Bottles of Beer Song

Write a Python program called `bottles.py` that takes a single option `-n|--num_bottles` which is an positive integer (default 10) and prints the " bottles of beer on the wall song." If the `-n` argument is less than 1, die with "N () must be a positive integer". The program should also respond to `-h|--help` with a usage statement.

I'd encourage you to think about the program as a formal algorithm. Read the introduction to Jeff Erickson's book *Algorithms* available here:

- http://jeffe.cs.illinois.edu/teaching/algorithms/#book
- http://jeffe.cs.illinois.edu/teaching/algorithms/book/00-intro.pdf

You are going to need to count down, so you'll need to consider how to do that. First, let's examine a list and see how it can be sorted and reversed. We've already used the `sorted` *function*, but we haven't really talked about the `list` class's `sort` *method*. Note that the former does not mutate the list itself:

```
>>> a = ['foo', 'bar', 'baz']
>>> sorted(a)
['bar', 'baz', 'foo']
>>> a
['foo', 'bar', 'baz']
```

But the `sort` method does:

```
>>> a.sort()
>>> a
['bar', 'baz', 'foo']
```

Also, note what is returned by `sort`:

```
>>> type(a.sort())
<type 'NoneType'>
```

So if you did this, you'd destroy your data:

```
>>> a = a.sort()
>>> a
```

As with `sort`/`sorted`, so it goes with `reverse`/`reversed`. The past participle version *returns a new copy of the data without affecting the original* and is therefore the safest bet to use:

```
>>> a = ['foo', 'bar', 'baz']
>>> a
['foo', 'bar', 'baz']
>>> reversed(a)
<listreverseiterator object at 0x10f0d61d0>
>>> list(reversed(a))
```

```
['baz', 'bar', 'foo']
>>> a
['foo', 'bar', 'baz']
```

Compare with:

```
>>> a.reverse()
>>> a
['baz', 'bar', 'foo']
```

Given that and your knowledge of how `range` works, can you figure out how to count down, say, from 10 to 1?

Expected Behavior

```
$ ./bottles.py -h
usage: bottles.py [-h] [-n INT]

Bottles of beer song

optional arguments:
  -h, --help            show this help message and exit
  -n INT, --num_bottles INT
$ ./bottles.py --help
usage: bottles.py [-h] [-n INT]

Bottles of beer song

optional arguments:
  -h, --help            show this help message and exit
  -n INT, --num_bottles INT
                        How many bottles (default: 10)
$ ./bottles.py -n 1
1 bottle of beer on the wall,
1 bottle of beer,
Take one down, pass it around,
0 bottles of beer on the wall!

$ ./bottles.py
10 bottles of beer on the wall,
10 bottles of beer,
Take one down, pass it around,
9 bottles of beer on the wall!

9 bottles of beer on the wall,
```

```
9 bottles of beer,
Take one down, pass it around,
8 bottles of beer on the wall!

8 bottles of beer on the wall,
8 bottles of beer,
Take one down, pass it around,
7 bottles of beer on the wall!

7 bottles of beer on the wall,
7 bottles of beer,
Take one down, pass it around,
6 bottles of beer on the wall!

6 bottles of beer on the wall,
6 bottles of beer,
Take one down, pass it around,
5 bottles of beer on the wall!

5 bottles of beer on the wall,
5 bottles of beer,
Take one down, pass it around,
4 bottles of beer on the wall!

4 bottles of beer on the wall,
4 bottles of beer,
Take one down, pass it around,
3 bottles of beer on the wall!

3 bottles of beer on the wall,
3 bottles of beer,
Take one down, pass it around,
2 bottles of beer on the wall!

2 bottles of beer on the wall,
2 bottles of beer,
Take one down, pass it around,
1 bottle of beer on the wall!

1 bottle of beer on the wall,
1 bottle of beer,
Take one down, pass it around,
0 bottles of beer on the wall!
```

Test Suite

A passing test suite looks like this:

```
$ 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/bottles_of_beer, inifile:
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_one PASSED                                                        [ 50%]
test.py::test_two PASSED                                                        [ 75%]
test.py::test_random PASSED                                                     [100%]

============================== 4 passed in 0.68 seconds ================================
```