RegularExpressions

February 1, 2024

1 Regular Expressions examples

[ ]:
```
import re

p = re.compile('[Pp]umas?|[Cc]ougars?')
p.findall('I saw a puma chasing two cougars.')
```

[ ]:
```
['puma', 'cougars']
```

[58]:
```
text = 'I saw a puma puma puma puma in the jungle.'
p = re.compile('(puma )+')
m = p.search(text)
print(m)
```

```
<re.Match object; span=(8, 28), match='puma puma puma puma '>
```

[ ]:
```
p = re.compile('[Ww]oodchuck')
m = p.match('Woodchucks ran after a woodchuck.')
```

[ ]:
```
m
```

[ ]:
```
<re.Match object; span=(0, 9), match='Woodchuck'>
```

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

[ ]:
```
(0, 9)
```

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

[ ]:
```
'Woodchuck'
```

[ ]:
```
len('Woodchuck'), 'Woodchuck ran ...'[8]
```

[ ]:
```
(9, 'k')
```

[ ]:
```
m.span(), m.start()
```

[ ]:
```
((0, 9), 0)
```

[ ]:
```
m = p.match('Three Woodchucks ran after a woodchuck.')
print(m)
```

```
None
```

[ ]:
```
m = p.search('Three Woodchucks ran after a woodchuck.')
m.group(), m.span(), 'Three Woodchucks'.find('Woodchuck')
```

[ ]:
```
('Woodchuck', (6, 15), 6)
```

[ ]:
```
matches = p.findall('Three Woodchucks ran after a woodchuck.')
matches
```

[ ]:
```
['Woodchuck', 'woodchuck']
```

[ ]:
```
matches = p.finditer('Three Woodchucks ran after a woodchuck.')
for m in matches:
    print(m.span())
```

```
(6, 15)
(29, 38)
```

[ ]:
```
p = re.compile('[Ww]oodchuck|[Gg]roundhog')
matches = p.findall('The woodchuck appears at the beginning in the movie␣
 ↪Groundhog Day')
matches
```

[ ]:
```
['woodchuck', 'Groundhog']
```

[ ]:
```
pd = re.compile(r'\d+')
matches = pd.findall("His GPA is 3.85. His age is 23, and he can swim 4000␣
 ↪yards without stopping")
print(matches)

pd = re.compile(r'[0-9]+')
matches = pd.findall("His GPA is 3.85. His age is 23, and he can swim 4000␣
 ↪yards without stopping")
print(matches)

pd = re.compile(r'[\d.]+')
matches = pd.findall("His GPA is 3.85. His age is 23, and he can swim 4000␣
 ↪yards without stopping")
print(matches)

pd = re.compile(r'[\d]+ [.]? \d+', re.VERBOSE)
matches = pd.findall("His GPA is 3.85. His age is 23, and he " \
                     "can swim 4000 yards without stopping." \
                     "How about 3.85.4?")
```

```
print(matches)
```

```
['3', '85', '23', '4000']
['3', '85', '23', '4000']
['3.85.', '23', '4000']
['3.85', '23', '4000', '3.85']
```

[ ]:
```
import re
p = re.compile('[Ww]oodchuck | [Gg]roundhog')
matches = p.findall('The woodchucks appears at the beginning in the movie␣
 ↪Groundhog Day')
matches
```

[ ]:
```
[' Groundhog']
```

[ ]:
```
p = re.compile('[Ww]oodchuck | [Gg]roundhog', re.VERBOSE)
matches = p.findall('The woodchucks appears at the beginning in the movie␣
 ↪Groundhog Day')
matches
```

[ ]:
```
['woodchuck', 'Groundhog']
```

[ ]:
```
p = re.compile(r'[Ww]oodchuck\ | [Gg]roundhog', re.VERBOSE)
matches = p.findall('The woodchuck appears at the beginning in the movie␣
 ↪Groundhog Day')
matches
```

[ ]:
```
['woodchuck ', 'Groundhog']
```

[ ]:
```
p = re.compile('[Ww]oodchucks?|[Gg]roundhogs?')
p.findall('Woodchucks, by any other name, such as groundhog, '
          'would woodchuck the same.')
```

[ ]:
```
['Woodchucks', 'groundhog', 'woodchuck']
```

[ ]:
```
p = re.compile('^[Hh]ow')
p.findall('How do you do? I do how I always do.')
```

[ ]:
```
['How']
```

[ ]:
```
p = re.compile('[Hh]ow')
p.findall('How do you do? I do how I always do.')
```

[ ]:
```
['How', 'how']
```

[ ]:
```
#p = re.compile('[^a-zA-Z][tT]he[^a-zA-Z]')
p = re.compile('[tT]he')
p.findall('The cat ran after the dog, but the other dog intervened.')
```

[ ]: ['The', 'the', 'the', 'the']

[ ]:
```
p = re.compile('[tT]he')
matches = p.finditer('The cat ran after the dog, '
                     'but the other dog intervened.')
for m in matches:
    print(m)

print()

matches = p.finditer('The cat ran after the dog, '
                     'but the other dog intervened.')
for m in matches:
    print(m.group(), m.start(), m.end())
```

```
<re.Match object; span=(0, 3), match='The'>
<re.Match object; span=(18, 21), match='the'>
<re.Match object; span=(31, 34), match='the'>
<re.Match object; span=(36, 39), match='the'>

The 0 3
the 18 21
the 31 34
the 36 39
```

[ ]:
```
p = re.compile('[^a-zA-Z][tT]he[^a-zA-Z]')
#p = re.compile('[tT]he')
p.findall('The cat ran after the dog, '
          'but the other dog intervened.')
```

[ ]: [' the ', ' the ']

[ ]:
```
s = 'The cat ran after the dog, but the other dog intervened.'

p1 = re.compile('[^a-zA-Z] ([tT]he) [^a-zA-Z]', re.VERBOSE)
r1 = p1.findall(s)
print(r1)

p2 = re.compile('^([tT]he) [^a-zA-Z]', re.VERBOSE)
r2 = p2.findall(s)
print(r2)

# Instead of trying to combine the two patterns (but try it as a homework␣
 ↪exercise).
r3 = p1.findall(' ' + s)
print(r3)
```

```
['the', 'the']
```

```
['The']
['The', 'the', 'the']
```

[ ]:
```
p = re.compile('a+b+')
p.findall('aabb aaabbb abcba aba aaaabb')
```

[ ]:
```
['aabb', 'aaabbb', 'ab', 'ab', 'aaaabb']
```

[ ]:
```
import re

p = re.compile(r'[pP]ythons?')
matches = p.findall('Python is a fun programming language. '
                    'There are many pythons in the jungle. '
                    'I like PYTHON!')
print(matches)
```

```
['Python', 'pythons']
```

[ ]:
```
p = re.compile(r'\s(cats?|dogs?)\W')
matches = p.findall('It is raining cats and dogs. '
                    'Her cat likes catfish.')
print(matches)
```

```
['cats', 'dogs', 'cat']
```

[ ]:
```
p = re.compile('colou?r')
p.sub('<color>', 'I would like to drive a blue coloured car.')
```

[ ]:
```
'I would like to drive a blue <color>ed car.'
```

1.1 Character classes \d, \D, ...

[ ]:
```
import re

text = 'I woke up at 8am this morning.'
p = re.compile('\D+')
p.findall(text)
```

[ ]:
```
['I woke up at ', 'am this morning.']
```

[ ]:
```
p = re.compile('[^0-9]+')
p.findall(text)
```

[ ]:
```
['I woke up at ', 'am this morning.']
```

Regular expression for recognizing time expressions, e.g. `8am`, `12:05pm`, ...

[ ]:
```
import re

p = re.compile('[0-9]+(:[0-9]+)?[ap]m')
```

```
text = 'I woke up at 8am and had lunch at 12:35pm, then went for a walk.'
m1 = p.search(text)
print(m1)
print(m1.group()) # this prints the matched string
print(m1.start()) # this prints the starting position
print(m1.end()) # this prints the end position
print(m1.span()) # this prints the (start, end) tuple
```

```
<re.Match object; span=(13, 16), match='8am'>
8am
13
16
(13, 16)
```

[ ]:
```
m2 = p.search(text[m1.end():])
print(m2)
```

```
<re.Match object; span=(18, 25), match='12:35pm'>
```

[ ]:
```
import re

p = re.compile('[0-9]+(:[0-9]+)?[ap]m')
text = 'I woke up at 8am and had lunch at 12:35pm, then went for a walk.'

# Find and print all matches.
m = p.search(text)
while m:
  print(m.group())
  text = text[m.end():]
  m = p.search(text)
```

```
8am
12:35pm
```

`Pattern.search()` has a keyword argument `pos` to specify where to start the search, by default 0.

[ ]:
```
text = 'I woke up at 8am and had lunch at 12:35pm, then went for a walk.'
p.search(text, pos = 16)
```

[ ]:
```
<re.Match object; span=(34, 41), match='12:35pm'>
```

[61]:
```
import re

p = re.compile('[0-9]+(:[0-9]+)?[ap]m')
text = 'I woke up at 8am and had lunch at 12:35pm, then went for a walk.'
# Find and print all matches.
m = p.search(text)
while m:
  print(m.group())
```

```
  m = p.search(text, pos = m.end())
```

```
8am
12:35pm
```

Use `re.VERBOSE` to indicate that spaces in the regular expression string are to be ignored.

[62]:
```
import re

p = re.compile('[0-9]+ (:[0-9]+)? [ap]m', re.VERBOSE)
text = 'I woke up at 8am and had lunch at 12:15pm, then went for a walk.'
m = p.search(text)
while m:
  print(m.group())
  m = p.search(text, pos = m.end())
```

```
8am
12:15pm
```

Let's make the regular expression more precise.

[72]:
```
p = re.compile(r'(?<=\D) (0?[0-9] | 1[012]) (:[0-5][0-9])? [ap]m', re.VERBOSE)
text = 'I woke up at 8am and had lunch at 12:15pm, then went for a walk. 34:
 ↪49am is not a valid time expression.'
m = p.search(text)
while m:
  print(m.group())
  m = p.search(text, pos = m.end())
```

```
8am
12:15pm
```

1.2 Use parantheses for *capturing* behavior

[ ]:
```
p = re.compile('[^a-zA-Z] [Tt]he [^a-zA-Z]', re.VERBOSE)
m = p.findall('Yes. The cat chases the dogs that bathe.')
print(m)
```

```
[' The ', ' the ']
```

[ ]:
```
p = re.compile('[^a-zA-Z] ([Tt]he) [^a-zA-Z]', re.VERBOSE)
m = p.findall('Yes. The cat chases the dogs that bathe.')
print(m)
```

```
[' The ', ' the ']
```

[ ]:
```
p = re.compile('( [0-9]+ )', re.VERBOSE)
p.sub(r'<\1> extra', 'the 35 boxes')
```

[ ]:
```
'the <35> extra boxes'
```

[ ]:
```
p = re.compile('( [0-9]+ )', re.VERBOSE)
p.sub(r'<\1> extra', '10 whiseky bottles and 35 boxes of gold')
```

[ ]:
```
'<10> extra whiseky bottles and <35> extra boxes of gold'
```

1.3 Use (?!) to indicate non-matching behavior.

[ ]:
```
p = re.compile(r'Isaac (?!Asimov)')
matches = p.finditer('I like reading Isaac Asimov '
                     'and listening to Isaac Perlman '
                     'and playing chess with Isaac .')
for m in matches:
    print(m.span(), m.group())
```

```
(45, 51) Isaac
(82, 88) Isaac
```

[ ]:
```
p = re.compile(r'Isaac (?!Asimov|Perlman)')
matches = p.finditer('I like reading Isaac Asimov '
                     'and listening to Isaac Perlman '
                     'and playing chess with Isaac .')
for m in matches:
    print(m.span(), m.group())
```

```
(82, 88) Isaac
```

1.4 Use (?:) to indicate parantheses are used for *grouping*, but not capturing behavior

[ ]:
```
import re

p = re.compile('[0-9]+ (?: :[0-9]+)? [ap]m', re.VERBOSE)
text = 'I woke up at 8am and had lunch at 12:35pm, then went for a walk.'
m = p.findall(text)
print(m)
```

```
['8am', '12:35pm']
```

1.5 Find-replace using regular expressions and p.sub()

[ ]:
```
import re

p = re.compile('\d+')
text = 'She ran for 3 miles, than she ate 2 apples and drank a 12 ounce can of␣
 ↪Coke.'
p.sub('<num>', text)
```

[ ]:
```
'She ran for <num> miles, than she ate <num> apples and drank a <num> ounce can
of Coke.'
```

Capture groups using parantheses and numbered registers.

[ ]:
```
import re

p = re.compile('(\d+)')
text = 'I ran for 3 miles, than I ate 2 apples and drank a 12 ounce can of Coke.
 ↪'
p.sub(r'\1 extra', text)
```

[ ]:
```
'I ran for 3 extra miles, than I ate 2 extra apples and drank a 12 extra ounce
can of Coke.'
```

[60]:
```
import re

p = re.compile(".*I am (depressed|sad).*")
text = "My cat is sick, I am sad, I don't know what to do!"
p.sub(r'I am sorry to hear you are \1.', text)
```

[60]:
```
'I am sorry to hear you are sad.'
```

[ ]:
```

```