file processing, lists

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

recall: reading files

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

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

line-based file processing

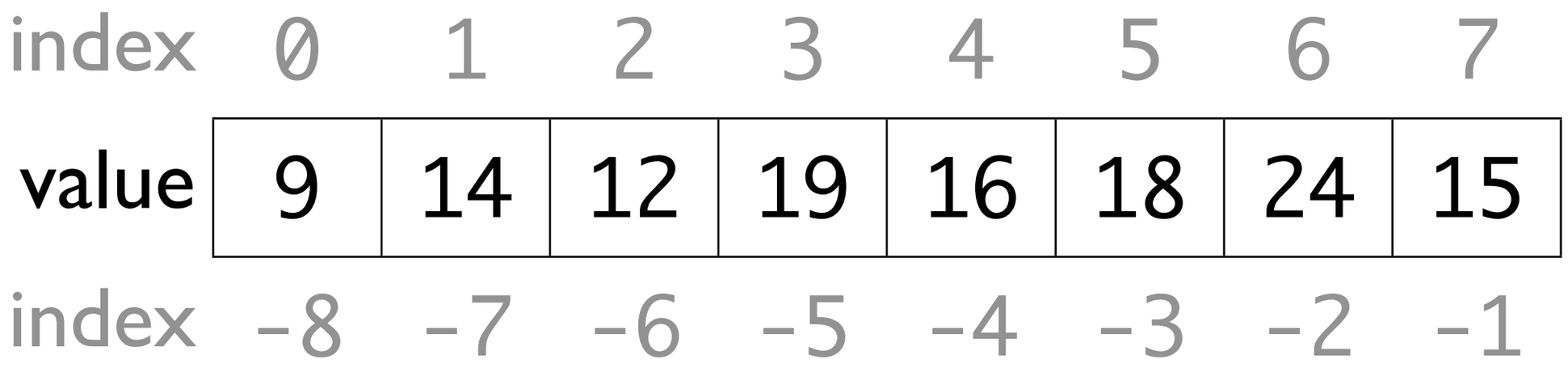
- `f.readline()`
  - Returns the next line in the file (like `nextLine` on a `Scanner`) or a blank string if there are no more lines
- `f.readlines()`
  - Returns a list of all lines in the file

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!)



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 open("scores.txt"):
    scores[int(line)]+=1

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

lists are everywhere!

- strings behave like lists of characters
  - `len`, indexing, `for` loops
- `range` may be cast to a list

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

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 = open("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
```

Exercise

- Recall the `hours.txt` data:

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

- Recreate the `Hours` program from lecture in Python:

```
Susan worked 31.4 hours, 7.85 / day, 2 days above average
Brad worked 36.8 hours, 7.36 / day, 2 days above average
Jenn worked 39.5 hours, 7.9 / day, 4 days above average
```

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 open("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!)