

Discussion #4

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 "[^"]") | "\d" match any <i>digit</i> character. "\D" is the complement. |
| "\$" matches the position at the end of string character. | "\w" match any <i>word</i> character (letters, digits, underscore). "\W" is the complement. |
| "?" match preceding literal or sub-expression 0 or 1 times. | "\s" match any <i>whitespace</i> character including tabs and newlines. \S is the complement. |
| "+" match preceding literal or sub-expression <i>one</i> or more times. | "*?" Non-greedy version of *. Not fully discussed in class. |
| "*" match preceding literal or sub-expression <i>zero</i> or more times | "\b" match boundary between words. Not discussed in class. |
| "." match any character except new line. | "+"? Non-greedy version of +. Not discussed in class. |
| "[]" match any one of the characters inside, accepts a range, e.g., "[a-c]" | "{m,n}" The preceding element or subexpression must occur between m and n times, inclusive. |
| "()" used to create a sub-expression | |

Some useful `re` package functions:

- | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>re.split(pattern, string)</code> split the <code>string</code> at substrings that match the <code>pattern</code> . Returns a list. | ing matching substrings with <code>replace</code> . Returns a string. |
| <code>re.sub(pattern, replace, string)</code> apply the <code>pattern</code> to <code>string</code> replac- | <code>re.findall(pattern, string)</code> Returns a list of all matches for the given <code>pattern</code> in the <code>string</code> . |

Regular Expressions

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

☐ A. What_is_1+1 ☐ B. Make_a_wish_at_11:11 ☐ C. 111_Ways_to_Succeed

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)?

☐ A. <.*@.*> ☐ B. <[^>]*@[^>]*> ☐ C. <.*@\w+\..*>

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]			
[^\s]+				
ab.*c				
[a-z1, 9]+				

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

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

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"`.

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."`.

8. Given `sometext = "I've_got_10_eggs,_20_goosees,_and_30_giants."`, use `re.findall` to extract all the items and quantities from the string. The result should look like `['10 eggs', '20 goosees', '30 giants']`. You may assume that a space separates quantity and type, and that each item ends in s.

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]
```