UE18CS306B
Python Application Programming

Question Answer

1) Why compile method is used in regular expressions? Explain it with an example?

re includes module-level functions for working with regular expressions as text strings, but it is more efficient to compile the expressions a program uses frequently. The compile() function converts an expression string into a RegexObject .

```
import re
# Precompile the patterns
regexes = [ re.compile(p)
for p in [ ’this’, ’that’ ]
]
text = ’Does this text match the pattern?’
print ’Text: %r\n’ % text
for regex in regexes:
print ’Seeking "%s" ->’ % regex.pattern,
if regex.search(text):
print ’match!’
else:
print ’no match’
```

The module-level functions maintain a cache of compiled expressions. However, the size of the cache is limited, and using compiled expressions directly avoids the cache lookup overhead. Another advantage of using compiled expressions is that by precompiling all expressions when the module is loaded, the compilation work is shifted to application start time, instead of to a point when the program may be responding to a user action.

```
$ python re_simple_compiled.py
Text: ’Does this text match the pattern?’
Seeking "this" -> match!
Seeking "that" -> no match
```

2) Differentiate between findall and search with an example?

search() is used for looking single instances of literal text strings. The findall() function returns all substrings of the input that match the pattern without overlapping.

```
import re
text = ’abbaaabbbbaaaaa’
pattern = ’ab’
for match in re.findall(pattern, text):
print ’Found "%s"’ % match
```

There are two instances of ab in the input string.

```
$ python re_findall.py
Found "ab"
Found "ab"
```

finditer() returns an iterator that produces Match instances instead of the
strings returned by findall() .

```
import re
text = ’abbaaabbbbaaaaa’
pattern = ’ab’
for match in re.finditer(pattern, text):
s = match.start()
e = match.end()
print ’Found "%s" at %d:%d’ % (text[s:e], s, e)
```

3) Discuss in detail about all the symbols used for repetition?

There are five ways to express repetition in a pattern. A pattern followed by the
metacharacter * is repeated zero or more times. (Allowing a pattern to repeat zero
times means it does not need to appear at all to match.) Replace the * with + and the
pattern must appear at least once. Using ? means the pattern appears zero times or one
time. For a specific number of occurrences, use {m} after the pattern, where m is the number of times the pattern should repeat. And, finally, to allow a variable but limited
number of repetitions, use {m,n} where m is the minimum number of repetitions and n
is the maximum. Leaving out n ({m,}) means the value appears at least m times, with
no maximum.

4) Mention breifly what the escape codes \d \D \s \S \w \W and anchoring in python regular expressions?

Ans)

Code

\d
\D
\s
\S
\w
\W

A digit
A nondigit
Whitespace (tab, space, newline, etc.)
Nonwhitespace
Alphanumeric
Nonalphanumeric

Code
^

$
\A
\Z
\b
\B

Meaning
Start of string, or line
End of string, or line
Start of string
End of string
Empty string at the beginning or end of a word
Empty string not at the beginning or end of a word

5) With a simple program explain groups in regular expressions?

Searching for pattern matches is the basis of the powerful capabilities provided by regular expressions. Adding groups to a pattern isolates parts of the matching text, expanding those capabilities to create a parser. Groups are defined by enclosing patterns in parentheses ((and)).

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 15 14:03:04 2020

@author: P Rama Devi
"""
'''
[] - matches characters in brackets
[^] - matches characters not in bracket
| - either or
() - group

quantifiers

* - 0 or more
+ - 1 or more
? - 0 or one
{1} - exact number
{1,3} - range of numbers(min,max)

groups - ()
'''
#regular expressions
'''
import re
text='''
```

```
Mr. Sachin
Mr Virat
Mrs Karanam
Ms Sindhu
Mr. R

'''

pattern = re.compile(r'(Mr|Ms|Mrs)\.?\s[A-Z]\w*')
matches=pattern.finditer(text)
for match in matches:
    print(match)
'''

emails='''
ramadevi@gmail.com
rama.devi@pes.edu
p-rama-devi@my-work.net
'''

pattern = re.compile(r'[a-zA-Z.]+@[a-zA-Z]+\.(com|edu)')
matches=pattern.finditer(emails)
for match in matches:
    print(match)
```