Chandrasekhar Limit

Decaying Omega minus particles

Reading a FASTA file with Python by Giuseppe

Since one of my tutoring duties is related to basic programming tasks for bioinformatics, I'm starting from there.

A good starting point is the exercise "Build a dictionary containing sequences from a FASTA file".

The FASTA file format (https://en.wikipedia.org/wiki/FASTA_format) is a text based representation of a biological sequences. A FASTA file contains one or multiple entries: each entry consists of an header like, starting with the character > (greater than), and a set of lines (each up to 60 characters long) containing the sequence.

Of course you can use one of the already made libraries for the task: Bio.SeqIO.to_dict() or a loop using other methods from BioPython (http://biopython.org/wiki/SeqIO#Sequence_Input) may be faster. However, this exercise is perfect in teaching how to think like a computer scientist.

First, the problem must be solved without using code: if you don't know how to solve a problem, you can't write a program that performs the solution.

In general, the FASTA file contains multiple entries, so we need a way to track the current entry name. We must distinguish the header from the sequence content.

Building from the inside-out:

```
1 #distinguish header from sequence
2 if line[0]=='>': #or line.startswith('>')
3     #it is the header
4 else:
5     #it is sequence
```

In the first case, we need to store the new sequence name and create its entry in the dictionary:

```
1 #distinguish header from sequence
2 if line[0]=='>': #or line.startswith('>')
3     #it is the header
```

```
        name = line[1:] #discarding the initial >
        seqs[name] = ''
    else:
        #it is sequence
```

We wrap everything inside a for loop, i.e. we are reading one line at the time:

```
for line in f:
    #let's discard the newline (if any)
    line = line.rstrip()
    #distinguish header from sequence
    if line[0]=='>': #or line.startswith('>')
        #it is the header
        name = line[1:] #discarding the initial >
        seqs[name] = ''
    else:
        #it is sequence
```

Now we have different ways to manage the sequence:

1. build a string and update the dictionary when a new header or the end of file is found
2. update the dictionary entry belonging to the current name

In the first case:

```
name = None
s = ''
for line in f:
    #let's discard the newline at the end (if any)
    line = line.rstrip()
    #distinguish header from sequence
    if line[0]=='>': #or line.startswith('>')
        #it is the header
        #if this is not the first sequence (i.e. we already read one)
        if name:
            seqs[name]=s
        name = line[1:] #discarding the initial >
        seqs[name] = ''
        s = ''
    else:
        #it is sequence
        s = s + line
#we must keep in mind the last entry: it was not saved
#since we didn't find a new header
if name: #just checking against an empty file
    seqs[name]=s
```

In the second approach:

```
name = None
for line in f:
    #let's discard the newline at the end (if any)
    line = line.rstrip()
    #distinguish header from sequence
    if line[0]=='>': #or line.startswith('>')
        #it is the header
        name = line[1:] #discarding the initial >
        seqs[name] = ''
    else:
        #it is sequence
        seqs[name] = seqs[name] + line
```

F can be a file object from **open()** or a list of lines from **readlines()** (I discourage the second one).

This entry was posted in Post in English, Turing machines and tagged fasta file, programming, python.