DS 100: Principles and Techniques of Data Science **Date: October 2, 2019**

Discussion #6 Solutions

Name:

Regular Expressions

Here's a complete list of metacharacters:

`.   ^   $   *   +   ?   { }   [ ]   \   |   ( )`

Some reminders on what each can do (this is not exhaustive):

"^" matches the position at the beginning of string (unless used for negation "[^]")

"$" matches the position at the end of string character.

"?" match preceding literal or sub-expression 0 or 1 times.

"+" match preceding literal or sub-expression *one* or more times.

"*" match preceding literal or sub-expression *zero* or more times

"." match any character except new line.

"[ ]" match any one of the characters inside, accepts a range, e.g., "[a-c]".

"( )" used to create a sub-expression

"\d" match any *digit* character. "\D" is the complement.

"\w" match any *word* character (letters, digits, underscore). "\W" is the complement.

"\s" match any *whitespace* character including tabs and newlines. \S is the complement.

"*?" Non-greedy version of *. Not fully discussed in class.

"\b" match boundary between words. Not discussed in class.

"+?" Non-greedy version of +. Not discussed in class.

Some useful `re` package functions:

`re.split(pattern, string)` split the `string` at substrings that match the `pattern`. Returns a list.

`re.sub(pattern, replace, string)` apply the `pattern` to `string` replacing matching substrings with `replace`. Returns a string.

`re.findall(pattern, string)` Returns a list of all matches for the given `pattern` in the `string`.

Regular Expressions

1. Which strings contain a match for the following regular expression, `"1+1$"`? The character `"␣"` represents a single space.

   ○ `What␣is␣1+1` [✓ `Make␣a␣wish␣at␣11:11`] ○ `111␣Ways␣to␣Succeed`

   Solution: Recall that `1+` matches on *at least one* occurrence of the character `1`, and `$` marks the end of the string.

2. Given the text:

   `"<record>␣Josh␣Hug␣<hug@cs.berkeley.edu>␣Faculty␣</record>"`
   `"<record>␣Manana␣Hakobyan␣<manana.hakobyan@berkeley.edu>␣TA␣</record>"`

   Which of the following matches exactly to the email addresses (including angle brackets)?
   ○ `<.*@.*>` [✓ `<[^>]*@[^>]*>`] ○ `<.*@\w+\..*>`

   Solution: Greediness matches too much in the first and third choices.

3. For each pattern specify the starting and ending position of the first match in the string. The index starts at zero and we are using closed intervals (both endpoints are included).

	`abcdefg`	`abcs!`	`ab␣abc`	`abc,␣123`
`abc*`	$[0, 2]$	$[0, 2]$	$[0, 1]$	$[0, 2]$
`[^\s]+`	$[0, 6]$	$[0, 4]$	$[0, 1]$	$[0, 3]$
`ab.*c`	$[0, 2]$	$[0, 2]$	$[0, 5]$	$[0, 2]$
`[a-z1,9]+`	$[0, 6]$	$[0, 3]$	$[0, 1]$	$[0, 3]$

4. Write a regular expression that matches strings (including the empty string) that only contain lowercase letters and numbers.

   Solution:

   ```
   ^[a-z0-9]*$
   ```

5. Write a regular expression that matches strings that contain exactly 5 vowels.

Solution:

```
^([^aeiouAEIOU]*[aeiouAEIOU]){5}[^aeiouAEIOU]*$
```

6. Given that `address` is a string, use `re.sub` to replace all vowels with a lowercase letter “o”. For example `"123␣Orange␣Street"` would be changed to `"123␣orongo␣Stroot"`.

Solution:

```
re.sub(r"[aeiuAEIOU]",  "o", address)
```

7. Given that `sometext` is a string, use `re.sub` to replace all clusters of non-vowel characters with a single period. For example `"a␣big␣moon,␣between␣us..."` would be changed to `"a.i.oo.e.ee.u."`.

Solution:

```
re.sub( r"[^aeoiuAEIOU]+", "." , sometext)
```

8. Given `sometext = "I've␣got␣10␣eggs,␣20␣gooses,␣and␣30␣giants."`, use `re.findall` to extract all the items and quantities from the string. The result should look like **`['10 eggs', '20 gooses', '30 giants']`**. You may assume that a space separates quantity and type, and that each item ends in s.

Solution:

```
re.findall(r"\d\d\s\w+s", sometext)
```

9. Given the following text in a variable `log`:

```
169.237.46.168 - - [26/Jan/2014:10:47:58 -0800]
"GET␣/stat141/Winter04/␣HTTP/1.1" 200 2585
"http://anson.ucdavis.edu/courses/"
```

Fill in the regular expression in the variable `pattern` below so that after it executes, day is 26, month is Jan, and year is 2014.

```
pattern = ...
matches = re.findall(pattern, log)
day, month, year = matches[0]
```

Solution:

```
pattern = "\[(.+)\/(.+)\/([^:]+).*\]"
matches = re.findall(pattern, log)
day, month, year = matches[0]
```