In [1]:
```
import re

# List of patterns to search for
patterns = [ 'term1', 'term2' ]

# Text to parse
text = 'This is a string with term1, but it does not have the other term.'

for pattern in patterns:
    print 'Searching for "%s" in: \n"%s"' % (pattern, text),

    #Check for match
    if re.search(pattern,  text):
        print '\n'
        print 'Match was found. \n'
    else:
        print '\n'
        print 'No Match was found.\n'
```

```
Searching for "term1" in:
"This is a string with term1, but it does not have the other term."

Match was found.

Searching for "term2" in:
"This is a string with term1, but it does not have the other term."

No Match was found.
```

In [2]:
```
pattern = 'term1'

# Text to parse
text = 'This is a string with term1, but it does not have the other term.'

match = re.search(pattern,  text)

type(match)
```

Out[2]:
```
_sre.SRE_Match
```

In [3]:
```
# Show start of match
match.start()
```

Out[3]:
```
22
```

In [4]:
```
# Show start of match
match.start()
```

Out[4]:
```
27
```

```
In [5]: # Term to split on
        split_term = '@'

        phrase = 'What is the domain name of someone with the email: hello@gmail.com'

        # Split the phrase
        re.split(split_term,phrase)

Out[5]: ['What is the domain name of someone with the email: hello', 'gmail.com']
```

```
In [6]: # Term to split on
        split_term = ''

        phrase = 'What is the domain name of someone with the email: hello@gmail.com'

        # Split the phrase
        re.split(split_term,phrase)

Out[6]: ['What is the domain name of someone with the email: hello@gmail.com']
```

```
In [7]: # Returns a list of all matches
        re.findall('match','test phrase match is in middle')

Out[7]: ['match']
```

In [11]:
```
def multi_re_find(patterns,phrase):
    '''
    Takes in a list of regex patterns
    Prints a list of all matches
    '''
    for pattern in patterns:
        print 'Searching the phrase using the re check: %r' %pattern
        print re.findall(pattern,phrase)
        print '\n'
test_phrase = 'sdsd..sssddd...sdddsddd...dsds...dsssss...sdddd'

test_patterns = [ 'sd*',     # s followed by zero or more d's
                'sd+',          # s followed by one or more d's
                'sd?',          # s followed by zero or one d's
                'sd{3}',        # s followed by three d's
                'sd{2,3}',      # s followed by two to three d's
                ]

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: 'sd*'
['sd', 'sd', 's', 's', 'sddd', 'sddd', 'sddd', 'sd', 's', 's', 's', 's', 's',
's', 'sdddd']


Searching the phrase using the re check: 'sd+'
['sd', 'sd', 'sddd', 'sddd', 'sddd', 'sd', 'sdddd']


Searching the phrase using the re check: 'sd?'
['sd', 'sd', 's', 's', 'sd', 'sd', 'sd', 'sd', 's', 's', 's', 's', 's', 's', 's
d']


Searching the phrase using the re check: 'sd{3}'
['sddd', 'sddd', 'sddd', 'sddd']


Searching the phrase using the re check: 'sd{2,3}'
['sddd', 'sddd', 'sddd', 'sddd']
```

```
In [12]: test_phrase = 'sdsd..sssddd...sdddsddd...dsds...dsssss...sdddd'

         test_patterns = [ '[sd]',    # either s or d
                     's[sd]+']   # s followed by one or more s or d


         multi_re_find(test_patterns,test_phrase)

         Searching the phrase using the re check: '[sd]'
         ['s', 'd', 's', 'd', 's', 's', 's', 'd', 'd', 'd', 's', 'd', 'd', 'd', 's',
         'd', 'd', 'd', 'd', 's', 'd', 's', 'd', 's', 's', 's', 's', 's', 's', 'd', 'd',
         'd', 'd']


         Searching the phrase using the re check: 's[sd]+'
         ['sdsd', 'sssddd', 'sdddsddd', 'sds', 'sssss', 'sdddd']
```

```
In [13]: test_phrase = 'This is a string! But it has punctuation. How can we remove it?'
         re.findall('[^!.? ]+',test_phrase)

Out[13]: ['This',
          'is',
          'a',
          'string',
          'But',
          'it',
          'has',
          'punctuation',
          'How',
          'can',
          'we',
          'remove',
          'it']
```

In [14]:
```
test_phrase = 'This is an example sentence. Lets see if we can find some letters.

test_patterns=[ '[a-z]+',      # sequences of lower case letters
                '[A-Z]+',      # sequences of upper case letters
                '[a-zA-Z]+',   # sequences of lower or upper case letters
                '[A-Z][a-z]+'] # one upper case letter followed by lower case let

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: '[a-z]+'
['his', 'is', 'an', 'example', 'sentence', 'ets', 'see', 'if', 'we', 'can', 'fi
nd', 'some', 'letters']


Searching the phrase using the re check: '[A-Z]+'
['T', 'L']


Searching the phrase using the re check: '[a-zA-Z]+'
['This', 'is', 'an', 'example', 'sentence', 'Lets', 'see', 'if', 'we', 'can',
'find', 'some', 'letters']


Searching the phrase using the re check: '[A-Z][a-z]+'
['This', 'Lets']
```

In [15]:
```
test_phrase = 'This is a string with some numbers 1233 and a symbol #hashtag'

test_patterns=[ r'\d+', # sequence of digits
                r'\D+', # sequence of non-digits
                r'\s+', # sequence of whitespace
                r'\S+', # sequence of non-whitespace
                r'\w+', # alphanumeric characters
                r'\W+', # non-alphanumeric
                ]

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: '\\d+'
['1233']


Searching the phrase using the re check: '\\D+'
['This is a string with some numbers ', ' and a symbol #hashtag']


Searching the phrase using the re check: '\\s+'
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']


Searching the phrase using the re check: '\\S+'
['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a', 's
ymbol', '#hashtag']


Searching the phrase using the re check: '\\w+'
['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a', 's
ymbol', 'hashtag']


Searching the phrase using the re check: '\\W+'
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' #']
```

In [ ]: