Lab 7

Modules And Files

Sup'Biotech 3

Python

Pierre Parutto

November 23, 2016

Preamble

Document Property

Authors	Pierre Parutto
Version	1.0
Number of pages	9

Contact

Contact the assistant team at: supbiotech-bioinfo-bt3@googlegroups.com

Copyright

Contents

1 Introduction

In this lab we will manipulate files

2 Warm-up - Modules

2.1 Testing your code

Put the code:

```
my_u0 = 3

def my_seq(n, u0):
    if n == 0:
        return u0
    return 3 * my_seq(n-1, u0) + 6
```

in the file *my_seq.py*.

Import the function `my_seq` and the variable `my_u0` in the file *my_seq_test.py*. Use the `assert` statement to test the function. **Try the three different ways to import the code.**

Correction:

```
from my_seq import my_seq

assert my_seq(1, 3) == 15
assert my_seq(3, 1) == 105
```

3 Reading Files

For each function, use all three reading methods.

3.1 DNA File

Write a function `read_dna(fname: str) -> list` that reads the content of the file named `fname` that contain one DNA sequence per line, and returns the DNA sequences in a list.

Example

With the file *0.dna*:

```
```

```
>>> read_dna("0.dna")
[]
```

With the file *a.dna*:

```
GTGTGTGTCA
```

```
>>> read_dna("a.dna")
["GTGTGTGTCA"]
```

With the file *b.dna*:

```
ATGCT
CACAA
GCGCA
```

```
>>> read_dna("b.dna")
["ATGCT", "CACAA", "GCGCA"]
```

Correction:

```python
#with readline
def read_dna(fname):
    res = []
    f = open(fname, "r")
    line = f.readline()
    while line != "":
        res.append(line.rstrip("\n"))
        line = f.readline()
    f.close()
    return res

#with readlines
def read_dna(fname):
    res = []
    f = open(fname, "r")
    for e in f.readlines():
        res.append(e.rstrip("\n"))
    f.close()
    return res

#with a for loop
def read_dna(fname):
    res = []
    f = open(fname, "r")
    for line in f:
        res.append(line.rstrip("\n"))
    f.close()
    return res
```

3.2 Fasta File

The fasta format allows to store DNA sequences and their ids, the format is the following:

```
> id1
seq1
> id2
```

```
seq2
...
```

The line starting with ">␣" gives the id of the sequence, the next line is the associated sequence.

Write a function `read_fasta(fname: string) -> list` that returns the list of tuples `(id, seq)` contained in the fasta file `fname`.

Example

With the file `f.fasta`:

```
```

```
>>> read_fasta("f.fasta")
[]
```

With the file `ff.fasta`:

```
> OZDJKOIFJZ
CGAGA
```

```
>>> read_fasta("ff.fasta")
[("OZDJKOIFJZ", "CGAGA")]
```

With the file `fff.fasta`:

```
> seq1
ATTATAGA
> seq2
ATT
```

```
>>> read_fasta("fff.fasta")
[("seq1", "ATTATAGA"), ("seq2", "ATT")]
```

Correction:

```
def read_fasta(fname):
    res = []
    f = open(fname, "r")
    id = None
    for line in f:
        if line[0:2] == ">␣":
            id = line[2:].rstrip("\n")
        else:
            res.append((id, line.rstrip("\n")))
    f.close()
    return res
```

3.3 One Every Two

Write a function `one_over_two(fname: str) -> list` that reads one line every two of the file named `fname`.

Example

With the file *stairway.txt*:

```
And if you listen very hard
The tune will come to you at last.
When all are one and one is all
To be a rock and not to roll.
```

```
>>> one_over_two("stairway.txt")
["And␣if␣you␣listen␣very␣hard", "When␣all␣are␣one␣and␣one␣is␣all"]
```

With the file *tornado.txt*:

```
Who's to say,what's for me to say...be...do
Cause a big nothing it'll be for me
The land of opportunity
The golden chance for me
My future looks so bright
Now I think I've seen the light
```

```
>>> one_over_two("tornado.txt")
["Who's␣to␣say,what's␣for␣me␣to␣say...be...do", "The␣land␣of␣opportunity", \
"My␣future␣looks␣so␣bright"]
```

Correction:

```
def one_over_two(fname):
    res = []
    f = open(fname, "r")
    i = 0
    for line in f:
        if i % 2 == 0:
            res.append(line.rstrip("\n"))
        i = i + 1
    f.close()
    return res
```

3.4 Read Sequence

Write a function `read_seq(fname: str) -> list` that reads the integer sequence from the file `fname`, where the numbers are separated by coma, and returns it.

Example

With the file *int_seq1.txt*:

```
1,2,2,3,4
```

```
>>> read_seq("int_seq1.txt")
[1,2,2,3,4]
```

With the file *int_seq2.txt*:

```
1,2,2,3,4,3,4,5,5
```

```
>>> read_seq("int_seq2.txt")
[1,2,2,3,4,3,4,5,5]
```

Correction:

```
def read_seq(fname):
    res = []
    f = open(fname, "r")
    line = f.readline()
    for e in line.rstrip("\n").split(","):
        res.append(int(e))
    f.close()
    return res
```

4 Writing files

4.1 Writing DNA Sequences

Write a function `write_DNA(seqs: list, fname: str) -> None` that write the sequences in the list `seqs` into the file named `fname`, one sequence per line.

Example

```
>>> write_DNA(["ATATATAGA"], "dna_seqs.dna")
```

Creates the file *dna_seqs.dna*:

```
ATATATAGA
```

```
>>> write_DNA(["ATA", "GCGAG"], "dna_seqs1.dna")
```

Creates the file *dna_seqs1.dna*:

```
ATA
GCGAG
```

Correction:

```
def write_DNA(seqs, fname):
    f = open(fname, "w")
    for s in seqs:
        f.write(s + "\n")
    f.close()
```

4.2 Write Sequence

Write a function `write_sequence(seq: list, fname: str) -> None` that writes the integer sequence given in `seq` in the file named `fname` with each integer separated by coma.

Example

```
>>> write_sequence([1,2,3], "yolo.txt")
```

Creates the file *yolo.txt*:

```
1,2,3
```

```
>>> write_sequence([1,-1,-3,2], "yolo.yolo")
```

Creates the file *yolo.yolo*:

```
1,-1,-3,2
```

Correction:

```
def write_sequence(seq, fname):
    f = open(fname, "w")
    i = 0
    while i < len(seq)-1:
        f.write(str(seq[i]) + ",")
        i = i + 1
    f.write(str(seq[i]) + "\n")
    f.close()
```

4.3 Write Fasta

Write a function `write_fasta(seqs: list, fname: str) -> None` that writes in the fasta format, the sequences from the list of tuples `(id, seq)`, `seqs` in the file named `fname`.

Example

```
>>> write_fasta([("seq1", "ATTAGA")], "1.fasta")
```

Creates the file *1.fasta*:

```
> seq1
ATTAGA
```

```
>>> write_fasta([("s1", "TATA"), ("s2", "GAC")], "2.fasta")
```

Creates the file *2.fasta*:

```
> s1
TATA
> s2
GAC
```

Correction:

```
def write_fasta(seqs, fname):
    f = open(fname, "w")
    for e in seqs:
        f.write("> " + e[0] + "\n")
        f.write(e[1] + "\n")
    f.close()
```