Python

Regular Expressions

Fall, 2018

I don't want to play golf. When I hit a ball, I want someone else to go chase it.

- Roger Hornsby

Administrivia

Notes on Regular Expressions

Motivation

We often need to verify data

Does this string hold a valid date?

```
import re
lst = re.split('/', '6/25/2018')
print(lst)
['6', '25', '2018']
```

Now we can check length and range of each component

We were able to split without RE - what do Regular Expressions add?

Methods

The power in Regular Expressions lies in their ability to describe general patterns

But let's start by looking at some key methods

```
import re
text = 'I yam what I yam'
m = re.match('yam', text)    # Find at start
m = re.search('yam', text)   # Finds first
m = re.findall('yam', text) # ['yam', 'yam']
m = re.split('yam', text)    # ['I ',' what I ','']
m = re.sub('yam', 'am', text) # 'I am what I am'
```

Wild Cards

We can match an arbitrary character with '.'

```
import string
import re
s = 'put the pot upon the spit'
m = re.findall('p.t', s)
```

```
['put', 'pot', 'pit']
```

Anchors

Match start of a string with ‘^’ and the end with ‘$’

```
        s = 'put the pot upon the spit'
        m = re.findall('^p.t', s)
['put']

        m = re.findall('p.t$', s)
['pit']
```

Set of matches

We can specify a set of characters to match

[aeou] means a, e, o, or u

```
s = 'put the pot upon the spit'
m = re.findall('p[aeou]t', s)
```

```
['put', 'pot']
```

Set of matches

We can negate a set with ^ - [^aeou]

We have also used ^ to anchor the start

```
s = 'put the pot upon the spit'
m = re.findall('p[^aeou]t', s)
```

```
['pit']
```

Classes of symbols

We look for digits (\d) in a list of printable characters

```
import string
import re
printable = string.printable
let = re.findall('\d', printable)
print(lst)
['0', '1', '2', '3', '4', '5', '6',
'7', '8', '9']
```

Some definitions

\d	Digit	[0-9]
\D	Non-digit	[^0-9]
\w	Alphanumeric	[a-zA-Z0-9]
\W	Non alphanumeric	[^a-zA-Z0-9]
\s	Whitespace	' \t\n\r\x0b\x0c'
\S	Non whitespace	
\b	Word boundary	
\B	Non word boundary	

Match a Social Security Number

```
# Looking for 9 digits in text 123-45-6789
# \d   Digit                    [0-9]


if re.search('\d\d\d-\d\d-\d\d\d\d', text):
    return True                 # Found SS
```

Repetition

```
# Looking for 9 digits: 123-45-6789
# \d   Digit                  [0-9]
# We can include a count {3}

if re.search('\d{3}-\d{2}-\d{4}', text):
    return True # Found SS
```

Non-explicit counts

```
    print(re.findall('[a-z]\d', 'a b1 c23'))
['b1', 'c2']            # one letter, one digit
    print(re.findall('[a-z]\d*', 'a b1 c23'))
['a', 'b1', 'c23']      # * zero or more digits
    print(re.findall('[a-z]\d+', 'a b1 c23'))
['b1', 'c23']           # + one or more
    print(re.findall('[a-z]\d?', 'a b1 c23'))
['a', 'b1', 'c2']       # ? zero or one
```

All words with only vowels

```
def find_words( ):
    lst = []
    with open('words.txt', 'r') as f:
        for line in f:
            word = line.strip()
            pattern = '^[aeiou]+$'
            if re.search(pattern, word):
                lst.append(word)
    return lst
```

Smith, C —> C Smith

```
pattern = '^[A-Z]\w+,?\s*[A-Z]$'

re.findall(pattern, 'Smith, C')
['Smith, C']

re.findall(pattern, 'C Smith')
[]

re.findall(pattern, 'Jones K')
['Jones K']
```

Break it down

```
pattern = ‘^[A-Z]\w+,?\s*[A-Z]$'
          ^      Start of string
          [A-Z]  One upper case letter
          \w+    One or more alphanumeric
          ,?     Optional comma
          \s*    Zero or more white space
          [A-Z]  One upper case letter
          $      End of string
re.findall(pattern, 'Smith, C’)
re.findall(pattern, 'Potter-Pirbright, C’)
```

Grouping

```
# Old pattern = '^[A-Z]\w+,?\s*[A-Z]$'
pattern = '^([A-Z]\w+)(,?\s*[A-Z])$'

match = re.search(pattern, 'Smith, C')

print(match.group(0))     # Smith, C
print(match.group(1))     # Smith
print(match.group(2))     # , C
```

Different Decomposition

```
# pattern = '^([A-Z]\w+)(,?\s*)([A-Z])$'

match = re.search(pattern, 'Smith, C')

match.group(0)     # Smith, C
match.group(1)     # Smith
match.group(2)     # ,
match.group(3)     # C
match.group(3) + match.group(1) # C Smith
```

More info

https://pymotw.com/3/re/

http://evc-cit.info/comsc020/python-regex-tutorial/#

One minute Reflection

What did you find the most surprising?

What questions do you have?