プログラミング演習の為のデータファイルを自分のディレクトリにコピー

cd : ホームディレクトリに移動

cd C_PROG : C_PROGディレクトリに移動

mkdir 05 : 05というディレクトリを作る

cd 05 : 05というディレクトリ内に移る

git clone https://github.com/NGS-maps/Practice03 : Practice03というディレクトリをgithubからダウンロード

```
cd Practice03
ls
```

この中に、hoge.txt ３つの.gbファイルalphabet.txt, numbers.txt, columns.txt, city_pop.dat, city_pop2.dat合計９個のファイルがある

まず、実習編の前半では３つの*.gb* ファイルについて考慮する
それぞれのファイルが大きいのでまずそのサイズを知る方法として普通に ls -l してみると、

```
ls -l *.gb
-rw-r--r--  1 kenske  kenske  3913605  May 26 11:18 Bacillus.gb
-rw-r--r--  1 kenske  kenske  3383496  May 26 11:17 Ecoli.gb
-rw-r--r--  1 kenske  kenske  4674633  May 26 11:18 Leuconostoc.gb
```

: Bacillus は3,913,605つまり約3.9MB
Ecoli は3,383,496つまり約3.4MB

ls -lh *.gb : 桁数が大きいと読みにくいので-h オプションで人に優しく*Human*

```
-rw-r--r--  1 kenske  staff   3.8M  May 26 11:18 Bacillus.gb
-rw-r--r--  1 kenske  staff   3.3M  May 26 11:17 Ecoli.gb
-rw-r--r--  1 kenske  staff   4.5M  May 26 11:18 Leuconostoc.gb
```

file :ファイルのタイプを調べる方法、
UNIX上のファイルにはテキストファイルとバイナリファイルがあり、
バイナリファイルはcat,less,more などで開けない

```
file *.gb
Bacillus.gb: ASCII text
Ecoli.gb:    ASCII text
Leuconostoc.gb:   ASCII text
```

いずれもASCIIテキストファイルであることが判るので、cat, less, moreなどで開いて見れる。
バイナリファイルをless, moreなどで開こうとするとすこし面倒なことになる
次にテキストファイルとしての行数を確認

wc :行数、語数、文字数をカウント *WordCount*

```
wc *.gb
 79713  179016 3913605 Bacillus.gb
 73008  125343 3383496 Ecoli.gb
 71788  291965 4674633 Leuconostoc.gb
224509  596324 11971734 total
```

数値は左から行数、語数、文字数である。とりあえず行数に注目すると、Bacillusは６万２百行すこし、Ecoliは７万６千７百行すこしあり、**cat**で表示しようとすると、スクリーン上を流れて、最後のニ十行程だけが残って見える。moreまたは**less**を用いると、始めの１ページ分のみが表示され、スペースバーで１ページずつ進めることが出来る。bと打つと前のページに戻る。/ の後に検索したい文字列を入れてreturnキーを押せばその文字列に一致する箇所まで

飛ぶこともできる。大文字のGを打つとファイルの末尾まで飛ぶ。最後までページを進めずに終了する場合は途中で q を押す(Quit)

```
% less Bacillus.gb
LOCUS       NC_000964             4215606 bp    DNA     circular BCT 31-MAR-2010
DEFINITION  Bacillus subtilis subsp. subtilis str. 168, complete genome.
ACCESSION   NC_000964
VERSION     NC_000964.3  GI:255767013
DBLINK      Project:76
KEYWORDS    complete genome.
SOURCE      Bacillus subtilis subsp. subtilis str. 168
ORGANISM  Bacillus subtilis subsp. subtilis str. 168
                        Bacteria; Firmicutes; Bacillales; Bacillaceae; Bacillus.
REFERENCE   1  (bases 1 to 4215606)
AUTHORS   Barbe,V., Cruveiller,S., Kunst,F., Lenoble,P., Meurice,G.,
             Sekowska,A., Vallenet,D., Wang,T., Moszer,I., Medigue,C. and
             Danchin,A.
                  ……
```

lessでスペースバーを打って、しばらく下の方へ下がっていくと、このファイルの大部分は次ページに示した様な、左側に、source, gene, CDS と並ぶ複数行の繰り返しであることがわかる。（CDSはrRNA, tRNAなどと入れ替わることもある）ここで、例えば

```
       gene   410..1750
```

は、対応するFASTAファイルの410塩基目から1750塩基目がある遺伝子であることを示している。
この様な行をカウントすれば、このファイルに含まれる遺伝子の数をカウントできる。

```
FEATURES             Location/Qualifiers
     source          1..4215606
                     /organism="Bacillus subtilis subsp. subtilis str. 168"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:224308"
     gene            410..1750
                     /gene="dnaA"
                     /locus_tag="BSU00010"
                     /db_xref="GeneID:939978"
     CDS             410..1750
                     /gene="dnaA"
                     /locus_tag="BSU00010"
                     /function="16.9: Replicate"
                     /protein_id="NP_387882.1"

     gene            1939..3075
                     /gene="dnaN"
                     /locus_tag="BSU00020"
                     /db_xref="GeneID:939970"
     CDS             1939..3075
                     /gene="dnaN"
                     /locus_tag="BSU00020"
                     /product="DNA polymerase III subunit beta"
                     /protein_id="NP_387883.1"
                     /db_xref="GI:16077070"
                     /db_xref="GeneID:939970"
     gene            3206..3421
                     /gene="yaaA"
                     /locus_tag="BSU00030"
                     /db_xref="GeneID:939444"
     CDS             3206..3421
                     /gene="yaaA"
                     /locus_tag="BSU00030"
                     /inference="ab initio prediction:AMIGene:2.0"
                     /note="Evidence 3: Function proposed based on presence of
                     conserved amino acid motif, structural feature or limited
```

ファイルの中で特定の文字列を含む行のみを抽出するにはgrepコマンドを使う。例えば

grep gene Bacillus.gb | less
とすればgeneという文字列を含む行が全て出力できる。たくさん出るのでlessにパイプでつないで１ページずつ見る

```
     gene            410..1750
                     /gene="dnaA"
                     /gene="dnaA"
                     multiple genes including itself."
     gene            1939..3075
                     /gene="dnaN"
                     /gene="dnaN"
     gene            3206..3421
```

ただし /gene=‘dnaA’の様な行も出力されてしまうので、前後をスペースで挟まれているgeneという文字列、を指定するためにシングルクォート（'）でスペースを含めた文字列を挟んで明示するために、以下の様にする

grep ‘ gene ‘ Bacillus.gb | less
^^ ^^ 'とgeneの前後の間にそれぞれ二個のスペース

```
     gene            410..1750
     gene            1939..3075
     gene            3206..3421
```

この結果をwcコマンドにパイプでつなぐことで gene エントリーの数を知ることができる。

grep ‘ gene ‘ Bacillus.gb | wc
^^ ^^

```
    4422    8844    193385
```

この結果が4422になることを確認したら、CDSと他の二つのファイルについてもカウントし、下の課題の表を埋める

課題1. 同様にして、Ecoli.gb, Leuconostocについて gene エントリーと CDS エントリーの数をカウントしメールの本文として以下の様な表を作り、、メールタイトル を、**名前_05_01** のようにして（**名前**は適宜置き換え）knakamura.maebit@gmail.com まで送ること

＝＝＝＝＝＝＝＝＝＝＝＝ 本文 ＝＝＝＝＝＝＝＝＝＝＝＝

氏名

	geneの数	CDSの数
Bacillus	4422	###
Ecoli	####	###
Leuconostoc	####	###

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

今までに学んだコマンド

wc ワードカウント： 行数 ・ 語数 ・ 文字数 を表示 wc ファイル名

grep 文字列検索 ： 特定の文字列を含む行のみを出力 grep 文字列 ファイル名
-v オプションをつけると含まない行のみを出力できる grep -v 文字列 ファイル名
文字列の部分はシングルクォート(')で囲むことで検索文字列に空白文字（スペース）を含められる grep ‘Hello world’ ファイル名

head ファイルの初めの何行かを表示する -10とすれば１０行 -100とすれば100行 head -10 ファイル名
tail ファイルの終わりの何行かを表示する tail -10 ファイル名

> リダイレクト ： コマンドの出力をファイルに流し込む >> と二回重ねることですでにあるファイルに書き足し（１回なら上書き）
grep CDS Bacillus.gb > Bacillus_CDS.gb
wc Bacillus.gb
とすればBacillus.gbの中の CDS を含む行数がカウントできる

| パイプ ： 前のコマンドの出力を次のコマンドに流し込む
grep CDS Bacillus.gb | wc
とすれば同じことが１行でできる

今日使ってみるコマンド

sort ソート：　　辞書順に行を並べ替える　　sort　ファイル名

```
% cat alphabet.txt
AA
AB
BA
BB
B
A
```

```
% sort alphabet.txt
A
AA
AB
B
BA
BB
```

```
% cat numbers.txt
11
12
21
22
2
1
```

```
% sort numbers.txt
1
11
12
2
21
22
```

辞書順の整列では数字の順序が予想通りに出来ない場合がある。数値順に並べたいときには -n オプションを　また逆順に並べたい場合には -r オプションを用いる。

```
% sort –n numbers.txt
1
2
11
12
21
22
```

```
% sort –r alphabet.txt
BB
BA
B
AB
AA
A
```

```
% sort –rn numbers.txt
22
21
12
11
2
1
```

複数の列（カラム）からなるファイルで、特定のカラムで並べ替えをしたい場合　-k#　（#に何カラム目かの数値、ここでは２）オプションを使う

```
% cat columns.txt
A  11
B  12
AA 21
ZZ 22
DD 11
X  2
Y  1
```

```
% sort –k2n columns.txt
Y  1
X  2
A  11
DD 11
B  12
AA 21
ZZ 22
```

awkというコマンドはテキストファイルからの情報抽出に非常に便利な多機能なコマンドである、ここでは特定のカラムを抽出する機能の簡単な使い方を示す。教科書8章の7節にも記述があるので、参考にされたい。 awkの基本的な使い方としては、
awk '{パターン　{アクション}　}　'　ファイル名
がある。他のフィルタコマンド同様ファイル名を省いて、標準入力を受け取ることもできる。awk の後に示す指示は '{　}' とシングルクォートと中括弧で囲む点に注意する。上記columns.txt のように複数のカラムからなるファイルについて左側の１番目のカラムが**$1**、右側２番めの数値のカラムが**$2**と表される。これにより、パターン無しで、awk '{print　$2} とすれば数値のカラムだけを抽出できる。また、パターンとして、数値が一定数より大きい場合、などとif文を用いて記述すると、２カラム目の数値が一定数より大きい行のみを抽出できる。行全体は **$0** で表される。

```
% awk '{print $2}' columns.txt
11
12
21
22
11
2
```

```
% awk '{if ($2 > 20) {print $0}}' columns.txt
AA 21
ZZ 22
```

課題１： alphabet.txt, numbers.txt, columns.txt について内容を確認してから上記の操作を試すこと

課題２：city_pop.dat は右に示すような、市の名前と人口が併記されたファイルである。
　　このファイルを1) 市名のアルファベット順、2) 人口の少ない順、3) 人口の多い順に並べ替えなさい

```
% cat city_pop.dat
MAEBASHI 342678
TAKASAKI 371127
SAITAMA 1232577
YOKOHAMA 3688624
NARA 364738
IKOMA 119258
```

課題３： city_pop2.dat は全国の市についてcity_pop.datと同様に市名と人口を記したファイルである
　　このファイルに基づいて、以下の市名と人口を調べ、
　　名前_05_02 （**名前**を自分の名で置き換え）というタイトルのメールの本文に記して送りなさい
　　(1) 最も人口の多い市名　　(2) 最も人口の少ない市名　　(3) アルファベット順で最初の市名
　　(4) アルファベット順で最後の市名　　(6)　人口が１００番目に多い市名　　（7）人口が500番目に多い市名
　　(8）アルファベット順で300番目の市名　(9）人口が１００番目に少ない市名

課題４： `citi_pop2.dat` から、awkを用いて人口が１００万人以上の市を抽出し、上から人口の多い順にソートしたファイルを作成し、**名前**_05_03 といったタイトルのメールの本文に、入力したコマンドを記述し、添付ファイルに作成したファイルを付けて提出しなさい