Tranpose ABC Notation

Write a Python program called `transpose.py` that will read a file in ABC notation (https://en.wikipedia.org/wiki/ABC_notation) and transpose the melody line up or down by a given `-s|--shift` argument. Like the `rot13` exercise, it might be helpful to think of the space of notes (`ABCDEFG`) as a list which you can roll through. For instance, if you have the note `c` and want to transpose up a (minor) third (`-s 3`), you would make the new note `e`; similarly if you have the note `F` and you go up a (major) third, you get `A`. You will not need to worry about the actual number of semitones that you are being asked to shift, as the previous example showed that we might be shifting by a major/minor/augmented/diminished/pure interval. The purpose of the exercise is simply to practice with lists.

Expected Behavior

```
$ ./transpose.py
usage: transpose.py [-h] [-s int] FILE
transpose.py: error: the following arguments are required: FILE
$ ./transpose.py -h
usage: transpose.py [-h] [-s int] FILE

Tranpose ABC notation

positional arguments:
  FILE                  Input file

optional arguments:
  -h, --help            show this help message and exit
  -s int, --shift int   Interval to shift (default: 2)
$ ./transpose.py foo
"foo" is not a file
$ ./transpose.py songs/legacy.abc -s 1
--shift "1" must be between 2 and 8
$ ./transpose.py songs/legacy.abc
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:A
AGA CBC | aga abc | AGA CBC | e2B BGE |
AGA CBC | aga abc | baf feC |1 eCB BGE :|2 eCB BCe |:
```

```
fgf feC | eCB BCe | fgf feC | aeC BCe |
fgf feC | e2e efg | agf feC |1 eCB BCe :|2 eCB BGE |]
</score>
```

Discussion

A sample ABC song is given:

```
$ cat songs/legacy.abc
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA AFD :|2 dBA ABd |:
efe edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
</score>
```

If you use `new_py.py` to create your new program with the `file` as a single positional argument, you can use this code to get the input file and check that it is, indeed, a file:

```
args = get_args()
file = args.file

if not os.path.isfile(file):
    die('"{}" is not a file'.formate(file))
```

Now that you have a file, you can use a `for` loop to read it. Each line will still have a newline attached to the end, so you can use `rstrip()` to remove it:

```
for line in open(file):
    line = line.rstrip()
```

If a line starts with `<` and ends with `>` (cf. `str.startswith` and `str.endswith`), you can just print the line as-is. If the line starts with `K:`, then you have the key signature and should transpose it, e.g., if you have `K:A` and you are shifting a fifth, you should print `K:E`. If you have a line that starts with any other single uppercase letter and a colon, just print the line as-is. Finally, if you have a line that doesn't match any of the above conditions, you have a line of melody that needs to be transposed.

If you are unfamiliar with musical transposition, you may be a bit confused by

the notion of a interval. A "second" equals a `--shift` of one note; that is, the distance from `A` to `B` is one note, but we call that a "second." Therefore, assume that the `--shift` argument is the name of the interval, e.g., `4` (a "fourth") is actually a move of three notes. That means the argument provided by the user should be in the range 2 to 8, inclusive, so complain if it is not.

Note that the transposition of a tune up a fourth is the same as down a fifth:

```
$ ./transpose.py songs/legacy.abc -s 4
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:C
CBC EDE | cbc cde | CBC EDE | g2D DBG |
CBC EDE | cbc cde | dca agE |1 gED DBG :|2 gED DEg |:
aba agE | gED DEg | aba agE | cgE DEg |
aba agE | g2g gab | cba agE |1 gED DEg :|2 gED DBG |]
</score>
$ ./transpose.py songs/legacy.abc -s -5
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:C
CBC EDE | cbc cde | CBC EDE | g2D DBG |
CBC EDE | cbc cde | dca agE |1 gED DBG :|2 gED DEg |:
aba agE | gED DEg | aba agE | cgE DEg |
aba agE | g2g gab | cba agE |1 gED DEg :|2 gED DBG |]
</score>
```

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/ch05-python-strings-li
```

```
plugins: remotedata-0.3.1, openfiles-0.3.2, doctestplus-0.2.0, arraydiff-0.3
collected 3 items

test.py::test_usage PASSED                                               [ 33%]
test.py::test_bad_input PASSED                                           [ 66%]
test.py::test_file PASSED                                                [100%]

========================== 3 passed in 0.53 seconds ==========================
```