Common Patterns in Python

Get positional command-line arguments

You can get the command-line arguments using `sys.argv` (argument vector), but it's annoying that the name of the Python program itself is in the first position (`sys.argv[0]`). To skip over this, take a slice of the argument vector starting at the second position (index `1`) which will succeed even if there are no arguments – you'll get an empty list, which is safe.

```
$ cat -n args.py
     1  #!/usr/bin/env python3
     2
     3  import os
     4  import sys
     5
     6  args = sys.argv[1:]
     7  num = len(args)
     8
     9  print('There are {} arg{}'.format(num, '' if num == 1 else 's'))
$ ./args.py
There are 0 args
$ ./args.py foo
There are 1 arg
$ ./args.py foo bar
There are 2 args
```

Put positional arguments into named variables

If you use `sys.argv[1]` and `sys.argv[2]` throughout your program, it degrades readability. It's better to copy the values into variables that have meaningful names like "file" or "num_lines".

```
$ cat -n name_args.py
     1  #!/usr/bin/env python3
     2
     3  import os
     4  import sys
     5
     6  args = sys.argv[1:]
     7
     8  if len(args) != 2:
     9      print('Usage: {} FILE NUM'.format(os.path.basename(sys.argv[0])))
    10      sys.exit(1)
    11
```

```
    12  file, num = args
    13
    14  file = args[0]
    15  num = args[1]
    16
    17  print('FILE is "{}", NUM is "{}"'.format(file, num))
$ ./name_args.py
Usage: name_args.py FILE NUM
$ ./name_args.py nobody.txt 10
FILE is "nobody.txt", NUM is "10"
```

Set defaults for optional arguments

```
$ cat -n default_arg.py
     1  #!/usr/bin/env python3
     2
     3  import os
     4  import sys
     5
     6  args = sys.argv[1:]
     7  num_args = len(args)
     8
     9  if not 1 <= num_args <= 2:
    10      print('Usage: {} FILE [NUM]'.format(os.path.basename(sys.argv[0])))
    11      sys.exit(1)
    12
    13  file = args[0]
    14  num = args[1] if num_args == 2 else 10
    15
    16  print('FILE is "{}", NUM is "{}"'.format(file, num))
$ ./default_arg.py
Usage: default_arg.py FILE [NUM]
$ ./default_arg.py nobody.txt
FILE is "nobody.txt", NUM is "10"
$ ./default_arg.py nobody.txt 5
FILE is "nobody.txt", NUM is "5"
```

Test argument is file and read

This program takes an argument, tests that it is a file, and then reads it. It's basically `cat`.

```
$ cat -n read_file.py
     1  #!/usr/bin/env python3
```

```
     2  """Read a file argument"""
     3
     4  import os
     5  import sys
     6
     7  args = sys.argv[1:]
     8
     9  if len(args) != 1:
    10      print('Usage: {} ARG'.format(os.path.basename(sys.argv[0])))
    11      sys.exit(1)
    12
    13  filename = args[0]
    14
    15  if not os.path.isfile(filename):
    16      print('"{}" is not a file'.format(filename), file=sys.stderr)
    17      sys.exit(1)
    18
    19  for line in open(filename):
    20      print(line, end='')
$ ./read_file.py foo
"foo" is not a file
$ ./read_file.py nobody.txt
I'm Nobody! Who are you?
Are you - Nobody - too?
Then there's a pair of us!
Don't tell! they'd advertise - you know!

How dreary - to be - Somebody!
How public - like a Frog -
To tell one's name - the livelong June -
To an admiring Bog!

Emily Dickinson
```

Test if an argument is a directory and list the contents

```
$ cat -n list_dir.py
     1  #!/usr/bin/env python3
     2  """Show contents of directory argument"""
     3
     4  import os
     5  import sys
     6
     7  args = sys.argv[1:]
     8
```

```
 9  if len(args) != 1:
10      print('Usage: {} DIR'.format(os.path.basename(sys.argv[0])))
11      sys.exit(1)
12
13  dirname = args[0]
14
15  if not os.path.isdir(dirname):
16      print('"{}" is not a directory'.format(dirname), file=sys.stderr)
17      sys.exit(1)
18
19  for entry in os.listdir(dirname):
20      print(os.path.abspath(entry))
$ ./list_dir.py nobody.txt
"nobody.txt" is not a directory
$ ./list_dir.py .
/Users/kyclark/work/biosys-analytics/lectures/08-python-patterns/examples/list_dir.py
/Users/kyclark/work/biosys-analytics/lectures/08-python-patterns/examples/nobody.txt
/Users/kyclark/work/biosys-analytics/lectures/08-python-patterns/examples/read_file.py
```

Skip an iteration of a loop

Sometimes in a loop (`for` or `while`) you want to skip immediately to the top of the loop. You can use `continue` to do this. In this example, we skip the even-numbered lines by using the modulus `%` operator to find those line numbers which have a remainder of 0 after dividing by 2. We can use the `enumerate` function to provide both the array index and value of any list.

```
$ cat -n skip_loop.py
 1  #!/usr/bin/env python3
 2
 3  import os
 4  import sys
 5
 6
 7  args = sys.argv[1:]
 8
 9  if len(args) != 1:
10      print('Usage: {} FILE'.format(os.path.basename(sys.argv[0])))
11      sys.exit(1)
12
13  file = args[0]
14
15  if not os.path.isfile(file):
16      print('"{}" is not a file'.format(file), file=sys.stderr)
17      sys.exit(1)
```

```
    18
    19  for i, line in enumerate(open(file)):
    20      if (i + 1) % 2 == 0:
    21          continue
    22
    23      print(i + 1, line, end='')
$ ./skip_loop.py
Usage: skip_loop.py FILE
$ ./skip_loop.py nobody.txt
1 I'm Nobody! Who are you?
3 Then there's a pair of us!
5
7 How public - like a Frog -
9 To an admiring Bog!
11 Emily Dickinson
```