Basic text processing

In [1]:
```
s = "hello world"
```

In [2]:
```
s.count("l")
```

Out[2]: 3

In [3]:
```
s.endswith("ld")
```

Out[3]: True

In [4]:
```
s.endswith("a")
```

Out[4]: False

In [5]:
```
"ell" in s
```

Out[5]: True

In [6]:
```
s.replace('o', '0')
```

Out[6]: 'hell0 w0rld'

In [7]:
```
s.split(' ')
```

Out[7]: ['hello', 'world']

In [8]:
```
"xx".join(["ABC","DEF"])
```

Out[8]: 'ABCxxDEF'

Complex motivational example

In [9]:
```
import re
s = "hello cool world seems"
```

In [10]:
```
# Find repeated characters that are one character from the end of the wo
rd
# "\2" denotes the second group repeated (i.e., the second parenthetical
part "(.)")
# "(?= )" is for ends with...though it's not part of the group
p = re.compile(r'((.)\2)(?=.\b)')
```

In [11]:
```
s1 = p.sub(lambda match : match.group(1).upper(), s)
print(s1)
```

```
heLLo cOOl world seems
```

Compile, match, search

In [12]:
```
p = re.compile("a|b|c")
```

In [13]:
```
p.match("a")
```

Out[13]:
```
<re.Match object; span=(0, 1), match='a'>
```

In [14]:
```
p.match("asdf")
```

Out[14]:
```
<re.Match object; span=(0, 1), match='a'>
```

In [15]:
```
p.search("sad")
```

Out[15]:
```
<re.Match object; span=(1, 2), match='a'>
```

In [16]:
```
p.search("dog")
```

In [17]:
```
p.search("abracadabra")
```

Out[17]:
```
<re.Match object; span=(0, 1), match='a'>
```

In [18]:
```
p = re.compile('(a|b|c)*')
```

In [19]:
```
p.search("abracadabra")
```

Out[19]:
```
<re.Match object; span=(0, 2), match='ab'>
```

In [20]:
```
p.search("dog")
```

Out[20]:
```
<re.Match object; span=(0, 0), match=''>
```

In [21]:
```
p = re.compile("ece (264|20875|368)")
```

In [22]:
```
p.match("ece 264")
```

Out[22]:
```
<re.Match object; span=(0, 7), match='ece 264'>
```

In [23]:
```
p.search("hello ece 368").group()
```

Out[23]:
```
'ece 368'
```

In [24]:
```
p = re.compile(r'hello (\w*)')
```

In [25]:
```
p.sub(r'goodbye \1', 'hello ece hello')
```

Out[25]:
```
'goodbye ece hello'
```

Literal strings

In [26]:
```
re.search(r'Back tail', 'Back tail').group()
```

Out[26]:
```
'Back tail'
```

In [27]:
```
try:
    re.search('Back\stail', 'Back\stail').group()
except AttributeError as e:
    print(e)
```

```
'NoneType' object has no attribute 'group'
```

In [28]:
```
re.search(r'Back\stail', 'Back tail').group()
```

Out[28]:
```
'Back tail'
```

In [29]:
```
x = 'hello \n world'
y = r'hello \n world'
```

In [30]:
```
print(x)
print(y)
```

```
hello
 world
hello \n world
```

Repetitions

In [31]:
```
re.search(r'Co+kie', 'Cooookie').group()
```

Out[31]:
```
'Cooookie'
```

In [32]:
```
re.search(r'elec*trica*l', 'eleccctricl').group()  #Checks for 0 or more
                                                   #'c' and 0 or more 'a'
```

Out[32]:
```
'eleccctricl'
```

In [33]:
```
re.search(r'Colou?r', 'Color').group()      #Checks for 0 or 1 'u'
```

Out[33]:
```
'Color'
```

In [34]:
```
re.search(r'\d{5,10}', '098765 4321').group()   #Checks for a digit
                                                #between 5 and 10 times
```

Out[34]:
```
'098765'
```

Groups

In [35]:
```
email_address = 'Please contact us at: support@datacamp.com'
match = re.search(r'([\w\.-]+)@([\w\.-]+)', email_address)

print(match.group()) # The whole matched text
print(match.group(1)) # The username (group 1)
print(match.group(2)) # The host (group 2)
```

```
support@datacamp.com
support
datacamp.com
```

In [36]:
```
p = re.compile('(a(b)c)d')   #Nested groups: Count opening parentheses
m = p.match('abcd')
print(m.group(0))
print(m.group(1))
print(m.group(2))
print(m.groups())
```

```
abcd
abc
b
('abc', 'b')
```

In [37]:
```
p = re.compile(r'\b(\w+)(\s\1)+\b')   #Using groups to detect repeated w
ords
p.search('Paris in the the the the spring').group()
```

Out[37]:
```
'the the the the'
```

Substitutions

In [38]:
```
searchstring = "<i>hello and goodbye</i> to <b>everyone</b>"
print(searchstring)
```

```
<i>hello and goodbye</i> to <b>everyone</b>
```

In [39]:
```
html = re.compile(r'<(i|b|strong|em)>(.*)?</\1>')
```

In [40]:
```
html.search(searchstring)
```

Out[40]:
```
<re.Match object; span=(0, 24), match='<i>hello and goodbye</i>'>
```

In [41]:
```
m = html.search(searchstring)
```

In [42]:
```
print(m.group(0))
```

```
<i>hello and goodbye</i>
```

In [43]:
```
print(m.group(1))
```

```
i
```

In [44]:
```
print(m.group(2))
```

```
hello and goodbye
```

In [45]:
```
html.sub(r'\2', searchstring)
```

Out[45]:
```
'hello and goodbye to everyone'
```

In [46]:
```
s = 'aaa@xxx.com bbb@yyy.net ccc@zzz.org'
```

In [47]:
```
y = re.compile('@[a-z]*.[a-z]*').sub('@abc.org', s)
```

In [48]:
```
print(y)
```

```
aaa@abc.org bbb@abc.org ccc@abc.org
```

File I/O

In [49]:
```
with open("test.txt", 'w') as f:
    f.write("my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")
```

In [50]:
```
f = open("test.txt",'r')
f.read(4)   # read the first 4 data
```

Out[50]:
```
'my f'
```

In [51]:
```
f.read(4)   # read the next 4 data
```

Out[51]:
```
'irst'
```

In [52]:
```
f.read()    # read in the rest until the end
```

Out[52]:
```
' file\nThis file\n\ncontains three lines\n'
```

In [53]:
```
f.close()
```

In [54]:
```
f = open("test.txt",'r')
```

In [55]:
```
f.readlines()
```

Out[55]:
```
['my first file\n', 'This file\n', '\n', 'contains three lines\n']
```

In [ ]:
```

```