Regular Expressions

Nov 3, 2015

Project Proposal Reminders

- Don’t forget to include the skeleton code
  - Define the functions you are using
  - Comments telling us the inputs and outputs

- Numbered list of steps
  - Describe how functions compute
  - If you’re testing a hypotheses, you should explain how the functions help test it.

Today

- Finish ACT 2-7
- Regular Expressions
- Using regular expressions in Python
- If we have time, a new data structure: *Iterator*

Today

- Finish ACT 2-7
- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- If we have time, a new data structure: *Iterator*

Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

```
myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file\n')
myFile.write(str(myNum))
myFile.write('\n')
myFile.close()
```

Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

```
myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file\n')
myFile.write(str(myNum))
myFile.write('\n')
myFile.close()
```

```
this is an output file
1
```

Distance Matrix

Do Task 4

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

```
myNum = 1
myFile = open('output.csv','w')
myFile.write('this is an output file\n')
myFile.write(str(myNum))
myFile.write('\n')
myFile.close()
```

```
this is an output file
1
```

Generating the file

```
#For each file, create a string row with all the values in the
#corresponding list row in distMatrix, with commas in between
for i in range(0,len(FILE_LIST))
     row = '' + FILE_LIST[i]

     # Loop through the columns in the current list row
     for val in distMatrix[i]:
          row = row + ',' + str(val)

     #At this point, we created our string row.
     #We want to write this row into our csv
     outFile.write(row)

     #Need a newline at the end of each string row
     outFile.write('\n')

# Finalize the new file by closing it
outFile.close()
```

Distance Matrix

This matrix looks kind of familiar...

Instead of printing to the screen, write it to a file in CSV (comma-separated value) format.

Open the CSV file in Google Spreadsheets. Use conditional formatting to look for pattern

Do Task 5

What’s Your Answer?

Discern the **Outlier**:
The one book that is NOT in the series of the others.

File	Title	Series	Author
file1.txt			
file2.txt			
file3.txt			
file4.txt			
file5.txt			
file6.txt			

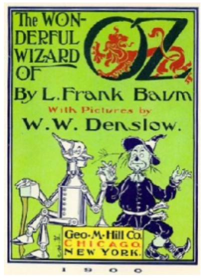
What’s Your Answer?

Discern the **Outlier**:
The one book that is NOT in the series of the others.

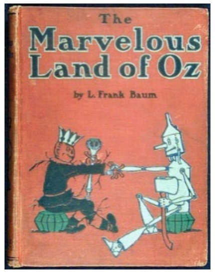
File	Title	Series	Author
file1.txt	Wonder Wizard of Oz	Oz	
file2.txt	Alice’s Adventures in Wonderland	Alice in Wonderland	
file3.txt	Dorothy and the Wizard in Oz	Oz	
file4.txt	Emerald City of Oz	Oz	
file5.txt	Royal Book of Oz	Oz	
file6.txt	Glinda of Oz	Oz	

The Wizard of OZ

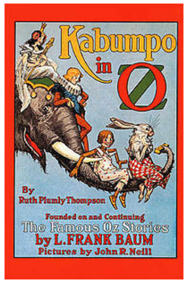
- About 40 Books, written by 7 different authors

#1 

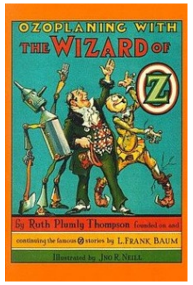
…

#14 

#15

#16 

…

#33 

Lyman Frank Baum
(1856-1919)

Published in
1921

Ruth Plumly Thompson

http://www.ssc.wisc.edu/~zzeng/soc357/OZ.pdf

What’s Your Answer?

Discern the **Outlier**:
The one book that is NOT in the series of the others.

File	Title	Series	Author
file1.txt	Wonder Wizard of Oz	Oz	Lyman Frank Baum
file2.txt	Alice’s Adventures in Wonderland	Alice in Wonderland	Lewis Carroll
file3.txt	Dorothy and the Wizard in Oz	Oz	Lyman Frank Baum
file4.txt	Emerald City of Oz	Oz	Lyman Frank Baum
file5.txt	Royal Book of Oz	Oz	Ruth Plumly Thompson
file6.txt	Glinda of Oz	Oz	Lyman Frank Baum

What’s Your Answer?

	file1.txt	file2.txt	file3.txt	file4.txt	file5.txt	file6.txt
file1.txt	0	0.16824579	0.08785724	0.08832557	0.13960696	0.10485192
file2.txt	0.16824579	0	0.1688084	0.1623172	0.18100267	0.17304682
file3.txt	0.08785724	0.1688084	0	0.07196412	0.11099609	0.07941847
file4.txt	0.08832557	0.1623172	0.07196412	0	0.13234903	0.09274778
file5.txt	0.13960696	0.18100267	0.11099609	0.13234903	0	0.14705212
file6.txt	0.10485192	0.17304682	0.07941847	0.09274778	0.14705212	0

Busted!

File	Title	Series	Author
file1.txt	Wonder Wizard of Oz	Oz	Lyman Frank Baum
file2.txt	Alice’s Adventures in Wonderland	Alice in Wonderland	Lewis Carroll
file3.txt	Dorothy and the Wizard in Oz	Oz	Lyman Frank Baum
file4.txt	Emerald City of Oz	Oz	Lyman Frank Baum
file5.txt	Royal Book of Oz	Oz	Ruth Plumly Thompson
file6.txt	Glinda of Oz	Oz	Lyman Frank Baum

Tools you've learned

- Reading and writing files **(ACT 2-2, 2-7)**
- String processing: split(), find(), etc. **(ACT 2-2, 2-6)**
- Lists and dictionaries **(ACT 2-5)**
- Iterating over data: two approaches to for-loops **(HW 2-4)**
- Summaries statistics like counts, averages, min/max **(ACT 2-3)**

Play Time!

Play Time!

Go to http://regexpal.com/ and
copy this text into the lower box:

```
Ahab was born in 1802.  Starbuck, in '98 --- 1998, that is.  And (as
everyone knows), Ishmael was born in 2036 but died in 1879 (making him
[1879-2036=]-57 years old at the time of his death, or 48, depending on
how you count).  And the white whale was immortal.

Am I telling this story right?  Anyhow, each of them had access to a time
machine, a harpoon, and 20+ sheep for barter.  Including the whale.  As
was common in his era, Starbuck had his own name tattooed on the back of
his arm: "*$", it read, "Star-buck", a kind of pun.

Our story begins in interstellar space, in the year 2000 BC...
```

Try entering different things (letters, numbers, symbols, brackets) in the upper box.

Any surprises?

Play Time!

Go to http://regexpal.com/ and
copy this text into the lower box:

```
Ahab was born in 1802.  Starbuck, in '98 --- 1998, that is.  And (as
everyone knows), Ishmael was born in 2036 but died in 1879 (making him
[1879-2036=]-57 years old at the time of his death, or 48, depending on
how you count).  And the white whale was immortal.

Am I telling this story right?  Anyhow, each of them had access to a time
machine, a harpoon, and 20+ sheep for barter.  Including the whale.  As
was common in his era, Starbuck had his own name tattooed on the back of
his arm: "*$", it read, "Star-buck", a kind of pun.

Our story begins in interstellar space, in the year 2000 BC...
```

Do Task 1

Regular Expressions

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

Regular Expressions

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

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

Regular Expressions

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

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

Regular Expressions

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

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

Regular Expressions

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

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

Regular Expressions

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

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

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	[qjd]u	qu ju
\w	Match any alphanumeric	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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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 Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	`qu1ck fox over lazy`

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	` qu1ck fox over lazy`
*	Match **zero or more** of the previous things.		

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	` qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	`qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	` qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`
.	Match any character		

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	` qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`
.	Match any character	`.`	

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	[qjd]u	qu ju
\w	Match any alphanumeric	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	qu1ck fox over lazy
*	Match **zero or more** of the previous things.	\w\d*\w	Th qu 1c br ow fo ju mp ed ov er th la zy d0g
.	Match any character	.	T h e ‘ ‘ q u ...

Regular Expressions

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

Special Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	` qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`
.	Match any character	`\s.+\s`	

Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

Syntax	Meaning	R.E Example	Result
[ ]	Match anything between brackets	[qjd]u	qu ju
\w	Match any alphanumeric	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	qu1ck fox over lazy
*	Match **zero or more** of the previous things.	\w\d*\w	Th qu 1c br ow fo ju mp ed ov er th la zy d0g
.	Match any character	\s.+\s	qu1ck brown fox jumped over the lazy

Today

- Regular Expressions – a way to **match strings**
- Using regular expressions in Python
- If time, new data structure: *Iterator*

Regular Expressions

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

Do Task 2

Special Syntax	Meaning	R.E	
[ ]	Match anything between brackets	`[qjd]u`	`qu ju`
\w	Match any alphanumeric	`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`	`qu1ck fox over lazy`
*	Match **zero or more** of the previous things.	`\w\d*\w`	`Th qu 1c br ow fo ju mp ed ov er th la zy d0g`
.	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

If time…

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

Today

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

Data Structures

Lists

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

Data Structures

Lists

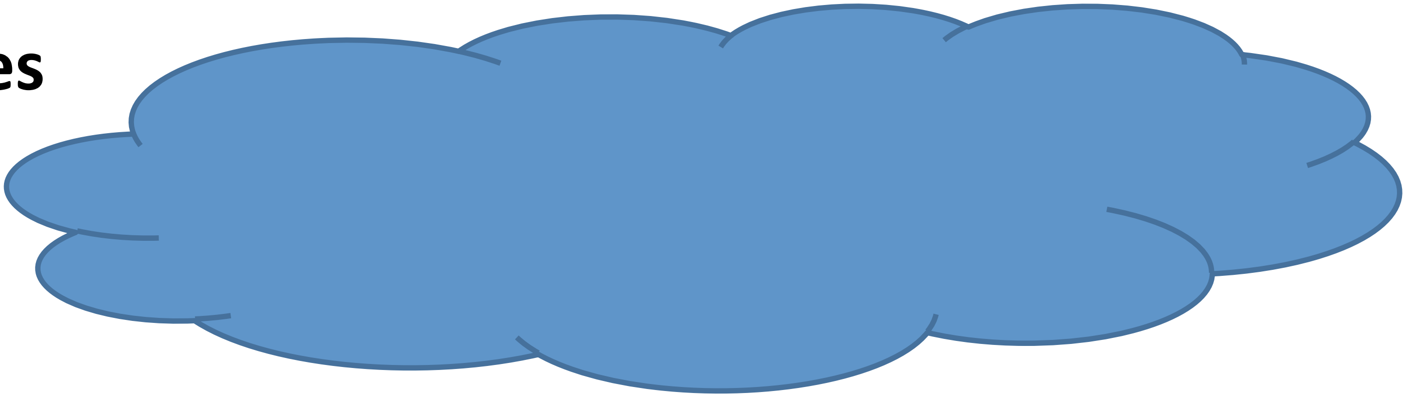
content	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’
indices	0	1	2	3	4	5	6	7	8	9

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



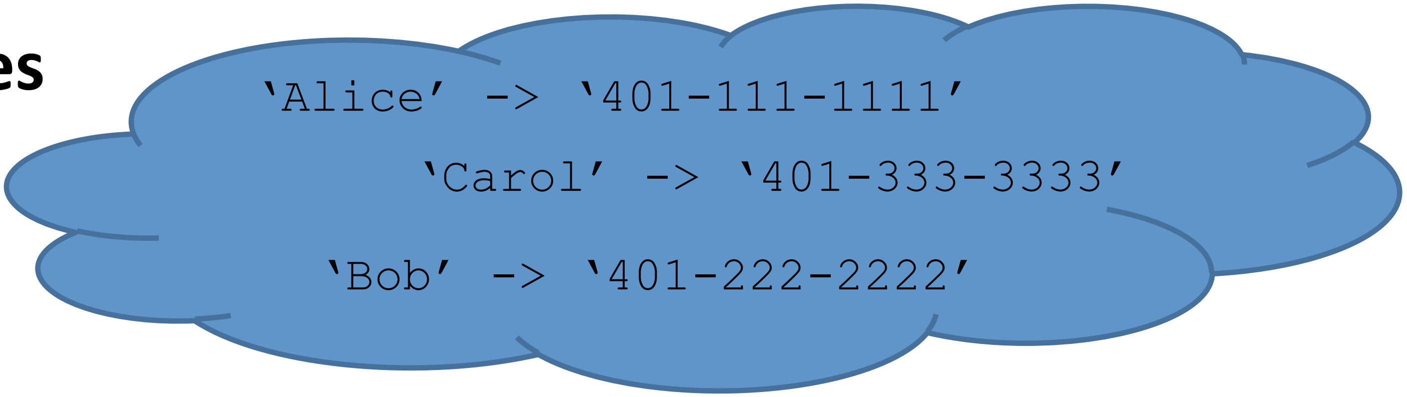
Data Structures

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



```
'Alice' -> '401-111-1111'
'Carol' -> '401-333-3333'
'Bob' -> '401-222-2222'
```

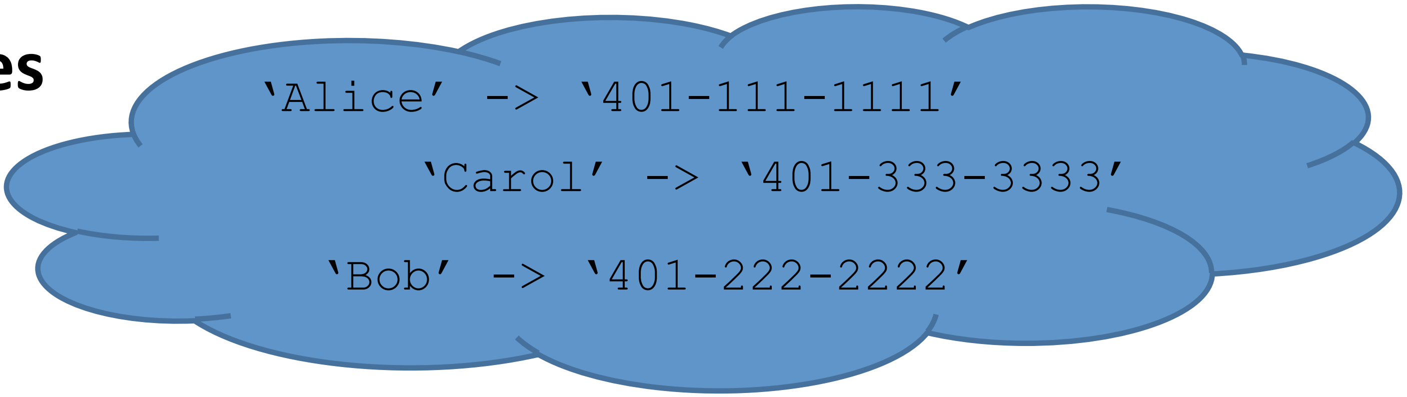
Data Structures

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

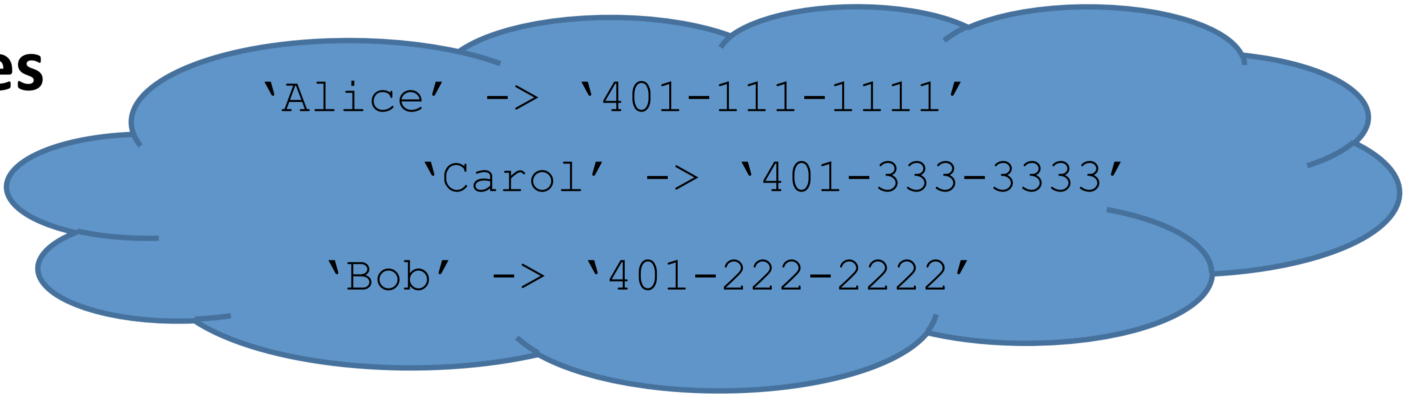
Data Structures

Lists

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

Dictionaries

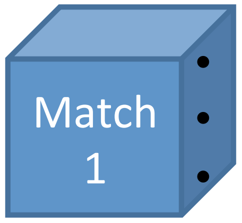
keys & values



```
'Alice' -> '401-111-1111'
'Carol' -> '401-333-3333'
'Bob' -> '401-222-2222'
```

Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

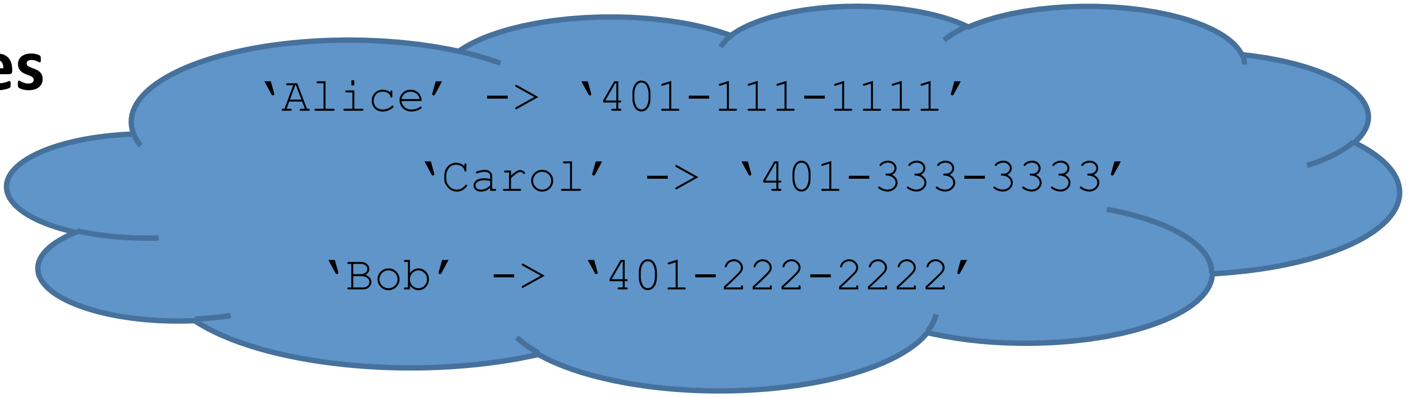
Data Structures

Lists

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

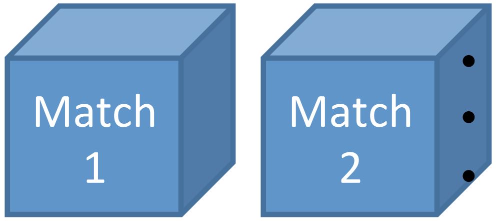
Dictionaries

keys & values



Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

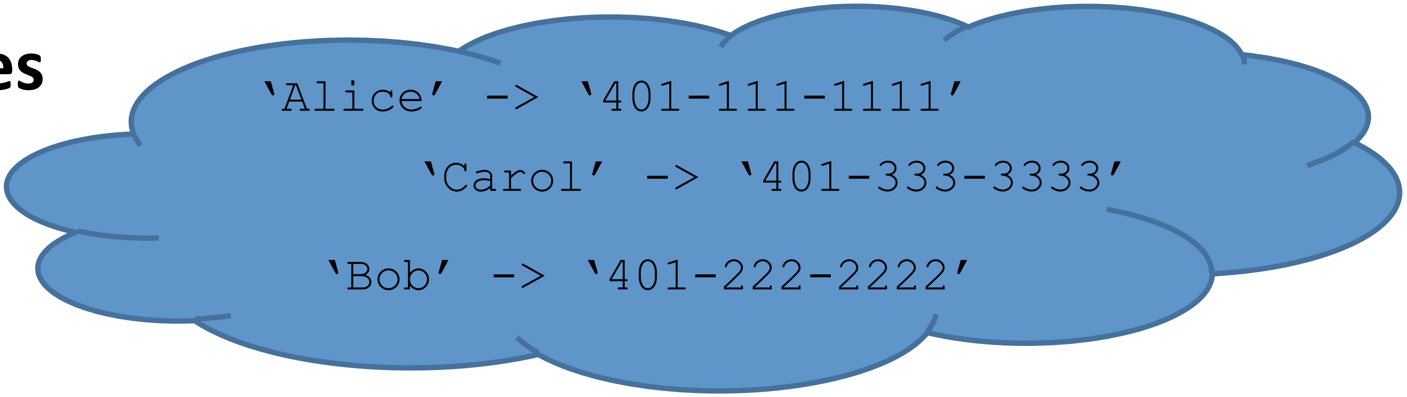
Data Structures

Lists

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

Dictionaries

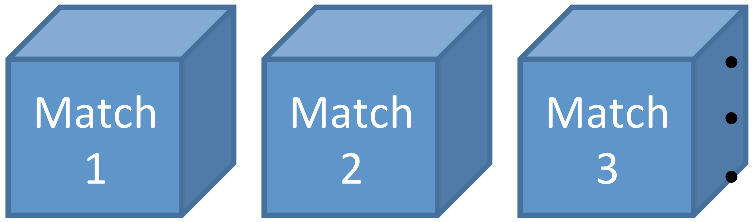
keys & values



```
‘Alice’ -> ‘401-111-1111’
    ‘Carol’ -> ‘401-333-3333’
  ‘Bob’ -> ‘401-222-2222’
```

Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

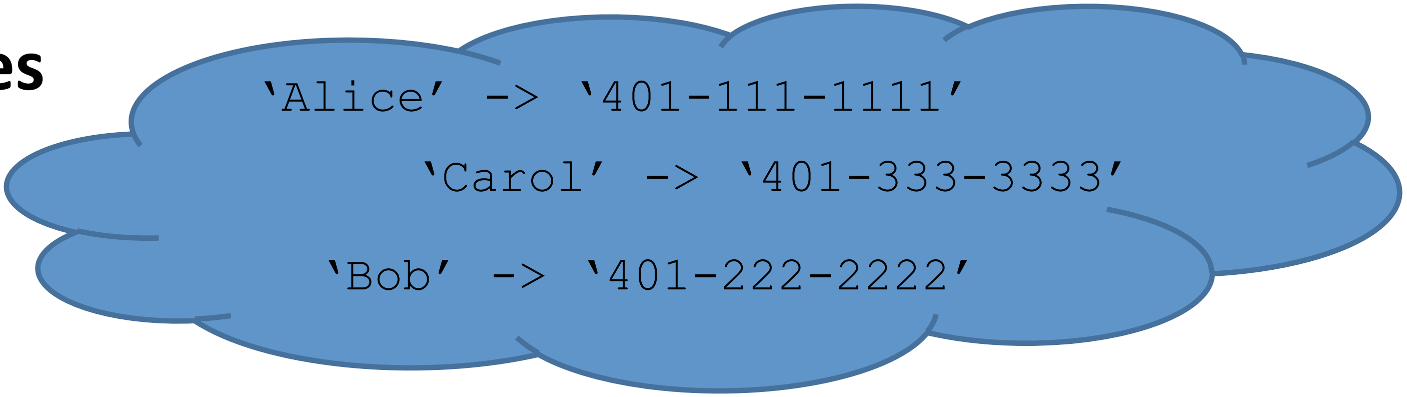
Data Structures

Lists

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

Dictionaries

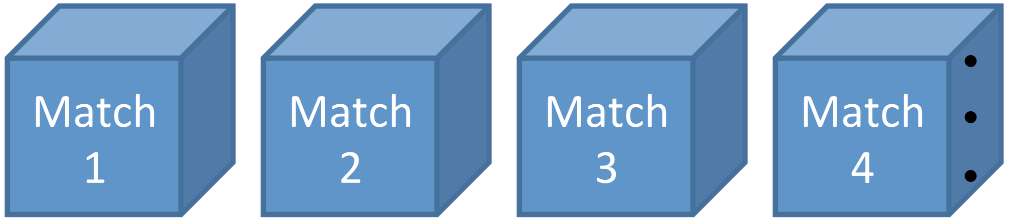
keys & values



```
'Alice' -> '401-111-1111'
'Carol' -> '401-333-3333'
'Bob' -> '401-222-2222'
```

Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

Iterators

`The cat in the hat sat on a mat`

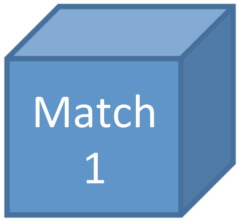
`Regular Expression:‘\wat’`

Iterators

The cat in the hat sat on a mat

`Regular Expression:'\wat'`

Iterator

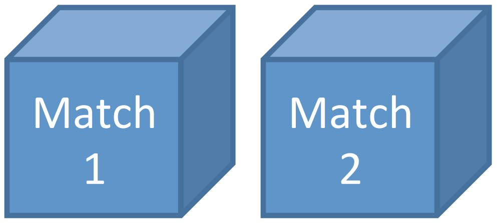


Iterators

The cat in the hat sat on a mat

`Regular Expression:'\wat'`

Iterator

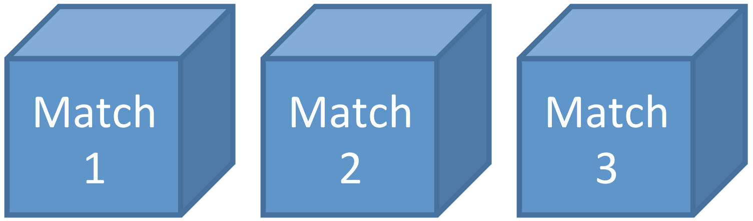


Iterators

The cat in the hat sat on a mat

`Regular Expression:'\wat'`

Iterator

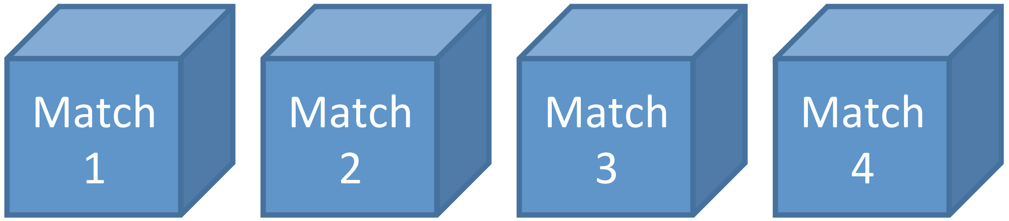


Iterators

`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
```