電子計算機入門

池田, 大輔
九州大学情報基盤センター

電子計算機入門
第11回

池田 大輔

daisuke@cc.kyushu-u.ac.jp

情報基盤センター

目次

- 文字列の検索と置換
- 文字列の分割・統合

検索と置換とは

- どちらもコンピュータ上で実現できる基本的な操作

検索と置換とは

- どちらもコンピュータ上で実現できる基本的な操作
- 検索
  - 入力：文字列 t と p
  - 問題：t 中に p が現われるかどうか調べる
    - 文字列 t 中にある文字 c が含まれるかどうかは

```
if c in t:
```

検索と置換とは

- どちらもコンピュータ上で実現できる基本的な操作
- 検索
  - 入力：文字列 t と p
  - 問題：t 中に p が現われるかどうか調べる
    - 文字列 t 中にある文字 c が含まれるかどうかは

      ```
      if c in t:
      ```
- 置換
  - 入力：文字列 t と、文字列 p_1, p_2
  - 問題：t 中に現われる p_1 を p_2 で置きかえること

Pythonでの検索と置換

- 基本的な機能は、“文字列” というデータが提供する
- 例：文字列 `str` に対し

  ```
  str.find(substr)
  ```

  で `str` 中の `substr` の位置を探す
- より高度な検索や置換は正規表現 (Regular Expression) モジュールで提供される (次回説明予定)

  ```
  import re
  ```

  して利用する

基本的な検索 find

- 文字列操作のためのモジュール

```
str.find(substr[, start[, end]])
```

  - [] は省略可能な部分
  - str 中で最初に substr が出現する位置 (整数) を返す
    - str は文字列を格納した変数や、文字列そのもの
  - 見つからなければ-1 を返す
  - 省略しなければ str[start:end] 中で探す

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

→2となる

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

→2となる

```
print "acgtaacgt".find("gt")# 文字列から
```

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

→2となる

```
print "acgtaacgt".find("gt")# 文字列から
```

→2となる

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

→2となる

```
print "acgtaacgt".find("gt")# 文字列から
```

→2となる

```
print var.find("gt", 3)
```

例：find

```
var = "acgtaacgt"
print var.find("gt")# 変数から探す
```

→2となる

```
print "acgtaacgt".find("gt")# 文字列から
```

→2となる

```
print var.find("gt", 3)
```

→7となる(後の“gt”の位置)

実際の使い方：find

```
p = var.find(pattern) # 位置を保存
if p >= 0:
    # pattern が見つかった時の処理
else:
    # そうでないときの処理
```

実習：find

- FASTA 形式のファイルの配列部分のみ出力させなさい
  - FASTA 形式は、遺伝子配列を表現するための形式で

```
>gi|16127994:190-255
ATGAAACGCATTAGCACCACCAT....
>gi|16127994:337-2799
ATGCGAGTGTTGAAGTTCGGCGG....
TGGAAAGCAATGCCAGGCAGGGG....
GGTGGCGATGATTGAAAAAACCA....
...
```

    という形式である
  - '>' で始まる行が DNA の位置などを表わし、それ以外の行が遺伝子等を表わす
  - ただし、長い場合は配列部分は適宜改行されている
- "NC_000913.ffn" で実際にやってみる

実習：find (Cont.)

- 入力：ファイル名
- 復習：ファイルを１行づつ読み込むには

```
f=open(file, 'r')
line=f.readline()
while line:
    print line, # 単に表示するだけ
    line=f.readline()
```

実習：解答例

```
import sys
file = sys.argv[1]
f=open(file, 'r')
line=f.readline()
while line:
    if line.find('>') >= 0:
        print line,
    line=f.readline()
```

その他の検索メソッド

- `index`
  - `find` と同じだが、見つからないときに `ValueError` 例外を返す
- `rfind`, `rindex`
  - `find` と `index` と同じだが、最後にある位置を返す

indexの使い方

```
try:
    p = var.index(pattern)
    # pattern が見つかった時の処理
except ValueError:
    # そうでないときの処理
```

その他のメソッド

- 部分文字列 sub の出現回数を求める

  ```
  count(sub[, start[, end]])
  ```

- 条件判定
  - 文字列が数字であるか、小文字から成るかなどを調べる

    ```
    isalnum(), isalpha(), isdigit(),
    islower(), isspace(), issuper()
    ```

    例：`if var.isalnum():`
  - 文字列の最初や最後がどうなっているか

    ```
    startswith(), endswith()
    ```

    例：`if line.startswith('>'):`

置換

- 文字列 old を new に置きかえる場合

```
replace(old, new)
```

  - replace は置きかえた後の文字列を返す

- 例

```
var="abcd"
var=var.replace("ab", "AB")
```

実習：置換

- 入力：ACGT からなる文字列
- 出力：入力文字列の相補な文字列を出力せよ
- 相補 (Complement)
  - A と T のペア、C と G のペアで文字をいれかえて反転させる
  - 例：ACCTG ー (反転) → GTCCA ー (入替) → CAGGT

文字列の分割と統合

- FASTA 形式では、長い配列は分割されている
- '>' で始まる行から次の'>' で始まる行までを 1 本の長い配列にしないといけない
- 文字列配列の統合：`join` メソッドを使う
  - 統合すべき個々の文字列は配列に格納されている

FASTA形式の場合...

- アルゴリズム

```
while 行がある間:
    if '>' で始まっている:
        ここまでに格納された配列 seq の中身を
        統合する
    else:
        配列 seq に現在の行を追加
```

FASTA形式の場合...

```
'>gi|161...'        # seq=[]
ATGAAACGCT          # seq.append()
'>gi|161...'        # join&seq=[]
AACGCATTAG          # seq.append()
AACGCATTAG          # seq.append()
ATAACATTAG          # seq.append()
'>gi|161...'        # seq=[]
AGAACGCTTA          # seq.append()
TAAGCATTAG          # seq.append()
'>gi|161...'        # seq=[]
```

join

- 配列 `array=["ab", "cd", "ef", ...]`

  `seq=''.join(array)`

  # join は文字列を対象にしたメソッドではないので

  # 何もない文字列" から呼びだす

  # `seq="abcdef...."`となる

実習：join(今日の課題)

- 入力：FASTA 形式のファイル
- 出力：各遺伝子を 1 本の文字列として出力

split

- join の逆で、規則性のある文字列をその規則で分ける

```
seq="ab:cde:af:dde"
array=seq.split(':')
# 何で分けるか明記する
# array=["ab", "cde", "af", "dde"] となる
```