

Regular expressions

Chapter 16

This chapter covers

- Understanding regular expressions
- Creating regular expressions with special characters
- Using raw strings in regular expressions
- Extracting matched text from strings
- Substituting text with regular expressions

What is a regular expression?

- A regular expression (regex) is a way of recognizing and often extracting data from certain patterns of text.
- A regex that recognizes a piece of text or a string is said to match that text or string.
- A regex is defined by a string in which certain characters (the so-called metacharacters) can have a special meaning, which enables a single regex to match many different specific strings.

```
1  import re
2
3  regexp = re.compile("hello")
4  count = 0
5
6  file = open("textfile", 'r')
7
8  for line in file.readlines():
9      if regexp.search(line):
10         count = count + 1
11
12  file.close()
13  print(count)
```

Example

- Here's a program with a regular expression that counts how many lines in a text file contain the word hello.
- A line that contains hello more than once is counted only once.

Regular expressions with special characters

- The previous example has a small flaw: It counts how many lines contain "hello" but ignores lines that contain "Hello" because it doesn't take capitalization into account.
- One way to solve this problem would be to use two regular expressions—one for "hello" and one for "Hello"—and test each against every line.
- A better way is to use the more advanced features of regular expressions.

```
3  regexp = re.compile("hello")
4  regexp = re.compile("hello|Hello")
5  regexp = re.compile("(h|H)ello")
6  regexp = re.compile("[hH]ello")
```

Regular expressions and raw strings

- A raw string looks similar to a normal string except that it has a leading r character immediately preceding the initial quotation mark of the string.
- Here are some raw strings:

```
r"Hello"
```

```
r"""\tTo be\n\tor not to be"""
```

```
r'Goodbye'
```

```
r' '12345' ' '
```

Extracting matched text from strings

- One of the most common uses of regular expressions is to perform simple pattern-based parsing on text.

```
surname, firstname middlename: phonenumber
```

```
✓ regexp = re.compile(r"(?P<last>[-a-zA-Z]+),"
                      r" (?P<first>[-a-zA-Z]+)"
                      r"( (?P<middle>([-a-zA-Z]+)))?"
                      r": (?P<phone>(\d{3}-)?\d{3}-\d{4})"
                      )
```

```

1  import re
2
3  regexp = re.compile(r"(?P<last>[-a-zA-Z]+),"
4                      r" (?P<first>[-a-zA-Z]+)"
5                      r"( (?P<middle>([-a-zA-Z]+)))?"
6                      r": (?P<phone>(\d{3}-)?\d{3}-\d{4})"
7  )
8
9  file = open("textfile", 'r')
10 for line in file.readlines():
11     result = regexp.search(line)
12     if result == None:
13         print("Oops, I don't think this is a record")
14     else:
15         lastname = result.group('last')
16         firstname = result.group('first')
17         middlename = result.group('middle')
18         if middlename == None:
19             middlename = ""
20         phonenumber = result.group('phone')
21         print('Name:', firstname, middlename, lastname, ' Number:', phonenumber)
22 file.close()

```


TRY THIS

- Making international calls usually requires a + and the country code.
- Assuming that the country code is two digits, how would you modify the code above to extract the + and the country code as part of the number? (Again, not all numbers have a country code.)
- How would you make the code handle country codes of one to three digits?

Substituting text with regular expressions

- In addition to extracting strings from text, you can use Python's regex module to find strings in text and substitute other strings in place of those that were found.

```
1  import re
2
3  string = "If the the problem is textual, use the the re module"
4
5  pattern = r"the the"
6
7  regexp = re.compile(pattern)
8
9  regexp.sub("the", string)
```

TRY THIS

- In the checkpoint in section 16.4, you extended a phone-number regular expression to also recognize a country code.
- How would you use a function to make any numbers that didn't have a country code now have +1 (the country code for the United States and Canada)?

Lab

Phone-Number Normalizer

Summary

- For a complete list and explanation of the regex special characters, refer to the Python documentation.
- In addition to the search and sub methods, many other methods can be used to split strings, extract more information from match objects, look for the positions of substrings in the main argument string, and precisely control the iteration of a regex search over an argument string.
- Besides the `\d` special sequence, which can be used to indicate a digit character, many other special sequences are listed in the documentation.
- There are also regex flags, which you can use to control some of the more esoteric aspects of how extremely sophisticated matches are carried out.