Faculty of Computers and Information
Cairo University
NLP-2019

NLP
Lab-1
Regular Expressions

Lab Objectives

- Learn about regular expressions
- Use python to implement regular expressions through a library named (re)

Regular expressions (Regex) (RE)

A regular expression is a sequence of characters that define a search pattern. Regular expression is used in search engines, search and replace dialogs of word processors and text editors, and in lexical analysis.

Syntax of regular expressions

Each character in a regular expression is either a metacharacter, having a special meaning, or a regular character that has a literal meaning.

Metacharacters in python re library

ad and e are constants (regular characters)

Meta character	Description	Examples
*	**zero or more:** Matches any number of occurrences of the previous characters	**ad*e** matches ae, ade, adde, addde,...
?	**zero or one:** Matches at most one occurrence of the previous characters	**ad?e** matches ae and ade
+	Matches **one or more** occurrences of the previous characters	**ad+e** matches ade,adde,addde, ...
+?, *? ,??	The '*', '+', and '?' qualifiers are all greedy; they match as much text as possible. Adding ? matches **as few characters as possible**	**Given <a> b <c>** **<.*>** matches **<a> b <c>** **<.*?>** matches **<a>**
{n}	Matches **exactly n** occurrences of the previous RE	**ad{2}e** matches adde
{n,}	**n or more:** Matches n or more occurrences of the previous RE	**ad{2,}e** matches Adde,addde,...
{n,m}	**from n to m:** Matches from n to m occurrences of the previous characters	**ad{2,4}e** matches Adde,addde,adddde.
{n,m}?	**fewer characters as possible** will be matched	
.	Matches **one occurrence of any character** of the alphabet or digit except new line (can include new line character with the re.DOTALL flag)	**a.e** matches aae, aAe, abe, aBe, a1e, etc.

Meta character	Description	Examples
.*	Matches **any string of characters** and until it encounters a new line character	
[…]	Matches **one occurrence of any character contained in the list**	**[abc]** means one occurrence of either a, b, or c
[^…]	Matches **one occurrence of any character not contained in the list**	**[^abc]** means one occurrence of any character that is not an a, b, and c
[A-Z]	Matches **one upper-case letter**	
[a-z]	Matches **one lower-case letter**	
[0-9]	Matches **one digit**	
\d	Matches **one digit**. Equivalent to [0-9]	**A\dC** matches A0C, A1C, A2C, A3C,…
\D	Matches **one non-digit**. Equivalent to [^0-9]	
\w	Match **one word character**: letter, digit, or underscore. Equivalent to [a-zA-Z0-9_]	**1\w2** matches 1a2, 1A2, 1b2, 1B2, …
\W	Matches **one special character**. Equivalent to [^\w]: non alphanumeric @! # …	
\s	Matches **one whitespace character**: space, tabulation, new line, form feed, etc.	
\S	Matches **one non-white space character**. Equivalent to [^\s]	
|	Or operator between two RE	**a|bc** matches a or bc
( )	Grouping of a pattern	**(a|b)c** matches ac or bc
^	Match **from the start of string** (or **start of line** with re.MULTILINE flag)	**^cat$** matches cat alone in a line
$	Match **the end of string or line**	
\	Match **special character**	Match * using *
\b	**To match whole word**. Matches the empty string, but only at the beginning or end of a word	**'\bfoo\b'** matches 'foo', 'foo.', '(foo)', 'bar foo baz' but not 'foobar' or 'foo3'

Python re library

- To import re library we use the statement: `import re`
- 3 main functions are provided:
  - `re.search()`: checks for a match to the pattern anywhere in the string
  - `re.match()`: checks for a match to the pattern only at the beginning of the string
  - `re.findall()`: returns list of all matches non-overlapping in the string

- Each function takes 3 parameters:
  - Pattern: the regex
  - String: the string that will be searched to find the pattern
  - Flag: The expression's behavior can be modified by specifying a flags value

Examples

	Code	output
1	`m= re.search('.*','abc133\\33') #Match any string` `print(m.group(0))`	abc133\33

2	`m= re.search('.*?','abc133\\33') #Note the change when adding ?`
	`print(m)`
	`print(m.group(0)) #prints nothing`
		<_sre.SRE_Match object; span=(0, 0), match=''>

3	`m= re.search('abcd*','abc133\\3') #Match any string has abc and d with any number of occurrences` `print(m.group(0))`	abc

4	`m= re.search('abcd+','abc133\\3')`
	`#Match any string has abc and d. d should appear 1 or more`
	`print(m)`
	`print(m.group(0)) #Runtime error`
		None

5	`m= re.search('abcd+',' abc31abcdd33\\3') #Match any string has abc and d. d should appear 1 or more` `print(m.group(0))`	Abcdd

6	`m= re.search('^abcd+',' abc31abcdd33\\3')`
	`#Match any string has abc and d appear in the beginning.`
	`#d should appear 1 or more`
	`print(m)`
	`m.group(0) #Runtime error`
		None

7	`m= re.search('^a{3}',' abc31abcdd33\\3') #Match any string has aaa appear in the beginning.` `print (m)`	None
8	`m= re.search('^a{3,5}','aaaabc31abcdd33\\3') #Match any string has aaa or aaaa or aaaaa appear in the beginning.` `print (m)`	<_sre.SRE_Match object; span=(0, 4), match='aaaa'>

9	`m= re.search('a{3,5}|b{2}|^c{2}','aaaabc31abcdd33\\3')`
	`#Match any string has aaa or aaaa or aaaaa or bb or cc in the beginning.`
	`print (m)`
		<_sre.SRE_Match object; span=(0, 4), match='aaaa'>

	Code	output
10	`m= re.search('[fek]',' aaaabc31fabcdd33\\3') #Match f or e or k.` `print (m)`	<_sre.SRE_Match object; span=(0, 4), match='f'>

match.group(n)

Returns one or more subgroups of the match.

Example:

```
m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
m.group(0)       # The entire match
#'Isaac Newton'
m.group(1)       # The first parenthesized subgroup.
#'Isaac'
m.group(2)       # The second parenthesized subgroup.
#'Newton'
m.group(1, 2)    # Multiple arguments give us a tuple.
#('Isaac', 'Newton')
```

Alternative method to define pattern

If the pattern will be used multiple times on different sequences, it is define a regex object storing the pattern and then use this object to call the match, search, or other methods.

Example:

```
import re
reObj = re.compile('\d+')#find all numbers
m= reObj.findall('aaaabc31fabcdd33\\3')
print (m)
```

```
['31', '33', '3']
```

Tasks

(1) Search on Vodafone number on the string where the length number should be 11 and start with 010. The String is 'My phone number is 01033192192'

(2) Find All adverbs in this text 'He was carefully disguised but captured quickly by police'

(3) Split The string based on number '20A50B1C19D', hint: try split function

Solutions

(1) Vodafone number:

```
re.search('010[0-9]{8}','My phone number is 01033192192.')
print (m.group(0))
```

(2) Adverbs

```
text = "He was carefully disguised but captured quickly by
police."
m = re.findall(r"\w+ly", text)
print (m)
```

(3) Split by numbers

```
m = re.split('\d+', '20A34B13C')
print (m)
```

Student task

Write a regex to extract all emails from a text.

For more Information

https://docs.python.org/2/library/re.html