Find Unclustered Proteins

Run `make data` to get the data you need for this exercise or manually download the data:

```
wget ftp://ftp.imicrobe.us/biosys-analytics/exercises/unclustered-proteins.tgz
```

Unpack the tarball with `tar xvf unclustered-proteins.tgz`.

Write a Python program called `find_unclustered.py` that will create a FASTA file of the unclustered proteins. The program will take a `-c|--cdhit` argument that is the name of the CD-HIT cluster file, a `-p|--proteins` FASTA file, and an `-o|--outfile` argument (default `unclustered.fa`) where to write the sequences.

```
$ ./find_unclustered.py
usage: find_unclustered.py [-h] -c str -p str [-o str]
find_unclustered.py: error: the following arguments are required: -c/--cdhit, -p/--proteins
$ ./find_unclustered.py -h
usage: find_unclustered.py [-h] -c str -p str [-o str]

Find unclustered proteins

optional arguments:
  -h, --help            show this help message and exit
  -c str, --cdhit str   Output file from CD-HIT (clustered proteins) (default:
                        None)
  -p str, --proteins str
                        Proteins FASTA (default: None)
  -o str, --outfile str
                        Output file (default: unclustered.fa)
```

If either of the CD-HIT or proteins files are not files, die with an error message:

```
$ ./find_unclustered.py -c foo -p unclustered-proteins/proteins.fa
--cdhit "foo" is not a file
$ ./find_unclustered.py -c unclustered-proteins/cdhit60.3+.clstr -p foo
--proteins "foo" is not a file
```

If successful, report the number of unclustered proteins written to the indicated output file:

```
$ ./find_unclustered.py -c unclustered-proteins/cdhit60.3+.clstr -p unclustered-proteins/pro
Wrote 204,262 of 220,520 unclustered proteins to "unclustered.fa"
$ ./find_unclustered.py -c unclustered-proteins/cdhit60.3+.clstr -p unclustered-proteins/pro
Wrote 204,262 of 220,520 unclustered proteins to "unprots.fa"
```

Discussion

The file `cdhit60.3+.clstr` contains all of the GI numbers for proteins that were clustered and put into HMM profiles. It is almost in FASTA format, but not quite.

```
$ head -5 cdhit60.3+.clstr
>Cluster_5086
0    358aa, >gi|317183610|gb|ADV... at 66.76%
1    361aa, >gi|315661179|gb|ADU... at 70.36%
2    118aa, >gi|375968555|gb|AFB... at 70.34%
3    208aa, >gi|194307477|gb|ACF... at 61.54%
```

The protein IDs are in the bit that looks like ">gi|317183610|gb|ADV..." where the ID is "317183610". A regex would be the perfect thing to extract this.

The file `proteins.fa` contains all proteins. The protein ID is usually the only thing in the header:

```
$ grep '>' proteins.fa | head -5
>388548806
>388548807
>388548808
>388548809
>388548810
```

But not always:

```
$ grep -e '^>' proteins.fa | sed "s/^>//" | grep -v -P '^\d+$' | head -5
26788002|emb|CAD19173.1| putative RNA helicase, partial [Agaricus bisporus virus X]
26788000|emb|CAD19172.1| putative RNA helicase, partial [Agaricus bisporus virus X]
985757046|ref|YP_009222010.1| hypothetical protein [Alternaria brassicicola fusarivirus 1]
985757045|ref|YP_009222011.1| hypothetical protein [Alternaria brassicicola fusarivirus 1]
985757044|ref|YP_009222009.1| polyprotein [Alternaria brassicicola fusarivirus 1]
```

I would recommend you first parse the CD-HIT file and get all the protein IDs that have been clustered. You only need to know that they were in some cluster, so a `set` is a good data structure although a dictionary is fine, too. Proteins may be clustered more than once, but you don't need to keep track of that.

Once you know which protein IDs were clustered, go through the `proteins.fa` file and print the unclustered proteins to the given outfile. You will need to remove anything from the ID starting with a `|` (pipe character). The `re` module has a `sub` function that "substitutes" some pattern with a replacement. Because the `|` is a metacharacter inside a regex, it must be backslash-escaped or put into a character class:

```
>>> re.sub('\|.*', '', 'foo|bar')
'foo'
>>> re.sub('[|].*', '', 'foo|bar')
```

```
'foo'
```

Test Suite

A passing test suite looks like this:

```
$ make test
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/python/practical_python_for_data_science/ch11-regular-expressio
plugins: remotedata-0.3.1, openfiles-0.3.2, doctestplus-0.2.0, arraydiff-0.3
collected 4 items

test.py::test_usage PASSED                                               [ 25%]
test.py::test_bad_input PASSED                                           [ 50%]
test.py::test_good_input1 PASSED                                         [ 75%]
test.py::test_good_input2 PASSED                                         [100%]

========================== 4 passed in 14.60 seconds ===========================
```