# Bash Scripting

In this exercise you will create two scripts that focus on using `for` loops and arguments in bash.

## cat -n

The `cat` program is used to "concatenate" files together, but it's often abused to simply spit out a short files so you can look at the contents without using a text editor or a pager like `more` or `less`. Another common task is to use the `-n` flag to "number" the lines of output:

```
$ cat -n files/sonnet-29.txt
     1  Sonnet 29
     2  William Shakespeare
     3
     4  When, in disgrace with fortune and men's eyes,
     5  I all alone beweep my outcast state,
     6  And trouble deaf heaven with my bootless cries,
     7  And look upon myself and curse my fate,
     8  Wishing me like to one more rich in hope,
     9  Featured like him, like him with friends possessed,
    10  Desiring this man's art and that man's scope,
    11  With what I most enjoy contented least;
    12  Yet in these thoughts myself almost despising,
    13  Haply I think on thee, and then my state,
    14  (Like to the lark at break of day arising
    15  From sullen earth) sings hymns at heaven's gate;
    16  For thy sweet love remembered such wealth brings
    17  That then I scorn to change my state with kings.
```

You will create a bash script called "cat_n.sh" that mimics this output. Here are the expectations of your program:

- If there are no arguments, it should print a "Usage" and exit *with an error code*
- Your program will expect to receive an argument in `$1`
- If the argument is not a file, it should notify the user and exit *with an error code*
- It will iterate over the lines in the file and print the line number, a space, and the line of the file
- Your output will differ from regular `cat -n` as I won't expect you to right-align the numbers.

Here is the expected behavior:

```
$ ./cat_n.sh
Usage: cat-n.sh FILE
$ ./cat_n.sh files/
files/ is not a file
$ ./cat_n.sh files/sonnet-29.txt
1 Sonnet 29
2 William Shakespeare
3
4 When, in disgrace with fortune and men's eyes,
5 I all alone beweep my outcast state,
6 And trouble deaf heaven with my bootless cries,
7 And look upon myself and curse my fate,
8 Wishing me like to one more rich in hope,
9 Featured like him, like him with friends possessed,
10 Desiring this man's art and that man's scope,
11 With what I most enjoy contented least;
12 Yet in these thoughts myself almost despising,
13 Haply I think on thee, and then my state,
14 (Like to the lark at break of day arising
15 From sullen earth) sings hymns at heaven's gate;
16 For thy sweet love remembered such wealth brings
17 That then I scorn to change my state with kings.
```

## Testing

You have been provided a `Makefile` that will run a test suite. This is what it should look like when all tests are passing:

```
$ 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/ch02-unix-exercises/ex
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_run PASSED                                                 [100%]

========================== 3 passed in 0.05 seconds ===========================
```

First the program is executed with no arguments to see if it produces a "usage"-type help messages. Then it is tested with bad input. Then it is tested that it

produces the expected output when given good input.

This is the basis of "test-driven design" (TDD). We define a set of tests that describe what working software should do. When the tests pass, the software is done. When you are passing all the tests, you are done!