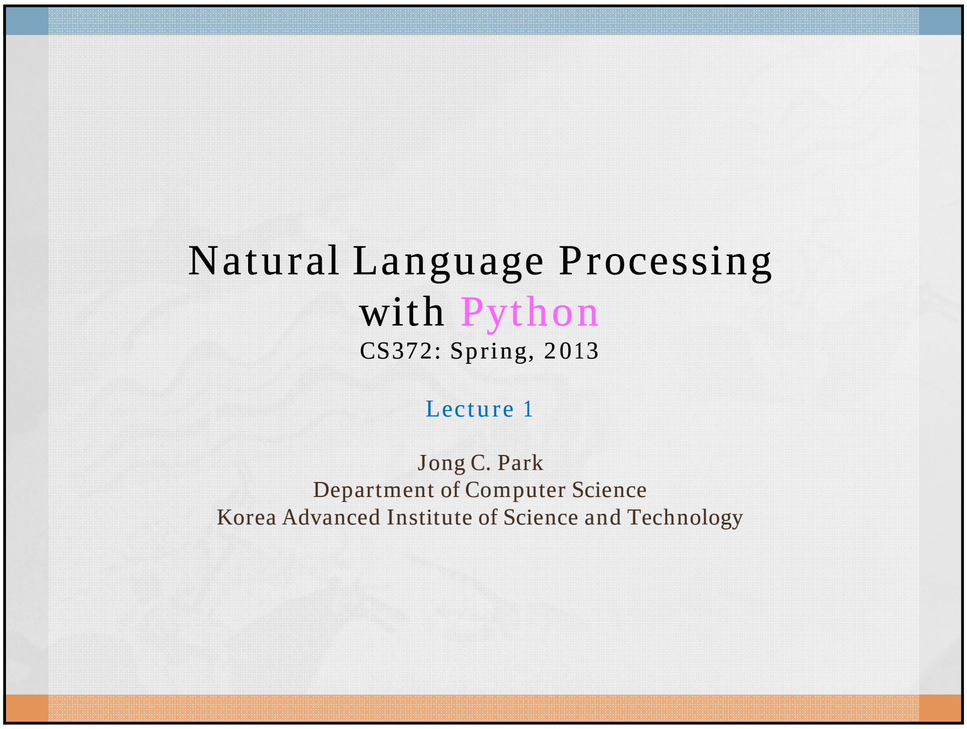
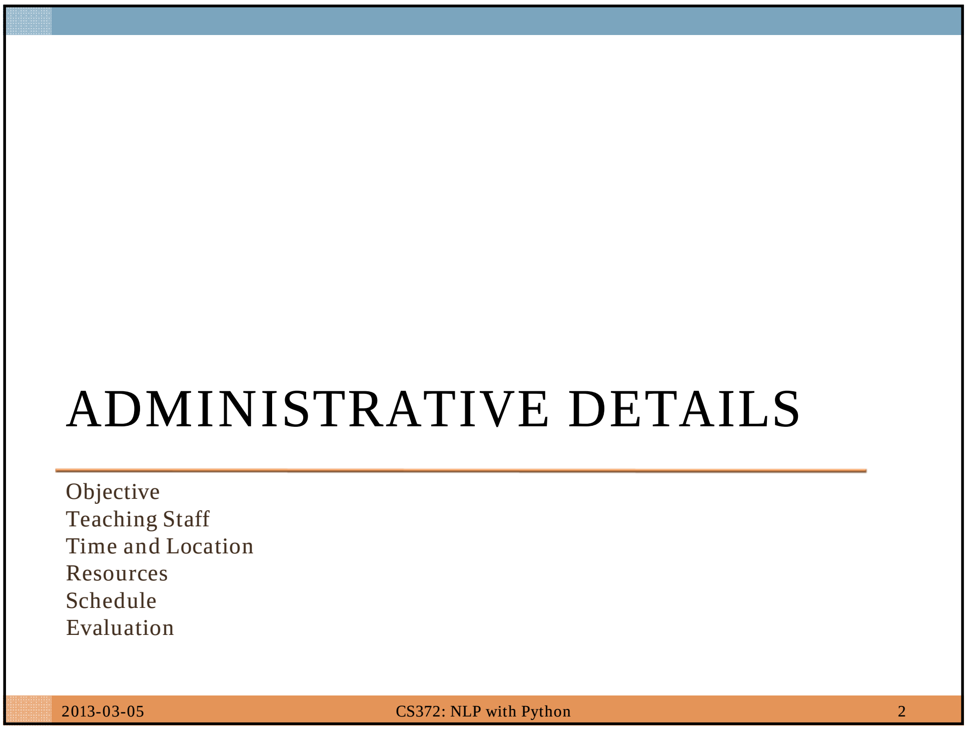


Natural Language Processing with Python

CS372: Spring, 2013

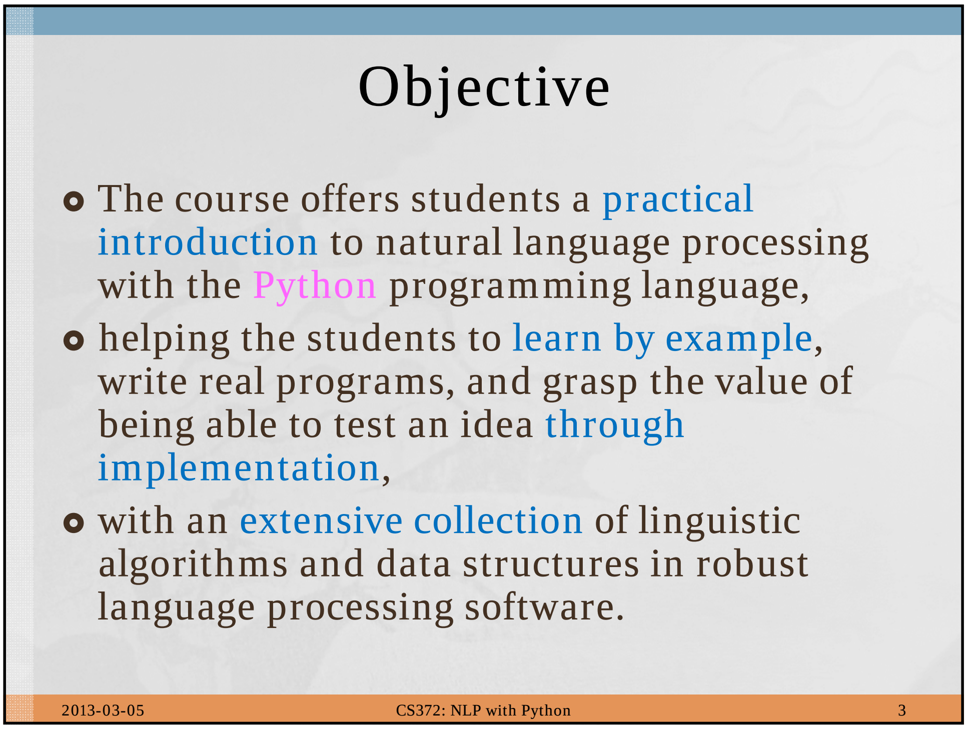
Lecture 1

Jong C. Park
Department of Computer Science
Korea Advanced Institute of Science and Technology



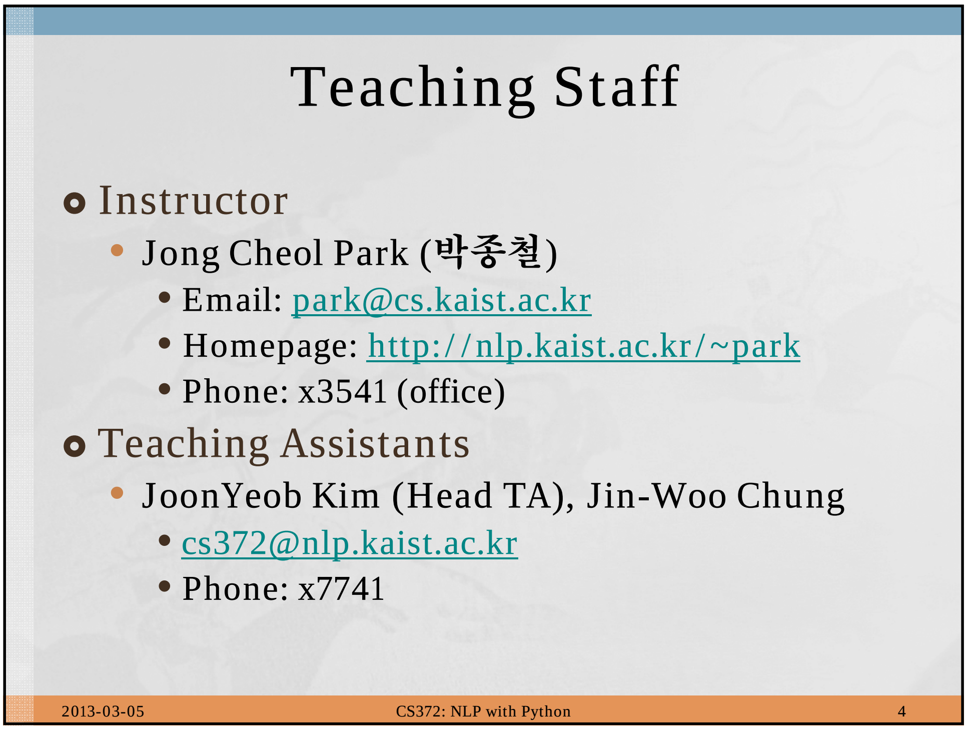
ADMINISTRATIVE DETAILS

Objective
Teaching Staff
Time and Location
Resources
Schedule
Evaluation



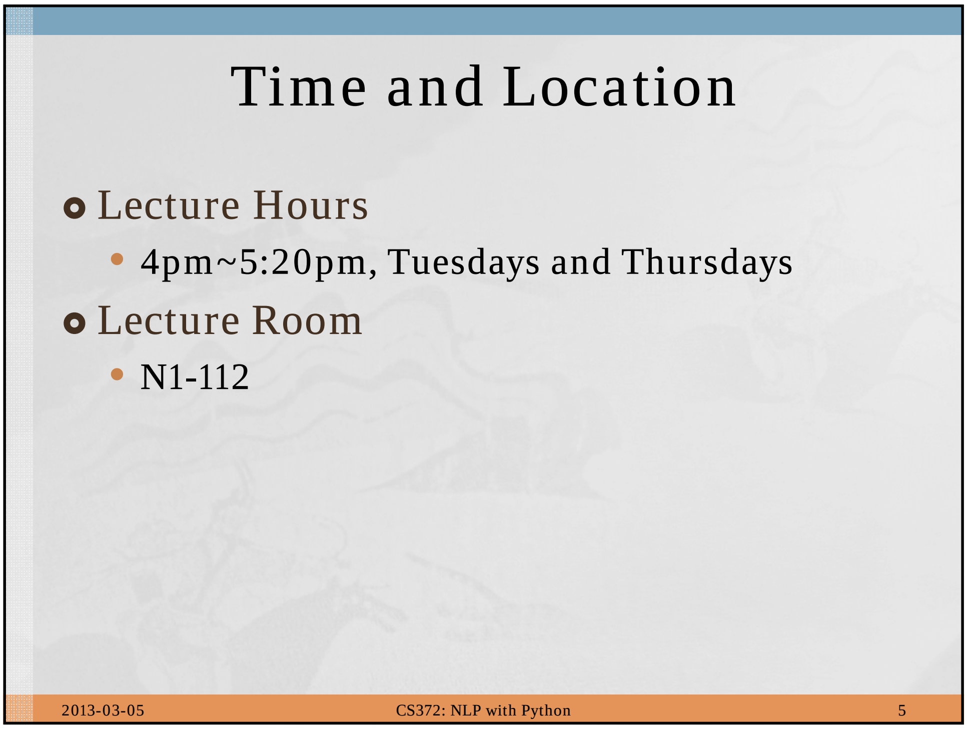
Objective

- The course offers students a practical introduction to natural language processing with the Python programming language,
- helping the students to learn by example, write real programs, and grasp the value of being able to test an idea through implementation,
- with an extensive collection of linguistic algorithms and data structures in robust language processing software.



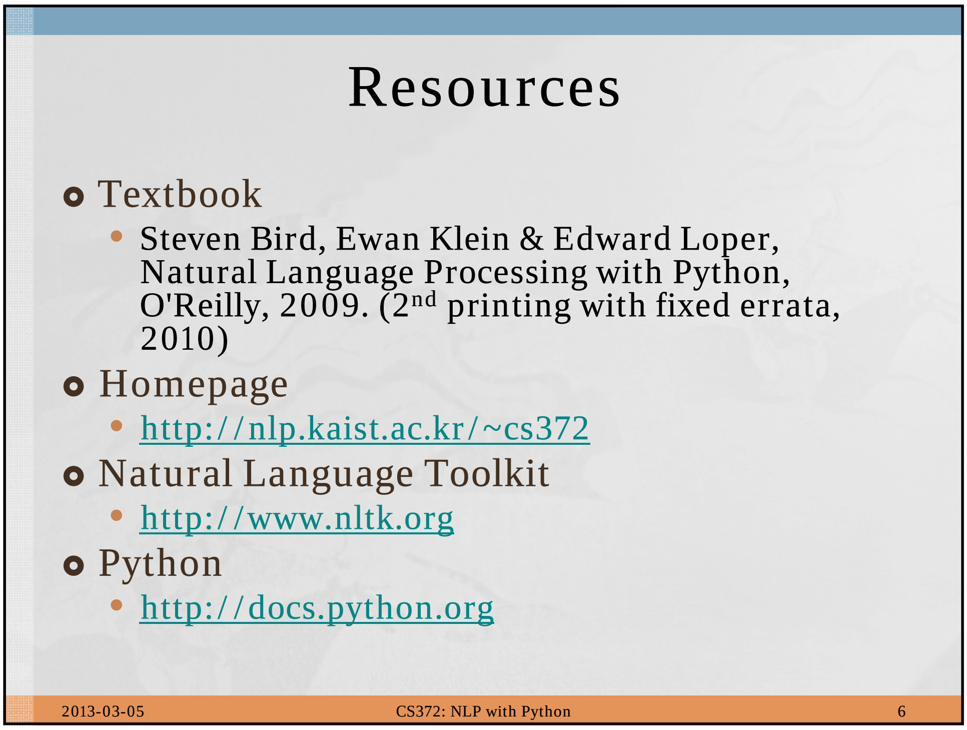
Teaching Staff

- Instructor
  - Jong Cheol Park (박종철)
    - Email: park@cs.kaist.ac.kr
    - Homepage: http://nlp.kaist.ac.kr/~park
    - Phone: x3541 (office)
- Teaching Assistants
  - JoonYeob Kim (Head TA), Jin-Woo Chung
    - cs372@nlp.kaist.ac.kr
    - Phone: x7741



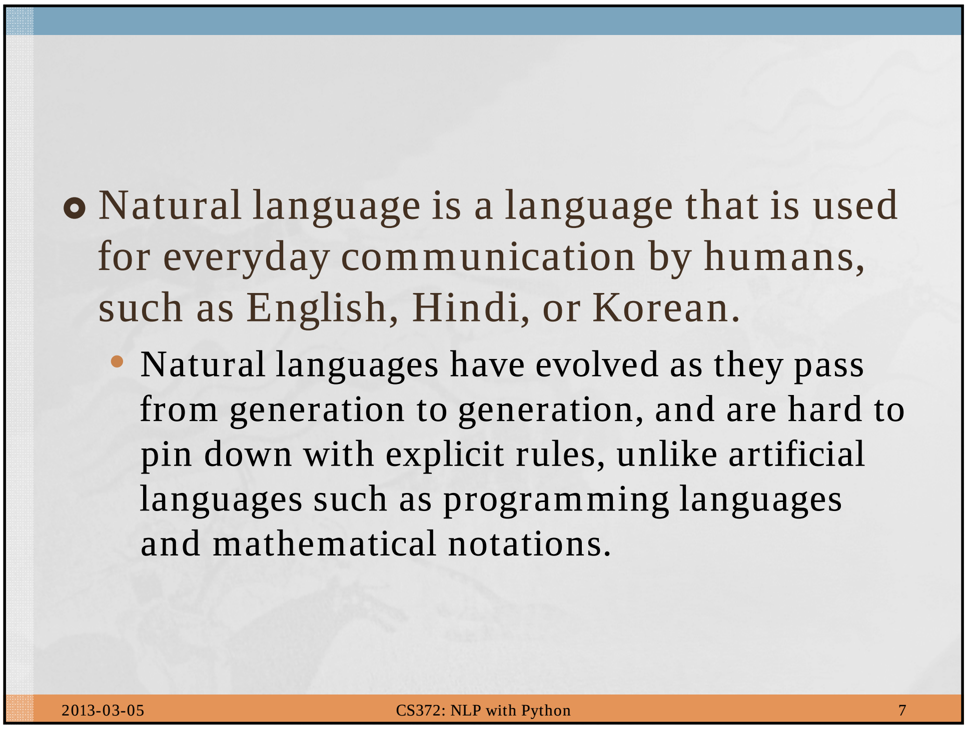
Time and Location

- Lecture Hours
  - 4pm~5:20pm, Tuesdays and Thursdays
- Lecture Room
  - N1-112

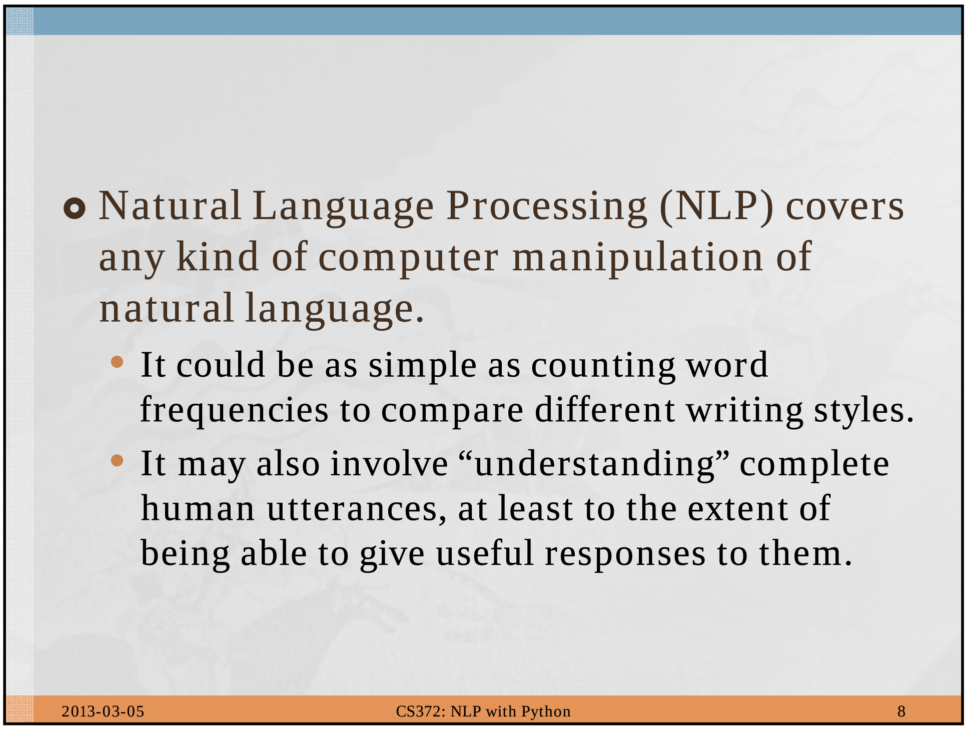


Resources

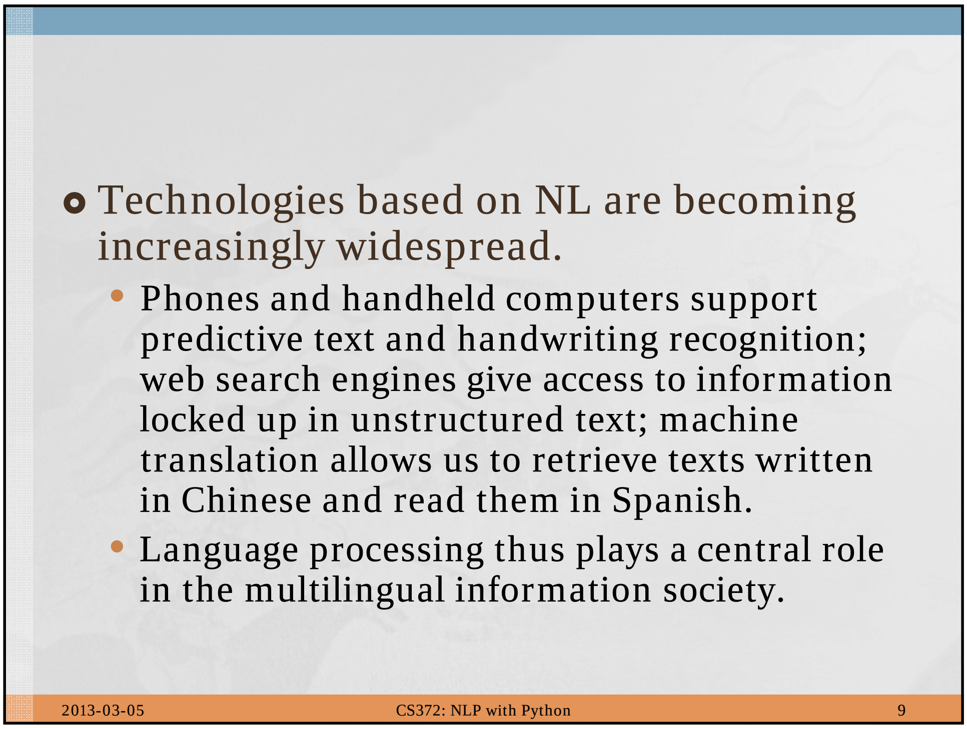
- Textbook
  - Steven Bird, Ewan Klein & Edward Loper, Natural Language Processing with Python, O'Reilly, 2009. (2nd printing with fixed errata, 2010)
- Homepage
  - http://nlp.kaist.ac.kr/~cs372
- Natural Language Toolkit
  - http://www.nltk.org
- Python
  - http://docs.python.org



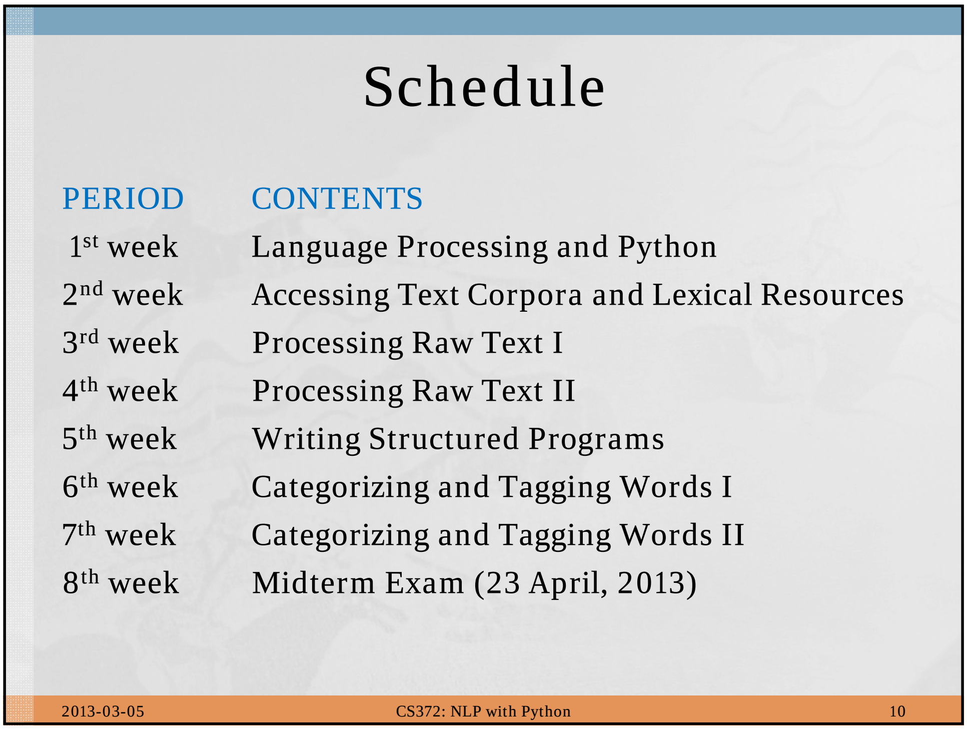
- Natural language is a language that is used for everyday communication by humans, such as English, Hindi, or Korean.
  - Natural languages have evolved as they pass from generation to generation, and are hard to pin down with explicit rules, unlike artificial languages such as programming languages and mathematical notations.



- Natural Language Processing (NLP) covers any kind of computer manipulation of natural language.
  - It could be as simple as counting word frequencies to compare different writing styles.
  - It may also involve “understanding” complete human utterances, at least to the extent of being able to give useful responses to them.

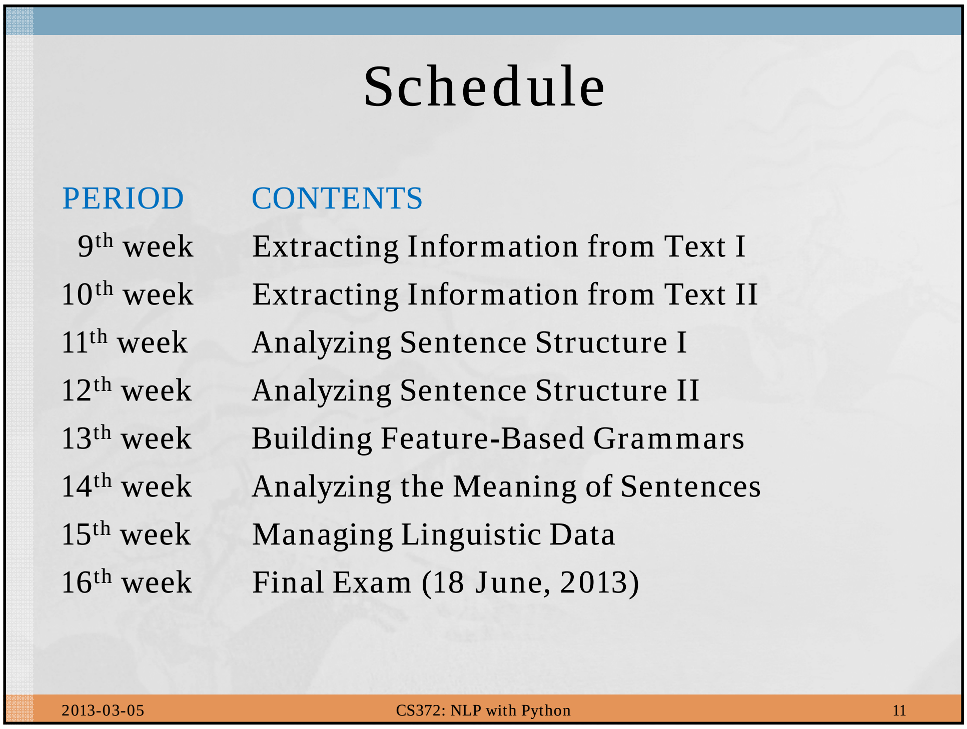


- Technologies based on NL are becoming increasingly widespread.
  - Phones and handheld computers support predictive text and handwriting recognition; web search engines give access to information locked up in unstructured text; machine translation allows us to retrieve texts written in Chinese and read them in Spanish.
  - Language processing thus plays a central role in the multilingual information society.



Schedule

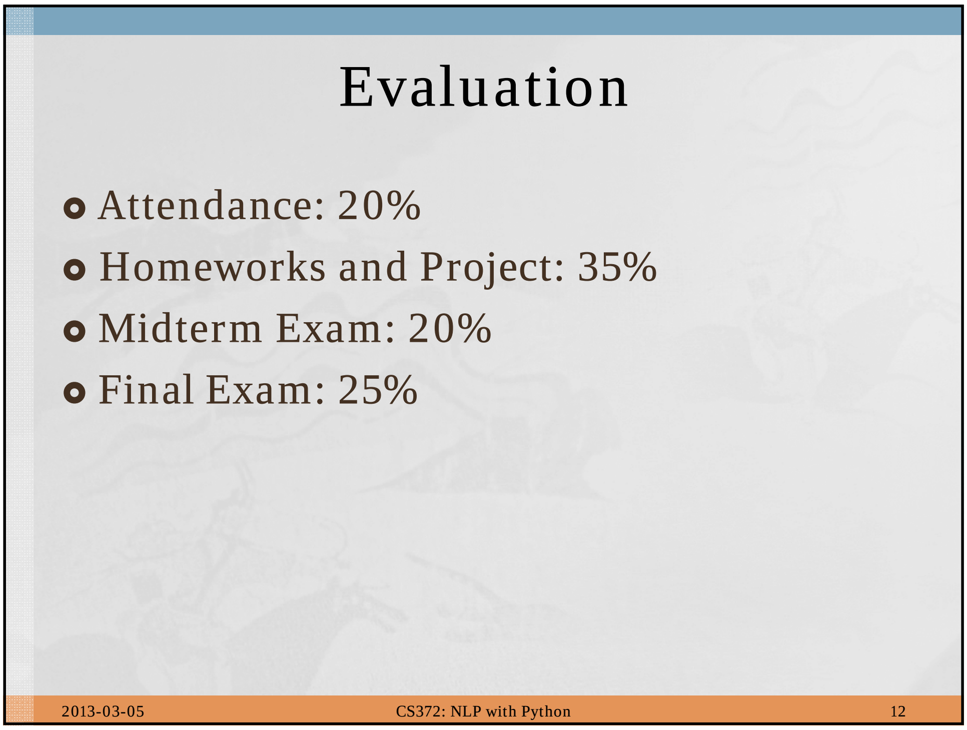
PERIOD	CONTENTS
1st week	Language Processing and Python
2nd week	Accessing Text Corpora and Lexical Resources
3rd week	Processing Raw Text I
4th week	Processing Raw Text II
5th week	Writing Structured Programs
6th week	Categorizing and Tagging Words I
7th week	Categorizing and Tagging Words II
8th week	Midterm Exam (23 April, 2013)



Schedule

PERIOD	CONTENTS
9th week	Extracting Information from Text I
10th week	Extracting Information from Text II
11th week	Analyzing Sentence Structure I
12th week	Analyzing Sentence Structure II
13th week	Building Feature-Based Grammars
14th week	Analyzing the Meaning of Sentences
15th week	Managing Linguistic Data
16th week	Final Exam (18 June, 2013)

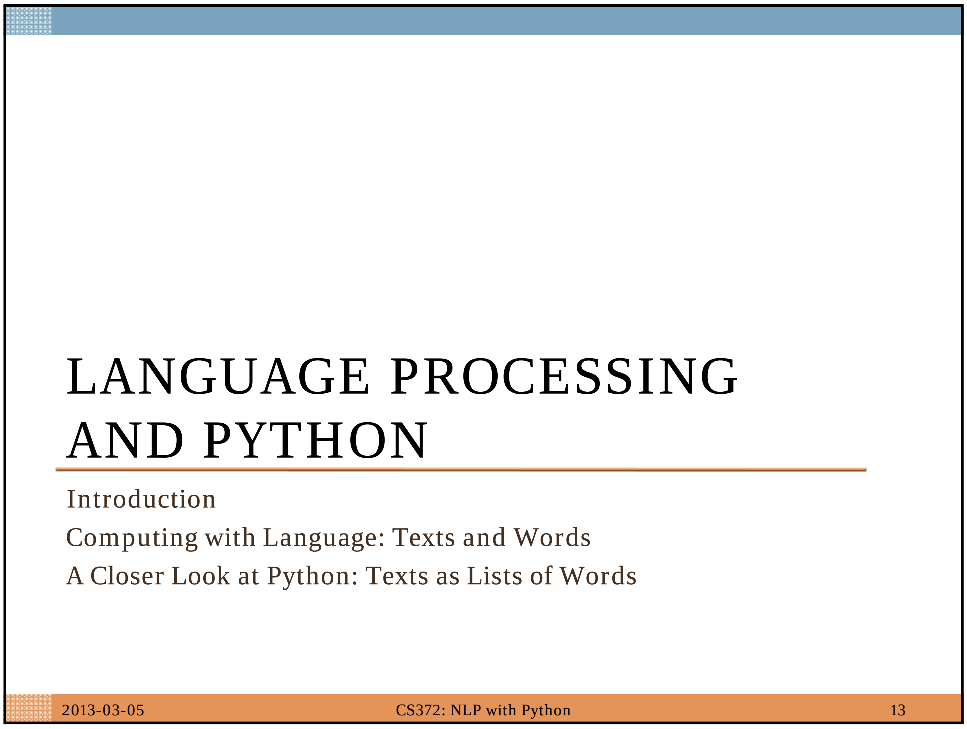
2013-03-05 CS372: NLP with Python 11



Evaluation

- Attendance: 20%
- Homeworks and Project: 35%
- Midterm Exam: 20%
- Final Exam: 25%

2013-03-05 CS372: NLP with Python 12

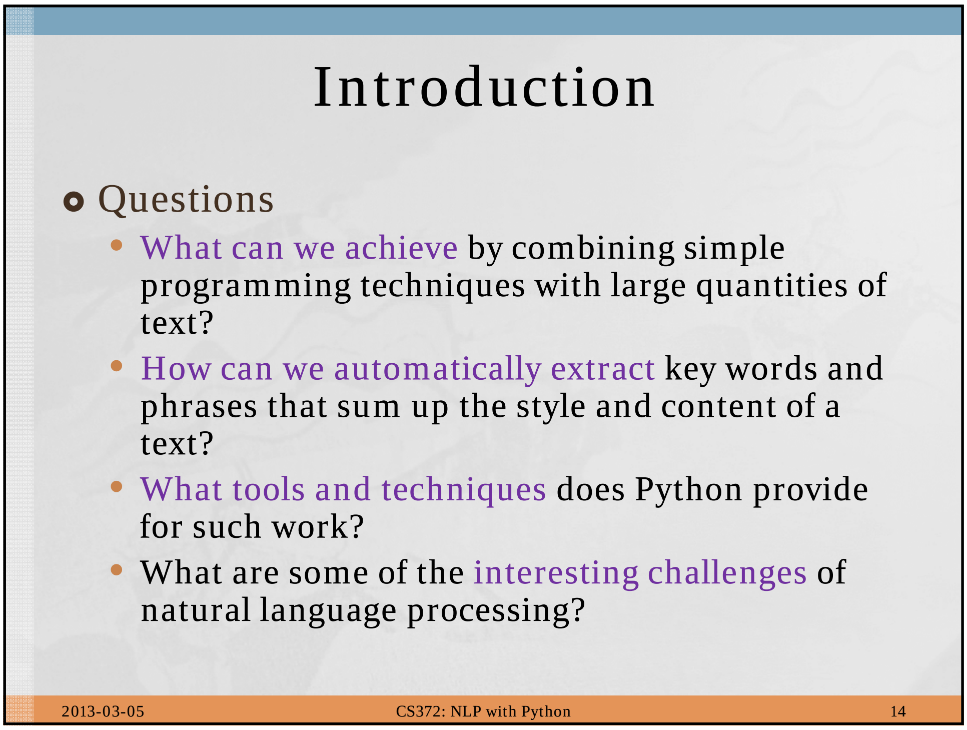


LANGUAGE PROCESSING AND PYTHON

Introduction

Computing with Language: Texts and Words

A Closer Look at Python: Texts as Lists of Words



Introduction

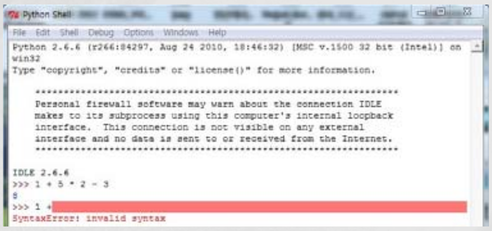
- Questions
  - What can we achieve by combining simple programming techniques with large quantities of text?
  - How can we automatically extract key words and phrases that sum up the style and content of a text?
  - What tools and techniques does Python provide for such work?
  - What are some of the interesting challenges of natural language processing?

Computing with Language:
Texts and Words

- Getting Started with Python
- Getting Started with NLTK
- Searching Text
- Counting Vocabulary

Getting Started with Python

- Python allows you to type directly into the interactive interpreter.
- We may access the interpreter with a simple graphical interface called the Interactive DeveLopment Environment (IDLE).

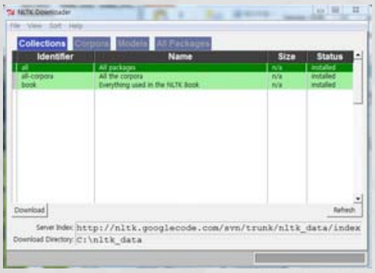


```
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.6
>>> 1 + 5 * 2 - 3
8
>>> 1 +
SyntaxError: invalid syntax
```

Getting Started with NLTK

- Install NLTK, start up the Python interpreter, and install the data.

```
>>> import nltk
>>> nltk.download()
```



Identifier	Name	Size	Status
all	All packages	n/a	installed
all-corpora	All the corpora	n/a	installed
book	Everything used in the NLTK Book	n/a	installed

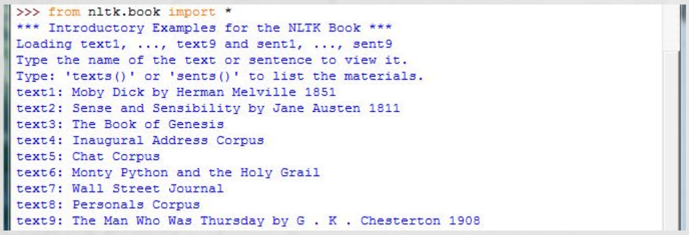
Server Index: http://nltk.googlecode.com/svn/trunk/nltk_data/index

Download Directory: C:\nltk_data

Getting Started with NLTK

- Load all items from NLTK's **book** module.

```
>>> from nltk.book import *
```

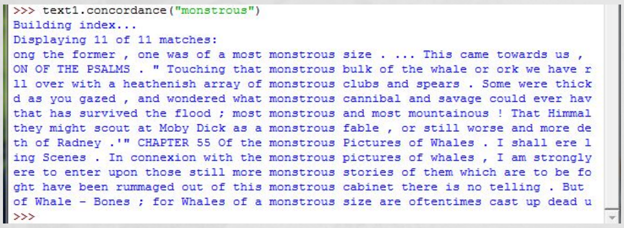


```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Searching Text

- A concordance view shows us every occurrence of a given word, together with some context.

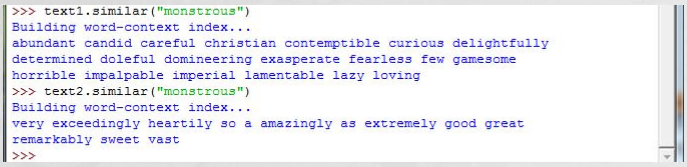
  >>> text1.concordance(“monstrous”)



```
>>> text1.concordance("monstrous")
Building index...
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
>>>
```

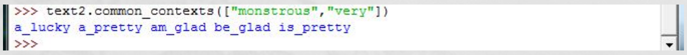
Searching Text

- Check for similar words in context.



```
>>> text1.similar("monstrous")
Building word-context index...
abundant candid careful christian contemptible curious delightfully
determined doleful domineering exasperate fearless few gamesome
horrible impalpable imperial lamentable lazy loving
>>> text2.similar("monstrous")
Building word-context index...
very exceedingly heartily so a amazingly as extremely good great
remarkably sweet vast
>>>
```

- Check for common contexts.



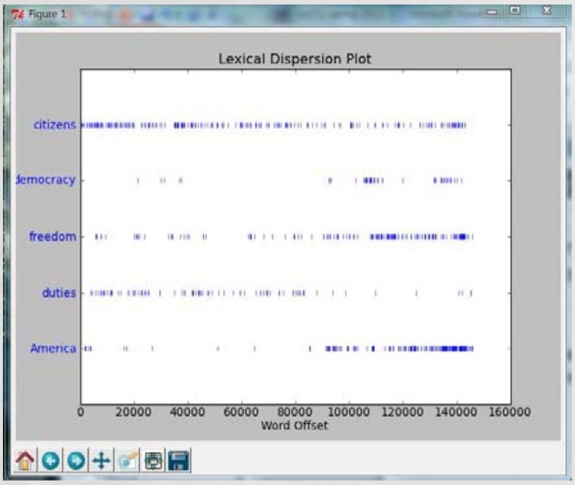
```
>>> text2.common_contexts(["monstrous","very"])
a_lucky a_pretty am_glad be_glad is_pretty
>>>
```

Searching Text

- Determine the location of a word in the text: how many words there are from the beginning it appears.
  - We use a dispersion plot, where each stripe represents an instance of a word, and each row represents the entire text.

```
>>> text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])
```

Searching Text

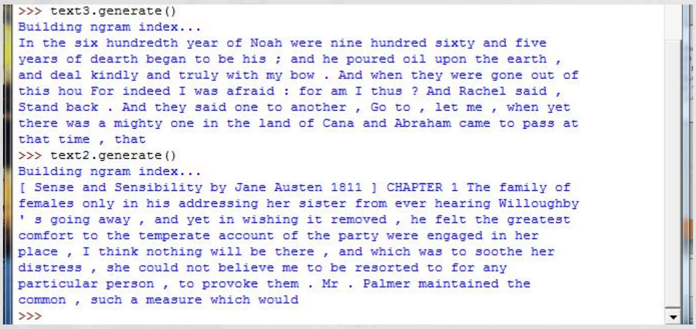


You need to have Python's NumPy and Matplotlib packages installed.

Searching Text

- Generate random text in various styles.

  >>> text3.generate()

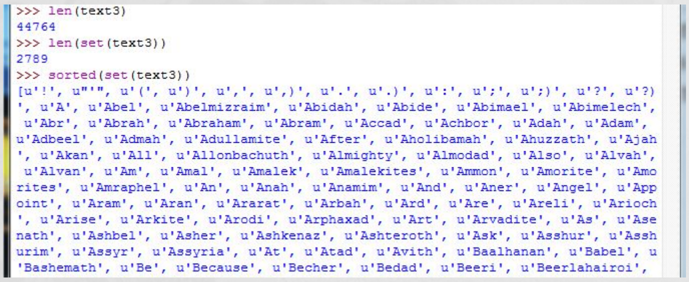


```
>>> text3.generate()
Building ngram index...
In the six hundredth year of Noah were nine hundred sixty and five
years of dearth began to be his ; and he poured oil upon the earth ,
and deal kindly and truly with my bow . And when they were gone out of
this hou For indeed I was afraid : for am I thus ? And Rachel said ,
Stand back . And they said one to another , Go to , let me , when yet
there was a mighty one in the land of Cana and Abraham came to pass at
that time , that
>>> text2.generate()
Building ngram index...
[ Sense and Sensibility by Jane Austen 1811 ] CHAPTER 1 The family of
females only in his addressing her sister from ever hearing Willoughby
' s going away , and yet in wishing it removed , he felt the greatest
comfort to the temperate account of the party were engaged in her
place , I think nothing will be there , and which was to soothe her
distress , she could not believe me to be resorted to for any
particular person , to provoke them . Mr . Palmer maintained the
common , such a measure which would
>>>
```

2013-03-05 CS372: NLP with Python 23

Counting Vocabulary

- Find out the length of a text from start to finish, in terms of the words and punctuation symbols that appear.

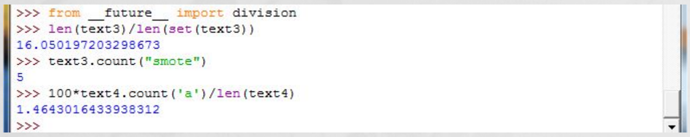


```
>>> len(text3)
44764
>>> len(set(text3))
2789
>>> sorted(set(text3))
[u'!', u"'", u'(', u')', u',', u',)', u'.', u'.)', u':', u';', u';)', u'?', u'?)
', u'A', u'Abel', u'Abelmizraim', u'Abidah', u'Abide', u'Abimael', u'Abimelech',
 u'Abr', u'Abrah', u'Abraham', u'Abram', u'Accad', u'Achbor', u'Adah', u'Adam',
u'Adbeel', u'Admah', u'Adullamite', u'After', u'Aholibamah', u'Ahuzzath', u'Ajah
', u'Akan', u'All', u'Allonbachuth', u'Almighty', u'Almodad', u'Also', u'Alvah',
 u'Alvan', u'Am', u'Amal', u'Amalek', u'Amalekites', u'Ammon', u'Amorite', u'Amo
rites', u'Amraphel', u'An', u'Anah', u'Anamim', u'And', u'Aner', u'Angel', u'App
oint', u'Aram', u'Aran', u'Ararat', u'Arbah', u'Ard', u'Are', u'Areli', u'Arioch
', u'Arise', u'Arkite', u'Arodi', u'Arphaxad', u'Art', u'Arvadite', u'As', u'Ase
nath', u'Ashbel', u'Asher', u'Ashkenaz', u'Ashteroth', u'Ask', u'Asshur', u'Assh
urim', u'Assyr', u'Assyria', u'At', u'Atad', u'Avith', u'Baalhanan', u'Babel', u
'Bashemath', u'Be', u'Because', u'Becher', u'Bedad', u'Beeri', u'Beerlahairoi',
```

2013-03-05 CS372: NLP with Python 24

Counting Vocabulary

- Calculate a measure of the lexical richness of the text.

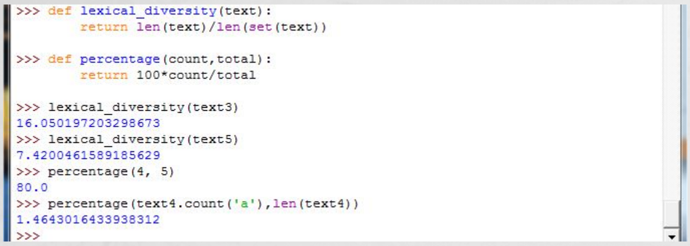


```
>>> from __future__ import division
>>> len(text3)/len(set(text3))
16.050197203298673
>>> text3.count("smote")
5
>>> 100*text4.count('a')/len(text4)
1.4643016433938312
>>>
```

2013-03-05 CS372: NLP with Python 25

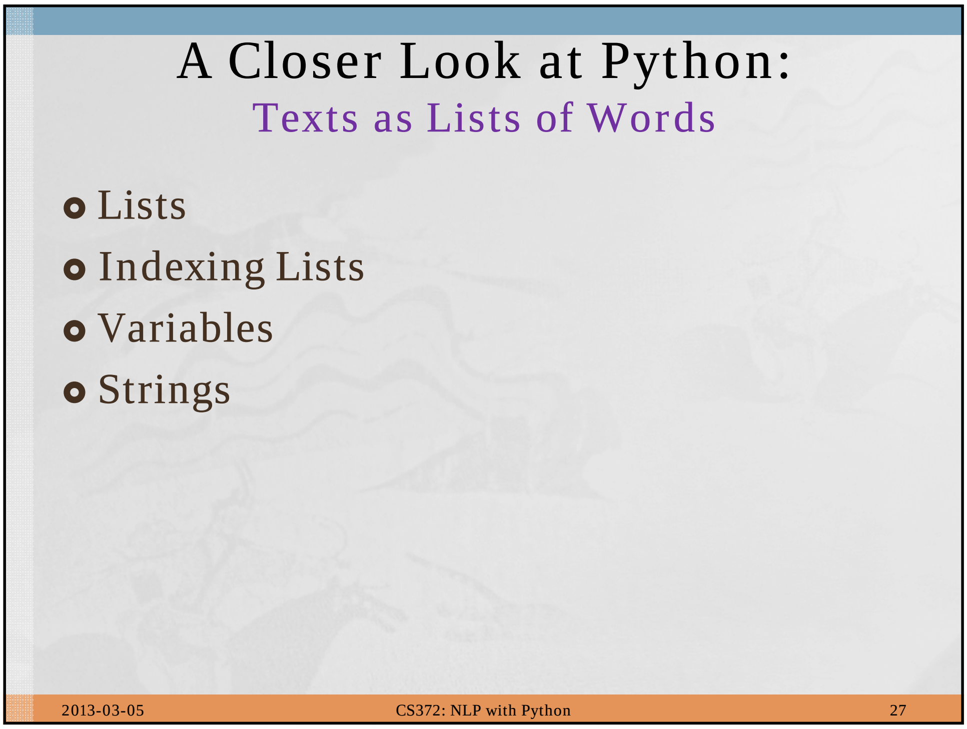
Counting Vocabulary

- Define functions and use them.



```
>>> def lexical_diversity(text):
        return len(text)/len(set(text))

>>> def percentage(count,total):
        return 100*count/total

>>> lexical_diversity(text3)
16.050197203298673
>>> lexical_diversity(text5)
7.4200461589185629
>>> percentage(4, 5)
80.0
>>> percentage(text4.count('a'),len(text4))
1.4643016433938312
>>>
```

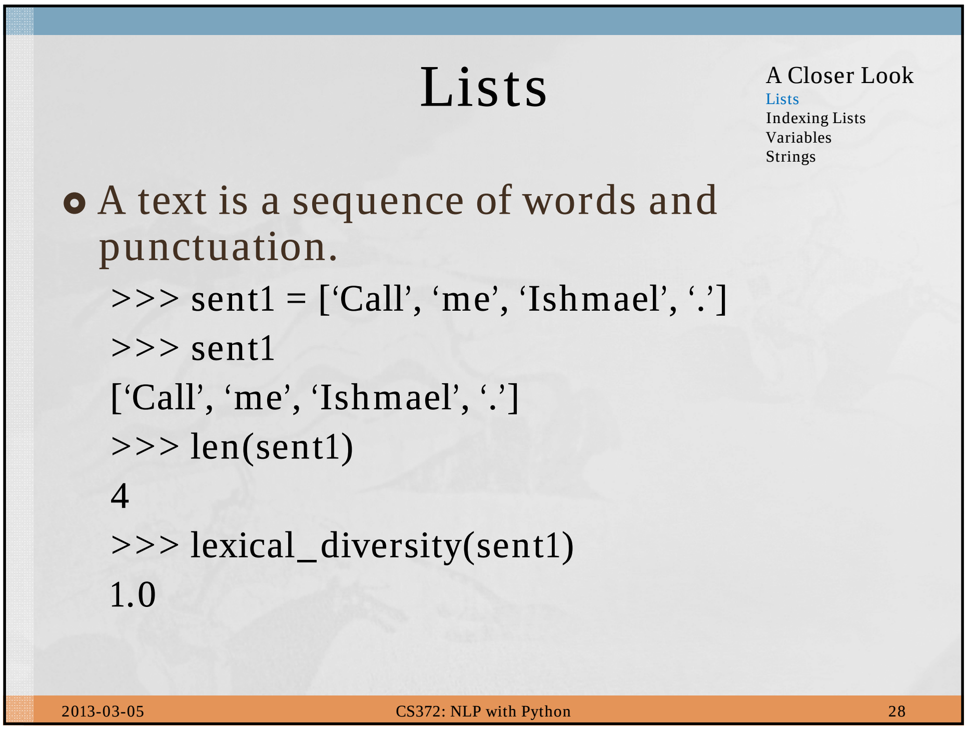
2013-03-05 CS372: NLP with Python 26



A Closer Look at Python:

Texts as Lists of Words

- Lists
- Indexing Lists
- Variables
- Strings

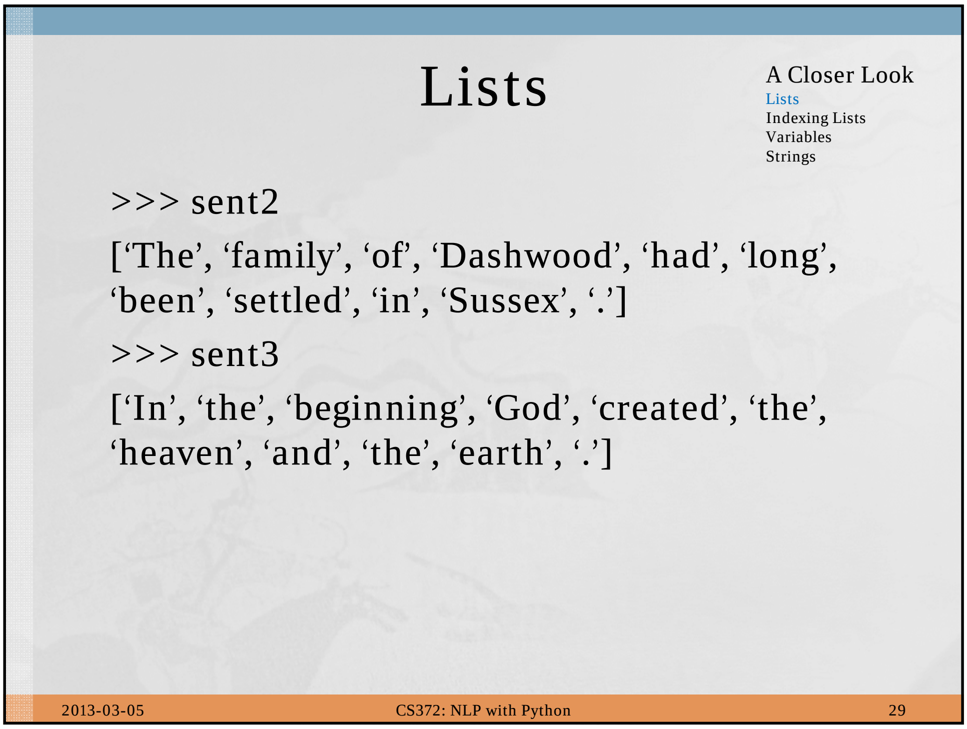


Lists

A Closer Look
Lists
Indexing Lists
Variables
Strings

- A text is a sequence of words and punctuation.

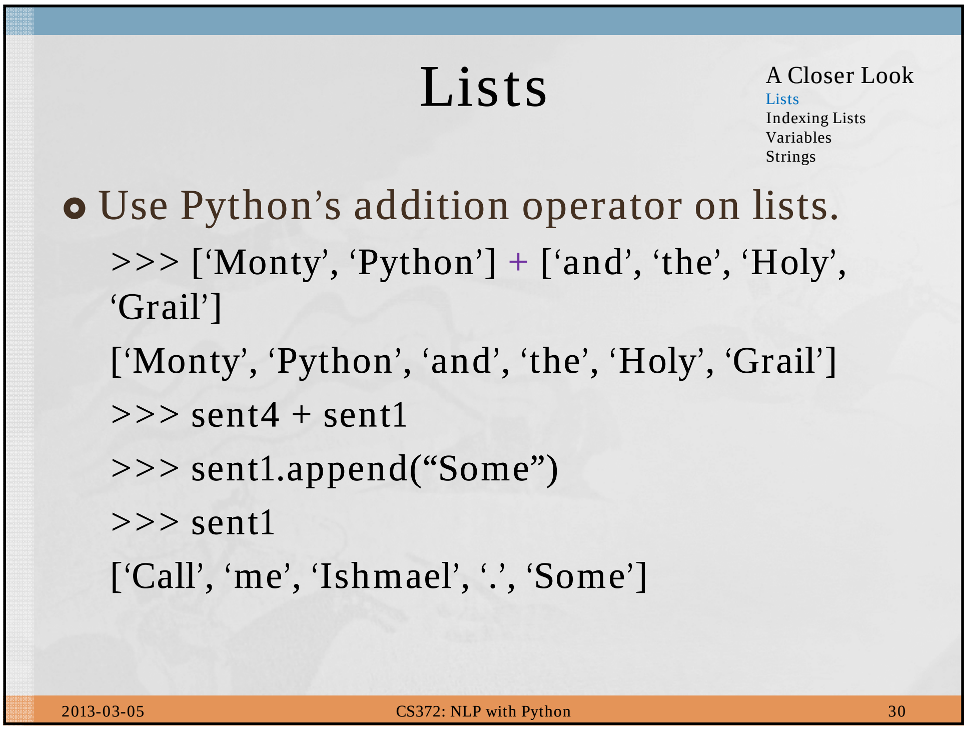
```
>>> sent1 = [‘Call’, ‘me’, ‘Ishmael’, ‘.’]
>>> sent1
[‘Call’, ‘me’, ‘Ishmael’, ‘.’]
>>> len(sent1)
4
>>> lexical_diversity(sent1)
1.0
```



Lists

A Closer Look
Lists
Indexing Lists
Variables
Strings

```
>>> sent2
['The', 'family', 'of', 'Dashwood', 'had', 'long',
'been', 'settled', 'in', 'Sussex', '.']
>>> sent3
['In', 'the', 'beginning', 'God', 'created', 'the',
'heaven', 'and', 'the', 'earth', '.']
```



Lists

A Closer Look
Lists
Indexing Lists
Variables
Strings

- Use Python's addition operator on lists.

```
>>> ['Monty', 'Python'] + ['and', 'the', 'Holy',
'Grail']
['Monty', 'Python', 'and', 'the', 'Holy', 'Grail']
>>> sent4 + sent1
>>> sent1.append("Some")
>>> sent1
['Call', 'me', 'Ishmael', '.', 'Some']
```

Indexing Lists

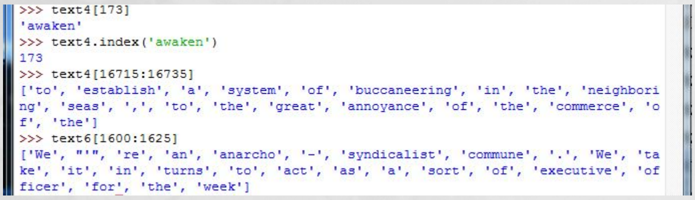
A Closer Look
Lists
Indexing Lists
Variables
Strings

- Identify the elements of a Python list by their order of occurrence in the list.

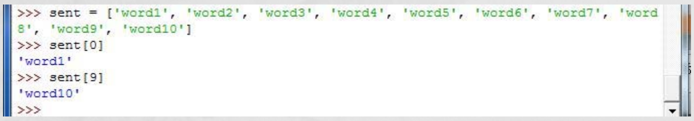
```
>>> text4[173]
>>> text4.index(‘awaken’)
>>> text4[16715:16735]
>>> text6[1600:1625]
>>> sent = [‘word1’, ‘word2’, ‘word3’, ‘word4’,
‘word5’, ‘word6’, ‘word7’, ‘word8’, ‘word9’, ‘word10’]
>>> sent[0]
>>> sent[9]
```

Indexing Lists

A Closer Look
Lists
Indexing Lists
Variables
Strings



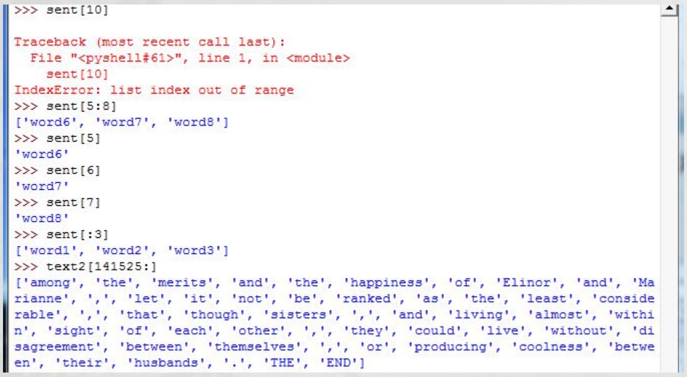
```
>>> text4[173]
'awaken'
>>> text4.index('awaken')
173
>>> text4[16715:16735]
['to', 'establish', 'a', 'system', 'of', 'buccaneering', 'in', 'the', 'neighbori
ng', 'seas', ',', 'to', 'the', 'great', 'annoyance', 'of', 'the', 'commerce', 'o
f', 'the']
>>> text6[1600:1625]
['We', "'", 're', 'an', 'anarcho', '-', 'syndicalist', 'commune', '.', 'We', 'ta
ke', 'it', 'in', 'turns', 'to', 'act', 'as', 'a', 'sort', 'of', 'executive', 'of
ficer', 'for', 'the', 'week']
```



```
>>> sent = ['word1', 'word2', 'word3', 'word4', 'word5', 'word6', 'word7', 'word
8', 'word9', 'word10']
>>> sent[0]
'word1'
>>> sent[9]
'word10'
>>>
```

Indexing Lists

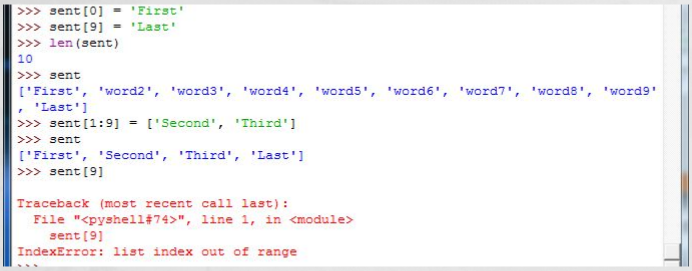
A Closer Look
Lists
Indexing Lists
Variables
Strings



```
>>> sent[10]

Traceback (most recent call last):
  File "<pyshell#61>", line 1, in <module>
    sent[10]
IndexError: list index out of range
>>> sent[5:8]
['word6', 'word7', 'word8']
>>> sent[5]
'word6'
>>> sent[6]
'word7'
>>> sent[7]
'word8'
>>> sent[:3]
['word1', 'word2', 'word3']
>>> text2[141525:]
['among', 'the', 'merits', 'and', 'the', 'happiness', 'of', 'Elinor', 'and', 'Ma
rianne', ',', 'let', 'it', 'not', 'be', 'ranked', 'as', 'the', 'least', 'conside
rable', ',', 'that', 'though', 'sisters', ',', 'and', 'living', 'almost', 'withi
n', 'sight', 'of', 'each', 'other', ',', 'they', 'could', 'live', 'without', 'di
sagreement', 'between', 'themselves', ',', 'or', 'producing', 'coolness', 'betwe
en', 'their', 'husbands', '.', 'THE', 'END']
```

Indexing Lists

A Closer Look
Lists
Indexing Lists
Variables
Strings



```
>>> sent[0] = 'First'
>>> sent[9] = 'Last'
>>> len(sent)
10
>>> sent
['First', 'word2', 'word3', 'word4', 'word5', 'word6', 'word7', 'word8', 'word9'
, 'Last']
>>> sent[1:9] = ['Second', 'Third']
>>> sent
['First', 'Second', 'Third', 'Last']
>>> sent[9]

Traceback (most recent call last):
  File "<pyshell#74>", line 1, in <module>
    sent[9]
IndexError: list index out of range
```

Variables

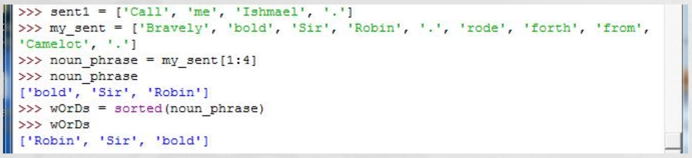
A Closer Look
Lists
Indexing Lists
Variables
Strings

- The variable assignment process does not generate any output.

```
>>> sent1 = [‘Call’, ‘me’, ‘Ishmael’, ‘.’]
>>> my_sent = [‘Bravely’, ‘bold’, ‘Sir’, ‘Robin’, ‘.’,
‘rode’, ‘forth’, ‘from’, ‘Camelot’, ‘.’]
>>> noun_phrase = my_sent[1:4]
>>> noun_phrase
[‘bold’, ‘Sir’, ‘Robin’]
>>> wOrDs = sorted(noun_phrase)
>>> wOrDs
[‘Robin’, ‘Sir’, ‘bold’]
```

Variables

A Closer Look
Lists
Indexing Lists
Variables
Strings

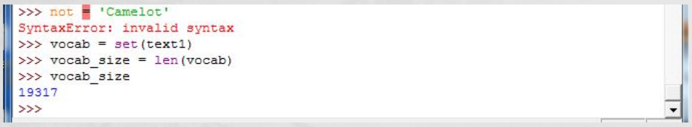


```
>>> sent1 = ['Call', 'me', 'Ishmael', '.']
>>> my_sent = ['Bravely', 'bold', 'Sir', 'Robin', '.', 'rode', 'forth', 'from',
'Camelot', '.']
>>> noun_phrase = my_sent[1:4]
>>> noun_phrase
['bold', 'Sir', 'Robin']
>>> wOrDs = sorted(noun_phrase)
>>> wOrDs
['Robin', 'Sir', 'bold']
```

Variables

A Closer Look
Lists
Indexing Lists
Variables
Strings

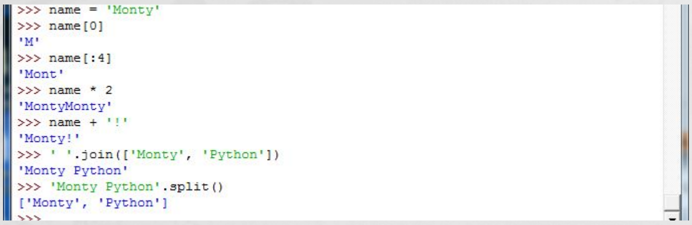
```
>>> not = ‘Camelot’
>>> vocab = set(text1)
>>> vocab_size = len(vocab)
>>> vocab_size
```



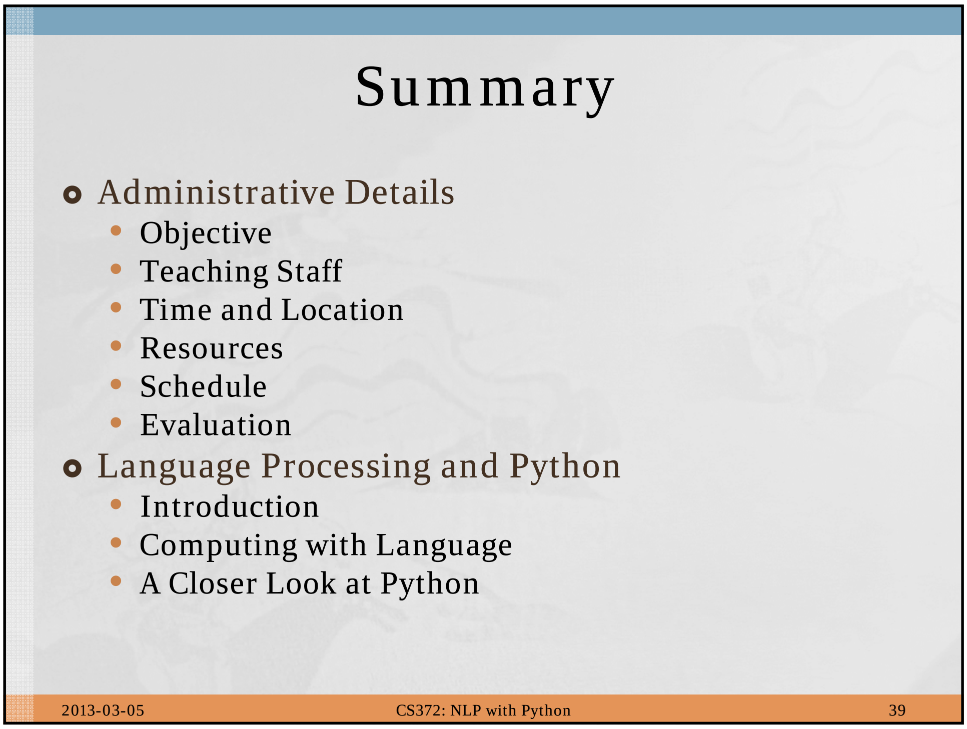
```
>>> not = 'Camelot'
SyntaxError: invalid syntax
>>> vocab = set(text1)
>>> vocab_size = len(vocab)
>>> vocab_size
19317
>>>
```

Strings

A Closer Look
Lists
Indexing Lists
Variables
Strings



```
>>> name = 'Monty'
>>> name[0]
'M'
>>> name[:4]
'Mont'
>>> name * 2
'MontyMonty'
>>> name + '!'
'Monty!'
>>> ' '.join(['Monty', 'Python'])
'Monty Python'
>>> 'Monty Python'.split()
['Monty', 'Python']
>>>
```



Summary

- Administrative Details
  - Objective
  - Teaching Staff
  - Time and Location
  - Resources
  - Schedule
  - Evaluation
- Language Processing and Python
  - Introduction
  - Computing with Language
  - A Closer Look at Python

2013-03-05 CS372: NLP with Python 39