Python Bash Utilities

Now we will revisit our "head.sh" and "cat-n.sh" utilities but writing them now in Python.

cat_n.py

Create a Python program called "cat_n.py" that does the following:

- It should expect exactly one argument which is a regular file; if either condition fails, print a "Usage" statement
- It should print each line of the file argument preceeded by the line number which is right-justified in spaces and a colon. You may use format strings to make it look exactly like the output below, but the test is just checking for a leading space, some number(s), a colon, and whatever else.

Expected behavior:

```
$ ./cat_n.py
Usage: cat_n.py FILE
$ ./cat_n.py foo
foo is not a file
$ ./cat_n.py files/issa.txt
    1: Selected Haiku by Issa
    2:
    3: Don't worry, spiders,
    4: I keep house
    5: casually.
    6:
    7: New Year's Day-
    8: everything is in blossom!
    9: I feel about average.
   10:
   11: The snow is melting
   12: and the village is flooded
   13: with children.
   14:
   15: Goes out,
   16: comes back-
   17: the love life of a cat.
   18:
   19: Mosquito at my ear-
   20: does he think
   21: I'm deaf?
   22:
   23: Under the evening moon
```

```
24: the snail
25: is stripped to the waist.
26:
27: Even with insects-
28: some can sing,
29: some can't.
30:
31: All the time I pray to Buddha
32: I keep on
33: killing mosquitoes.
34:
35: Napped half the day;
36: no one
37: punished me!
```

head.py

Create a Python program called "head.py" that does the following:

- It should expect one or two arguments; if there are no arguments, print a "Usage" statement
- The first argument is required and much be a regular file; if it is not, print " is not a file" and exit *with an error code*
- The second argument is optional. If given, it must be a positive number (non-zero); if it is not, then print "lines () must be a positive number". If no argument is provided, use a default value of 3. You can expect that the test will only give you a value that can be safely converted to a number using the `int` function.
- If given good input, it should act like the normal `head` utility and print the expected number of lines from the file

Expected behavior:

```
$ ./head.py
Usage: head.py FILE [NUM_LINES]
$ ./head.py foo
foo is not a file
$ ./head.py files/issa.txt
Selected Haiku by Issa

Don't worry, spiders,
$ ./head.py files/issa.txt 5
Selected Haiku by Issa

Don't worry, spiders,
I keep house
```

casually.

Test Suite

A passing test suite looks like the following:

```
$ make test
python3 -m pytest -v test.py
============================= test session starts ==============================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1 -- /anaconda3/bin/pyth
cachedir: .pytest_cache
rootdir: /Users/kyclark/work/worked_examples/04-python-bash, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 7 items

test.py::test_usage_catn PASSED                                          [ 14%]
test.py::test_usage_head PASSED                                          [ 28%]
test.py::test_bad_input_catn PASSED                                      [ 42%]
test.py::test_bad_number_head PASSED                                     [ 57%]
test.py::test_bad_input_head PASSED                                      [ 71%]
test.py::test_catn_run PASSED                                            [ 85%]
test.py::test_head_run PASSED                                            [100%]

========================== 7 passed in 0.57 seconds ===========================
```