テキスト処理

田中哲
産業技術総合研究所
情報技術研究部門
tanaka.akira+textprocess@gmail.com
http://staff.aist.go.jp/tanaka-akira/textprocess/

今日の内容

- この授業の概要
- Ruby の使いかた
- 正規表現の使いかた
- Rubyのインストール

この授業の概要

- 質問について
- 授業の資料
- 授業の狙い
- テキスト処理とは?
- 授業の構成
- 評価
- 参考書

質問について

- 授業中はいつでも可
- 大学に常駐していないので口頭での質問は授業の後のみ
- メールでの宛先:
  tanaka.akira+textprocess@gmail.com

授業の資料

- 前日までにプレゼン資料を用意する
  http://staff.aist.go.jp/tanaka-akira/textprocess/

授業の狙い

- 日常生活でテキストを処理するプログラムを書く力を身につける
- テキスト処理の仕組を理解する
- 形式言語理論の基礎を学ぶ

テキスト処理とは?

- テキストを処理する (文字どおり)
  - テキスト
    - プレインテキスト
    - HTML
    - XML
    - etc.
  - 処理
    - 検索
    - 集計
    - etc.

授業の構成

- Rubyを使ってみる
- 正規表現を使ってみる
- 正規表現エンジンを作ってみる
- 正規表現の理論を学ぶ
- 理論にあわないところを学ぶ
- XMLを使ってみる
- XMLの理論を学ぶ

評価

- レポート6割
  最大のレポートは正規表現エンジンを作るものになる?
- 試験4割

参考文献

- 必要というわけではないが、興味があれば
- Rubyについて
  - たのしいRuby
    http://www.notwork.org/sbp-ruby/
  - Rubyプログラミング基礎講座
    http://www.gihyo.co.jp/books/4-7741-2645-4
  - Rubyリファレンスマニュアル (web)
    http://www.ruby-lang.org/ja/man/
- 理論について
  - オートマトン 言語理論 計算論 I
    http://www.saiensu.co.jp/books-htm/ISBN4-7819-0374-6.htm

Rubyの使いかた

- プログラミング言語Ruby
- Rubyの起動
- データの表示

プログラミング言語Ruby

- オブジェクト指向スクリプト言語
- 正規表現をサポートしている
- XMLのライブラリが添付されている
- 開発元: http://www.ruby-lang.org/

Hello World(1)

- Hello World と表示するプログラム
- print “Hello World\n”

Hello World (2)

- プログラム: print “Hello World\n”
- 引数から実行

```
% ruby -e 'print "Hello World\n"'
Hello World
```

- ファイルから実行

```
% cat hello.rb
print "Hello World\n"
% ruby hello.rb
Hello World
```

- irbから実行

```
% irb
irb(main):001:0> print "Hello World\n"
Hello World
=> nil
```

Rubyの起動

- 引数から実行
  % ruby -e 'コード'
- ファイルから実行
  % ruby コードの入ったファイル名
- irb (対話型Ruby) から実行
  % irb
  irb(main):001:0> コード

データの表示: p

- p 式

ruby -e 'p 1'	1	整数
ruby -e 'p 1+1'	2	整数な式
ruby -e 'p “a”*3'	“aaa”	文字列
ruby -e 'p [1, 2+3, “z”]'	[1,5,”z”]	配列
ruby -e 'p true'	true	真
ruby -e 'p false'	false	偽
ruby -e 'p nil'	nil	nil

正規表現

- 文字列照合の道具
- 以下のようなことを調べられる
  - 文字列の中に cde という文字列が含まれているか?
  - 文字列の中に c が3個以上並んでいて、
    その次に d があるところがあるか?
  - 文字列の中に bc という文字列か、
    de という文字列か、すくなくともどちらかは含まれて
    いるか?
  - 文字列は、abc の繰り返しか?
- このようなことが成立したら、
  「正規表現がマッチした」という

正規表現マッチの例

- 文字列 “abcdef” の中に cde が含まれているか?

  ruby -e 'p /cde/ =~ “abcdef”'

  2　　“abcdef” の 2文字目からマッチ (0-origin)

  cde は含まれている

  cde の最初の文字が 2文字目

- 文字列 “abcdef” の中に xyz が含まれているか?

  ruby -e 'p /xyz/ =~ “abcdef”'

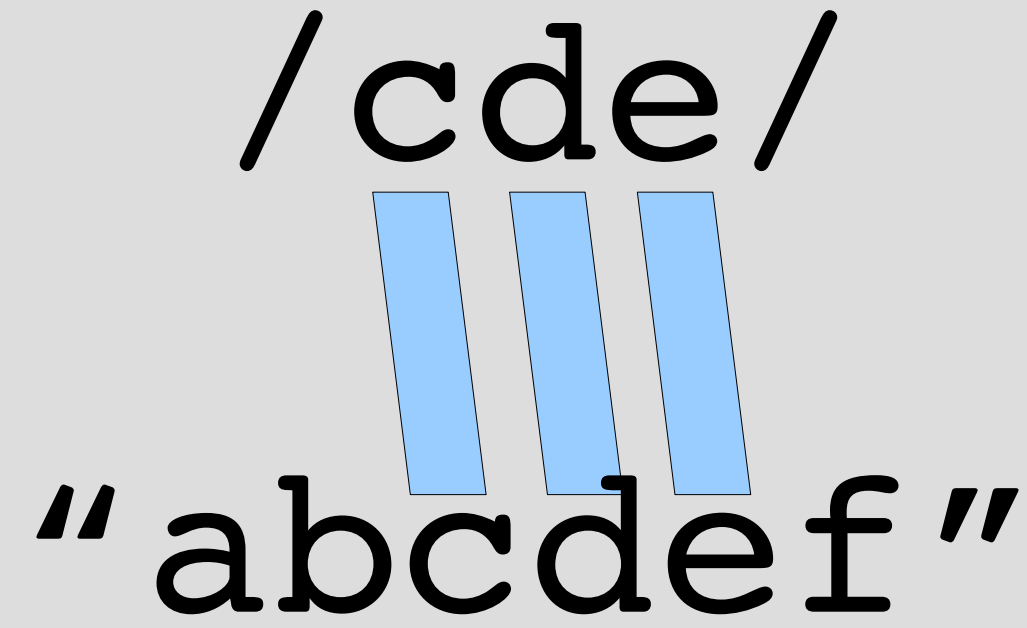
  nil　　マッチしない

  xyz は含まれていない

正規表現マッチ

- /正規表現/ =~ 文字列
- マッチしたらマッチした位置を返す
  - 文字列の先頭からマッチしたら 0 を返す (0文字目)
  - 次の文字からマッチしたら 1 を返す (1文字目)
  - 以下同様
- マッチしなかったら nil を返す

/cde/ =~ “abcdef”

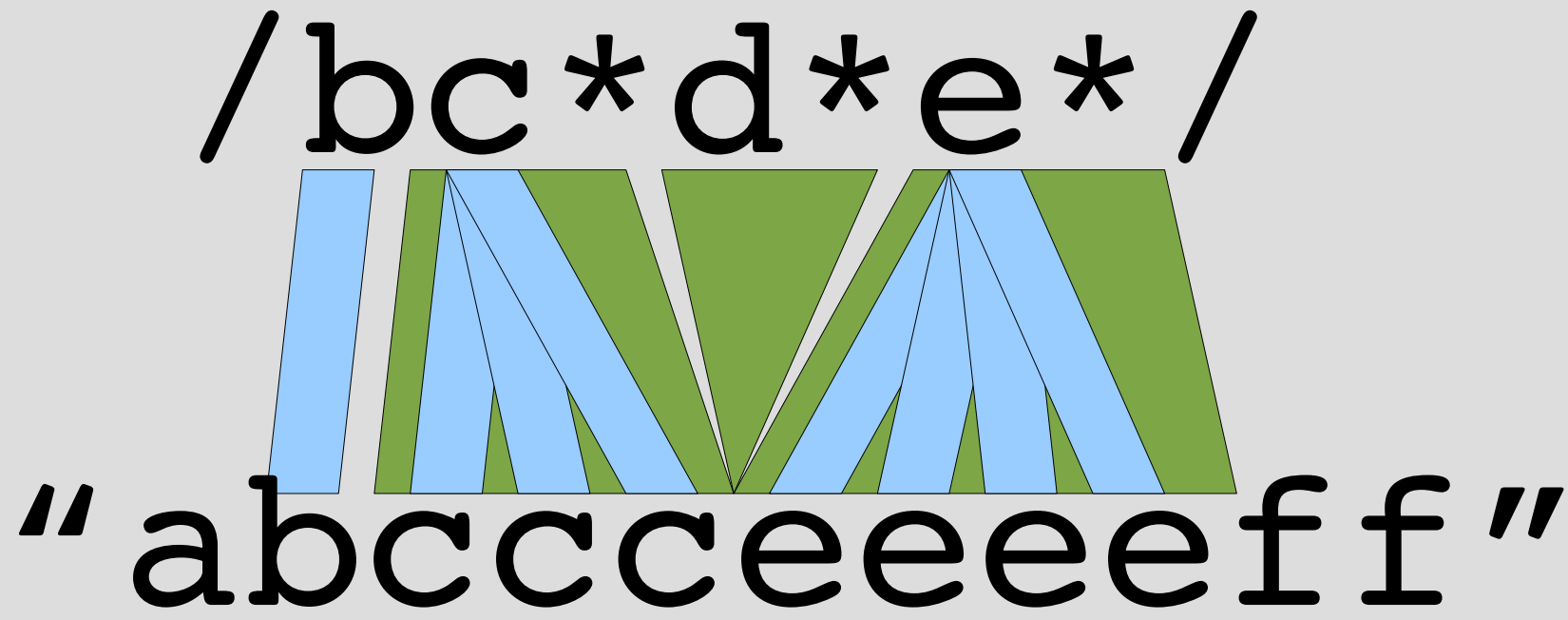


正規表現の要素

- 文字 c
- 連接 rr
- 繰り返し r*
- 選択 r|r
- 文字列の先頭 \Ar
- 文字列の最後 r\z
- グループ化 (r)
- etc.

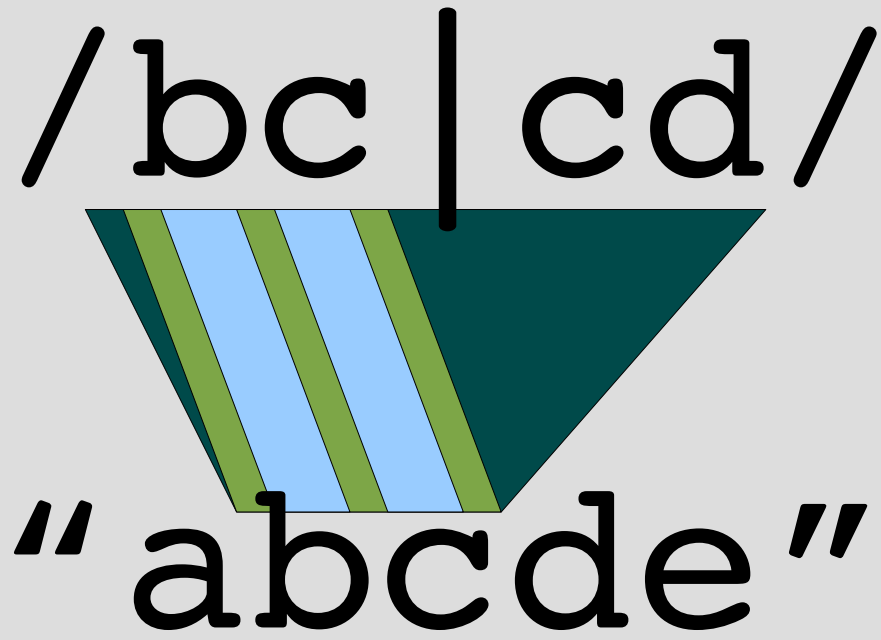
繰り返し /bc*d*e*/ =~ “abcccceeeeff”

- r* は r の 0回以上の繰り返しにマッチする



選択 /bc|cd/ =~ “abcde”

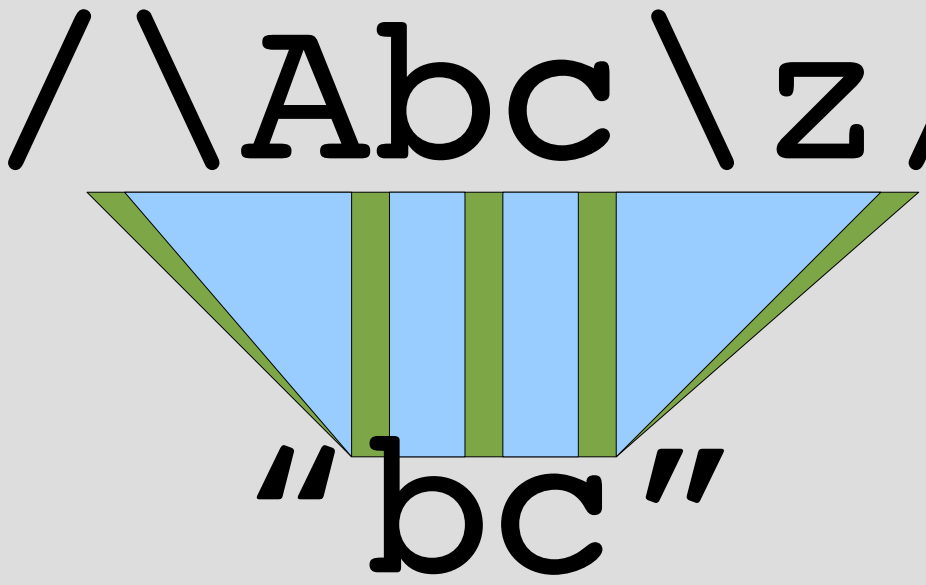
- r1|r2 は r1 と r2 のどちらかにマッチする



- 文字列の左から試していくので cd はマッチしない

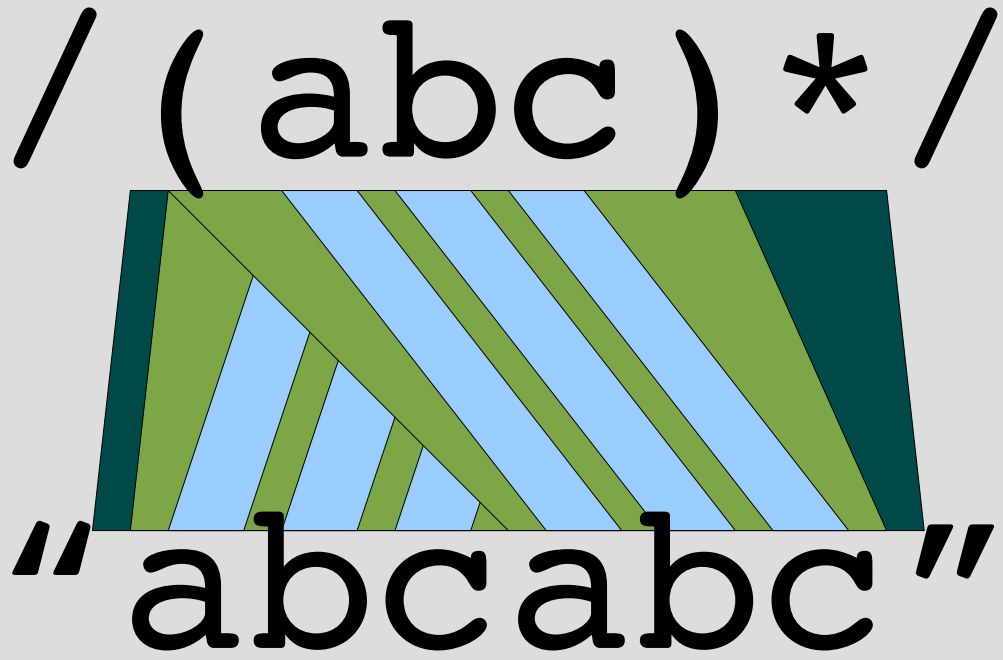
文字列の先頭・最後 /\Abc\z/ =~ “bc”

- \A は文字列の先頭にしかマッチしない
- \z は文字列の最後にしかマッチしない
- /\Abc\z/ =~ “bc” はマッチする (0 を返す)
- /\Abc\z/ =~ “abcd” はマッチしない (nil を返す)



グループ化 /(abc)*/ =~ “abcabc”

- (r) は r にマッチする



いくらでも組合せ可能

- \A((a|b)*|(de|f*g)*)
- abc(abc)*\z
- (\Aabc)|(def\z)
- etc.

- 疑問:
  表現できないものはあるか?
  なにが表現できてなにが表現できないのだろう?
  →形式言語理論が答になる

Rubyのインストール

- この授業では最新安定版 Ruby 1.8.4 を想定する
- 「Rubyの歩き方」が参考になる
  http://jp.rubyist.net/magazine/?FirstStepRuby
- Windows 上で ActiveScriptRuby を使用する場合、インストーラを実行すれば良い
- Unix では、多くの場合パッケージがある
- パッケージがなくても、自分でコンパイルすることはできる
- それもできなければ質問すること

まとめ

- この授業の概要
- Ruby の使いかた
- 正規表現の使いかた
- Rubyのインストール