

正規表現オブジェクトについて もっと詳しく

ー チェリー本 第6章（正規表現を理解する） ー

P.197 6.5 - P.201 6.6

6.5.1 正規表現オブジェクトを作成するさまざまな方法

P.197～

- 正規表現オブジェクトを作成する様々な方法

今まで学習した内容： 「/」 を利用した方法

- ①Regexp.newの引数にパターンの文字列を渡す方法

```
Regexp.new('\d{3}-\d{4}')
```

- ②「%r」を使う方法（%記法） スラッシュをエスケープする必要がない

```
%r!http://example\.com!  
%r{http://example\.com}
```

※エスケープしている場合

```
/http:\\example\.com/
```

変数の中身展開もできるよ！

`#{ }`

6.5.2 case文で正規表現を使う

P.197～

- case文ではwhen節で正規表現を使う

```
text = '03-1234-5678'
text = '2021/7/19'
text = '000-0000'

case text
when /\d{3}-\d{4}$/
  puts '郵便番号です'
when /\d{4}\/\d{1,2}\/\d{1,2}$/
  puts '日付です'
when /\d+-\d+-\d+$/
  puts '電話番号です'
end
```

郵便番号 : When /\d{3}-\d{4}\$/

「^は文頭\$は文末、半角数字3文字 ハイフン
半角数字4文字をスラッシュでかこんている」

日付 : when /\d{4}\/\d{1,2}\/\d{1,2}\$/

「文頭文末はさっきと一緒に、半角数字4文字、エスケープしてからスラッシュ、
半角数字1～2文字、エスケープしてからスラッシュ、半角数字1～2文字」

電話番号 : when /\d+-\d+-\d+\$/

「半角数字1文字以上 ハイフン 半角数字1文字以上 ハイフン半角数字1文字以上

Regexp.newの場合

```
#Regexp.newのやり方
case text
when Regexp.new('^d{3}-d{4}$')
  puts '郵便番号です'
when Regexp.new('^d{4}/d{1,2}/d{1,2}$') #エスケープ外し
  puts '日付です'
when Regexp.new('^d+-d+-d+$')
  puts '電話番号です'
end
```

%記法の場合

```
##%記法
case text
when %r!^d{3}-d{4}$!
  puts '郵便番号です'
when %r!^d{4}/d{1,2}/d{1,2}$! #エスケープ外し
  puts '日付です'
when %r!^d+-d+-d+$!
  puts '電話番号です'
end
```

6.5.3 正規表現オブジェクト作成時のオプション

P198～

【参考】 <https://docs.ruby-lang.org/ja/latest/method/Regexp/c/IGNORECASE.html>

- iオプション

大文字、小文字を区別しない

```
#iオプション  
'HELLO' =~ /hello/i  
  
'HELLO' =~ %r{hello}i  
  
regexp = Regexp.new('hello', Regexp::IGNORECASE)  
'HELLO' =~ regexp
```

```
irb(main):003:0> regexp = Regexp.new('hello', Regexp::IGNORECASE)  
=> /hello/i
```

Regexpを使う場合は
Regexp::IGNORECASEという定数を渡す
下の画像がターミナルirb上の結果

- mオプション

任意の文字を表すドット(.)が改行文字にもマッチするようになる。

Regexpを使う場合は
Regexp::MULTILINEという定数を渡す
下の画像がターミナルirb上の結果

```
#mオプション
"Hello\nBye" =~ /Hello.Bye/m
"Hello\nBye" =~ %r{Hello.Bye}m

regexp = Regexp.new('Hello.Bye', Regexp::MULTILINE)
"Hello\nBye" =~ regexp
```

```
irb(main):009:0> regexp = Regexp.new('Hello.Bye', Regexp::MULTILINE)
=> /Hello.Bye/m
```

- xオプション

空白文字（半角スペースや改行文字）が無視 # を使って正規表現中にコメントが書ける。

xオプションを付けている時に空白を無視せず正規表現の一部として扱いたい場合はバックスラッシュでエスケープする。

```
#xオプション
regexp = /
  \d{3} #郵便番号の先頭3桁
  -    #区切り文字のハイフン
  \d{4} #郵便番号の末尾4桁
/x

'123-4567' =~ regexp

regexp = /
  \d{3}
  \ #区切り文字のハイフン
  \d{4}
/x

'123 4567' =~ regexp
```


- xオプション (Regexp.newを使う場合)

```
pattern = <<'TEXT'  
  \d{3} #郵便番号の先頭3桁  
  -    #区切り文字のハイフン  
  \d{4} #郵便番号の末尾4桁  
TEXT  
regexp = Regexp.new(pattern, Regexp::EXTENDED)  
'123-4567' =~ regexp
```

Regexpを使う場合は
Regexp::EXTENDEDという定数を渡す
下の画像がターミナルirb上の結果

```
irb(main):001:0> pattern = <<'TEXT'  
  \d{3} #郵便番号の先頭3桁  
  -    #区切り文字のハイフン  
  \d{4} #郵便番号の末尾4桁  
TEXT  
irb(main):002:0' irb(main):003:0' irb(main):004:0' irb(main):005:0'  
=> "  \\d{3} #郵便番号の先頭3桁\\n  -    #区切り文字のハイフン\\n  \\d{4} #郵便番号の末尾4桁\\n"  
irb(main):006:0> regexp = Regexp.new(pattern, Regexp::EXTENDED)  
=> /  \d{3} #\u90F5\u4FBF\u756A\u53F7\u306E\u5148\u982D3\u6841  
  -    #\u533A\u5207\u308A\u6587\u5B57\u306E\u30CF\u30A4\u30D5\u30F3  
  \d{4} #\u90F5\u4FBF\u756A\u53F7\u306E\u672B\u5C3E4\u6841  
/x  
irb(main):007:0> '123-4567' =~ regexp  
=> 0
```

返ってきている値が
xオプションの形になっている

※これらのオプションは同時に使うこともできる (テキスト参照)

6.5.4 組み込み変数でマッチの結果を取得する

P.199

- \$で始まる特殊な変数（組み込み変数）が存在する。

```
text = '私の誕生日は1977年7月17日です。'  
text =~ /(\d+)年(\d+)月(\d+)日/  
$~ #<MatchData "1977年7月17日" 1:"1977" 2:"7" 3:"17">
```

正規表現に () を使うと、
その部分がキャプチャ（捕捉）され連番がつけられる

```
# マッチした部分全体を取得する  
$& #=> "1977年7月17日"  
  
# 1番目～3番目のキャプチャを取得する  
$1 #=> "1977"  
$2 #=> "7"  
$3 #=> "17"  
  
# 最後のキャプチャ文字列を取得する  
$+ #=> "17"
```

組み込み変数を使う場合はどの記号がどの意味になるのかを
きちんと理解して使う必要がある

6.5.5 Regexp.last_matchでマッチの結果を取得する

P.200

【参考】 https://docs.ruby-lang.org/ja/latest/method/Regexp/s/last_match.html

- 先程の組み込み演算子のように =~ 演算子などで最後にマッチした結果を取得できる。

```
text = '私の誕生日は1977年7月17日です。'
text =~ /(\d+)年(\d+)月(\d+)日/

Regexp.last_match #<MatchData "1977年7月17日" 1:"1977" 2:"7" 3:"17">

# マッチした部分全体を取得する
Regexp.last_match(0) #=> "1977年7月17日"

# 1番目～3番目のキャプチャを取得する
Regexp.last_match(1) #=> "1977"
Regexp.last_match(2) #=> "7"
Regexp.last_match(3) #=> "17"

# 最後のキャプチャ文字列を取得する
Regexp.last_match(-1) #=> "17"
```


6.5.6 組み込み変数を書き換えないmatch?メソッド

P.200

- match?メソッド マッチすればtrue マッチしなければfalseを返す
=~演算子やmatchメソッドよりも高速に動作する。

```
/\d{3}-\d{4}/.match?('123-4567') #=> true  
'123-4567'.match?(/\d{3}-\d{4}/) #=> true  
$~ #nil  
Regexp.last_match #nil
```

文字列と正規表現の順番
は入れ替えてもOK

マッチしても組み込み変数や
Regexp.last_matchの内容を書き換えない

最後に

正規表現を使う場合は自分が利用しているRubyのバージョンに応じた公式ドキュメントをチェックしよう！

まとめ

- ・ 正規表現を / / 以外で書く方法 (Regex.new %r)
- ・ iオプション：大文字小文字を区別しない
- ・ mオプション：(.)が改行文字にもマッチするようになる
- ・ xオプション：正規表現中にコメントが書ける
- ・ どのオプションを使う場合でもRegex.newを利用してコードを書く時は引数に注意 (IGNORECASEやEXTENDEDなど)
- ・ \$：組み込み変数を使ってマッチ結果を取得する方法
- ・ Regex.last_matchを使ってマッチ結果を取得する方法
- ・ match?メソッド：true falseで返すメソッド 高速で、マッチしても組み込み変数やRegex.last_matchメソッドの内容を書き換えない

みんなのRubyレベルが
1上がった！！👑

