file processing, lists

http://www.youtube.com/watch?v=5zey8567bcg

recall: while loops

```
while test:
    statements
```

reading files

```
f = file("filename")
text = f.read()
```

- Opens a file for reading, and stores its contents in a variable named `text`.

line-based file processing

- `f.readline()`
  - Returns the next line in the file (like `nextLine` on a `Scanner`) or a blank string if there are no more lines
- `f.readlines()`
  - Returns a list of all lines in the file

loop-based file input

- a file object can be the target of a for loop

```
>>> for line in file("carroll.txt"):
...     print line.strip()   # strip() removes \n

Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

exercise

- write a function `stats` that accepts a filename and reports the file’s longest line

```
>>> stats("carroll.txt")
longest line = 42 characters
the jaws that bite, the claws that catch,
```

solution

```python
def stats(filename):
    longest = ""
    for line in file(filename):
        if len(line) > len(longest):
            longest = line

    print "Longest line =", len(longest)
    print longest
```

writing files

- `f = file("filename", "w")`
  - opens a file to be written (overwriting any existing contents)
- `f.write(s)`
  - writes the contents of string s to the file
- `f.close()`
  - closes the file - call this when you’re done writing

example

```
>>> out = file("output.txt", "w")
>>> out.write("Hello, world!\n")
>>> out.write("How are you?")
>>> out.close()

>>> file("output.txt").read()
'Hello, world!\nHow are you?'
```

exercise

- write a function `remove_lowercase` that accepts two filenames, and copies all lines that do not start with a lowercase letter from the first file into the second

```
>>> remove_lowercase("carroll.txt", "out.txt")
>>> print file("out.txt").read()
Beware the Jabberwock, my son,
Beware the JubJub bird and shun
```

solution

```python
def remove_lowercase(infile, outfile):
    output = file(outfile, "w")
    for line in file(infile):
        if not line[0] in "abcdefghijklmnopqrstuvwxyz":
            output.write(line)
    output.close()
```

lists

- like Java's arrays (but way cooler)
- declaring:
    - `name = [value1, value2, ...]` or
    - `name = [value] * length`
- accessing/modifying:
    - `name[index] = value`

list indexing

- lists can be indexed with positive or negative numbers (we’ve seen this before!)

index	0	1	2	3	4	5	6	7
value	9	14	12	19	16	18	24	15
index	-8	-7	-6	-5	-4	-3	-2	-1

list slicing

```
name[start:end]          # end is exclusive
name[start:]             # to end of list
name[:end]               # from start of list
name[start:end:step]     # every step'th value
```

other list abilities

- lists can be printed (or converted to string with `str()`)
- `len(list)` returns a list's length

exercise

Using data from midterm.txt:

```
58
89
94
77
78
```

Recreate `Histogram.java` in python

```
94: *****************
95: *************
96: *******************
97: *******
98: **
99: ******
100: ***
```

midterm.py

```
scores = [0]*101

#for each line in the file increment count of that score
for line in file("scores.txt"):
    scores[int(line)]+=1

for i in range(len(scores)):
    if scores[i] > 0:
        print str(i) + ": " + "*" * scores[i]
```

lists are everywhere!

- `range` returns a list

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- strings behave like lists of characters
  - `len`, indexing, `for` loops

splitting

- split breaks a string into a list of tokens

```
>>> name = "Brave Sir Robin"
>>> tokens = name.split()    # break by whitespace
>>> tokens
['Brave', 'Sir', 'Robin']
>>> name.split("r")          # break by delimiter
['B', 'ave Si', ' Robin']
```

- join does the opposite of a split

```
>>> "||".join(tokens)

'Brave||Sir||Robin'
```

tokenizing file input

- use `split` on lines when reading files
- remember typecasting: `type(value)`

```
>>> f = file("example.txt")
>>> line = f.readline()
>>> line
'hello world 42\n'

>>> tokens = line.split()
>>> tokens
['hello', 'world', '42']

>>> word = tokens[0]
>>> word
'hello'
>>> answer = int(tokens[2])
>>> answer
42
```

example

cities.txt

```
371.3839576  299.9969436
377.4026844  298.2579679
378.0258114  298.1785059
381.4240249  295.9413698
382.4046042  294.9817241
382.7161681  290.1804379
382.7306589  289.9512235
383.1509076  289.6578281
383.5590794  288.9182286
383.8682278  288.7195753
383.573571   288.5331478
383.8078469  288.4506304
384.1822063  288.3406073
383.6750096  288.1602916
```

...

(13510 more lines)

map.py

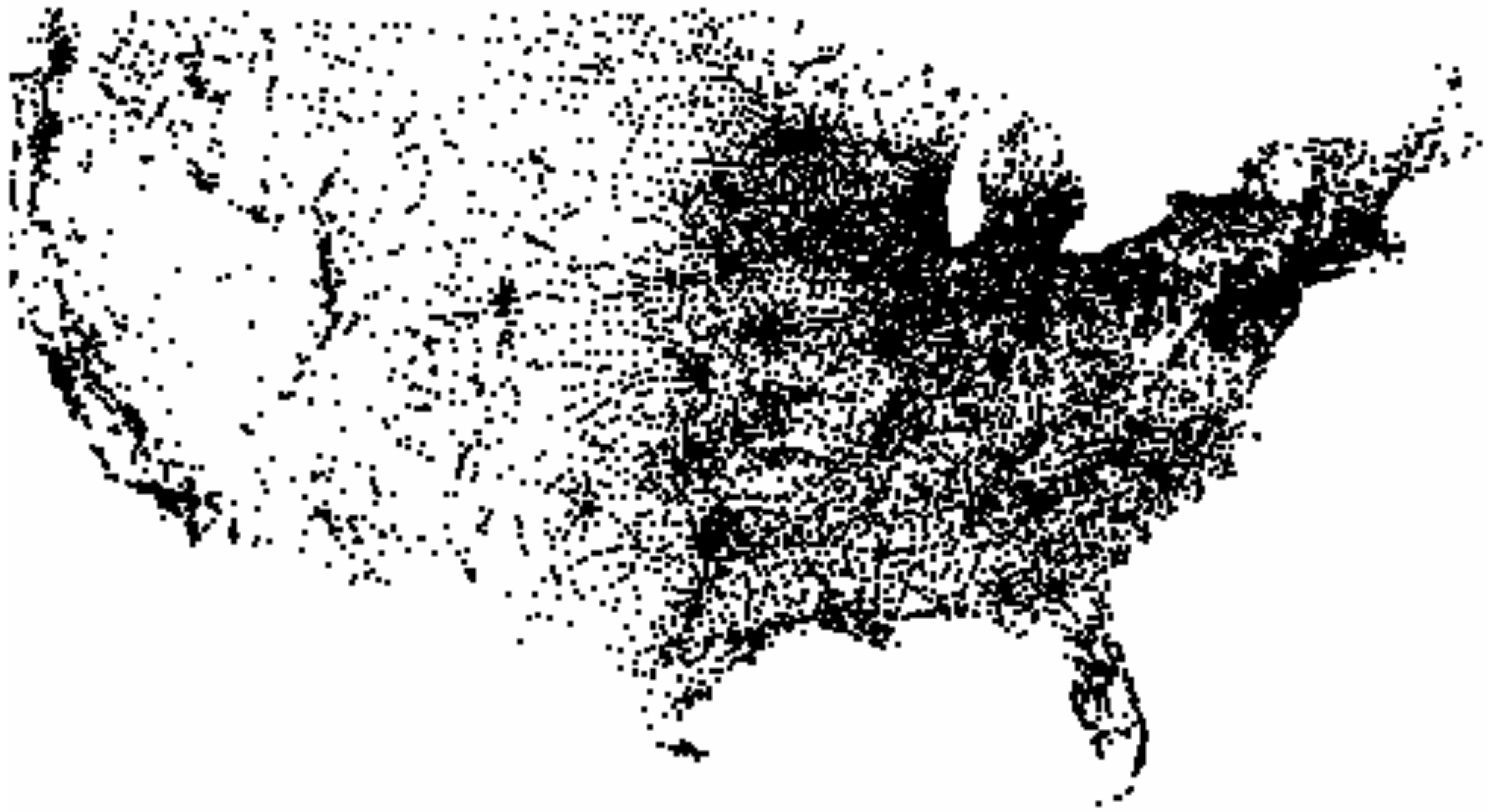
```
from drawingpanel import *

panel = DrawingPanel(500, 300)

for line in file("cities.txt"):
    parts = line.split()
    x = int(round(float(parts[0])))
    y = int(round(float(parts[1])))
    panel.canvas.create_rectangle(x, y, x, y)

panel.mainloop()
```

output



(file processing is awesome!)