プロジェクト第３回

正規表現

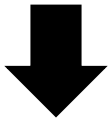
正規表現

正規表現

- 文字列の集合を１つの文字列で表現する

- 例

集合 {a,aa,aaa,aaaa,aaaaa,...}



正規表現 a+

+は直前の文字が１つ以上繰り返される

正規表現の種類

文字	説明	同様	例	マッチする	マッチしない
\d	任意の数字	[0-9]			
\D	任意の数字以外	[^0-9]			
\s	任意の空白文字	[\t\n\r\f\v]			
\S	任意の空白文字以外	[^\t\n\r\f\v]			
\w	任意の英数字	[a-zA-Z0-9_]			
\W	任意の英数字以外	[\a-zA-Z0-9_]			
\A	文字列の先頭	^			
\Z	文字列の末尾	$			
.	任意の一文字	-	a.c	abc, acc, aac	abbc, accc
^	文字列の先頭	-	^abc	abcdef	defabc
$	文字列の末尾	-	abc$	defabc	abcdef
*	０回以上の繰り返し	-	ab*	a, ab, abb, abbb	aa, bb
+	１回以上の繰り返し	-	ab+	ab, abb, abbb	a, aa, bb
?	０回または１回	-	ab?	a, ab	abb
{m}	m回の繰り返し	-	a{3}	aaa	a, aa, aaaa
{m,n}	m〜n回の繰り返し	-	a{2, 4}	aa, aaa, aaaa	a, aaaaa
[]	集合	-	[a-c]	a, b, c	d, e, f
縦線	和集合（または）	-	a縦線b	a, b	c, d
()	グループ化	-	(abc)+	abc, abcabc	a, ab, abcd

Ex3_1. reモジュール

```
import re

content = "hello python, 123, end."
pattern = "he”

# compileしてからmatch
repatter = re.compile(pattern)
result = repatter.match(content)

print(result.group())
```

Ex3_2. 数字の抽出

```
import re

content = "hello python, 123,end"
# ()で取りたい文字を
pattern = '.*?(¥d+).*'
result = re.match(pattern, content)

if result: #matchした時のみ
    # group()で全文字を
    print(result.group()) # hello python, 123,end
    # group(1)で数字を
    print(result.group(1)) # 123
```

Ex3_3. search関数

```
import re

s = "aaa@xxx.com, bbb@yyy.com, ccc@zzz.net"

m = re.search("[a-z]+@[a-z]+¥.net", s)

print(m.group())
```

演習の準備

- http://www.cl.ecei.tohoku.ac.jp/nlp100/data/jawiki-country.json.gzにアクセスする

- このファイルを展開して、実行するファイルと同じ場所に置く

今回のデータ説明

- Wikipediaのデータ

- １行に１記事の情報がJSON形式で格納される
- 各行には記事名が”title”キーに、記事本文が”text”キーの辞書オブジェクトに格納され、そのオブジェクトがJSON形式で書き出される
- ファイル全体はgzipで圧縮される

データの中身

```
{{イギリス関連の項目}}
{{ヨーロッパ}}
{{EU}}
{{国連安全保障理事会理事国}}
{{G8}}
{{OECD}}
{{イギリス連邦}}

{{デフォルトソート:いきりす}}
[[Category:イギリス|*]]
[[Category:英連邦王国|*]]
[[Category:G8加盟国]]
[[Category:欧州連合加盟国]]
[[Category:海洋国家]]
[[Category:君主国]]
[[Category:島国|くれいとふりてん]]
[[Category:1801年に設立された州・地域]]
```

Ex3_4. イギリスの記事抽出

```
import json

with open("jawiki-country.json") as f:
    articles=f.readlines()
    for con in articles:
        a_dic=json.loads(con)
        if a_dic["title"]=="イギリス":
            texts=a_dic["text"]
            print(texts)
```

Ex3_5. カテゴリ名の行抽出

- Ex3_4の続き

```
categories=[]

for c in texts.split("¥n"):
    if "Category" in c:
        categories.append(c)
print(categories)
```

演習1. カテゴリ名の抽出

Ex3_5ではカテゴリ名を行単位で抽出していた。行単位ではなく名前で抽出せよ。reモジュールのsearch関数を使用する。

ヒント：赤文字の正規表現を考える（１文字）

^¥[¥[Category:(.*?)(|¥|.*)¥]¥]__

演習1のコード

```
categories2=[]

for c in texts.split("¥n"):
    c_obj=re.search(正規表現 ,c)
    if not(c_obj ==None):
        categories2.append(c_obj.group(1))

print(categories2)
```

演習2. ファイル参照の抽出

イギリスの記事から参照されているメディアファイルを全て抽出せよ。メディアファイルは”File”または”ファイル”で表現されている。

ヒント：赤文字の正規表現を考える（３文字）

(File|ファイル):(___)¥|

演習2のコード

```
for f in texts.split("¥n"):
    f_obj=re.search(正規表現,f)
    if not(f_obj ==None):
        print(f_obj.group(2))
```