |W will not solve this problem.

Fill in the function `problem2` . This function should search an input string for any valid street address, then return *just the street name* from that address: not the street number, and not the road type. So if you pass in:

```
The EE building is at 465 Northwestern Ave.
```

you should return:

```
Northwestern
```

If you pass in:

```
Meet me at 201 South First St. at noon
```

you should return:

```
South First
```

Also, if you pass in:

```
123 Mayb3 Y0u 222 Did not th1nk 333 This Through Rd. Did Y0u Ave.
```

you should return:

`This Through` .

(Note: Existence of any character which interferes with any upper case letter being followed by zero or more lowercase letters, after one or more digits invalidates the address name. This should indicate that you begin looking in the rest of the input)

Be careful not to return extra spaces in the return value. You may need to do a little bit of extra processing of the string captured by your group to ensure this. You will receive partial credit for having spaces. Please remove extra spaces for full credit.

Problem 3: Substitution

Fill in the function `problem3`. This function should *garble* addresses by returning the original address but with the street name reversed. For some of the above examples, you should return:

```
The EE building is at 465 nretsewhtroN Ave.
```

```
Meet me at 201 tsriF htuoS St. at noon
```

(Note that the entire street name is reversed, not word by word)

and, if your input is `Go West on 999 West St.`, you should return:

```
Go West on 999 tseW St.
```

*(Note that **only** the street name is reversed)*.

Two hints:

1. Think about creating three groups in your regular expression. One that captures the street number, one that captures the street name, and one that captures the road type. You can then use those three groups to assemble the appropriate
2. If you have a string `s`, `s[::-1]` is the reversed string.

You should feel free to write helper functions to help solve this problem.

Be careful not to return extra spaces in the final output. You may need to do a little bit of extra processing of the string captured by your group to ensure this. You will receive partial credit for having unwanted spaces. Please remove extra spaces for full credit.

What to Submit

Please submit `problems.py` with all the functions filled in.

Submitting your code

Please add, commit and push the latest version of your code, as you did in the previous HW. Do not make any modifications to this post submission and prevent the late penalty.