PROXECTO

APLICACIÓN PARA MANIPULAR FICHEIROS EN FORMATO FASTA

PARTE 1

Versión documento: 2025.04.01.v1

Táboa de contidos

1. Descrición do proxecto

O obxectivo deste proxecto é a creación dunha aplicación para o manexo de ficheiros en formato FASTA[1], un formato de texto para representar secuencias de ADN ou proteínas moi utilizado en bioinformática.

Un ficheiro FASTA almacena información de unha ou varias secuencias. Cada secuencia está composta por unha liña de cabeceira que comeza polo carácter "maior que" (`'>'`) e por unha ou máis liñas que conteñen o texto coa secuencia de ADN ou proteínas correspondente. A cabeceira proporciona o identificador da secuencia.

Vexamos un exemplo de contido en formato FASTA:

```
>S1
ACTG
```

Neste caso o ficheiro contén unha única secuencia cuxo identificador é `'S1'` e a secuencia de ADN é `'ACTG'`.

Como se comentou anteriormente, un ficheiro pode ter varias secuencias e, á súa vez, cada secuencia pode ter o seu contido espallado en varias liñas. Vexamos outro exemplo:

```
>S1
ACTG
AT
>S2
AACC
GC
```

Neste caso, representado na Figura 1, o ficheiro contén dúas secuencias cuxo identificadores son `'S1'` e `'S2'`, e a secuencias de ADN son `'ACTGAT'` e `'AACCGC'` respectivamente (como podes observar, as liñas que seguen á cabeceira xúntanse nunha única cadea que representa a secuencia completa).

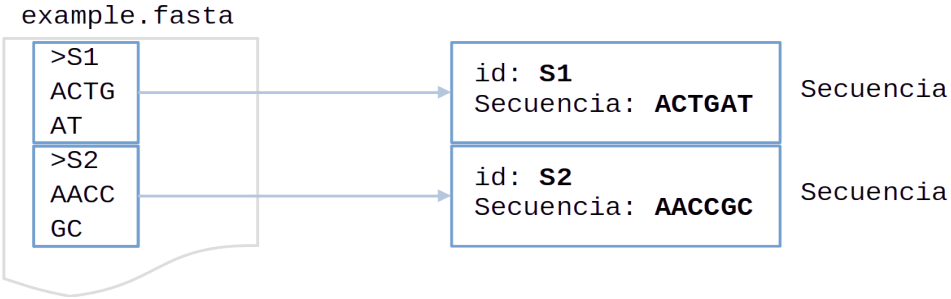

Figura 1: Estruturación da información contida nun ficheiro en formato FASTA.

1 https://en.wikipedia.org/wiki/FASTA_format

No proxecto crearás:

- Unha API (funcións, clases, módulos, etc.) para representar e manipular secuencias, incluíndo funcións relacionadas coa lectura e escritura de ficheiros en formato FASTA. Chamaremos a esta parte o "núcleo" ou *core* do proxecto.
- Varios *scripts* que ofrecerán funcionalidades concretas aos usuarios finais. Estes *scripts* empregarán a API creada anteriormente para levar a cabo o seu cometido. Esta parte será a interface de usuario por liña de comandos (CLI).

A continuación descríbense, paso a paso, os distintos exercicios que compoñen a primeira entrega do proxecto. Tanto na primeira como na segunda entrega, nalgúns exercicios crearás as funcionalidades básicas do *core* e noutros crearás os *scripts* que os usuarios poderán utilizar para manipular os seus ficheiros FASTA.

2. Cargar ficheiros

Comeza creando o *core* para a lectura de ficheiros, é dicir, as funcións/clases necesarias para realizar a carga dos ficheiros `test_1.fasta` e `test_2.fasta` (que podes atopar dentro da carpeta `test_data`).

Comeza co ficheiro `test_1.fasta`, que ten un formato "simple" en que cada secuencia ocupa exactamente dúas liñas (a primeira comeza por '>' e contén o identificador e a segunda contén a propia secuencia). O obxectivo é crear unha función que reciba como argumento a ruta ao ficheiro e que devolva unha **lista** de **obxectos** na que cada obxecto represente unha secuencia (identificador e a propia secuencia correspondente en cada un). Suxírese crear unha **clase** para representar obxectos de tipo "Secuencia" (`class Sequence`) onde almacenes a información da que se compón cada secuencia (identificador e a cadea da secuencia en si).

Para probar o programa, podes facer un `print(seq)` de cada secuencia lida para comprobar que a carga se fixo correctamente. Lembra probar sempre programa dentro dun bloque `if __name__ == '__main__'` para que estas probas non se executen cando uses o módulo desde outros *scripts* ou módulos.

Unha vez o teñas feito, modifica a lóxica (é dicir, o algoritmo) do cargador de ficheiros para poder cargar tamén o ficheiro `test_2.fasta`, no que algunhas secuencias conteñen máis de dúas liñas. A idea é xuntar as liñas nunha única cadea que represente o contido da

secuencia. Remata o exercicio probando a cargar os ficheiros `test_3.fasta` e `test_4.fasta`.

3. Formatar secuencias

O seguinte paso é seguir aumentando as funcionalidades dispoñibles no *core* e agora debes permitir que as secuencias sexan formatadas de acordo con dous parámetros:

- `case`: que permite especificar o tipo de letra (minúscula, maiúscula ou manter o orixinal). Suxestión: emprega un enumerado para almacenar estes posibles valores.
- `max_length`: lonxitude máxima que pode ocupar cada liña da secuencia (sen contar a cabeceira). Á hora de mostrar a secuencia por consola (ou de escribila nun ficheiro), o valor `0` significa que a liña pode ocupar todo o ancho que queira e un valor `X` significa que cada liña só pode ter `X` caracteres como máximo.

Tomando como exemplo a primeira secuencia do ficheiro `test1.fasta`:

```
>S1
ACTG
```

Se se indica `case = minúscula` (con `max_length = 0`), o formato obtido para a secuencia anterior debería ser o seguinte:

```
>S1
actg
```

Se se indica `max_length = 2` (con `case = manter o formato orixinal`) significa que cada liña da secuencia pode ter como máximo 2 letras, co cal hai que "partir" a secuencia en liñas. No exemplo obteríase:

```
>S1
AC
TG
```

Polo tanto, neste apartado deberás crear as funcións/clases necesarias para **formatar** un **obxecto** que represente unha **secuencia** (feito no apartado anterior) **creando** unha **cadea** que o represente de acordo cos parámetros anteriores. Proba o código que crees empregando **datos fixos**, é dicir, sen empregar o código creado no exercicio anterior.

Unha aclaración importante é que estas funcións/clases non modifican os obxectos secuencia creados, tan só crean representacións textuais dos mesmos empregando cadeas

de caracteres. É dicir, este exercicio é **totalmente independente** do anterior.

Ademais, considera que o formato por defecto debe ser o seguinte:

- `case = manter o formato orixinal`.
- `max_length = 0`, o que significa que a liña pode ocupar todo o ancho que queira (non é necesario dividila en varias liñas).

4. Escritura de ficheiros

Comeza creando o *core* para a escritura de ficheiros, é dicir, as funcións/clases necesarias para escribir unha lista de secuencias (obxectos "Secuencia" como os teñas definidos no proxecto) nun ficheiro. Para realizar a escritura emprega un formatador de secuencias, un obxecto/función encargado de converter unha secuencia a unha cadea de texto, que é o que debes escribir no ficheiro. Por tanto, neste exercicio:

- Debes procesar unha lista de secuencias.
- Para cada secuencia debes obter a súa representación como cadea de texto empregando o formatador creado anteriormente.
- Debes escribir ditas representación textuais nun ficheiro de texto.

Proba o código que crees empregando datos fixos e escribindo os ficheiros para comprobar que todo está funcionando ben.

5. Script para cambiar o formato un ficheiro

Neste paso crearás o primeiro script para usuarios finais, que servirá para cambiar o formato dun ficheiro dado como entrada e xerar un novo ficheiro. A idea é empregar todas as funcións e tipos de datos creados nos apartados anteriores desde este script.

O script debe chamarse `fasta_format.py` e debe poder executarse da seguinte maneira na liña de comandos:

```
fasta_format.py --input=/path/to/input.fasta --output=/path/to/output.fasta --case=lower/upper/original --max_length=0
```

Os parámetros `--case` e `--max_length` son opcionais, de xeito que se non se reciben deben tomar os valores `original` e `0` respectivamente.

Os argumentos que recibe por liña de comandos un script en Python están na lista `sys.argv`

(`import sys`). Para a realización do proxecto, proporciónase o módulo `argsparser.py` que define a función `parse_args`. Esta función procesa a lista (`sys.argv`) e crea un dicionario no que cada clave é cada nome de parámetro (sen os `--`) e ten asociado o seu valor correspondente. Ante a invocación anterior, o método de uso sería:

```
args = parse_args(sys.argv[1:])
```

De xeito que o dicionario args contería:

```
{'input': '/path/to/input.fasta', 'output': '/path/to/output.fasta', 'case':
'lower/upper/original', 'max_length': '0'}
```

Este script ten que facer o seguinte:

- Cargar o ficheiro de entrada usando as funcións do apartado 2. Neste punto terás unha variable que apunte a unha lista de secuencias.
- Crear un formatador de secuencias cos parámetros axeitados usando o código do apartado 3.
- Escribir a lista de secuencias cargada no ficheiro de saída usando o código do apartado 4 e o formatador anterior.

Configurar Visual Studio Code para almacenar os comandos de proba

Para probar os scripts que vaias desenvolvendo, coma o do apartado anterior, resulta útil gardar os comandos completos nalgún lugar, de xeito que se poidan executar en calquera momento facilmente.

Unha boa forma de facelo é crear "Run tasks" en Visual Studio Code, que non son máis que listados de tarefas (por exemplo: comandos na terminal) que se poden executar a través dun menú.

Para crear unha tarefa para o apartado anterior, segue estes pasos:

1. Vai ao menú "Terminal / Configure tasks".
2. Cando se abra o menú, preme na opción "Create tasks.json file from template" a continuación na opción "Others".
3. Aparecerá o ficheiro ".vscode/tasks.json" no que podes engadir as túas tarefas, por exemplo:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "test fasta_format.py [1]",
            "type": "shell",
            "command": "/usr/bin/python3
/home/user/Projects/FASTA/fasta_format.py --input=test_data/test_1.fasta --
output=/tmp/test-out.fasta --max-length=2 --case=3 && cat /tmp/test-out.fasta",
            "problemMatcher": []
        }
    ]
}
```

4. Agora poderás executar a tarefa sempre que queiras usando o menú “Terminal / Run task”.