前回の残り ～ 数えてみよう

もう１がんばり～codonをアミノ酸に変換してみる

どんなアミノ酸列ができるのか？　codon(3文字)をアミノ酸に置換え

第一塩基	第二塩基 U	第二塩基 C	第二塩基 A	第二塩基 G
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Try UAC Try UAA 終結 UAG 終結	UGU Cys UGC Cys UGA 終結 UGG Trp
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg
A	AUU Ile AUC Ile AUA Ile AUG Met	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly

アミノ酸	表記 3文字	1文字	pH7での側鎖の性質
Alanine(アラニン)	Ala	A	非極性(疎水性)
Cysteine(システイン)	Cys	C	極性
Asparatic acid (アスパラギン酸)	Asp	D	極性(親水性、酸性)
Glutamic acid (グルタミン酸)	Glu	E	極性(親水性、酸性)
Phenylalanine (フェニルアラニン)	Phe	F	非極性(疎水性)
Glycine (グリシン)	Gly	G	非極性
Histidine (ヒスチジン)	His	H	極性(親水性、塩基性)
Isoleucine (イソロイシン)	Ile	I	非極性(疎水性)
Lysine (リジン)	Lys	K	極性(親水性、塩基性)
Leucine (ロイシン)	Leu	L	非極性(疎水性)
Methionine (メチオニン)	Met	M	非極性(疎水性)
Asparagine (アスパラギン)	Asn	N	極性(親水性、中性)
Proline (プロリン)	Pro	P	非極性
Glutamine (グルタミン)	Gln	Q	極性(親水性、中性)
Arginine (アルギニン)	Arg	R	極性(親水性、塩基性)
Serine (セリン)	Ser	S	極性
Threonine (スレオニン)	Thr	T	極性
Valine (バリン)	Val	V	非極性(疎水性)
Tryptophan (トリプトファン)	Trp	W	非極性
Tyrosine (チロシン)	Tyr	Y	極性

前回の残り ～ 数えてみよう

もう１がんばり～codonをアミノ酸に変換してみる

どんなアミノ酸列ができるのか？
codon(3文字)をアミノ酸に置換え

if で1つ1つ条件で分けるのも一法

```
if codon=='AAA':
    aa = 'K'          Lysine
elif codon=='AAC':
    aa = 'N'          Asparagine
elif codon=='AAG':
```

ここでは「辞書型」を使ってみる

辞書型：　対応表のようなもの

takasa

'東京タワー'	333
'スカイツリー'	634
'富士山'	3776
'通天閣'	108

takasa['通天閣']は108を返す

予め辞書codonsに
codon⇒AAの対応表を入れてある

```
aa = codons[codon]
```

辞書codonsはimportできるように
用意してある
但しおまじないが必要

```
import sys                  # おまじない
sys.path.append('.')  # おまじない
from codons import codons
```

こうしておけば

```
aa = codons[好きなcodon]
```

でaaにアミノ酸が得られる

同じフォルダ内のcodons.py
をファイル一覧から開いてみよ

数えてみよう

もう１がんばり～codonをアミノ酸に変換してみる

では、コドンのリスト cdn から
　アミノ酸に変換してみよう

cdn から１つずつ取り出して、
　codons[. . .]
で変換しよう
（結果はアミノ酸を表す１文字）

変換結果はアミノ酸の配列にしたい
空の文字列 aa を用意しておき
cdnから1つずつ取り出して
　　codons[…] で変換するが
　　その結果は aa の後にくっつける
　　（文字列の結合でよい）
できたら aa を表示・確認　　TAチェック

732485.aa.txtに正解があるので比較してみよ

別のDNA配列でも試してみよう？
⇒　X00226.1.fasta

今回を始める前に ～ 準備

今回は少しややこしいので
よく注意を聞いてステップごとに

ファイルをダウンロードしよう

①下記のホームページ
　http://pepper.is.sci.toho-u.ac.jp/DL
から、デスクトップへ、mb-9.zip をダウンロード

②デスクトップで、mb-9.zipをダブルクリックすると、解凍される
　⇒ デスクトップにフォルダmb-9ができる
（右欄へ続く）

慣れない学生に対して
TAはサポートをお願いします

③フォルダmb-9の中にできたファイルを
　全部選択して、
　右クリックでコピーを選んでコピーして
④ D(USB):¥Wpy-3662¥notebookフォルダの中に貼り付ける
　主に使うファイルは
　　The_Adventure_Of_The_Dancing_Men.txt
　　（ホームズ「踊る人形」テキスト）
　　neko.txt（「吾輩は猫である」テキスト）
　　MeCabtest.ipynb

⑤テスト：　Jupyter Notebookで
　MeCabtest.ipynb を開き、run
　　語のリストができればＯＫ

文字数を数える続き

普通の文でも数えてみよう

英文（英語）では
　文字の出現頻度に偏りがある

シャーロックホームズ「踊る人形」
　英字→人形の形　に１対１変換した
　暗号があった。
　それを、出現頻度の偏りを頼りに
　解読する　という話

（英語に限るが）eが最も頻度が高い
　第二次大戦の本職の暗号解読（英国）でも
　第一歩として使われたらしい

ここでは、「eが最も頻度が高い」が
　本当かどうか、確かめてみよう

例題テキスト：　ホームズの
　The Adventure Of The Dancing Men
http://www.gutenberg.org/files/108/108-0.txt

何をするか？

①ファイルを読込む

　ファイルは既にフォルダにあるはず

　今回は行に関係なく全部読込みしよう

②全部小文字に変換する

　大文字小文字を別々に数えると後で面倒

　初めに全部小文字にしてしまおう

③改行文字¥nは要らないので消す

　まあ あっても大丈夫なのだけれど

　うっとうしいので消し方を覚えよう

④後は数えて結果を表示

　全文の文字列をCounterで数えればよい

文字数を数える続き

もう少し詰めてゆこう

①ファイルを読込む

The_Adventure_Of_The_Dancing_Men.txt

今回はファイル全体を一括で読込む

```
with open('ファイル名', 'r') as f:
    str = f.read()
```

'ファイル名' のファイルを、'r' モードで
　open き、それを f と呼ぶことにする
そのfに対して、read() 手続きを実行し
　その結果を str に代入する
　　結果の str は文字列で、改行は
　　改行文字 ¥n で表されている

```
This is a pen.¥nHe is a boy.¥n
```

② str の内容を全部小文字にする

```
str = str.lower()
```

lower() は、strに対して全部小文字にする手続きで、得られた結果をもう一度変数 str に書き戻している

```
This is a pen.¥nHe is a boy.¥n
```

⇩

```
this is a pen.¥nhe is a boy.¥n
```

③改行文字¥nは要らないので消す

```
str = str.replace('¥n', '')
```

replace() は、strに対して文字列の置換をする手続き。'¥n' を ''（空の文字列）に置換えて、結果を str に書き戻す

```
This is a pen.¥nHe is a boy.¥n
```

⇩

```
this is a pen.he is a boy.
```

文字数を数える続き

もう少し詰めてゆこう

④最後に文字数を数える

先頭で

```
from collections import Counter
```

としておいて、

```
Counter(str)
```

とすれば、文字数を数えることができる

得られた結果は：

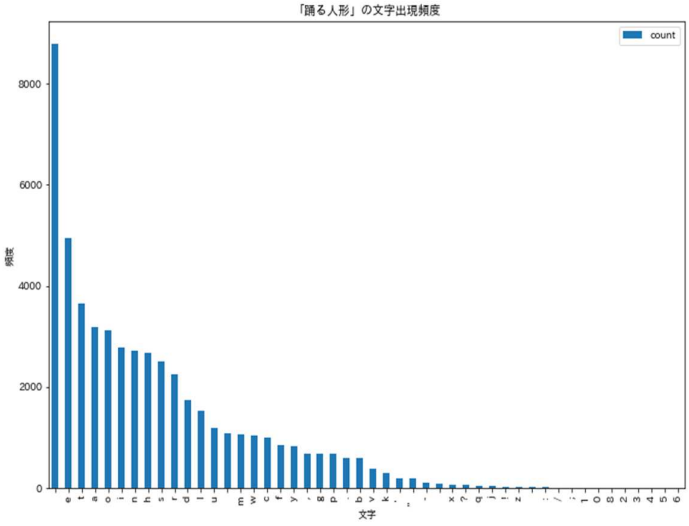
```
Counter({' ': 8795, 'e': 4941, 't': 3657, 'a': 3191, 'o': 3127, 'i': 2788, 'n': 2717, 'h': 2673, 's': 2513, 'r': 2244, 'd': 1732, 'l': 1532, 'u': 1198, '¥n': 1081, 'm': 1065, 'w': 1048, 'c': 1008, 'f': 843, 'y': 829, ',': 676, 'g': 674, 'p': 669, '.': 595, 'b': 590, 'v': 388, 'k': 306, '“': 200, '”': 186, '-': 101, '’': 78, 'x': 64, '?': 59, 'q': 35, 'j': 33, '!': 19, 'z': 17, '‘': 17, ':': 14, '/': 5, '1': 3, '0': 3, ';': 3, '8': 2, '2': 1, '3': 1, '4': 1, '5': 1, '6': 1})
```

確かに e が多く、t, a, o などが続いている

TAチェック

おまけ ～ グラフを描いてみた

書き方はあとで（１１ページ）



日本語で単語カウント

日本語の単語分割はやや面倒

単語への分割：

英語 ⇒ 空白で切ればいい

記号に対して考慮は必要…

日本語 ⇒ 切れ目がない

吾輩|は|猫|で|ある|。

形態素解析

吾輩 は 猫 で ある 。

形態素解析

辞書・アルゴリズム⇒大変

出来合いパッケージを使う

⇒ 無償のMeCabやJumanなど

PythonでMeCabを使う：

```
from natto import MeCab
str = '今日は良い天気です'
m = MeCab()
p = m.parse(str)
print(p)
```

```
今日	名詞,副詞可能,*,*,*,*,今日,キョウ,キョー
は	助詞,係助詞,*,*,*,*,は,ハ,ワ
良い	形容詞,自立,*,*,形容詞・アウオ段,基本形,良い,ヨイ,ヨイ
天気	名詞,一般,*,*,*,*,天気,テンキ,テンキ
です	助動詞,*,*,*,特殊・デス,基本形,です,デス,デス
EOS
```

```
for pl in p.split('¥n')[:-1]:
    word = pl.split('¥t')[0]
    print(word)
```

⇒

```
今日
は
良い
天気
です
```

split('...') ： 文字列を'...'で分割 ⇒ 結果はリスト

'abc¥ndef¥ngh¥n' ⇒ ['abc', 'def', 'gh', '']

日本語で単語カウント

「吾輩は猫である」を単語分割 ⇒ 出現頻度をカウント

元のテキスト： neko.txt

出典＝青空文庫　https://www.aozora.gr.jp/
⇒ 著作権の切れた作品をオンライン化

今回の処理用に整形済み

読込み：　readlinesで

```
from natto import MeCab
m = MeCab()
with open('neko.txt', 'r') as f:
    lines = f.readlines()
for l in lines:
    １行ごとの処理をここへ書く
        １行ごとに単語を取出して
        単語リストwlistにappendする
```

TAチェック

```
wlist = []   # 空リストを作っておく
for l in lines:
    # １行ごとの処理をここへ書く
    p = m.parse(l)  # MeCab処理をする
    for pl in p.split('¥n')[:-1]:
        # MeCab結果を行ごとに分ける→pl
        word = pl.split('¥t')[0]
        # word=行内をタブ¥tで分けた0番目
        wlist.append(word)
        # できたwordをwlistにappendする

# lのループが終われば全単語がwlistに入る
print(wlist[:20])  # wlistの先頭20個

# あとはwlist上の単語の頻度を数えるだけ
```


吾輩は猫である
夏目漱石

【テキスト中に現れる記号について】
《》:ルビ
(例)吾輩《わがはい》は猫である

～～中略～～

[＃8字下げ]一[＃「一」は中見出し]

　吾輩《わがはい》は猫である。名前はまだ無い。
　どこで生れたかとんと見当《けんとう》がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番｜獰悪《どうあく》な種族であったそうだ。この書生というのは時々我々を捕《つかま》えて煮《に》て食うという話である。しかしその当時は何という考もなかったから別段恐しいとも思わなかった。ただ彼の掌《てのひら》に載せられてスーと持ち上げられた時何だかフワフワした感じがあったばかりである。掌の上で少し落ちついて書生の顔を見たのがいわゆる人間というものの見始《みはじめ》であろう。この時妙なものだと思った感じが今でも残っている。第一毛をもって装飾されべきはずの顔がつるつるしてまるで薬缶《やかん》だ。

整形してある

吾輩は猫である
名前はまだ無い
どこで生れたかとんと見当がつかぬ
何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している
吾輩はここで始めて人間というものを見た
しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ
この書生というのは時々我々を捕えて煮て食うという話である
しかしその当時は何という考もなかったから別段恐しいとも思わなかった
ただ彼の掌に載せられてスーと持ち上げられた時何だかフワフワした感じがあったばかりである
掌の上で少し落ちついて書生の顔を見たのがいわゆる人間というものの見始であろう
この時妙なものだと思った感じが今でも残っている
第一毛をもって装飾されべきはずの顔がつるつるしてまるで薬缶だ

グラフを描きたければ

matplotlib を使えばよい

簡単に描くために、pandas を導入して、その plot 機能を使う

```
%matplotlib inline                                  # 画面内に描画するおまじない
from collections import Counter
import pandas as pd                                 # pandasを使う

# 出てくる単語をリスト wlist にリストしてあるとしよう。
# つまり wlist の中身は ['吾輩', 'は', '猫', 'で', 'ある', '名前'...] のような感じ

c = Counter(wlist)     # Counterでwlist上の出現頻度を数える
df = pd.DataFrame(list(c.items()), columns=['word', 'count'])  # cをpandasへ移す
df = df.sort_values(by='count', ascending=False)           # countの大きい順に並べる
#print(df)
ax1 = df[:50].plot(x='word', kind='bar', figsize=(12, 9))       # dfを棒グラフに描く
plt.title('「吾輩は猫である」の単語出現頻度')
plt.xlabel('単語')
plt.ylabel('頻度')
# plt.savefig('wagahai-word.png')                   # 画像をファイルにセーブしたいとき
plt.show()
```