

# More Regular Expressions

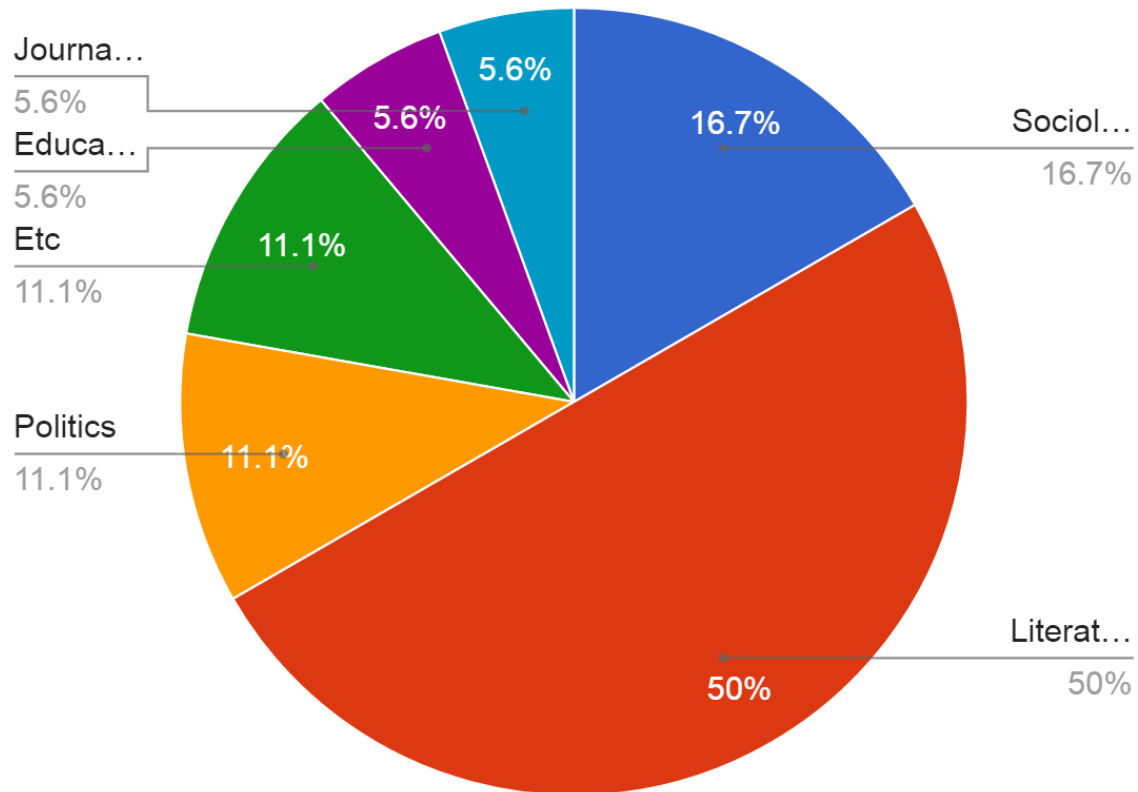
Apr 5 2016

# Class Today

- Project 2 Statistics
- Continue with regular expressions
- Match iterators
- More special characters
- Working with match *groups*
- Let's talk about the project first

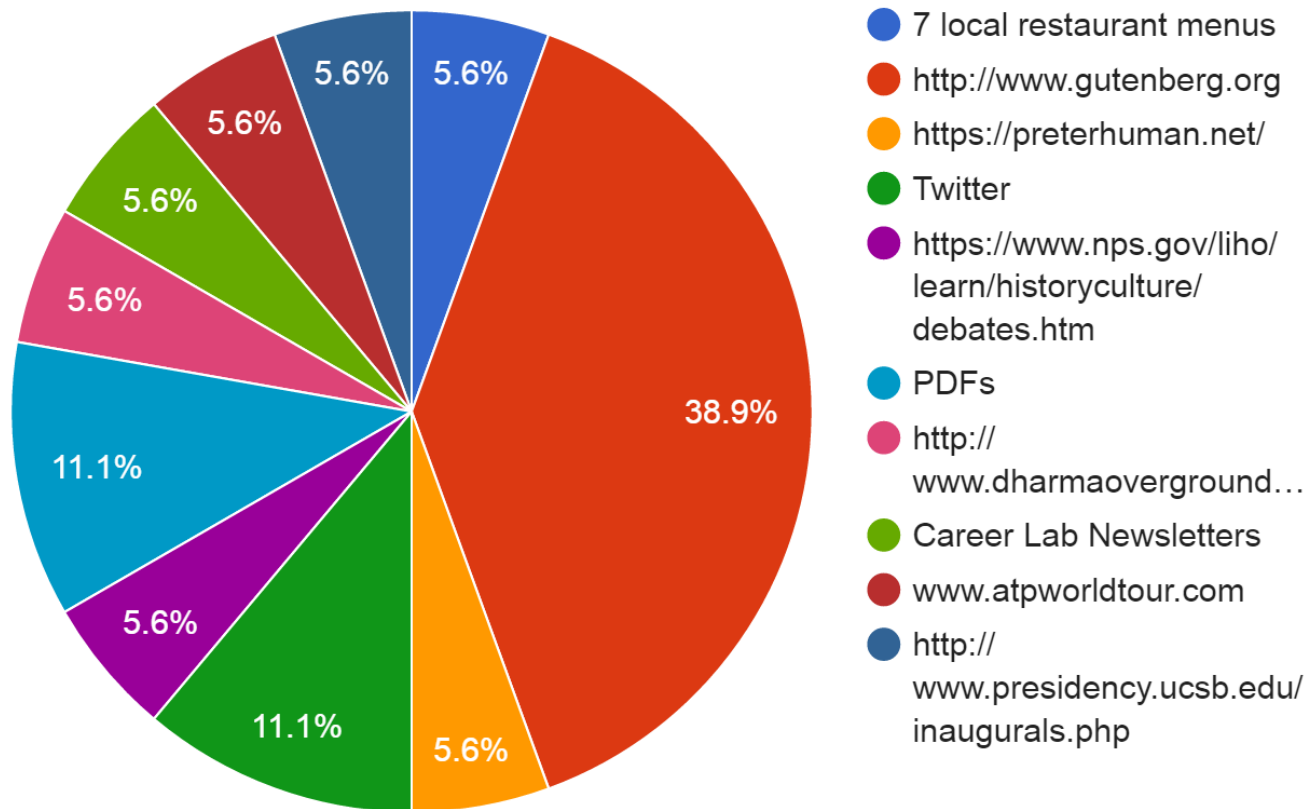
# Project 2 Statistics

**Count of Category**



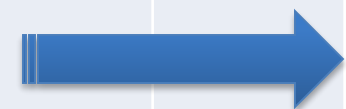
# Project 2 Statistics

Count of Data Source



# Calendar

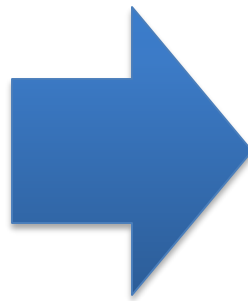
| Sun  | Mon  | Tues                                               | Wed  | Thurs                                                                      | Fri  | Sat  |
|------|------|----------------------------------------------------|------|----------------------------------------------------------------------------|------|------|
| 4/3  | 4/4  | 4/5                                                | 4/6  | 4/7                                                                        | 4/8  | 4/9  |
|      |      |                                                    |      | <b>HW 3-1 out</b><br><b>Project 2 due</b><br><b>Work on your projects!</b> |      |      |
| 4/10 | 4/11 | 4/12                                               | 4/13 | 4/14                                                                       | 4/15 | 4/16 |
|      |      | <b>HW 3-1 due</b><br><b>Work on your projects!</b> |      | <b>Final Proj Out</b>                                                      |      |      |
| 4/17 | 4/18 | 4/19<br><b>HW 3-2 out</b>                          |      |                                                                            |      |      |



# Loading dictionary words into Python

**dictionary.txt**

```
Aam\nAaronic\nAaronical\nAb\nAbaca\nAbacinate\n
```



**dict**

| Word      | Freq. |
|-----------|-------|
| Aam       | 1     |
| Aaronic   | 1     |
| Aaronical | 1     |
| Ab        | 1     |
| Abaca     | 1     |
| Abacinate | 1     |

# Loading dictionary words into Python

```
# read contents of dictionary
file = open('dictionary.txt', 'r')
contents = file.read()
file.close()

# get list of words
words = contents.split('\n')

# add all words into dict
# note that dictionaries have unique keys (not duplicated)
dict = {}
for word in words[:10]:
    dict[word] = 1
```

To search in the dictionary:

key in dict      (evaluates to `True` or `False`)

# Class Today

- Continue with Regular Expressions
- Match iterators
- More special characters
- Working with match *groups*

# Rules for Regular Expression Search

- Sequentially visits each character in the search string
- Finds the longest match available.
- Restart the search at the position right next to the last match

`\wat`

The **cat** in the **hat**at **sat** on a **mat**

# Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

[illegible]

# Regular Expressions

The qu1ck brown fox jumped over the lazy d0g.

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

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                         | R.E Example | Result      |
|----------------|---------------------------------|-------------|-------------|
| [ ]            | Match anything between brackets | [qj d]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 quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                         | R.E Example | Result      |
|----------------|---------------------------------|-------------|-------------|
| [ ]            | Match anything between brackets | [qj d]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 quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                          | R.E Example | Result                                  |
|----------------|--------------------------------------------------|-------------|-----------------------------------------|
| [ ]            | Match anything between brackets                  | [qj d]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 <b>one or more</b> of the previous things. | \s\w+\s     | ' quick ' ' fox<br>' over ' '<br>lazy ' |
|                |                                                  |             |                                         |
|                |                                                  |             |                                         |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                           | R.E Example | Result                                              |
|----------------|---------------------------------------------------|-------------|-----------------------------------------------------|
| [ ]            | Match anything between brackets                   | [qj d]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 <b>one or more</b> of the previous things.  | \s\w+\s     | quick fox over<br>lazy                              |
| *              | Match <b>zero or more</b> of the previous things. | \w\d*\w     | Th qu 1c br ow<br>fo ju mp ed ov<br>er th la zy d0g |
|                |                                                   |             |                                                     |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                           | R.E Example | Result                                              |
|----------------|---------------------------------------------------|-------------|-----------------------------------------------------|
| [ ]            | Match anything between brackets                   | [qj d]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 <b>one or more</b> of the previous things.  | \s\w+\s     | quick fox over<br>lazy                              |
| *              | Match <b>zero or more</b> of the previous things. | \w\d*\w     | Th qu 1c br ow<br>fo ju mp ed ov<br>er th la zy d0g |
| .              | Match any character                               |             |                                                     |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                           | R.E Example | Result                                              |
|----------------|---------------------------------------------------|-------------|-----------------------------------------------------|
| [ ]            | Match anything between brackets                   | [qj d]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 <b>one or more</b> of the previous things.  | \s\w+\s     | quick fox over<br>lazy                              |
| *              | Match <b>zero or more</b> of the previous things. | \w\d*\w     | Th qu 1c br ow<br>fo ju mp ed ov<br>er th la zy d0g |
| .              | Match any character                               | .           |                                                     |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                           | R.E Example | Result                                              |
|----------------|---------------------------------------------------|-------------|-----------------------------------------------------|
| [ ]            | Match anything between brackets                   | [qj d]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        | lc 0g                                               |
| +              | Match <b>one or more</b> of the previous things.  | \s\w+\s     | quick fox over<br>lazy                              |
| *              | Match <b>zero or more</b> of the previous things. | \w\d*\w     | Th qu lc br ow fo<br>ju mp ed ov er th<br>la zy d0g |
| .              | Match any character                               | .           | T h e ' ' q u ...                                   |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Syntax | Meaning                                           | R.E<br>Example | Result                                              |
|--------|---------------------------------------------------|----------------|-----------------------------------------------------|
| [ ]    | Match anything between brackets                   | [qj d]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 <b>one or more</b> of the previous things.  | \s\w+\s        | quick fox over<br>lazy                              |
| *      | Match <b>zero or more</b> of the previous things. | \w\d*\w        | Th qu 1c br ow<br>fo ju mp ed ov<br>er th la zy d0g |
| .      | Match any character                               | \s.+ \s        | ' quick brown<br>fox jumped over<br>the lazy '      |

# Regular Expressions

The quick brown fox jumped over the lazy dog.

| Special Syntax | Meaning                                           | R.E     |                                                     |
|----------------|---------------------------------------------------|---------|-----------------------------------------------------|
| [ ]            | Match anything between brackets                   | [qj d]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 <b>one or more</b> of the previous things.  | \s\w+\s | quick fox over lazy                                 |
| *              | Match <b>zero or more</b> of the previous things. | \w\d*\w | Th qu 1c br ow<br>fo ju mp ed ov<br>er th la zy d0g |
| .              | Match any character                               | \s.+ \s | quick brown fox                                     |

Do Task 2

# Regular Expressions

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

| Pattern                | Description                                                           |
|------------------------|-----------------------------------------------------------------------|
| <code>^</code>         | Matches beginning of line.                                            |
| <code>\$</code>        | Matches end of line.                                                  |
| <code>.</code>         | Matches any single character except newline.                          |
| <code>[...]</code>     | Matches any single character in brackets.                             |
| <code>[^...]</code>    | Matches any single character not in brackets                          |
| <code>re*</code>       | Matches 0 or more occurrences of preceding expression.                |
| <code>re+</code>       | Matches 1 or more occurrence of preceding expression.                 |
| <code>re?</code>       | Matches 0 or 1 occurrence of preceding expression.                    |
| <code>re{ n}</code>    | Matches exactly n number of occurrences of preceding expression.      |
| <code>re{ n,}</code>   | Matches n or more occurrences of preceding expression.                |
| <code>re{ n, m}</code> | Matches at least n and at most m occurrences of preceding expression. |
| <code>a  b</code>      | Matches either a or b.                                                |
| <code>(re)</code>      | Groups regular expressions and remembers matched text.                |
| <code>...</code>       | ...                                                                   |

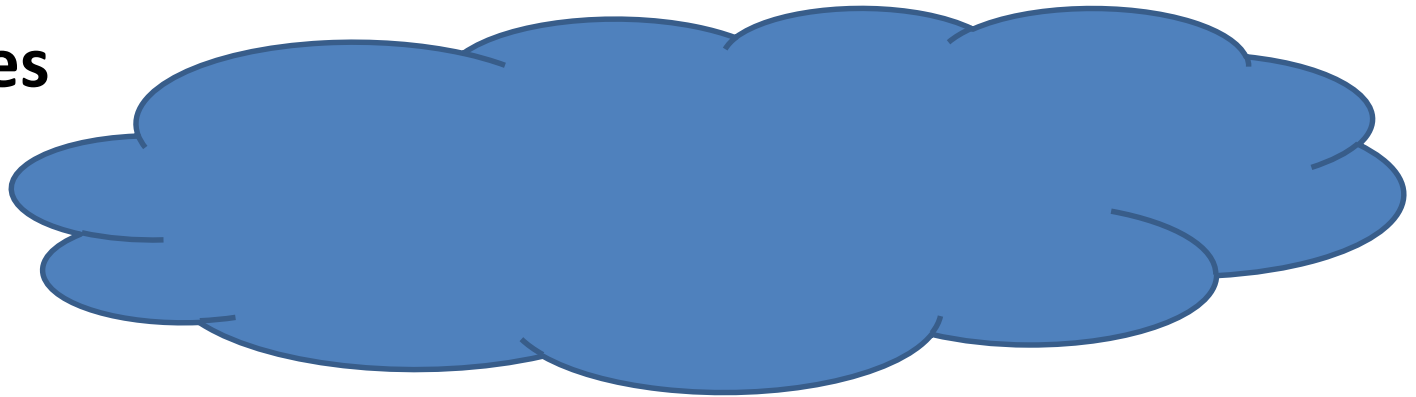
# Data Structures

## Lists

content  
indices

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

## Dictionaries



# Data Structures

## Lists

content  
indices

|     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' |
| 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  
indices

|     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' |
| 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

# Data Structures

## Lists

content  
indices

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

## Dictionaries

keys & values

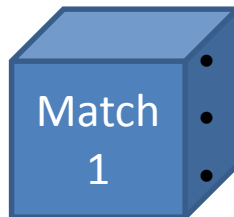
'Alice' -> '401-111-1111'

'Carol' -> '401-333-3333'

'Bob' -> '401-222-2222'

## Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

# Data Structures

## Lists

content  
indices

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

## Dictionaries

keys & values

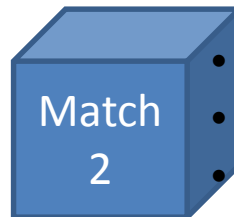
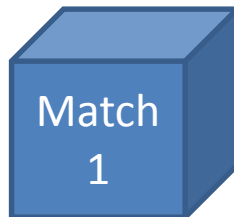
'Alice' -> '401-111-1111'

'Carol' -> '401-333-3333'

'Bob' -> '401-222-2222'

## Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

# Data Structures

## Lists

content  
indices

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

## Dictionaries

keys & values

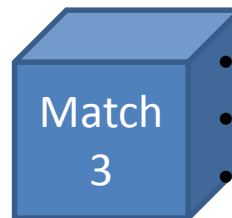
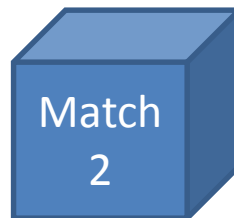
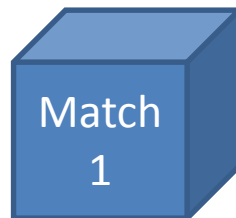
'Alice' -> '401-111-1111'

'Carol' -> '401-333-3333'

'Bob' -> '401-222-2222'

## Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

# Data Structures

## Lists

content  
indices

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

## Dictionaries

keys & values

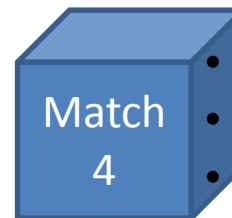
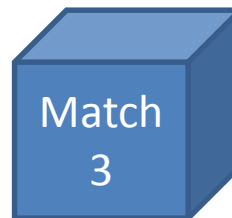
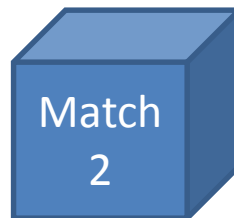
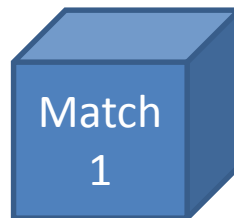
'Alice' -> '401-111-1111'

'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**

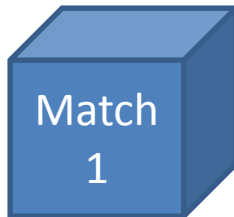
Regular Expression: `'\wat'`

# Iterators

The **cat** in the hat sat on a mat

Regular Expression: `'\wat'`

**Iterator**

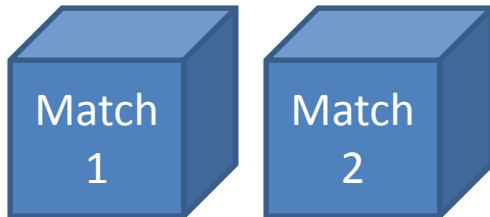


# Iterators

The cat in the **hat** sat on a mat

Regular Expression: `'\wat'`

**Iterator**

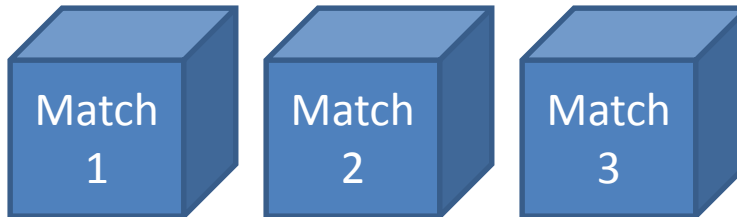


# Iterators

**The cat in the hat sat on a mat**

Regular Expression: `'\wat'`

**Iterator**

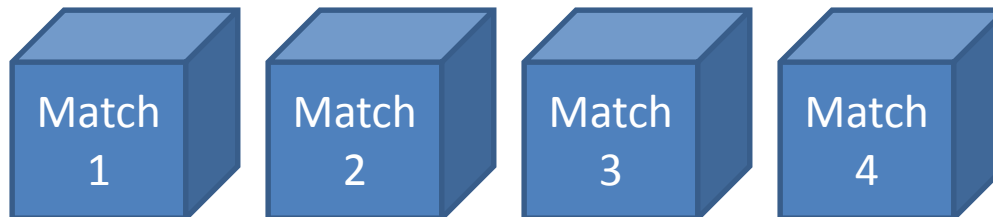


# Iterators

**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
```

# Search using RegEx in Python

- To use, type `import re` in your Python code, then function
  - `re.search('a\w+a', 'anica')`
    - Evaluates to `True`
      - You can find 'anica' in the string
  - `re.search('x\w+i', 'mechanical')`
    - Evaluates to `False`
      - You cannot find any substring starting with 'x' and finishing with 'i' in the string

**In groups,  
do Task 1**

# Match using RegEx in Python

- To use, type `import re` in your Python code, then function
  - `re.match('a\w+a', '1')`
    - Evaluates to `False`
      - The word is not in the specified form
  - `re.match('a\w+a', 'abracadabra')`
    - Evaluates to `True`
      - The word is in the specified form

**In groups,  
do Task 2**

# Task 2

```
for word in myList:  
    # print word only if it rhymes with 'ping pong'
```

# Task 2

```
for word in myList:  
    match = re.match('\w*ing\s*\w*ong', word)  
    if match:  
        print word
```

# Search and Match return MatchObjects

```
match = re.search('\s\w+\s', 'the cat jumped')
```



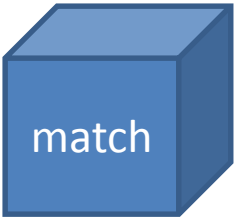
`' cat '`

Not really, you don't  
want the spaces

# Search and Match return MatchObject

**In groups,  
do Task 3**

```
match = re.search('\s(\w+)\s')
```

`if`  `:`

```
print match.group(0)  
print match.group(1)
```

```
' cat '  
'cat'
```

→ evaluates to `True`

→ Gives you back

0: the whole match

1...n: the corresp.  
parenthesis match

# Task 3

```
for word in myList:
    match = re.match('(\w*ing)\s*(\w*ong)', word)
    if match:
        print 'Whole match: ', match.group(0)
        print 'First sub-match: ', match.group(1)
        print 'Second sub-match: ', match.group(2)
```

# Match Iterators

- Before, we just checked to see whether a pattern existed in the text...
- Or if the text started with a pattern
- Let's now do some computing with those parts of the text that match

# Match Iterators

- Before, we just checked to see whether a pattern existed in the text...
- Or if the text started with a pattern
- Let's now do some computing with those parts of the text that match

# Data Structures

## Lists

content  
indices

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

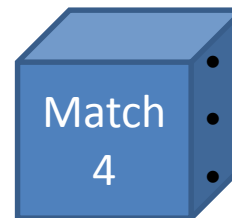
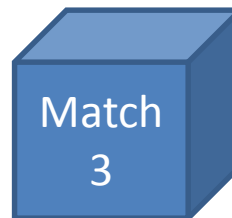
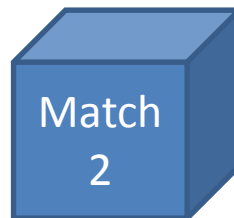
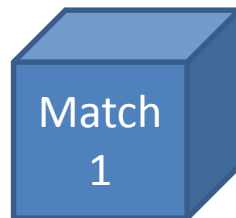
## Dictionaries

keys & values

| Key     | Value          |
|---------|----------------|
| 'alice' | '401-111-1111' |
| 'carol' | '401-222-2222' |
| 'bob'   | '401-333-3333' |

## Iterators

Match Objects



- Matched String
- Matched String Start
- Matched String End

# Matching iterative

## Do Task 4

```
def printRegex(regex, myStr):  
    '''Prints all occurrences of the regular expression'''  
  
    match_iterator = re.finditer(regex, myStr) # Iterator!  
  
    # This iterator contains MATCHES of the regex.  
    # The following functions work for each match:  
    #   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 is stored  
    # in the variable 'match', in sequence  
    for match in match_iterator:  
        print('matches', match.group(0), \  
              'at positions', match.start(0), '-', match.end(0))  
    return
```

| Pattern       | Description                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------|
| \s            | Match a whitespace character (except newline, if you use the "re.MULTILINE" option).                 |
| \w            | Match a letter, number, or underscore character.                                                     |
| \d            | Match a digit.                                                                                       |
| ^             | Matches beginning of string (or beginning of line with the "re.MULTILINE" option).                   |
| \$            | Matches end of string (or end of line with the "re.MULTILINE" option).                               |
| .             | Matches any single character except newline (also including newline with the "re.MULTILINE" option). |
| [aeiou]       | Matches any single character in brackets.                                                            |
| [A-Z]         | Matches any single character in the range indicated in the brackets.                                 |
| [^aeiou]      | Matches any single character not in brackets.                                                        |
| [aeiou]*      | Matches 0 or more occurrences of preceding expression.                                               |
| [aeiou]+      | Matches 1 or more occurrence of preceding expression.                                                |
| [aeiou]?      | Matches 0 or 1 occurrence of preceding expression.                                                   |
| [aeiou]{n}    | Matches exactly n number of occurrences of preceding expression.                                     |
| [aeiou]{n,}   | Matches n or more occurrences of preceding expression.                                               |
| [aeiou]{n, m} | Matches at least n and at most m occurrences of preceding expression.                                |
| a b           | Matches either a or b.                                                                               |
| ([aeiou]+)    | Groups regular expressions and remembers matched text.                                               |
| ...           | ...                                                                                                  |

# Check how we got the words from the *Online Plain Text English Dictionary*

```
# opens output file for writing
outputFile = open(outputFilename, 'w')

# for each letter
for letter in 'abcdefghijklmnopqrstuvwxyz':
    # download dictionary contents for the letter
    # urlLetter is the URL 'constructed' based on it
    remoteFile = urllib.request.urlopen(urlLetter)
    contents = remoteFile.read().decode('utf-8')
    remoteFile.close()

    # collect all the words (made only by characters)
    # that are inside '<B> </B>' and write to the output file
    for match in re.finditer('<[Bb]>(\w+)</[Bb]>', contents):
        outputFile.write(match.group(1))
        outputFile.write('\n')

outputFile.close()
```

# *group(1)...group(n)* also work for each match in the iterator

re\_tester.py

```
def showMatches(pattern, text):
    print('Text: ' + text)
    print('Patt: ' + pattern)
    match_iterator = re.finditer(pattern, text)
    for match in match_iterator:
        print('----- New Match -----')
        print('  match.start() = ' + str(match.start()))
        print('  match.end()   = ' + str(match.end()))
        print('  Whole match: ' + match.group(0))
        if match.lastindex is not None:
            print('  Match groups:')
            for i in range(1, match.lastindex+1):
                print('    match.group(' + str(i) + '): '
                    + match.group(i))
```

# Match Iterators

- Compare the output of these expressions:
  - `showMatches ( 'a\w+b.z' , 'foobarbaz' )`
  - `showMatches ( ' (a) \w+ (b.z) ' , 'foobarbaz' )`