PYTHON LAB BOOK

Python For Programmers

UCSC Extension Online

Lab 20 re Syntax

Topics

- Regular expression syntax (Optional)
- Testing regular expressions (Optional)

v.4.0

lab19_1.py

```
#!/usr/bin/env python
"""lab19_1.py Finds and adds up all the numbers in a file."""
import re

number_finder = re.compile(r"""
\d+       # 1 or more digits
\.        # followed by a dot
\d*       # and maybe some more digits
|         # or
\d*       # maybe some digits
\.        # followed by a dot
\d+       # and one or more digits
|         # or
\d+       # just some digits
""", re.VERBOSE) # re.VERBOSE flag allows all those comments

def TotalLine(line):
    """Returns the sum of all the numbers in the line."""
    # print number_finder.findall(line)
    return sum([float(each) for each in number_finder.findall(line)])

def TotalIt(stream):
    """Returns the sum of all the numbers in the stream."""
    return sum([TotalLine(line) for line in stream])

def TotalFile(name):
    """Returns the sum of all the numbers in the file."""
    try:
        open_file = open(name)
        try:
            return TotalIt(open_file)
        finally:
            open_file.close()
    except IOError:
        print "I can't read '%s'." % (name)

def main():
    while True:
        try:
            name = raw_input('File name: ')
        except (KeyboardInterrupt, EOFError):
            print
            break
        if name == '':
            break
        total = TotalFile(name)
        if total:
            print name, 'totals to', total

if __name__ == '__main__':
    main()

"""
$ lab19_1.py
File name: ../lab_10_File_IO_and_Packages/numbers.txt
../lab_10_File_IO_and_Packages/numbers.txt totals to 117.5
File name:
$
"""
```

lab19_2.py

```
#!/usr/bin/env python
"""lab19_2.py Swapping from a dictionary"""
import re

def DoSwap(text, replace_d):
    """Swaps all the replace_d.keys() in the text for their values."""
    # Using VERBOSE and named groups for readability
    compiled_re = re.compile(r"""
    \b                # matches a word boundary
    (?P<found>%s)     # matches apple or orange and puts the match
                      # in a group named "found" if the whole thing matches
    (?P<rest>s?)\b    # matches a word boundary or 's' and a word boundary
                      # and puts the 's', or not, into a group named "rest"
    """ % ('|'.join([re.escape(k) for k in replace_d])),
                              re.IGNORECASE | re.VERBOSE)

    def MatchCase(answer, like_string):
        if like_string.isupper():
            return answer.upper()
        if like_string.islower():
            return answer.lower()
        if like_string.istitle():
            return answer.title()
        return answer

    def SwapMatch(match_object):
        found = match_object.group('found')
        return MatchCase(replace_d[found.lower()], found)
    fixed = compiled_re.sub(SwapMatch, text)
    return fixed

def main():
    replace_d = {
        'general':'band leader',
        'china':'mexico',
        'zen':'mariachi',
        'master':'teacher',
        'sword':'baton',
        'through':'around'}

    text = open('zen.story').read()
    print DoSwap(text, replace_d)

if __name__ == '__main__':
    main()
"""
$ lab19_2.py
A band leader in ancient Mexico came to see a Mariachi
teacher.

He drew his baton and pointed it at the teacher, and
announced: "Don't you know that I am a man who can
run you around without blinking an eye?"

To which the Mariachi teacher responded instantly: "Don't
you know that I am a man who can be run around
without blinking an eye?"

Deeply impressed, the band leader sheathed his baton and
remained for the teaching.
$
"""
```

lab19_3.py
```
#!/usr/bin/env python
"""lab19_3.py This evil program crawls around on the web and
collects email addresses. """
import sys
import re
import urllib
import httplib
import signal
if __name__ == '__main__':
    sys.path.insert(0, "..")
else:
    sys.path.insert(0, os.path.join(os.path.split(__file__)[0], '..'))
import lab_12_Function_Fancies.time_out_decorator as time_out

@time_out.TimeOut(3)
def CollectPage(url):
    text=None
    try:
        urlpage=urllib.urlopen(url)
    except (IOError, httplib.InvalidURL):
        return []
    try:
        try:
            text=urlpage.read()
        finally:
            urlpage.close()
    except:
        pass
    return text

def HarvestEmailAddresses(start_with, addresses, max_pages=100):
    pages = [start_with]
    page_working_on = 0
    while page_working_on < max_pages:
        try:
            pages += [url for url in ParsePage(pages[page_working_on],
                                                   addresses)\
                          if url not in pages]
        except IndexError:
            print "Only %d url pages found starting at %s"\
                          % (page_working_on, start_with)
            break
        page_working_on += 1
    return addresses
```

```

def ParsePage(url, addresses):
    try:
        text = CollectPage(url)
    except RuntimeError:
        text = []
    if not text:
        return []
    email_catcher = re.compile(r"""
    [\s,:;(<\['\"]+([^@\s,;:\"]+)@
    ([^@\-\s,:;>\)'\"]+\.[a-z,A-Z]{2,3})[\s,;:\"'>\)]""",
                               re.VERBOSE)
    for new_one in [local_part + '@' + domain \
                    for local_part, domain \
                    in email_catcher.findall(text)]:
        if new_one not in addresses:
            print new_one
            addresses += [new_one]
    url_catcher=re.compile(r"http://[^\s\"<>()']*\b")
    return Uniquify(url_catcher.findall(text))

def Uniquify(a_list):
    a_list.sort()
    try:
        new_list = [a_list[0]]
    except IndexError:
        return []
    for each in a_list[1:]:
        if each not in new_list:
            new_list += [each]
    return new_list

def main():
    start_with='www.ngc.com'
    if len(sys.argv)>1:
        start_with=sys.argv[1]
    if start_with[:7] !='http://':
        start_with='http://' + start_with
    addresses = []
    HarvestEmailAddresses(start_with, addresses, 500)

if __name__=='__main__':
    main()
"""
$ lab19_3.py
```

```
CollectPage(http://www.astro-aerospace.com) timed out at 3 seconds.
Intel-Info@ngc.com
MobileShield@ngc.com  [and many more, and lots of junk.] """
```

```
import re

r"use raw string notation for regular expressions"
r'\n' sees  2 characters: a '\' and a 'n'

In Python, you are looking for "match objects".  There are two ways to get them:
c = re.compile(pattern, flags) -or- match_object = re.search(pattern, string, flags)
match_object = c.search(string)

You can '|' together mode-flags:
 S or DOTALL       '.' matches newline characters too
 I or IGNORECASE   Caseless
 L or LOCALE       In Mexico, e.g., includes accented characters in \w
 M or MULTILINE    affects ^ and $ (above)
 X or VERBOSE      Ignores spaces, newlines and # comments in the pattern.
                   Escape white space or put it in a [ ] to get it seen

match_object = None if there's not a match.

Otherwise, you can query the MatchObject for all you need:

MatchObject.group()        the string matched by the pattern
           .group(i)       the ith subgroup
           .group("found") the subgroup named "found"
           .start()        starting index of the match
           .end()          ending index of the match
           .span()         tuple:  (start_index, end_index) of the match

Functions/methods:                 c = re.compile(pattern, flags)
re.search(pattern, str, flags)     c.search(str)   Finds all matches in the str
 .match(pattern, str, flags)       c.match(str)    Match at the beginning of str
 .findall(pattern, str, flags)     c.findall(str)  Returns all matches as a list
 .finditer(pattern, str, flags)    c.finditer(str) Returns an iterator of matches
re.split(pattern, str, max_split)  c.split(str)    Split str where pattern is found
 .sub(pattern, rpl*, str, count)   c.sub(rpl*, str, count) Replace matches with rpl
 .subn(pattern, rpl*, str, count)  c.subn(rpl*, str, count, n)
                                       returns (new_string, number_of_replacements)
* rpl can be a string or a function
  If it is a string, it can contain: \i or \g<i> to put the i-th subgroup here
                                  \g<found> to put the subgroup named "found" here
  If it is a function, it will be called with the MatchObject as the only argument
  and the function should return the substitution string for that match.
```

```
[ ]   character class -> [abc123] = [a-c1-3]  matches any character listed
      Most metacharacters are not active inside a character class except:
      ^    at the beginning [^abc] means not a or b or c.
           [a^bc] still means a or ^ or b or c
      and special \ sequences which are good inside or outside character classes:
      \d   any decimal character [0-9]
      \D   [^0-9]
      \s   any white-space [ \t\n\r\f\v]
      \S   [^ \t\n\r\f\v]
      \w   any character that might be in a word [a-zA-Z0-9_]
      \W   [^a-zA-Z0-9_]
.     matches any character except newline, unless in DOTALL mode, where it also
      matches newline.  \. or [.] matches literal '.'
\b    current spot must be a word boundary to match
\B    current spot must not be a word boundary to match
^ or \A  anchors match to the beginning of the string or beginning of
         lines in MULTILINE mode (no issue for literal '^')
$ or \Z  anchors match to the end of the string or before newline in
         MULTILINE mode. \$ or [$] matches literal '$'

Greedy repetition: matches as much text as possible:
|     Non-greedy: matches as little text as possible:
|     |      previous character to be matched:
*     *?     0 or more times
?     ??     0 or 1 time
+     +?     1 or more times
{n}   {n}?   exactly n
{n,m} {n,m}? between n and m times
             {,} defaults to {0,2billion}

|     "or" dog|elephant matches either dog or elephant
      (\| or [|] to match literal '|')

()    capture the enclosed pattern as a numbered, and maybe named, subgroup
      The first '(' captures as group 1, etc.
      \i must match previously captured subgroup numbered i
      Extensions to '(':
        (?:              don't capture after-all, and don't number
        (?=             this group must match here, but so must the
                         stuff that follows this group
        (?!              this group must not match here, but the stuff
                         that follows this group must match here
        (?P<found>      names group "found" -- Python extension
        (?P=found        must match what was found above
      \( or [(] to match literal '(' : \) or [)] to match literal ')'
```

re_test.py

```
#!/usr/bin/env python
"""This exercise is from the book "Perl by Example" by Ellie Quigley.
The exercise in Ellie's book asks us to print the city and state
where Norma lives.

I used this little program to develop the regular expression.
"""

import re
import sys

def ReTest(re_str, data, flags):
    """Test the re_str against the data with flags.

    If it doesn't find a hit, try again with one character trimmed off
    the end, and again, and again, until a hit is found.  Then give
    a report.
    """
    for i in range(len(re_str), 0, -1):
        try:
            m = re.search(re_str[:i], data, flags)
            m.groups() # generate an error
        except:
            continue
        else:
            print "This much worked:"
            print re_str[:i]
            print "It broke here:"
            print re_str[i:]
            break

def main():
    data = """
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Lesle Kerstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
JonDeLoach:408-253-3122:123 Park St., San Jose, CA 94086:7/25/53:85100
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56700
Betty Boop:245-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Wilhelm Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35200
Barbara Kerz:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:26850
"""
```

```
    re_str = r"""
    ^%s    # Line starts with the name
    \b     # followed by a non-word character
    (?:    # Un-captured group
    [^:]+? # of non-colons
    :){2}  # followed by a colon, twice
    x      # a mistake!!!
    \d+?   # some digits
    [ ]+   # one or spaces in []
    (?P<town># capturing a group
    # named town. This sequence cannot be
    # split for comments.
    [^:\d] # with no colons or digits
    +?)    # one or more times
    \d      # a digit ends the match
    """ % 'Norma'
    ReTest(re_str, data, re.VERBOSE + re.MULTILINE)

if __name__ == '__main__':
    main()
"""
$ ./re_test.py
This much worked:

    ^Norma    # Line starts with the name
    \b     # followed by a non-word character
    (?:    # Un-captured group
    [^:]+? # of non-colons
    :){2}  # followed by a colon, twice

It broke here:
x      # a mistake!!!
    \d+?   # some digits
    [ ]+   # one or spaces in []
    (?P<town># capturing a group
    # named town. This sequence cannot be
    # split for comments.
    [^:\d] # with no colons or digits
    +?)    # one or more times
    \d      # a digit ends the match
"""
```

norma.py

```
#!/usr/bin/env python
"""Using a regular expression to print the city and state where Norma
lives."""
import re
def GetTownState(text, first_name):
    m = re.search(r"""
    ^%s     # Line starts with the name
    \b      # followed by a non-word character
    (?:     # Un-captured group
    [^:]+?  # of non-colons
    :){2}   # followed by a colon, twice
    [^, ]+?, # some non-commas
    (?P<town># capturing a group
    # named town. This sequence cannot be
    # split for comments.
    [^:\d]  # with no colons or digits
    +?)     # one or more times
    \d      # a digit ends the match
    """ % first_name, text, re.VERBOSE | re.MULTILINE)
    if m:
        return m.group("town")
    return "Not Found"

def ReadFile(file_name):
    try:
        fp = open(file_name)
        try:
            text = fp.read()
        finally:
            fp.close()
    except IOError:
        print "Can't open %s" % file_name
        return None
    return text

def main():
    print GetTownState(RealFile('address.data'), "Norma")

if __name__ == '__main__':
    main()
"""
$ norma.py
 Dearborn, MI
$"""
```

Lab 20

This sample data from *Perl by Example* by Ellie Quigley is in `labs_20_re_syntax/address.data`:

```
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Lesle Kerstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
JonDeLoach:408-253-3122:123 Park St., San Jose, CA 94086:7/25/53:85100
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56700
Betty Boop:245-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Wilhelm Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35200
Barbara Kerz:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:26850
```

The colon separated fields are name:phone:address:DOB:salary.

1. Use Regular Expressions to rewrite the data, giving everyone a $250 raise.
2. Print names and numbers of those in the 408 area code.
3. Change *CA* to *California*.
4. Print the names of people born in March.
5. Print all lines that don't contain *Boop*.
6. Print the file with the first and last names reversed.

PYTHON LAB BOOK

Python For Programmers

UCSC Extension Online

Lab 20 re Syntax

Lab Solutions

v.4.0

lab20_1.py

```
#!/usr/bin/env python
"""lab20_1.py - gives a raise to everyone in the address.data file"""
import re

salary_finder = re.compile("""(\d+)$""", re.VERBOSE | re.MULTILINE)

def GiveRaise(amount, text):
    """Gives a raise to the salaries in the text"""
    def UpIt(match_object):
        return str(int(match_object.group()) + amount)
    return salary_finder.sub(UpIt, text)

def main():
    import norma
    print GiveRaise(250, norma.ReadFile('address.data'))

if __name__ == '__main__':
    main()

"""$ lab20_1.py
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34450
Lesle Kerstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52850
JonDeLoach:408-253-3122:123 Park St., San Jose, CA 94086:7/25/53:85350
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56950
Betty Boop:245-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43750
Wilhelm Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43750
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245950
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45250
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35450
Barbara Kerz:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:27100

$ """
```

lab20_2.py

```
#!/usr/bin/env python
"""lab20_2.py Print names and numbers of those in the 408 area code."""

import re
import norma  # to collect the data

def GetWhoInAreaCode(text, area_code):
    return re.findall(
        r"""^(?P<name>[^:]+?)   # captured the name
        :%s-                    # must match area code
        (?P<number>\d{3}-\d{4}) # captured the phone number
        :""" \
        % area_code, text, re.MULTILINE | re.VERBOSE)

def main():
    print '\n'.join(["%25s: %s" % (name, number) \
                     for name, number in \
                     GetWhoInAreaCode(norma.read_file('address.data'), 408)])

if __name__ == '__main__':
    main()


"""
$ lab20_2.py
             Tommy Savage: 724-0140
            Lesle Kerstin: 456-1234
               JonDeLoach: 253-3122
$ """
```

lab20_3.py

```
#!/usr/bin/env python
"""lab20_3.py Change "CA" to "California"."""
import re
import norma  # to collect the data

def WholeCa(text):
    return re.sub("""CA""", "California", text)

def main():
    print WholeCa(norma.read_file('address.data'))

if __name__ == '__main__':
    main()

"""
$ lab20_3.py
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, California 94087:5/19/66:34200
Lesle Kerstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
JonDeLoach:408-253-3122:123 Park St., San Jose, California 94086:7/25/53:85100
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56700
Betty Boop:245-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Wilhelm Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35200
Barbara Kerz:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:26850

$ """
```

lab20_4.py

```
#!/usr/bin/env python
"""lab20_4.py Print the names of people born in March."""
import re
import norma  # to collect the data

def BornIn(month, text):
    return re.findall(
        r"""^([^:]+?)     # capture the name
        (?::[^:]+?)       # don’t capture colon followed by non colons
        {2}               # twice
        :%s               # follwed by colon and month number
        """ % month, text, re.MULTILINE|re.VERBOSE)

def main():
    print BornIn(3, norma.read_file(’address.data’))

if __name__ == ’__main__’:
    main()

"""
$ lab20_4.py
[’Norma Corder’]
$ """
```

lab20_5.py
```
#!/usr/bin/env python
"""lab20_5.py Print all lines that don't contain "Boop"."""
import re
import norma  # to collect the data

def GetName(name, text):
    return re.findall(
        r"""^      # at the beginning of a line
        (?!        # must not match
        .*?%s.*?)  # the name anywhere
        (.*?)      # but do mach the whole line
        $          # to the end
        """ % name, text, re.MULTILINE|re.VERBOSE)

def main():
    print '\n'.join(GetName('Boop', norma.read_file('address.data')))

if __name__ == '__main__':
    main()

"""
$ lab20_5.py
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Lesle Kerstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
JonDeLoach:408-253-3122:123 Park St., San Jose, CA 94086:7/25/53:85100
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56700
Wilhelm Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35200
Barbara Kerz:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:26850
$
"""
```

lab20_6.py

```
#!/usr/bin/env python
"""lab20_6.py Print the file with the first and last names reversed."""
import re
import norma  # to collect the data

def ReverseNames(text):
    compiled_re = re.compile(r"""^(?P<name>[^:]+?)(?P<rest>:)""", re.MULTILINE)
    def DoTheReverse(match_object):
        try:
            first, last = match_object.group('name').split()
        except ValueError:
            return match_object.group('name') +  match_object.group('rest')
        return ', '.join((last, first)) + match_object.group('rest')
    return compiled_re.sub(DoTheReverse, text)

def main():
    print ReverseNames(norma.read_file('address.data'))

if __name__ == '__main__':
    main()

"""
$ lab20_6.py
Savage, Tommy:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Kerstin, Lesle:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
JonDeLoach:408-253-3122:123 Park St., San Jose, CA 94086:7/25/53:85100
Hardy, Ephram:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/20:56700
Boop, Betty:245-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Kopf, Wilhelm:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Corder, Norma:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
Ikeda, James:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Gortz, Lori:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/76:35200
Kerz, Barbara:385-573-8326:832 Pnce Drive, Gary, IN 83756:12/15/46:26850
$
"""
```