第 6 回 Python 正規表現ゼミ

ゼミ担当者　：　西井　琢真, 水野　珠季
開催日　　：　2009 年 5 月 21 日

ゼミ内容: 正規表現 (regular expression) とは，ある文字列のパターンを表現するために利用される表現手法のことであり，プログラミングにおいて文字列の検索や置換を行うために利用される．本報告では，まず正規表現とは何かについて説明を行い，その後 Python における正規表現の基礎的な使用方法について解説する．

1　はじめに

正規表現とは，ある文字列のパターンを表現するために利用される表現手法のことである．通常の文字列とメタ文字，特殊シーケンスと呼ばれる文字を組み合わせた文字列（パターンという）により調べたい文字列の規則性を定義し，その規則に合致する文字列を調べることを目的としている．例えば，URL や html，連続した n 桁の数字など，一定のルールを持つ文字列を処理 (検索，置換，分割) するのに有効な手段である．正規表現を使用可能なプログラミング言語は，Python，Java，JavaScript，Peal，Ruby，PHP など様々である．本報告では正規表現についての概要を説明した後，Python における正規表現を使った文字列処理 (検索，置換，分割) の基礎について解説する．

2　正規表現

Python で正規表現を使用する前に，まず正規表現とは何かについて述べる．以下によく使う用語とその意味を示す．

- パターン:
  正規表現によって表わされる文字列. 正規表現パターンともいう．
- 文字:
  正規表現においては全角文字，半角文字，記号，空白文字を指す (改行は除く)．
- メタ文字:
  パターンに用いられる”.”や”[ ]”などの特殊な文字 (Table1 を参照)
- 特殊シーケンス:
  \d や\w などの数字，小文字・大文字などを表す特殊な意味を持つ文字である（Table2 を参照）
- マッチ:
  ある文字列と正規表現パターンが一致すること．

2.1　正規表現のメリット

正規表現を使うことで固定された文字列だけでなくあいまいな文字列を表現することが可能になる．例えば，”http://〜”で始まる URL は，正規表現パターンでは”http://[A-Za-z0-9.?/]+”と表現でき，ある文章中から URL を抜き出すプログラムを作る際に有用である．

2.2　正規表現のデメリット

正規表現自体を覚えることが難しい，複雑な正規表現のパターンが多用されたプログラムは読みづらいといったデメリットがある．また，Python は正規表現関数の処理に時間を要するので，固定された文字列を処理する場合は文字列関数を使用することが推奨されている．

2.3　正規表現パターンの説明

Python では，re モジュール関数に正規表現パターンと処理対象となる文字列を渡すことで，文字列の中から該当する単語などを抜き出し置換や分割を行うことが可能である．正規表現パターンに用いられるメタ文字一覧を Table 1 に示す．

以下に正規表現パターンを使った例を示す．

1 回以上の繰り返しにマッチする例

正規表現パターン:`"x+"`
マッチする文字列の例:`x,xx,xxx,…`,

x が 1 回以上連続した文字列にマッチする．

固定文字列＋任意の文字列にマッチする例

正規表現パターン:`"http://www\..*/"`
マッチする文字列の例:
`http://www.microsoft.com/`

”http://www.”と”/”で挟まれた任意の文字列にマッチする．”.”はメタ文字なので”.”をただの文字として扱う場合は，”\”でエスケープすることが必要になる．

Table 1　メタ文字一覧

メタ文字 (全て半角)	意味
.	任意の 1 文字を表す．改行を除く全角，半角，空白文字
^	文字列の先頭にマッチする
$	文字列の末尾にマッチする
[文字列]	文字集合の中の 1 文字にマッチする
*	直前にある 1 文字の 0 回以上の繰り返し．ワイルドカードとは異なる
+	直前にある 1 文字の 1 回以上の繰り返し
?	直前にある 1 文字の 0 回か 1 回の繰り返し
|	二種類のパターンの間に挟んで用い，OR を表す
(文字列)	( ) 内の文字列を 1 つとして扱う
{m}	直前のパターンの m 回の繰り返し
{m,n}	直前のパターンの m 回以上の n 回以下の繰り返し
[^文字列]	…以外の文字にマッチする
\	メタ文字をエスケープする

二桁の数字にマッチする例

正規表現パターン:`"[0-9]{2},"`
マッチする文字列の例:00,01,02,…,99

[ ] 内の文字が {} 内の数字分繰り返されるという意味になる．”[0123456789][0123456789]”も同義.[a-z] は英小文字，[A-Z] は英大文字,[0-9] は 1 桁の数字，[A-Za-z0-9] は英数字という意味になる．

”第 n 章”にマッチする例 (n は数字)

正規表現パターン:`"第 [0-9]+章"`
マッチする文字列の例:第 1 章，第 2 章，…,

第+数字+章の組み合わせにマッチする．

どちらかの文字列にマッチする例

正規表現パターン:`"(僕たち|私たち) は健康だ"`
マッチする文字列の例:僕たちは健康だ，私たちは健康だ

() 内で | を使うことにより，僕たち，私たちのどちらかにマッチすればという意味になる．

ある文字列の文末にマッチする例

正規表現パターン:`"です。$"`
マッチする文字列の例:りんごです。

文末が”です。”で終わる文字列にマッチする．

3　Python における正規表現

3.1　正規表現を使う 2 つの方法

Python で正規表現を扱うには，re モジュールを用いる．正規表現は大きく分けて二つの方法で使用することができる．

まず 1 つ目は，事前に正規表現パターンをコンパイルして正規表現オブジェクトを作成し，作成したオブジェクトに対するメソッド呼び出しで処理を行う方法である (主なメソッドは Table 2 を参照).

事前に正規表現オブジェクトを作成する方法

```
rc = re.compile("正規表現パターン")
result = rc.search("処理対象とする文字列")
```

同じパターンを複数回利用する際は，コンパイルを一度しか行う必要がなく，処理時間を短縮できる．

2 つ目は，re モジュールに定義された関数に，正規表現パターンを引数として渡し，マッチ処理を行う方法である．

正規表現パターンを引数として渡す方法

```
result = re.search("正規表現パターン", "処理対象とする文字列")
```

こちらは事前のコンパイルが必要なく，一度に処理を記述できるというメリットがある．ソースコードにより前者と後者のどちらを適用するべきかは異なる．

結果は，マッチオブジェクトや文字列のリストとして返される．ここでは，search 関数は結果をマッチオブジェクトとして返し，結果は result 変数に格納される．マッチオブジェクトは正規表現にマッチした文字列のインデックスや，マッチした文字列などの情報を持つ．例えば，result.group() でマッチした文字列を取得することができる (マッチオブジェクトに対するメソッドは Table 3 を参照).

Table 3　マッチオブジェクトに対するメソッド一覧

メソッド	意味
group()	正規表現にマッチした文字列を返す
start()	マッチした文字列が始まる位置を返す
end()	マッチした文字列が終わる位置を返す

Table 2　正規表現オブジェクトに対するメソッド

メソッド	意味	返り値
findall	string 中からパターンにマッチする文字列を全て探す	文字列リスト
search	文字列を走査し，正規表現がマッチする位置を調べる	マッチオブジェクト
match	正規表現が，文字列の先頭でマッチするかどうか調べる	マッチオブジェクト
split	正規表現がマッチする全ての位置で，文字列をリストに分割して返す	文字列リスト
sub	正規表現がマッチする全ての部分文字列を見つけ出し，別の文字列に置換して返す	string

4　検索

文字列の検索には search() メソッドや findall() メソッドを用いる．以下で例を用いて使用法を説明する．

url を含む文字列から url を抽出する

```
# -*- coding:  utf-8 -*-
import re
url = "ここ ('http://www.google.com/') をクリックしてください"
pattern = "http://[A-Za-z0-9.?/]+"
regExp = re.search(pattern, url)
if regExp :
    print regExp.group()
else :
    print "no pattern"
```

正規表現を使うため，re モジュールをインポートする必要がある．代入用変数 regExp には search 関数からオブジェクトが返され，url に pattern が含まれていれば，マッチした文字列を参照することができる．もし，該当するパターンがなければ search 関数は None を返す．search は，文字列を走査し正規表現が最初にマッチした位置で探索を終了するので，1 つの文字列しか返すことができない．

url を含む文字列から複数の url を抽出する

```
(上記のプログラムに加えて記述する)
multiUrl = "('http://www.google.com/') と ('http://www.yahoo.co.jp/') をクリック"
secondRegExp=re.findall(pattern,multiUrl)
if secondRegExp :
    print secondRegExp
else :
    print "no pattern"
```

multiUrl 内には複数の url が含まれている．パターンにマッチする文字列を全て探したいときは，findall を使う必要がある．

文字の出現回数を調べる

```
# -*- coding:  utf-8 -*-
import re
str = "apple orange superman"
pattern = "[a-e]"
regExp = re.findall(pattern, str)
if regExp :
    print "%d 回出現"%len(regExp)
else :  print "no pattern"
```

パターンで a,b,c,d,e を指定し，それが str 中に何回出現するかを調べるプログラムである．findall で返された文字列リストの要素数から出現回数を得ることができる．

5　置換

パターンにマッチする部分を見付け出し、別の文字列と置換する場合には sub() メソッドを用いる．

正規表現オブジェクトを使用する場合

```
rc = re.compile("正規表現パターン")
result = rc.sub("置換する文字列", "処理対象とする文字列")
```

正規表現パターンを引数として渡す場合

```
result = re.sub("正規表現パターン", "置換する文字列", "処理対象とする文字列")
```

sub() メソッドは置換した文字列を返す．以下に sub() メソッドを使った簡単な例を示す．

置換の例

```
p = re.compile( '(blue|white|red)')
print p.sub( 'colour', 'blue socks and red shoes')
#出力：'colour socks and colour shoes'
```

6　分割

文字列の分割には split() メソッドを使用する．split() メソッドは正規表現がマッチした全ての部分で文字列を分割し，各部分を文字列のリストとして返す．

Table 4　特殊シーケンス一覧

特殊シーケンス	意味
\d, \D	\d は数字とマッチ，\D は数字以外とマッチする
\s, \S	\s は空白や水平タブなどの空白文字列とマッチする．\S は空白文字列以外とマッチする
\w, \W	\w は英数字とマッチし，[a-zA-Z0-9] と同義．\W は英数字以外とマッチする

Table 5　フラグ一覧

フラグ (ショートネーム)	意味
DOTALL,S	. が改行も含めて、全ての文字とマッチするように指定する
IGNORECASE,I	大文字小文字を区別しない
LOCALE, L	ロケールを考慮してマッチングを行う
MULTILINE, M	複数行にマッチング（^と $ に影響する）
VERBOSE, X	冗長な正規表現を有効にする

正規表現オブジェクトを使用する場合

```
rc = re.compile("正規表現パターン")
result = rc.split("処理対象とする文字列")
```

正規表現パターンを引数として渡す場合

```
result = split("正規表現パターン", "処理対象
とする文字列")
```

以下に split() メソッドを使った簡単な例を示す.

分割の例

```
p = re.compile( '(blue|white|red)')
print p.split( 'color', 'blue shoes and
yellow soxes.')
#出力：'color shoes and yellow soxes.'
```

今回は触れなかったが，特殊シーケンスを用いることで正規表現パターンをよりスマートに記述することができる．re モジュールで使用できる特殊シーケンスの一覧を Table 4 に示す．また，正規表現オブジェクトのコンパイル時に，引数にフラグを指定することで特別な設定や文法の変更を実現することも可能である．使用可能なフラグの一覧を Table 5 に示す.

7　演習問題

以下のようなファイル名の一覧があるとき，ファイルの拡張子が.txt のものを.bak に変更し，変更後のファイル名一覧を出力せよ.

ファイル一覧

```
atxt.exe
contents.txt.gz
data.dat
index.html
readme.txt
```

参考文献

1) 柴田 淳: みんなの Python，ソフトバンク クリエイティブ株式会社，2006

2) Python Japan Users Group
http://www.python.jp/Zope/articles/tips/regex_howto/regex_howto_1

3) サルにもわかる正規表現入門
http://www.mnet.ne.jp/~nakama/