swisstake: Parsing SwissProt records

> “Without requirements or design, programming is the art of adding bugs to an empty text file." - Louis Srygley

Create a Python program called `swisstake.py` that processes a SwissProt-formatted file. The program’s parameters should be:

- A single positional argument for the input file.
- `-k|--keyword` argument of the keyword to match in the "keyword" field of the input record in order to determine which sequences to "take" (hence the name). This is **required**.
- `-s|--skiptaxa` option to list one or more taxa that will be used to skip/reject records.
- `-o|--output` option to where to write the output in FASTA format (default `out.fa`).

Here is the expected usage:

```
$ ./swisstake.py -h
usage: swisstake.py [-h] -k keyword [-s [taxa [taxa ...]]] [-o FILE] FILE

Filter SwissProt file for keywords, taxa

positional arguments:
  FILE                  SwissProt file

optional arguments:
  -h, --help            show this help message and exit
  -k keyword, --keyword keyword
                        Keyword to take (default: None)
  -s [taxa [taxa ...]], --skiptaxa [taxa [taxa ...]]
                        Taxa to skip (default: None)
  -o FILE, --outfile FILE
                        Output filename (default: out.fa)
```

SwissProt format

Read the docs on the SwissProt format:

```
http://athina.biol.uoa.gr/FT/format_SwissProt.html
```

We will use the `Bio.SeqIO` to parse the file. Read the docs on the `Bio.SwissProt.Record` package:

```
https://biopython.org/DIST/docs/api/Bio.SwissProt.Record-class.html
```

Look at the example in `inputs/single.txt` and compare what we see there with the result of parsing the file like so:

```
>>> from Bio import SeqIO
>>> parser = SeqIO.parse('./inputs/single.txt', 'swiss')
>>> rec = list(parser)[0]
>>> print(rec)
ID: P13813
Name: 110KD_PLAKN
Description: RecName: Full=110 kDa antigen; AltName: Full=PK110; Flags: Fragment;
Database cross-references: EMBL:M19152, PIR:A54527, ProteinModelPortal:P13813,
GeneDB:PKNH_1448000.1:pep, eggNOG:ENOG410JB7W, eggNOG:ENOG41118GC
Number of features: 17
/accessions=['P13813']
/protein_existence=2
/date=01-JAN-1990
/sequence_version=1
/date_last_sequence_update=01-JAN-1990
/date_last_annotation_update=20-JAN-2016
/entry_version=42
/organism=Plasmodium knowlesi
/taxonomy=['Eukaryota', 'Alveolata', 'Apicomplexa', 'Aconoidasida', 'Haemosporida',
'Plasmodiidae', 'Plasmodium', 'Plasmodium (Plasmodium)']
/ncbi_taxid=['5850']
/references=[Reference(title='Cloning and characterization of an abundant Plasmodium
knowlesi antigen which cross reacts with Gambian sera.', ...)]
/keywords=['Malaria', 'Repeat']
Seq('FNSNMLRGSVCEEDVSLMTSIDNMIEEIDFYEKEIYKGSHSGGVIKGMDYDLED...VEP', ProteinAlphabet())
```

Filtering the records

You can use a `for` loop as with FASTA files to read each record from the input file:

```
for rec in SeqIO.parse(args.file, "swiss"):
    print(rec)
```

The goal of the program is to:

1. Take any record where the `keywords` overlaps one of the `--keyword` arguments
2. Skip any record where the `taxonomy` overlaps any of the `--taxa` arguments

Note that these pieces of information are contained in the `rec.annotations` attribute which is a `dict`:

```
{'accessions': ['P13813'],
 'date': '01-JAN-1990',
 'date_last_annotation_update': '20-JAN-2016',
 'date_last_sequence_update': '01-JAN-1990',
 'entry_version': 42,
 'keywords': ['Malaria', 'Repeat'], ①
 'ncbi_taxid': ['5850'],
 'organism': 'Plasmodium knowlesi',
 'protein_existence': 2,
 'references': [Reference(title='Cloning and characterization of an abundant
Plasmodium knowlesi antigen which cross reacts with Gambian sera.', ...)],
 'sequence_version': 1,
 'taxonomy': ['Eukaryota', ②
              'Alveolata',
              'Apicomplexa',
              'Aconoidasida',
              'Haemosporida',
              'Plasmodiidae',
              'Plasmodium',
              'Plasmodium (Plasmodium)']}
```

① The `keywords` is a `list` of `str` values.

② The `taxonomy` is a `list` of `str` values.

NOTE There is no guarantee that the `keywords` and `taxonomy` exist for every record, so proceed with caution.

I recommend you use a `set()` to determine if there is overlap between the keywords and taxonomy values. For instance, the `inputs/single.txt` record has the following `taxonomy`:

```
>>> taxonomy = ['Eukaryota', 'Alveolata', 'Apicomplexa', 'Aconoidasida',
'Haemosporida', 'Plasmodiidae', 'Plasmodium', 'Plasmodium (Plasmodium)']
```

We can create a new `set` called `taxa` of the lowercased values:

```
>>> taxa = set(map(str.lower, taxonomy))
>>> taxa
{'haemosporida', 'aconoidasida', 'plasmodium (plasmodium)', 'alveolata',
'plasmodiidae', 'apicomplexa', 'plasmodium', 'eukaryota'}
```

Assume that we're running our program with `--skip` values of "plasmodium" and "Homo sapiens". We can create a similar `set` with those lowercased values:

```
>>> skip = set(map(str.lower, ['plasmodium', 'Homo sapiens']))
>>> skip
{'plasmodium', 'homo sapiens'}
```

The `set.intersection()` method will return those values that are shared:

```
>>> skip.intersection(taxa)
{'plasmodium'}
```

You can do likewise to find intersections between the record’s (possible) `keywords`. Remember:

1. If there is an intersection in the `taxonomy` from `rec.annotation` and `args.skiptaxa`, then **skip** the record.
2. If there is an intersection in the `keywords` from `rec.annotation` and `args.keyword`, then **take** the record.

Output file/format

The output should be in FASTA format, and the default output file name should be "out.fa." You can use the `SeqIO.write()` function like so:

```
if ...:
    SeqIO.write(rec, args.outfile, 'fasta')
```

The `args.outfile` parameter should be defined as a writable file for this code to work properly! If you define it as a `str`, then `SeqIO.write()` will **overwrite the output file** with each record and your file will end up with the *last* sequence.

If you run your program like so, you should see the following result:

```
$ ./swisstake.py inputs/swiss2.txt -k "complete proteome" -s Metazoa FUNGI
viridiplantae
Done, skipped 14 and took 1. See output in "out.fa".
```

We can use this to verify the number of records in "out.fa":

```
$ grep '^>' out.fa | wc -l ①
       1
```

① Use `grep` (global regular expression print — maybe?). This will test each line of the input from with the pattern `^>` (a greater than sign anchored to the start of the line). When a line matches the pattern, it will be printed. This will find the header lines in a FASTA file. Pipe that to `wc -l` to count the number of lines.

Here is a different example:

```
$ ./swisstake.py inputs/swiss2.txt -k "complete proteome" -s METAZOA fungi
Done, skipped 13 and took 2. See output in "out.fa".
```

And we can verify the correct number of sequences in the output file:

```
$ grep '^>' out.fa | wc -l
       2
```

Test Suite

A passing test suite looks like this:

```
$ make test
pytest -xv --disable-pytest-warnings test.py
============================ test session starts =============================
...

test.py::test_exists PASSED                                             [ 12%]
test.py::test_usage PASSED                                              [ 25%]
test.py::test_bad_file PASSED                                           [ 37%]
test.py::test_missing_keyword PASSED                                    [ 50%]
test.py::test_02 PASSED                                                 [ 62%]
test.py::test_03 PASSED                                                 [ 75%]
test.py::test_04 PASSED                                                 [ 87%]
test.py::test_01 PASSED                                                 [100%]

======================== 8 passed, 1 warning in 1.60s ========================
```