

## 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 [26]: "xx".join(["ABC", "DEF"])
```

```
Out[26]: 'ABCxxDEF'
```

## Complex motivational example

```
In [3]: import re  
s = "hello cool world seems"
```

```
In [9]: p = re.compile(r'((.)\2)(?=.+\b)')
```

```
In [10]: s1 = p.sub(lambda match : match.group(1).upper(), s)  
print(s1)
```

```
heLLo cOOl world sEEms
```

## Compile, match, search

```
In [11]: p = re.compile("a|b|c")
```

```
In [12]: p.match("a")
```

```
Out[12]: <re.Match object; span=(0, 1), match='a'>
```

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

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

```
In [14]: p.search("sad")
```

```
Out[14]: <re.Match object; span=(1, 2), match='a'>
```

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

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

```
Out[16]: <re.Match object; span=(0, 1), match='a'>
```

```
In [17]: p = re.compile('(a|b|c)*')
```

```
In [18]: p.search("abracadabra")
```

```
Out[18]: <re.Match object; span=(0, 2), match='ab'>
```

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

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

```
In [20]: p = re.compile("ece (264|20875|368)")
```

```
In [21]: p.match("ece 264")
```

```
Out[21]: <re.Match object; span=(0, 7), match='ece 264'>
```

```
In [22]: p.search("hello ece 368").group()
```

```
Out[22]: 'ece 368'
```

```
In [23]: p = re.compile(r'hello (\w*')
```

```
In [24]: p.sub(r'goodbye \1', 'hello ece hello')
```

```
Out[24]: 'goodbye ece hello'
```

## Literal strings

```
In [27]: re.search(r'Back tail', 'Back tail').group()
```

```
Out[27]: 'Back tail'
```

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

```
-----
----
AttributeError                                Traceback (most recent call 1
ast)
<ipython-input-28-3330a5ad815e> in <module>
----> 1 re.search('Back\stail', 'Back\stail').group()

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

```
In [34]: re.search(r'Back\stail', 'Back tail').group()
```

```
Out[34]: 'Back tail'
```

```
In [37]: x = 'hello \n world'
         y = r'hello \n world'
```

```
In [39]: print(x)
         print(y)
```

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

## Repetitions

```
In [35]: re.search(r'Co+kie', 'Cooookie').group()
```

```
Out[35]: 'Cooookie'
```

```
In [40]: re.search(r'elec*trica*1', 'elecctricl').group()  #Checks for 0 or more
                                                         #'c' and 0 or more 'a'
```

```
Out[40]: 'elecctricl'
```

```
In [41]: re.search(r'Colou?r', 'Color').group()           #Checks for 0 or 1 'u'
```

```
Out[41]: 'Color'
```

```
In [42]: re.search(r'\d{5,10}', '098765 4321').group() #Checks for a digit
#between 5 and 10 times
```

```
Out[42]: '098765'
```

## Groups

```
In [44]: 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 [45]: 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 [52]: 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[52]: 'the the the the'
```

## Substitutions

```
In [1]: searchstring = "<i>hello and goodbye</i> to <b>everyone</b>"
print(searchstring)
```

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

```
In [4]: html = re.compile(r'<(i|b|strong|em)>(.*?)</\1>')
```

```
In [5]: html.search(searchstring)
```

```
Out[5]: <re.Match object; span=(0, 24), match='<i>hello and goodbye</i>'>
```

```
In [6]: m = html.search(searchstring)
```

```
In [7]: print(m.group(0))
```

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

```
In [8]: print(m.group(1))
```

```
i
```

```
In [9]: print(m.group(2))
```

```
hello and goodbye
```

```
In [10]: html.sub(r'\2', searchstring)
```

```
Out[10]: 'hello and goodbye to everyone'
```

```
In [16]: s = 'aaa@xxx.com bbb@yyy.net ccc@zzz.org'
```

```
In [17]: y = re.compile('[a-z]*.[a-z]*').sub('@abc.org', s)
```

```
In [18]: print(y)
```

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

## File I/O

```
In [19]: 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 [20]: f = open("test.txt", 'r')  
         f.read(4)    # read the first 4 data
```

```
Out[20]: 'my f'
```

```
In [21]: f.read(4)    # read the next 4 data
```

```
Out[21]: 'irst'
```

```
In [22]: f.read()    # read in the rest until the end
```

```
Out[22]: ' file\nThis file\n\ncontains three lines\n'
```

```
In [23]: f.close()
```

```
In [24]: f = open("test.txt", 'r')
```

```
In [25]: f.readlines()
```

```
Out[25]: ['my first file\n', 'This file\n', '\n', 'contains three lines\n']
```

```
In [ ]:
```