Randomly Subset a FASTA file

Write a Python program called `sampler.py` that will probabilistically sample one or more input FASTA files into an output directory.

The inputs for this program will be generated by your `moog.py` program. You can run `make fasta` to create files of 1K, 10K, 100K, and 1M reads in this directory. You can then use these files for testing your program.

The parameters for your program are:

- One or more positional FILE arguments
- `-p|--pct`: a `float` value between 0 and 1 which is the percentage of reads to take (default `0.1` or 10%)
- `-s|--seed`: an `int` value to use for the random seed (default `None`)
- `-o|--outdir`: an `str` value to use for the output directory (default `'out'`). You will need to create this directory if it does not exist. Consult your `transcribe.py` program to see how to do that.

Here is the usage your program should create for `-h` or `--help`:

```
$ ./sampler.py -h
usage: sampler.py [-h] [-p reads] [-s seed] [-o DIR] FILE [FILE ...]

Probabalistically subset FASTA files

positional arguments:
  FILE                  Input FASTA file(s)

optional arguments:
  -h, --help            show this help message and exit
  -p reads, --pct reads
                        Percent of reads (default: 0.1)
  -s seed, --seed seed  Random seed value (default: None)
  -o DIR, --outdir DIR  Output directory (default: out)
```

When run with the `n1k.fa`, it should print this:

```
$ ./sampler.py n1k.fa -s 1
  1: n1k.fa
Wrote 95 sequences from 1 file to directory "out"
```

Here is an example of the output for multiple files:

```
$ ./sampler.py -p .25 -s 4 n1k.fa n10k.fa n100k.fa -o sampled
  1: n1k.fa
  2: n10k.fa
  3: n100k.fa
Wrote 27,688 sequences from 3 files to directory "sampled"
```

To parse the FASTA files, you will need to add this import statement:

```
from Bio import SeqIO
```

If you have not already install BioPython, you will need to do so:

```
$ python3 -m pip install biopython
```

Or run:

```
$ python3 -m pip install -r requirements.txt
```

Here is the basic structure for your program:

```
def main():
    args = get_args()
    random.seed(args.seed) ①

    for fh in ...:         ②
        basename = os.path.basename(fh.name)           ③
        out_file = os.path.join(args.outdir, basename) ④
        print(...)         ⑤

        out_fh = ...       ⑥
        for rec in SeqIO.parse(fh, 'fasta'):      ⑦
            if ...:                               ⑧
                SeqIO.write(rec, out_fh, 'fasta') ⑨

        out_fh.close()     ⑩

    print(...) ⑪
```

① Set your random seed right away.

② Iterate over the input file handles. Consider using the `enumerate()` function to get both the index and the value for the file handles.

③ You'll need the `os.path.basename()` to construct the output path.

④ The output file is the `basename` plus the output directory.

⑤ Update the user on the progress.

⑥ You'll need a file handle for writing the output.

⑦ Use the `SeqIO.parse()` function to parse the file handle's contents in FASTA format. Each `rec` is an object representing a sequence in the file.

⑧ Use `random.random()` in conjunction with `args.pct` to decide whether to take this sequence.

⑨ Write the sequence to the output file handle in FASTA format.

⑩ Close the output file handle.

⑪ Print a summary of how many sequences in how many files were written to what directory.

The `enumerate` function will give you both the index and value of elements in a sequence:

```
>>> list(enumerate('abc'))
[(0, 'a'), (1, 'b'), (2, 'c')]
```

You can start counting at a number other than 0!

```
>>> for i, letter in enumerate('abc', start=1):
...     print(f'{i:3}: {letter}')
...
  1: a
  2: b
  3: c
```

All tests should pass:

```
$ make test
pytest --disable-pytest-warnings -xv test.py
============================= test session starts ==============================
...
collected 6 items

test.py::test_exists PASSED                                              [ 16%]
test.py::test_usage PASSED                                               [ 33%]
test.py::test_bad_file PASSED                                            [ 50%]
test.py::test_bad_pct PASSED                                             [ 66%]
test.py::test_defaults PASSED                                            [ 83%]
test.py::test_options PASSED                                             [100%]

========================== 6 passed, 1 warning in 2.23s ========================
```