PYTHON ADVANCED PROGRAMMING

Chitra G M and P Rama Devi
Department of Computer Science Engineering

PYTHON ADVANCED PROGRAMMING

P Rama Devi
Department of Computer Science and Engineering

PYTHON ADVANCED PROGRAMMING

Search Options

The way the matching engine processes an expression can be changed using option flags. The flags can be combined using a bitwise OR operation, then passed to compile() , search() , match() , and other functions that accept a pattern for searching.

Case-Insensitive Matching

IGNORECASE causes literal characters and character ranges in the pattern to match both uppercase and lowercase characters.

```
import re
text = ’This is some text -- with punctuation.’
pattern = r’\bT\w+’
with_case = re.compile(pattern)
without_case = re.compile(pattern, re.IGNORECASE)
print(text)
print(pattern)
print ’Case-sensitive:’
for match in with_case.findall(text):
    print(match)
print ’Case-insensitive:’
for match in without_case.findall(text):
    print(match)
```

Since the pattern includes the literal T , without setting IGNORECASE , the only match is the word This . When case is ignored, text also matches.

Input with Multiple Lines

Two flags affect how searching in multiline input works: MULTILINE and DOTALL .

The MULTILINE flag controls how the pattern-matching code processes anchoring instructions for text containing newline characters. When multiline mode is turned on, the anchor rules for ^ and $ apply at the beginning and end of each line, in addition to the entire string.

PYTHON ADVANCED PROGRAMMING

```
import re
text = ’This is some text -- with punctuation.\nA second line.’
pattern = r’(^\w+)|(\w+\S*$)’
single_line = re.compile(pattern)
multiline = re.compile(pattern, re.MULTILINE)
print(‘text’)
print(‘pattern’)
print(’Single Line’)
for match in single_line.findall(text):
        print ’ %r’ % (match,)
print(‘Multiline’)

for match in multiline.findall(text):
        print(match)
```

The pattern in the example matches the first or last word of the input. It matches line. at the end of the string, even though there is no newline.

PYTHON ADVANCED PROGRAMMING

The previous code can print output for single line and multiline something like this

Single Line :

```
('This', '')
('', 'line.')
```

Multiline:

```
('This', '')
('', 'punctuation.')
('A', '')
('', 'line.')
```

PYTHON ADVANCED PROGRAMMING

DOTALL is the other flag related to multiline text. Normally, the dot character (.) matches everything in the input text except a newline character. The flag allows dot to match newlines as well.

```
import re
text = ’This is some text -- with punctuation.\nA second line.’
pattern = r’.+’
no_newlines = re.compile(pattern)
dotall = re.compile(pattern, re.DOTALL)
print(‘text’)
print(‘pattern’)
print ’No newlines :’
for match in no_newlines.findall(text):
        print(‘match’)
print ’Dotall:’
for match in dotall.findall(text):
        print(‘match’)
```

PYTHON ADVANCED PROGRAMMING

Without the flag, each line of the input text matches the pattern separately. Adding the flag causes the entire string to be consumed.

The output for the previous program can be something like this:

```
No newlines :
’This is some text -- with punctuation.’
’A second line.’

Dotall

’This is some text -- with punctuation.\nA second line.’
```

THANK YOU

Chitra G M and P Rama Devi
Department of CSE

pramadevi@pes.edu
chitragm@pes.edu