Transcribing DNA into RNA

For this exercise, we'll be applying what we learned about modifying strings to a variation on this Rosalind exercise that transcribes DNA into RNA:

http://rosalind.info/problems/rna/

You will write a Python program called `transcribe.py` that will accept:

- One or more positional arguments which must be readable files
- An optional `-o` or `--outdir` argument that names an output directory (default `'out'`)

You can use the `os.path.isdir` to check if the output directory exists. It works just like the `os.path.isfile` function we've used that will return `True` or `False` if a given string names an existing file, only this checks for a directory. Here assuming that "blargh" does not exist on your system:

```
>>> import os
>>> os.path.isdir('blargh')
False
```

If the directory does not exist, you should use the `os.makedirs` function to create it. Here is a bit of code you can put into your program:

```
if not os.path.isdir(out_dir):
    os.makedirs(out_dir)
```

Your program will read each of the input files which will contain a single DNA sequence on each line. The sequences will need to replace the `T` bases with `U`. For instance, the `input1.txt` file contains a single sequence `'GATGGAACTTGACTACGTAAATT'` which will become `'GAUGGAACUUGACUACGUAAAUU'`.

The new sequences from each input file will be written to a new output file in the `--outdir`. The name of the file will be the "basename" of the input file which you can get by using the `os.path.basename` function. For instance, the "basename" of `'./inputs/input1.txt'` is `'input1.txt'`:

```
>>> base = os.path.basename('./inputs/input1.txt')
>>> base
'input1.txt'
```

If the output directory is `'out'`, you can create a new path for the output file by using the `os.path.join` function with the basename of the input file's basename:

```
>>> out_dir = 'out'
>>> os.path.join(out_dir, base)
'out/input1.txt'
```

If you declare your `args.file` parameter using `type=argparse.FileType('r')`, then you'll be iterating over a list of *open file handles*. You can use the `fh.name` to get the name of the file:

```
for fh in args.file:
    out_file = os.path.join(out_dir, os.path.basename(fh.name))
    out_fh = open(out_file, 'wt')
```

You will have two levels of iteration:

- Each `file` argument
- Each line in each file

This is almost identical to the `wc.py` program.

You will need to `open` the output file for writing text, iterate over each line in the input file, and print the transcribed sequences to the output file.

Your program should print a brief usage when given no arguments:

```
$ ./transcribe.py
usage: transcribe.py [-h] [-o DIR] FILE [FILE ...]
transcribe.py: error: the following arguments are required: FILE
```

And a longer usage for `-h` and `--help`:

```
$ ./transcribe.py -h
usage: transcribe.py [-h] [-o DIR] FILE [FILE ...]

Transcribing DNA into RNA

positional arguments:
  FILE                  Input file(s)

optional arguments:
  -h, --help            show this help message and exit
  -o DIR, --outdir DIR  Output directory (default: out)
```

The output from the program should summarize how many sequences and files were processed. The `input1.txt` file contains a single line/sequence:

```
$ ./transcribe.py inputs/input1.txt
Done, wrote 1 sequence in 1 file to directory "out".
```

The `input2.txt` file contains two longer lines/sequences:

```
$ ./transcribe.py inputs/input2.txt
Done, wrote 2 sequences in 1 file to directory "out".
```

When you process both together, it should summarize for all the inputs:

```
$ ./transcribe.py inputs/*
Done, wrote 3 sequences in 2 files to directory "out".
```

I would recommend you revisit the "Words Count" to remember how to count lines of the input files. Note that you must use the correct singular/plural for both "sequence(s)" and "file(s)."

A passing test suite looks like this:

```
$ make test
pytest -xv test.py
============================= test session starts ==============================
...
collected 7 items

test.py::test_exists PASSED                                              [ 14%]
test.py::test_usage PASSED                                               [ 28%]
test.py::test_no_args PASSED                                             [ 42%]
test.py::test_bad_file PASSED                                            [ 57%]
test.py::test_good_input1 PASSED                                         [ 71%]
test.py::test_good_input2 PASSED                                         [ 85%]
test.py::test_good_multiple_inputs PASSED                                [100%]

============================== 7 passed in 0.36s ===============================
```