Regular Expressions -- Practice Code

In [58]:
```
import re
```

re.seacrh()

In [59]:
```
text1 = "This is a beautiful day"
```

In [60]:
```
re.search(r'is', text1)
```

Out[60]:
```
<_sre.SRE_Match object; span=(2, 4), match='is'>
```

In [61]:
```
m = re.search(r'is', text1)
```

In [62]:
```
m.group()
```

Out[62]:
```
'is'
```

In [63]:
```
m.start(), m.end()
```

Out[63]:
```
(2, 4)
```

In [64]:
```
m.span()
```

Out[64]:
```
(2, 4)
```

re.match()

In [65]:
```
m = re.match(r'is', text1)
```

In [66]:
```
print(m)
```

```
None
```

In [67]:
```
text1
```

Out[67]:
```
'This is a beautiful day'
```

In [68]:
```
m = re.match(r'Th', text1)
```

In [69]:
```
print(m)
```

```
<_sre.SRE_Match object; span=(0, 2), match='Th'>
```

In [70]:
```
m.group(), m.start(), m.end(), m.span()
```

Out[70]:
```
('Th', 0, 2, (0, 2))
```

re.findall()

In [71]:
```
text1
```

Out[71]:
```
'This is a beautiful day'
```

In [72]:
```
re.findall(r'is', text1)
```

Out[72]:
```
['is', 'is']
```

In [73]:
```
text2 = "abbbaaabbbbabababa"
```

In [74]:
```
re.findall(r'ba', text2)
```

Out[74]:
```
['ba', 'ba', 'ba', 'ba', 'ba']
```

re.finditer()

In [75]:
```
mat = re.finditer(r'ba', text2)
```

In [76]:
```
type(mat)
```

Out[76]:
```
callable_iterator
```

In [77]:
```
for m in mat:
    print(m.group(), m.start(), m.end())
```

```
ba 3 5
ba 10 12
ba 12 14
ba 14 16
ba 16 18
```

In [78]:
```
text2
```

Out[78]:
```
'abbbaaabbbbabababa'
```

```
In [79]: re.sub(r'ba', 'xy', text2)
Out[79]: 'abbxyaabbbxyxyxyxy'

In [81]: re.sub(r'ba', 'xy', text2, count=2)
Out[81]: 'abbxyaabbbxybababa'
```

re.compile()

```
In [82]: pat = re.compile(r'ba')

In [83]: re.findall(pat, text2)
Out[83]: ['ba', 'ba', 'ba', 'ba', 'ba']
```

re.split()

```
In [97]: text3 = "akaks ksdkdkd; aksakks: ajsjss, shshs; ususu;     hshs"

In [98]: text3.split()
Out[98]: ['akaks', 'ksdkdkd;', 'aksakks:', 'ajsjss,', 'shshs;', 'ususu;',
         'hshs']

In [99]: re.split(r'[ ;:,]\s*', text3)
Out[99]: ['akaks', 'ksdkdkd', 'aksakks', 'ajsjss', 'shshs', 'ususu', 'hsh
         s']
```

Writing REs: Repetition Coding Examples

```
In [100]: text = "ab abb a a a abbbb abbbbbbb"

In [101]: re.findall(r'ab*', text)
Out[101]: ['ab', 'abb', 'a', 'a', 'a', 'abbbb', 'abbbbbbb']

In [102]: re.findall(r'ab+', text)
Out[102]: ['ab', 'abb', 'abbbb', 'abbbbbbb']
```

```
In [105]: text
Out[105]: 'ab abb a a a abbbb abbbbbbb'

In [106]: re.findall(r'ab?', text)
Out[106]: ['ab', 'ab', 'a', 'a', 'a', 'ab', 'ab']

In [107]: re.findall(r'ab{2}', text)
Out[107]: ['abb', 'abb', 'abb']

In [108]: text
Out[108]: 'ab abb a a a abbbb abbbbbbb'

In [109]: re.findall(r'ab{3,5}', text)
Out[109]: ['abbbb', 'abbbbb']

In [110]: text
Out[110]: 'ab abb a a a abbbb abbbbbbb'

In [111]: re.findall(r'ab*', text)
Out[111]: ['ab', 'abb', 'a', 'a', 'a', 'abbbb', 'abbbbbbb']

In [112]: re.findall(r'ab*?', text)
Out[112]: ['a', 'a', 'a', 'a', 'a', 'a', 'a']

In [113]: re.findall(r'ab+', text)
Out[113]: ['ab', 'abb', 'abbbb', 'abbbbbbb']

In [114]: re.findall(r'ab+?', text)
Out[114]: ['ab', 'ab', 'ab', 'ab']

In [115]: text
Out[115]: 'ab abb a a a abbbb abbbbbbb'

In [116]: re.findall(r'ab?', text)
Out[116]: ['ab', 'ab', 'a', 'a', 'a', 'ab', 'ab']

In [117]: re.findall(r'ab??', text)
Out[117]: ['a', 'a', 'a', 'a', 'a', 'a', 'a']
```

Writing REs: Character Sets and Ranges

```
In [118]: text = "xyyxxyyyzzzx"
```

```
In [123]: re.findall(r'[xy]', text)
```

```
Out[123]: ['x', 'y', 'y', 'x', 'x', 'y', 'y', 'y', 'x']
```

```
In [124]: re.findall(r'x[xy]', text)
```

```
Out[124]: ['xy', 'xx']
```

```
In [125]: re.findall(r'x[xy]+', text)
```

```
Out[125]: ['xyyxxyyy']
```

```
In [126]: re.findall(r'x[xy]+?', text)
```

```
Out[126]: ['xy', 'xx']
```

```
In [127]: text = "xxy xyxyx xaxb xxyy aaxz"
```

```
In [128]: re.findall(r'x[^xy]+', text)
```

```
Out[128]: ['x ', 'xa', 'xb ', 'xz']
```

```
In [129]: text
```

```
Out[129]: 'xxy xyxyx xaxb xxyy aaxz'
```

```
In [130]: re.findall(r'x[^xy]+?', text)
```

```
Out[130]: ['x ', 'xa', 'xb', 'xz']
```

```
In [131]: text = "This a sample text. -- with some Punctuation marks!!!"
```

```
In [133]: re.findall(r'[A-Z][a-z]*', text)
```

```
Out[133]: ['This', 'Punctuation']
```

```
In [138]: re.findall(r'[^.\-! ]+', text)
```

```
Out[138]: ['This', 'a', 'sample', 'text', 'with', 'some', 'Punctuation', 'ma
          rks']
```

Writing REs: Escape Codes

In [140]:
```
text = "The cost of Python course is $125."
```

In [141]:
```
re.findall(r'\d', text)
```

Out[141]:
```
['1', '2', '5']
```

In [142]:
```
re.findall(r'\d+', text)
```

Out[142]:
```
['125']
```

In [144]:
```
re.findall(r'\D+', text)
```

Out[144]:
```
['The cost of Python course is $', '.']
```

In [145]:
```
re.findall(r'\s', text)
```

Out[145]:
```
[' ', ' ', ' ', ' ', ' ', ' ']
```

In [147]:
```
re.findall(r'\S+', text)
```

Out[147]:
```
['The', 'cost', 'of', 'Python', 'course', 'is', '$125.']
```

In [148]:
```
text
```

Out[148]:
```
'The cost of Python course is $125.'
```

In [151]:
```
re.findall(r'\w+', text)
```

Out[151]:
```
['The', 'cost', 'of', 'Python', 'course', 'is', '125']
```

In [152]:
```
text
```

Out[152]:
```
'The cost of Python course is $125.'
```

In [153]:
```
re.findall(r'\W+', text)
```

Out[153]:
```
[' ', ' ', ' ', ' ', ' ', ' $', '.']
```

Writing REs: Anchoring

In [154]:
```
text = "This is a beautiful day."
```

In [155]:
```
re.findall(r'is', text)
```

Out[155]:
```
['is', 'is']
```

```
In [156]: re.findall(r'^is', text)
Out[156]: []

In [157]: re.findall(r'^T', text)
Out[157]: ['T']

In [162]: re.findall(r'\.$', text)
Out[162]: ['.']

In [163]: text
Out[163]: 'This is a beautiful day.'

In [164]: re.findall(r'is', text)
Out[164]: ['is', 'is']

In [165]: re.findall(r'\bis\b', text)
Out[165]: ['is']

In [166]: re.search(r'\bis\b', text)
Out[166]: <_sre.SRE_Match object; span=(5, 7), match='is'>
```

Writing REs: Flags

```
In [174]: text = "Python python PYTHON"

In [175]: re.findall(r'Python', text)
Out[175]: ['Python']

In [176]: re.findall(r'Python', text, re.IGNORECASE)
Out[176]: ['Python', 'python', 'PYTHON']

In [177]: re.findall(r'Python', text, re.I )
Out[177]: ['Python', 'python', 'PYTHON']

In [178]: re.findall(r'Python', text, 2)
Out[178]: ['Python', 'python', 'PYTHON']
```

In [179]:
```
re.I
```

Out[179]: 2

In [180]:
```
re.S
```

Out[180]: 16

In [181]:
```
text = "Py\nthon"
re.findall(r'.+', text)
```

Out[181]: ['Py', 'thon']

In [183]:
```
re.findall(r'.+', text, re.DOTALL )
```

Out[183]: ['Py\nthon']

In [184]:
```
text = "Python is fun. Learning python."
```

In [185]:
```
re.sub(r'Py', 'My', text )
```

Out[185]: 'Mython is fun. Learning python.'

In [187]:
```
re.sub(r'Py', 'My', text, flags=re.I )
```

Out[187]: 'Mython is fun. Learning Mython.'

In [188]:
```
re.sub(r'Py', 'My', text, count=1, flags=re.I )
```

Out[188]: 'Mython is fun. Learning python.'

Writing REs: Grouping and Named groups.

In [189]:
```
text = '123-4567 is my telephone.'
```

In [190]:
```
re.findall(r'[\d]{3}-[\d]{4}', text)
```

Out[190]: ['123-4567']

In [191]:
```
m = re.search(r'([\d]{3})-([\d]{4})', text)
```

In [192]:
```
print(m)
```

```
<_sre.SRE_Match object; span=(0, 8), match='123-4567'>
```

In [193]:
```
m.group()
```

Out[193]: '123-4567'

In [194]:
```
m.groups()
```

Out[194]:
```
('123', '4567')
```

In [195]:
```
m.group(1)
```

Out[195]:
```
'123'
```

In [196]:
```
m.group(2)
```

Out[196]:
```
'4567'
```

In [197]:
```
m = re.search(r'(?P<first3>[\d]{3})-(?P<last4>[\d]{4})', text)
```

In [198]:
```
m.group('first3')
```

Out[198]:
```
'123'
```

In [199]:
```
m.group('last4')
```

Out[199]:
```
'4567'
```

Writing REs: A practical example -- Step by step

In [200]:
```
text = [ '123 456 7890', '(123) 456 7890']
```

In [204]:
```
pat = r'\(?\d{3}\)?\s\d{3}\s\d{4}'
```

In [205]:
```
for d in text:
    m = re.search(pat, d)
    if m:
        print(m.group())
```

```
123 456 7890
(123) 456 7890
```

In [217]:
```
text = [ '123 456 7890', '(123) 456 7890', '123-456-7890', '123.45
6.7890']
pat = r'\(?\d{3}\)?[\s\-\.]\d{3}[\s\-\.]\d{4}'
```

In [218]:
```
for d in text:
    m = re.search(pat, d)
    if m:
        print(m.group())
```

```
123 456 7890
(123) 456 7890
123-456-7890
123.456.7890
```

In [221]:
```
text = [ '123 456 7890', '(123) 456 7890', '123-456-7890', '123.45
6.7890', '1234567890']
pat = r'\(?\d{3}\)?[\s\-\.]?\d{3}[\s\-\.]?\d{4}'
```

In [222]:
```
for d in text:
    m = re.search(pat, d)
    if m:
        print(m.group())
```

```
123 456 7890
(123) 456 7890
123-456-7890
123.456.7890
1234567890
```

In [232]:
```
text = [ '1 123 456 7890', '+1 (123) 456 7890', '123-456-7890', '12
3.456.7890', '1234567890']
pat = r'\+?\d?\s?\(?\d{3}\)?[\s\-\.]?\d{3}[\s\-\.]?\d{4}'
patc = re.compile(pat)
```

In [233]:
```
for d in text:
    m = re.search(patc, d)
    if m:
        print(m.group())
```

```
1 123 456 7890
+1 (123) 456 7890
123-456-7890
123.456.7890
1234567890
```

Grouping

In [237]:
```
text = ['1 123 456 7890', '+1 (123) 456 7890', '123-456-7890', '12
3.456.7890', '1234567890']
pat = r'(\+?\d?)\s?(\(?\d{3}\)?)[\s\-\.]?(\d{3})[\s\-\.]?(\d{4})'
patc = re.compile(pat)

for dt in text:
    m = re.search(patc, dt)
    if m:
        print(m.group(), "\t", m.group(1), "\t", m.group(2), "\t",
m.group(3), "\t", m.group(4))
```

```
1 123 456 7890   1       123     456     7890
+1 (123) 456 7890        +1      (123)   456     7890
123-456-7890             123     456     7890
123.456.7890             123     456     7890
1234567890               123     456     7890
```

Naming Groups

In [241]:
```
text = ['1 123 456 7890', '+1 (123) 456 7890', '123-456-7890', '12
3.456.7890', '1234567890']
pat = r'(?P<add1>\+?\d?)\s?(?P<area>\(?\d{3}\)?)[\s\-\.]?(?P<first3
>\d{3})[\s\-\.]?(?P<last4>\d{4})'
patc = re.compile(pat)

for dt in text:
    m = re.search(patc, dt)
    if m:
        print(m.group(), "\t", m.group('add1'),"\t", m.group('are
a'),"\t",
              m.group('first3'),"\t", m.group('last4') )
```

```
1 123 456 7890   1       123     456     7890
+1 (123) 456 7890        +1      (123)   456     7890
123-456-7890             123     456     7890
123.456.7890             123     456     7890
1234567890               123     456     7890
```

In [ ]:
```
```