De Bruijn Graphs in Python

We will find paths through sequences that could aid in assembly (cf http://rosalind.info/problems/grph/). For this exercise, we will only attempt to join any two sequences together. To do this, we will look at the last `k` characters of every sequence and find where the first `k` character of a *different* sequence are the same.

For example, in this file:

```
$ cat sample1.fa
>Rosalind_0498
AAATAAA
>Rosalind_2391
AAATTTT
>Rosalind_2323
TTTTCCC
>Rosalind_0442
AAATCCC
>Rosalind_5013
GGGTGGG
```

If `k` is 3, then the last 3-mer of sequence 498 is "AAA" which is also the first 3-mer of 2391 and 442. "TTT" ends 2391 and starts 2323, so the graphs we could create from 3-mers would be:

```
$ ./grph.py -k 3 sample1.fa
Rosalind_0498 Rosalind_2391
Rosalind_0498 Rosalind_0442
Rosalind_2391 Rosalind_2323
```

You will write a Python program called `grph.py` which will take a `-k|--overlap` option with a default value of `3` and a single positional argument of an input file which will be in FASTA format. I would recommend you read all the sequences and build two data structures that hold the k-mers at the beginnings and ends of your sequences. You should go through all the ending kmers and see if there are any sequences that begin with that string. It does not matter what order you emit the pairs as they will be sorted on the command line for the tests.

Similar to the HAMM assignment, `test.py` will `import grph` and specifically test for a `grph.find_kmers` function which should be defined as taking a string and a integer representing `k` for the returned kmers. If you inspected the test, you should see how to write the function and what it should return:

```
def test_find_kmers():
    """Test the `find_kmers` function"""
    assert grph.find_kmers('ACTG', 2) == ['AC', 'CT', 'TG']
    assert grph.find_kmers('ACTG', 3) == ['ACT', 'CTG']
    assert grph.find_kmers('ACTG', 4) == ['ACTG']
```

Note that the code you need has already been written (https://github.com/hurwitzlab/biosys-analytics/blob/master/lectures/08-python-patterns/python-common-patterns.md#extract-k-mers-from-a-string).

Expected behavior

```
$ ./grph.py
usage: grph.py [-h] [-k int] str
grph.py: error: the following arguments are required: str
$ ./grph.py -h
usage: grph.py [-h] [-k int] str

Graph through sequences

positional arguments:
  str                   FASTA file

optional arguments:
  -h, --help            show this help message and exit
  -k int, --overlap int
                        K size of overlap (default: 3)
$ ./grph.py sample1.fa
Rosalind_0498 Rosalind_2391
Rosalind_0498 Rosalind_0442
Rosalind_2391 Rosalind_2323
[cholla@~/work/worked_examples/13-grph]$ ./grph.py -k 4
usage: grph.py [-h] [-k int] str
grph.py: error: the following arguments are required: str
$ ./grph.py -k 4 sample1.fa
Rosalind_2391 Rosalind_2323
$ ./grph.py -k 5 sample2.fa
Rosalind_7364 Rosalind_6988
Rosalind_5867 Rosalind_6515
Rosalind_7148 Rosalind_6515
Rosalind_9983 Rosalind_2130
Rosalind_3470 Rosalind_0281
Rosalind_0378 Rosalind_7135
Rosalind_2095 Rosalind_0045
Rosalind_2840 Rosalind_4493
Rosalind_4581 Rosalind_4374
Rosalind_1144 Rosalind_5870
Rosalind_0914 Rosalind_2840
Rosalind_7751 Rosalind_1517
Rosalind_2943 Rosalind_1144
```

Test Suite

A passing test suite looks like this (NB: you can use `pytest -v test.py` instead of `make test`):

```
$ make test
python3 -m pytest -v test.py
============================= test session starts ==============================
platform darwin -- Python 3.6.8, pytest-4.2.0, py-1.7.0, pluggy-0.8.1 -- /anaconda3/bin/pyth
cachedir: .pytest_cache
rootdir: /Users/kyclark/work/worked_examples/13-grph, inifile:
plugins: remotedata-0.3.1, openfiles-0.3.2, doctestplus-0.2.0, arraydiff-0.3
collected 13 items

test.py::test_usage PASSED                                               [  7%]
test.py::test_bad_k PASSED                                               [ 15%]
test.py::test_bad_file PASSED                                            [ 23%]
test.py::test_find_kmers PASSED                                          [ 30%]
test.py::test_01 PASSED                                                  [ 38%]
test.py::test_02 PASSED                                                  [ 46%]
test.py::test_03 PASSED                                                  [ 53%]
test.py::test_04 PASSED                                                  [ 61%]
test.py::test_05 PASSED                                                  [ 69%]
test.py::test_06 PASSED                                                  [ 76%]
test.py::test_07 PASSED                                                  [ 84%]
test.py::test_08 PASSED                                                  [ 92%]
test.py::test_09 PASSED                                                  [100%]

========================== 13 passed in 2.35 seconds ===========================
```