Regular Expressions (Not Gamebox :( )

***If there's anything Upsorn can do to help prep for the final, send her an email

***Just don't ask what the questions will be--anything else, she's looking to see what we need and what we want to work on

Useful when we don't have well-structured data

Let's us recognize patterns, what the data represents, etc

If we are looking for phone numbers, we can grab anything formatted +1 (###) ### - ####

Very specific symbols for "regular expressions" (patterns) recognized by computer, in slides

An incomplete table of these symbols:

Note: the square bracket means "any one of the things in here", like a visual "or"

Syntax	What it means
[abc]	Either "a" or "b" or "c"
[a-z]	Any lowercase letter
[a-z0-9]	Any lowercase letter or digit
.	Any single character
\.	A period (same escape as in \' and \")
*	Everything ("0 to many")
?	Either 0 or 1 (see exercise)
+	Everything but 0 ("1 to many")

We use these 'regular expressions' both to find data of a specific format and to verify that our data is of a specific format we are looking for

CSV-files are "structured", which makes them easy to work with; not everything is like that

Start regular expressions with "r", meaning "raw data", signalling to the computer to compile the data before reading it (otherwise it'll get confused)

Next, put the pattern we are looking for in quotes

Regular expressions are used alongside the "re" library, must be imported

Next, we must compile the expression we are looking for

We actually don't always need to compile, but it nevers hurts to, so we'll just always do it

Things regex can do:

regex.search(text) returns the first occurrence of that object

regex.findall(text) returns every occurrence of that object

Once we find the object, it will look weird, and it comes with functions

object_name.group() shows us the string that matched

object_name.start() shows us the index of the first character

object_name.end() shows us the index of the last one

To watch our regular expressions grab data (and thus practice writing them), go to http://regexr.com/

An exercise which uses the Simpsons' phonebook (http://www.cs.virginia.edu/~up3f/cs1110/practice-of-the-day/simpsons_phone_book.txt):

```
# always import to start
import re
# a phone number takes the form digit, digit, digit, dash, digit, digit, digit,
digit
# in a regular expression, that looks like r"[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]"
# each time we write [0-9], we are saying "any single digit between 0 and 9,
inclusive"

# there's a problem: sometimes, it starts with four digits
# putting a question mark after it says, if it's there, take it, if it's not
there, no problem
# it's either there 0 times or 1 time
# new expression r"[0-9]?[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]"

# in function form
def find_all_phone_numbers(filename, regex):
   infile = open(filename, 'r')
   for line in infile:
       obj = regex.search(line)
       # find all would also work in this example
       # we know there's only one number on each line, but if we didn't, we'd need
to use .findall()
       if obj != None:
           print(obj, obj.group(), obj.start(), obj.end())
           # this prints a mess!!!
           # obj.group() is the part that gave us what we want, just the phone
number
# call the function for our file and expression
find_all_phone_numbers("simpsons_phone_book",
re.compile(r"[0-9]?[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]"))
```