Exercises with regular expressions

8-10 Feb. 2017

- Checking your regular expressions in real time is really convenient. You can do so with https://regex101.com/ (https://regex101.com/) or a widget tool in mac computers.

Imagine you just got your (simple) gbk annotation file from RAST

```
....

FEATURES             Location/Qualifiers
     source          1..498
                     /mol_type="genomic DNA"
                     /genome_id="6666666.225369"
                     /organism="SOX smybiont"
     CDS             complement(104..454)
                     /db_xref="SEED:fig|6666666.225369.peg.189"
                     /translation="MRTAAWANGGANDICPSGFSVPTEAEITADTVHDGTYTGS
NDIT
                    NSATAFSSFLKIPVAGFRNRTNGALGSVGSGASLWSRSAGGANGRVLSVGSGYV
VFGS
                    VDRTGGFSVRCIKD"
                     /product="hypothetical protein"
                     /transl_table=11
BASE COUNT      137 a    126 c    120 g    115 t
ORIGIN
        1 gtggtgcgtt tatgatgtga atttaagata accatttcag gctaaagcca tgaaatggta
       61 aagtgacgaa agtgtacaaa tgaatcaaag tgtcacgctg cgcttaatcc ttaatgcaac
      121 gaacactaaa gccgccggtg cgatcaacgc tgccgaagac cacatagcca ctaccgacgc
      181 tcaaaacgcg accattcgcg ccaccagcag accgactcca caagctggcg ccagagccaa
      241 cactgccaag tgcgccattc gtacgattgc ggaagccagc aactgggatt ttaaggaaac
      301 tggagaaggc tgtggcgctg ttagtaatat cattgctgcc cgtgtaagta ccgtcatgaa
      361 cagtgtcagc agtaatttca gcttccgttg gtacactaaa gcctgatggg caaatgtcat
      421 tagcgccgcc attcgcccaa gcggctgtgc gcaatgcacc actatcgtct atattgttgc
      481 cgtcttgagt gttatttt
//
```

Your tasks:

- Change the tag db_xref to locus_tag
- Make the ID of the protein sho rter You have the format SEED:fig|6666666.225369.peg.189 and y ou would like to change it to the format SOX189

- Fetch the protein sequence
- Fetch the nucleotide sequen ce

How to open a file? (for now)

- We will learn later the details of how to read a file. For now, we will use a rather simplistic syntax to pipe text files to our perl scripts:

```
while (<>){
    # print "$_";
}
```

- To run the script you can redirect the contents of a file to execute them in perl like this:

```
cat file_name.txt | myperlscript.pl
```

Let’s create the regular expression to get db_xref

```
In [4]: %%perl
        use strict;

        my $line;
        my $locus_tag;

        while ($line = <>){   #While the file is being read. Perl reads the file lin
        e by line!
            if ($line=~ /(.*)\/db_xref\=.*peg\.(\S+)\"/ ){
                $line = $1 . "locus_tag=\"SOX" . $2 ;
                $locus_tag = "SOX" . $2;
            }
            print $line;
        }
```

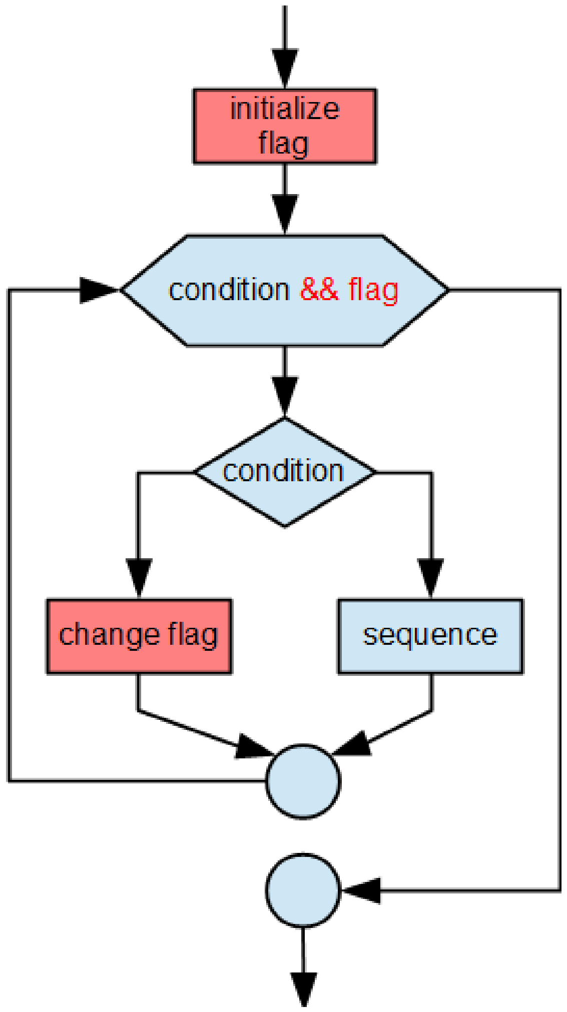
Now let’s get the protein sequence

Tips and tricks:

- Tip 1: You can use a flag to signalize that the file section containing the protein sequence is over (or as a mather of fact, you can use flags for any changing condition)

- It's not this kind of flag!

- Flags for programming



- Tip 2: Two string can be concatenated like this

```
my $this = 'This';
my $that = 'That';
$string = join("", $this, $that);
# $string now contains 'ThisThat'.
```

```
/translation="MRTAAWANGGANDICPSGFSVPTEAEITADTVHDGTYTGSNDIT
NSATAFSSFLKIPVAGFRNRTNGALGSVGSGASLWSRSAGGANGRVLSVGSGYVVFGS
VDRTGGFSVRCIKD"
/product="hypothetical protein"
```

We need a flag to indicate that the section in the file with protein information is starting and ending!

- What patterns can you use?

```
In [ ]: %%perl
        my $line;
        my $flag=0;
        my $seq;

        while ($line = <>){   #While the file is being read. Perl reads the file lin
        e by line!
            chomp $line;
            if ($line=~/(.*\/translation=\")(\S+)\"*/ ){
                $flag =1;
                $seq=$2;
            }
            if (($flag== 1) && ($line!~/.*\//) ){
                 $line=~ s/ //g;
                 $seq = join ("", $seq, $line);
            }
            if ($line=~/\/product/){
                $flag =0;
            }
        }

        print "The protein sequence is $seq \n" ;
```

And finally, let’s get the nucleotide sequence

```
ORIGIN
        1 gtggtgcgtt tatgatgtga atttaagata accatttcag gctaaagcca tgaaatggta
       61 aagtgacgaa agtgtacaaa tgaatcaaag tgtcacgctg cgcttaatcc ttaatgcaac
      121 gaacactaaa gccgccggtg cgatcaacgc tgccgaagac cacatagcca ctaccgacgc
      181 tcaaaacgcg accattcgcg ccaccagcag accgactcca caagctggcg ccagagccaa
      241 cactgccaag tgcgccattc gtacgattgc ggaagccagc aactgggatt ttaaggaaac
      301 tggagaaggc tgtggcgctg ttagtaatat cattgctgcc cgtgtaagta ccgtcatgaa
      361 cagtgtcagc agtaatttca gcttccgttg gtacactaaa gcctgatggg caaatgtcat
      421 tagcgccgcc attcgcccaa gcggctgtgc gcaatgcacc actatcgtct atattgttgc
      481 cgtcttgagt gttatttt
//
```

```
In [1]: %%perl
        my $line;
        my $flag=0;
        my $seq;
        my @seq;
        my $nt_seq="";
        my $locus_tag;

        while ($line = <>){   #While the file is being read. Perl reads the file lin
        e by line!
            chomp $line;
            if ($line=~ /\d+\s+(\D+\s*)/ ){   ##will fetch the nucleotide sequences o
        f the gbk file
                $nt_seq = join ("", $nt_seq, $1);
            }
        }

        $nt_seq =~ s/ //g;


        print "The nucleotide sequence is $nt_seq \n" ;
```

```
The nucleotide sequence is
```

- The code above is written assuming that there is only one scaffold
- What would you modify if you have multiple scaffolds?

Homework: Work with you nucleotide sequence

- Change lower case to upper case
- Reverse complement the sequence

Tips:

- Use the function reverse
- Use tr