Regular Expressions (for real this time)

March 20, 2012

Today

- Regular Expressions
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Regular Expression Demo

- In `ACT2-7.py`, Run `nameGame` with your name

```
def nameGame(name):
 name = name.lower()
 c = name[0]
 if (c == 'a') or (c == 'e') or (c == 'i') or (c == 'o') or (c == 'u'):
   suffix = name
 else:
   suffix = name[1:len(name)]

 myStr = name + ' ' + name + ' bo b' + suffix + '\n'
 myStr = myStr + 'banana fana fo f' + suffix + '\n'
 myStr = myStr + 'me mi mo m' + suffix + '\n'
 myStr = myStr + name
 return myStr
```

Regular Expression Demo

- Save `redemo.py` and open it in `IDLE`
- Press `F5`
- Copy the string from `nameGame` into the middle box
- Click `'Highlight all matches'`
- Type the suffix of your name in the top box

Do Task 1

Regular Expressions

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets		

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter		

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace		

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit		

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.		

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.		

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character		

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`.`	

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`.`	`T h e ' ' q u ...`

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`\s.+\s`	

Regular Expressions

`The qu1ck brown fox jumped over the lazy d0g.`

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`\s.+\s`	`qu1ck brown fox ...`

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`\s.+\s`	`qu1ck brown fox ...`

Do Task 2.1

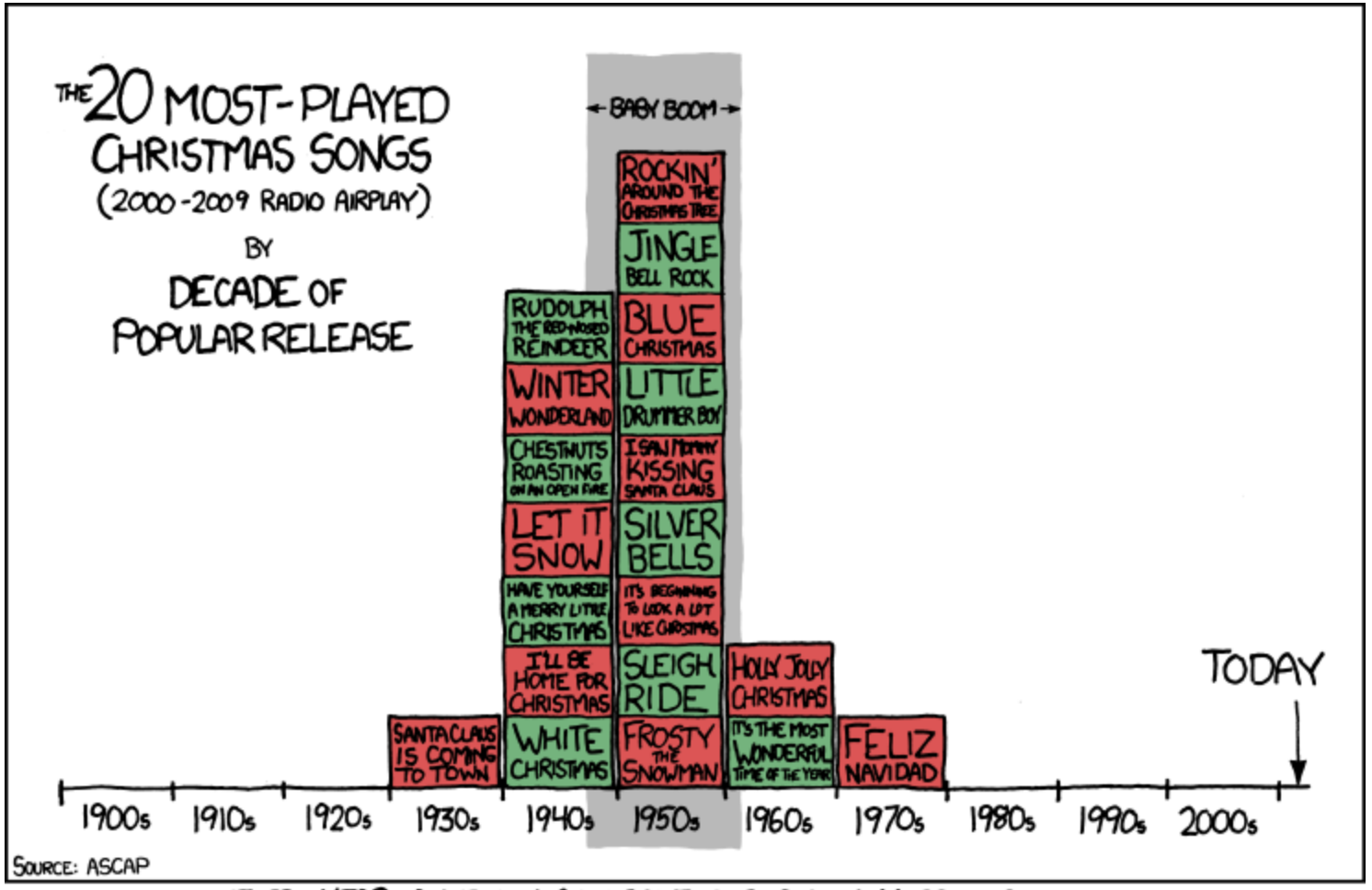
Break

david mccandless
informationisbeautiful.net

source: top ten lists from 120 music websites, magazines & retailers
via metacritic.com/feature/music-critic-top-ten-lists-best-albums-of-2011

Break (Cont'd)

http://xkcd.com/988/



EVERY YEAR, AMERICAN CULTURE EMBARKS ON A MASSIVE PROJECT TO CAREFULLY RECREATE THE CHRISTMASES OF BABY BOOMERS' CHILDHOODS.

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any letter	`o\w`	`ow ox ov`
\s	Match any whitespace	`e\s\w`	`'e q' 'e l'`
\d	Match any digit	`\d\w`	`1c 0g`
+	Match **one or more** of the previous things.	`\s\w+\s`	`The brown fox ...`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`The qu1ck brown ...`
.	Match any character	`\s.+\s`	`qu1ck brown fox ...`

Regular Expressions

Just the beginning… see the 'PythonRE' link for lots more:

Pattern	Description
^	Matches beginning of line.
$	Matches end of line.
.	Matches any single character except newline. Using m option allows it to match newline as well.
[...]	Matches any single character in brackets.
[^...]	Matches any single character not in brackets
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more occurrence of preceding expression.
re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a| b	Matches either a or b.
(re)	Groups regular expressions and remembers matched text.
…	…

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Data Structures

Lists

‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’

Data Structures

Lists

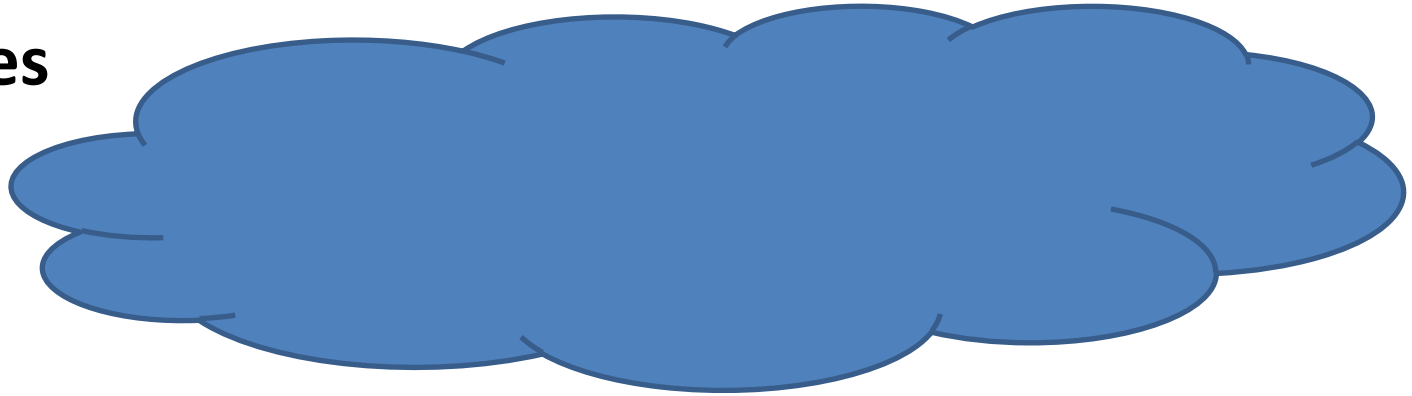
content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

Data Structures

Lists

content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

Dictionaries



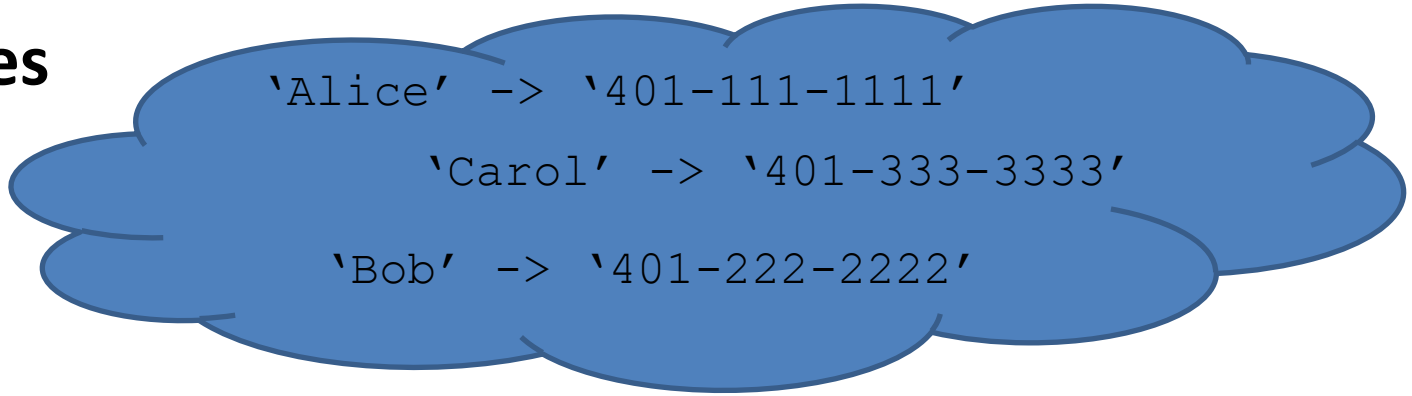
Data Structures

Lists

content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

Dictionaries

keys & values



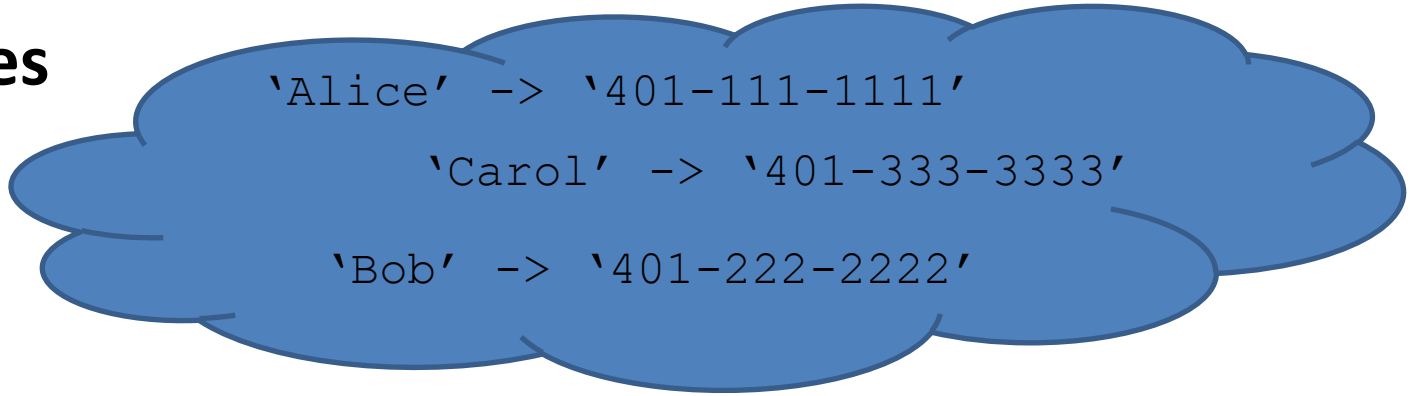
Data Structures

Lists

content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

Dictionaries

keys & values



Iterators

Match Objects

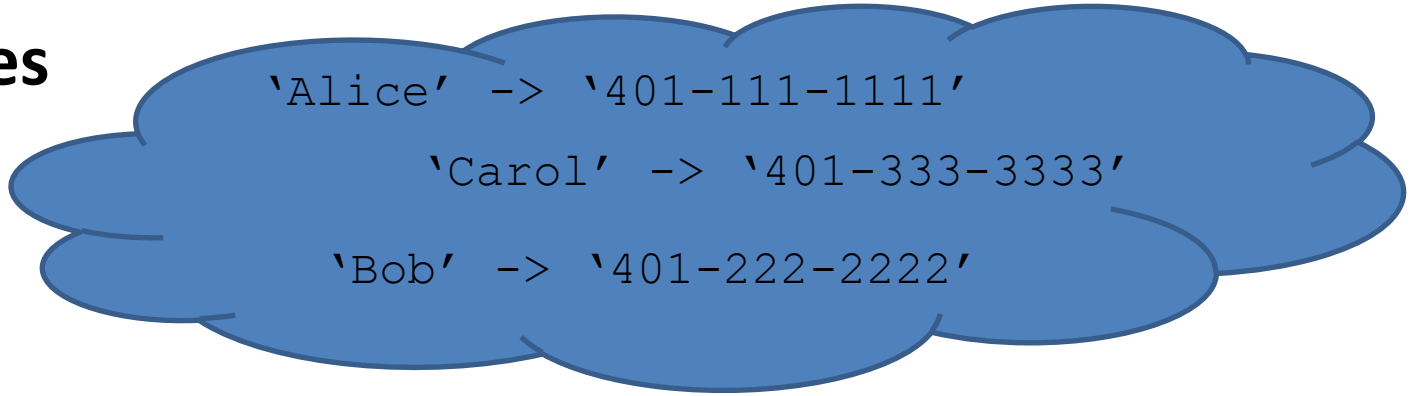
Data Structures

Lists

content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

Dictionaries

keys & values



Iterators

Match Objects

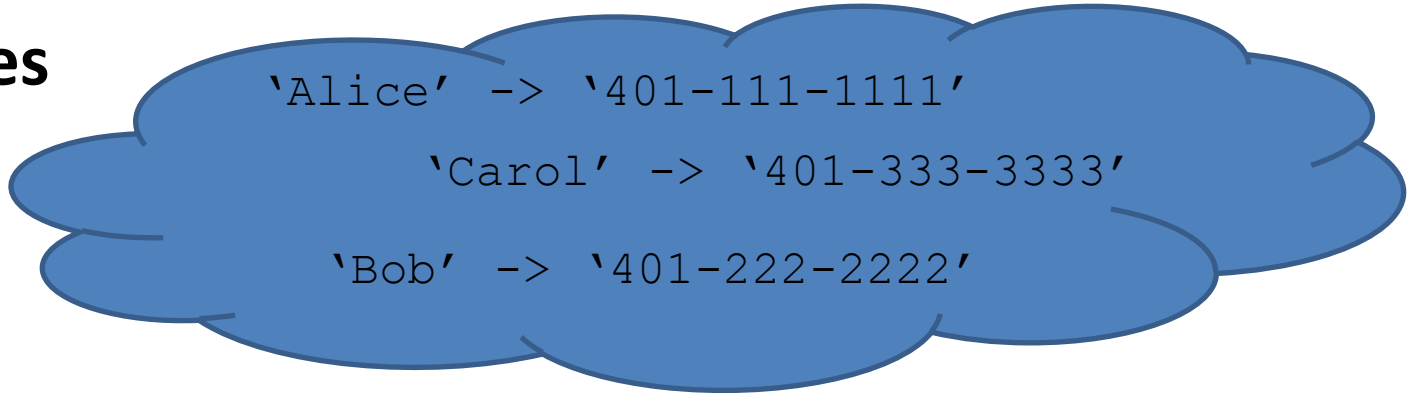
Data Structures

Lists

content	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'
indices	0	1	2	3	4	5	6	7	8	9

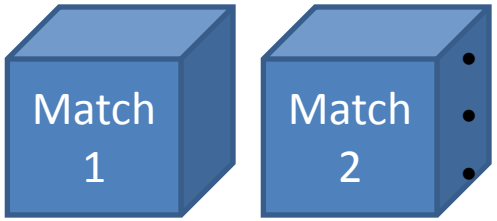
Dictionaries

keys & values



Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

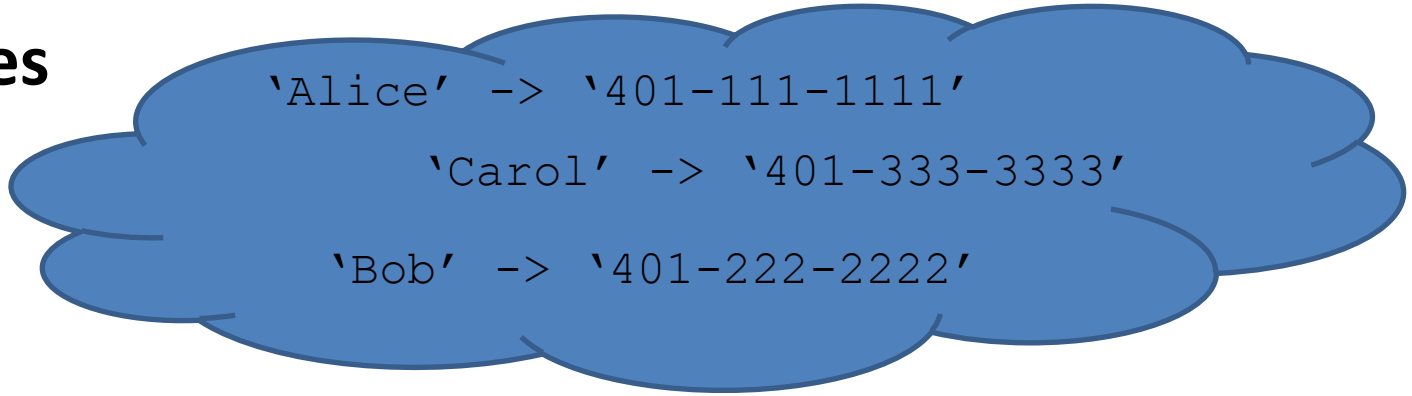
Data Structures

Lists

content	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'
indices	0	1	2	3	4	5	6	7	8	9

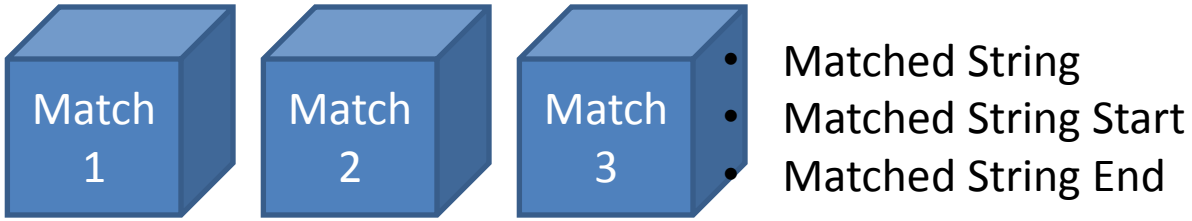
Dictionaries

keys & values



Iterators

Match Objects



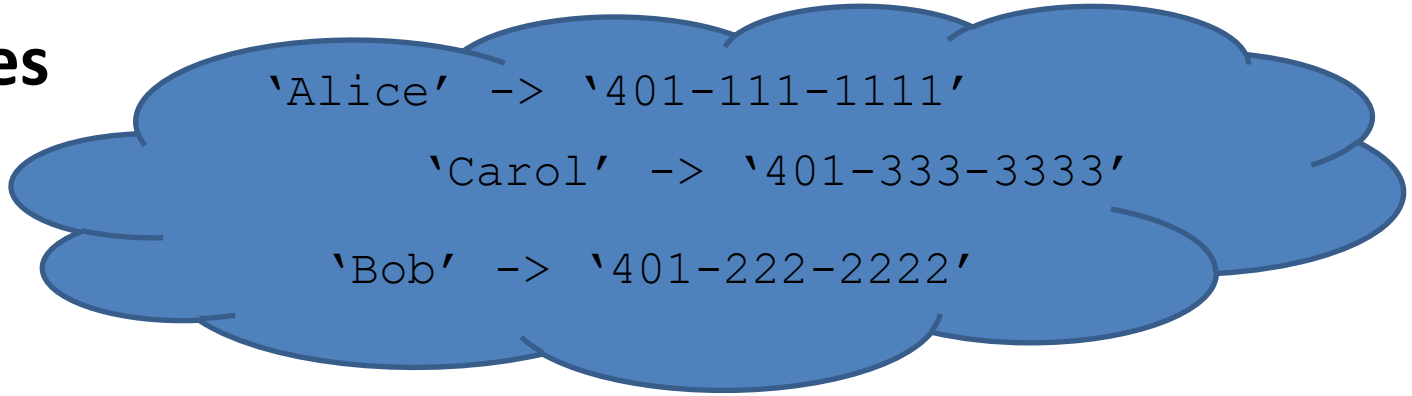
Data Structures

Lists

content	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'
indices	0	1	2	3	4	5	6	7	8	9

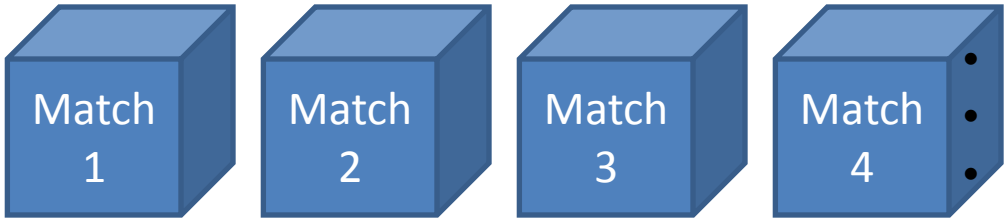
Dictionaries

keys & values



Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

Iterators

The cat in the hat sat on a mat

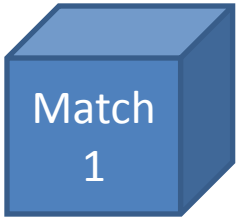
Regular Expression:'\wat'

Iterators

The cat in the hat sat on a mat

Regular Expression:'\wat'

Iterator

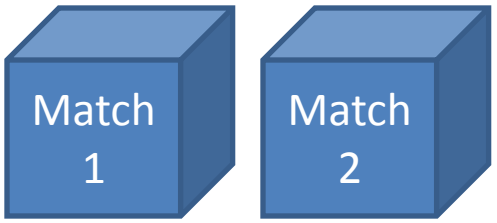


Iterators

The cat in the hat sat on a mat

Regular Expression:'\wat'

Iterator

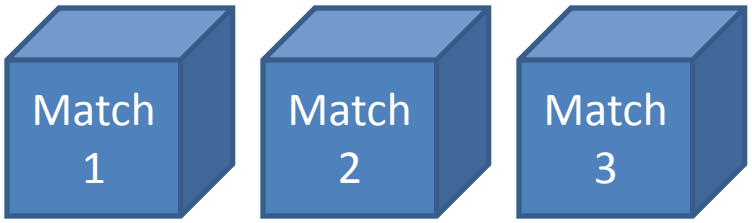


Iterators

The cat in the hat sat on a mat

Regular Expression:'\wat'

Iterator

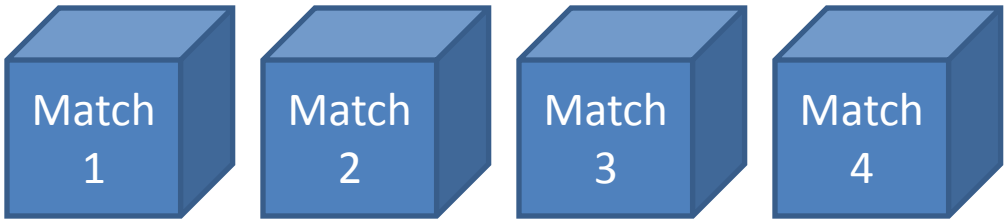


Iterators

The cat in the hat sat on a mat

Regular Expression:'\wat'

Iterator



Using Regular Expressions in Python

```
def printRegEx(regex,myStr):
    '''Prints all occurrences of the regular expression.'''

    myIter = re.finditer(regex,myStr) # Iterator!

    # This iterator contains MATCHES of the regex.
    # The following functions work on regex matches:
    #  group(0): returns the string that matches the regex
    #  start(0): the starting position of the string in myStr
    #  end(0): the ending position of the string in myStr

    # For loops work on iterators in addition to lists
    # Each match of the regex in myStr is stored in
    # a variable called 'occ'
    for occ in myIter:
        print 'matches',occ.group(0), \
            'at positions',occ.start(0),'-',occ.end(0)
    return
```

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Random Numbers

```
>>> import random
>>> random.random()
0.9367803948981343 6
```

Random Numbers

```
>>> import random
>>> random.random()
0.9367803948981343 6
```

`random.random()` returns a `float` between `0.0` (inclusive) and `1.0` (exclusive)

Write a program that reports the smallest number you've seen after running `random.random()` 20 times.

Random Numbers

```
import random

def smallestAfter20():
    '''Returns the smallest random # after 20 calls'''
    minval = 1.0
    for i in range(0,20):
        val = random.random()
        if val < minval:
            minval = val
    return minval
```

Write a program that reports the smallest number you've seen after running `random.random()` 20 times.

Random Numbers

```
import random

def smallestAfter20():
    '''Returns the smallest random # after 20 calls'''
    minval = 1.0
    for i in range(0,20):
        val = random.random()
        if val < minval:
            minval = val
    return minval
```

For HW: How would you simulate coin-flipping?

Write a program that reports the smallest number you've seen after running `random.random()` 20 times.

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- New data structure: *Iterator*
- Random Numbers

Next Few Weeks

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
3/18	3/19	3/20	3/21	3/22	3/23	3/24
		HW2-4 Due **HW2-5 Out**		**Project 2 Out**		
3/25	3/26	3/27	3/28	3/29	3/30	3/31
Spring Break! Wooooooo!						
4/1	4/2	4/3	4/4	4/5	4/6	4/7
		HW2-5 Due		**Proposal Due**		
4/8	4/9	4/10	4/11	4/12	4/13	4/14
				Project 2 Due		