文字列処理，ファイルの読み書き，正規表現

高橋　弘喜

2018-11-19

文字列処理

パソコンで扱うファイルは，テキストファイルとバイナリファイルに大別される．テキストファイル内に記載されている文字列情報の中から，目的の情報を抽出することが有益となることが多い．また，頻繁にその対応に迫られる．

生命科学においては，様々なフォーマットに様々な情報が記載されており，一つのファイルで全ての情報を得ることが容易ではない．多くの場合，複数のファイルから相互参照などを行うことで目的の情報を抽出することになる．

そのためにも，Python などのスクリプト言語によって，テキストファイル処理を習得することは，生物情報をより広範に整理・収集する有効な手立てとなる．

テキストファイル

テキストファイル (Text File) は，文字など文字コードによって表されるデータだけが含まれるファイルのことで，ファイルフォーマットの一種と見なすこともできる．互換性が高く幅広い環境でデータを利用できる利点がある一方，単純な文字だけしか扱えないという制限がある [1]．

- gff
- sam
- fastq
- fasta

gff や sam のように，複数列から成るテキストファイルを扱うことが多い．列の区切り文字としては，`comma(,)`，タブ (`\t`)，スペース ( ) が主に使用されている．

バイナリファイル

テキストとはデータの内容すべてを人間が読んで理解できる (human-readable) もの，バイナリとはそうでないものを指す [1]．

- bam
- gz

ファイルの読み書き

プログラムで処理した結果を保存しておきたい場合や，外部から Python にデータを取り込む場合にファイルを使用する．

Python でファイルを操作するためには，組み込み型の**ファイル型**を使う．

ファイルを読み込む

ファイルの読み込みについていくつかスクリプトを作成して，実行する．

```
with open (filename, 'r') as f: ## 'r'は省略化
```

ファイルに書き込む

ファイルの書き込みは，読み込み同様 open を使用する．ただし，mode='w' とする．

```
with open (filename, 'w') as f:
```

表 1: open のモード

オプション	解説
r	ファイルを読み込み専用に開く． 存在しないファイルを開こうとするとエラーになる．
w	ファイルを書き出し専用に開く． ファイルが存在しない場合は新規に作成する． 既に存在している場合は，ファイルの内容を空にする．
a	ファイルの最後に追記する．
+	読み込みと書き出しを両方可能にする． r+: 読み書き両対応で開く．ファイルが存在しない時はエラー． w+: 読み書き両対応で開く．ファイルが存在しない時は新規作成される． a+: 追加書き込み・読み書き両対応で開く． ファイルが存在しない時は新規作成される．
b	ファイルをバイナリモードで開くオプション．

GFF3

パーセントエンコーディング (URL エンコード)

URI において使用できない文字を使う際に行われるエンコード（一種のエスケープ）の名称である[1]．

GFF3 files are nine-column, tab-delimited, plain text files. Literal use of tab, newline, carriage return, the percent (%) sign, and control characters must be encoded using RFC 3986 Percent-

Encoding; no other characters may be encoded. Backslash and other ad-hoc escaping conventions that have been added to the GFF format are not allowed. The file contents may include any character in the set supported by the operating environment, although for portability with other systems, use of Latin-1 or Unicode are recommended.

tab (%09) newline (%0A) carriage return (%0D) % percent (%25) control characters (%00 through %1F, %7F) In addition, the following characters have reserved meanings in column 9 and must be escaped when used in other contexts:

; semicolon (%3B) = equals (%3D) & ampersand (%26) , comma (%2C) Note that unescaped spaces are allowed within fields, meaning that parsers must split on tabs, not spaces. Use of the "+" (plus) character to encode spaces is depracated from early versions of the spec and is no longer allowed.

Undefined fields are replaced with the "." character, as described in the original GFF spec[2].

read0.py

s288c_n20.gff を読み込んで出力する．一行目だけ出力する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
  line=f.readline()
  print(line)
```

write0.py

test1.txt に出力する．

```
output="./output/test1.txt"
out="Hello world!\n"
with open(output, 'w') as f:
    f.write(out)
```

read1.py

s288c_n20.gff を読み込んで出力する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
  for line in f:
    print(line)
```

read2.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
```

```
  for line in f:
    line=line.rstrip()
    print(line)
```

read3.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．#で始まる行だけを出力する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
  for line in f:
    line=line.rstrip()
    if line.startswith("#"):
      print(line)
```

read4.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．#で始まる行以外を出力する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
  for line in f:
    line=line.rstrip()
    if line.startswith("#"):
      continue
    print(line)
```

read5.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．#で始まる行以外を出力する．9 列目のデータのみを出力する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
  for line in f:
    line=line.rstrip()
    if line.startswith("#"):
      continue
    s=line.split("\t")
    print(s[8])
```

read6.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．#で始まる行以外を出力する．9 列目のデータのみを出力する．；で区切る．

```
input="./input/s288c_n20.gff"
with open(input) as f:
    for line in f:
        line=line.rstrip()
        if line.startswith("#"):
            continue
        s=line.split("\t")
        items=s[8].split(";")
        for item in items:
            print(item)
```

read7.py

s288c_n20.gff を読み込んで出力する．改行コード削除を実行する．#で始まる行以外を出力する．9 列目のデータのみを出力する．；で区切る．product を抽出する．

```
input="./input/s288c_n20.gff"
with open(input) as f:
    for line in f:
        line=line.rstrip()
        if line.startswith("#"):
            continue
        s=line.split("\t")
        items=s[8].split(";")
        for item in items:
            if item.startswith("product="):
                print(item)
```

SAM

２列目の FLAG 情報は，リードのマッピング状況を知ることができる．

pysam には，様々な関数が用意されている [3]．

sam0.py

どのような FLAG が存在するか集計してみる．下記のサイトで検索することができる．
https://broadinstitute.github.io/picard/explain-flags.html

```
input="../1-1/sam/SRR453566.sam.aa"
dict={}
with open(input) as f:
    for line in f:
        line=line.rstrip()
        if line.startswith("@"):
            continue
```

```
        s=line.split("\t")
        flg=s[1]
        if flg in dict:
            dict[flg]+=1
        else :
            dict[flg]=1
for k, v in dict.items():
    print(k+"\t"+str(v))
```

sam1.py

特定の FLAG を有するリード情報を抽出する．マッピングされなかったリードを抽出する．read unmapped は，0x4，4 で表現される．

ビット演算子&を使用する．

表 2: ビット演算子の一覧

ビット演算子	説明
x|y	x と y の論理和（OR）を取る．
x&Y	x と y の論理積（AND）を取る．
x^Y	x と y の排他的論理和（XOR）を取る．

```
input="../1-1/sam/SRR453566.sam.aa"
readList=[]
with open(input) as f:
    for line in f:
        line=line.rstrip()
        if line.startswith("@"):
            continue
        s=line.split("\t")
        FLG=int(s[1])
        if FLG & 4:
            readList.append(s[0])
for item in readList:
    print(item)
```

正規表現

正規表現（せいきひょうげん、英: regular expression）とは，文字列の集合を一つの文字列で表現する方法の一つである．正則表現（せいそくひょうげん）とも呼ばれ，形式言語理論の分野では比較的こちらの訳語の方が使われる．まれに正規式と呼ばれることもある [1]．

正規表現とは，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．通常の文字列とメタ文字と呼ばれる特殊な文字を組み合わせてパターンを作り，パターンに指定された法則で並ぶ文字

列検索を実現できる [4].

課題 1

../1-1/reference/s288c_e.gff を処理して，gene_id と product の一覧表を作成する．product は URL エンコードされている．

```
import urllib.parse
product="glutamate dehydrogenase %28NADP%28%2B%29%29 GDH3"
product=urllib.parse.unquote(product)
print(product) ## glutamate dehydrogenase (NADP(+)) GDH3
```

```
gene_0001       seripauperin PAU8
gene_0002       hypothetical protein
gene_0003       putative permease SEO1
gene_0004       hypothetical protein
gene_0005       hypothetical protein
gene_0006       Tda8p
```

課題 2

../1-1/sam/SRR453566.sam.aa を処理して，異なる染色体にマッピングされたリードペアを探索する．同じ染色体にマップされた場合，7 列目に=が記載されている．アンマップのリードの場合，*が記載されている．

参考文献

1. Wikipedia. Available at: https://ja.wikipedia.org/wiki/メインページ.

2. Generic feature format version 3 (gff3). Available at: https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md.

3. Pysam. Available at: https://pysam.readthedocs.io/en/latest/api.html.

4. 柴田淳. みんなの *Python* 第 4 版. (SB クリエイティブ株式会社, 2017).