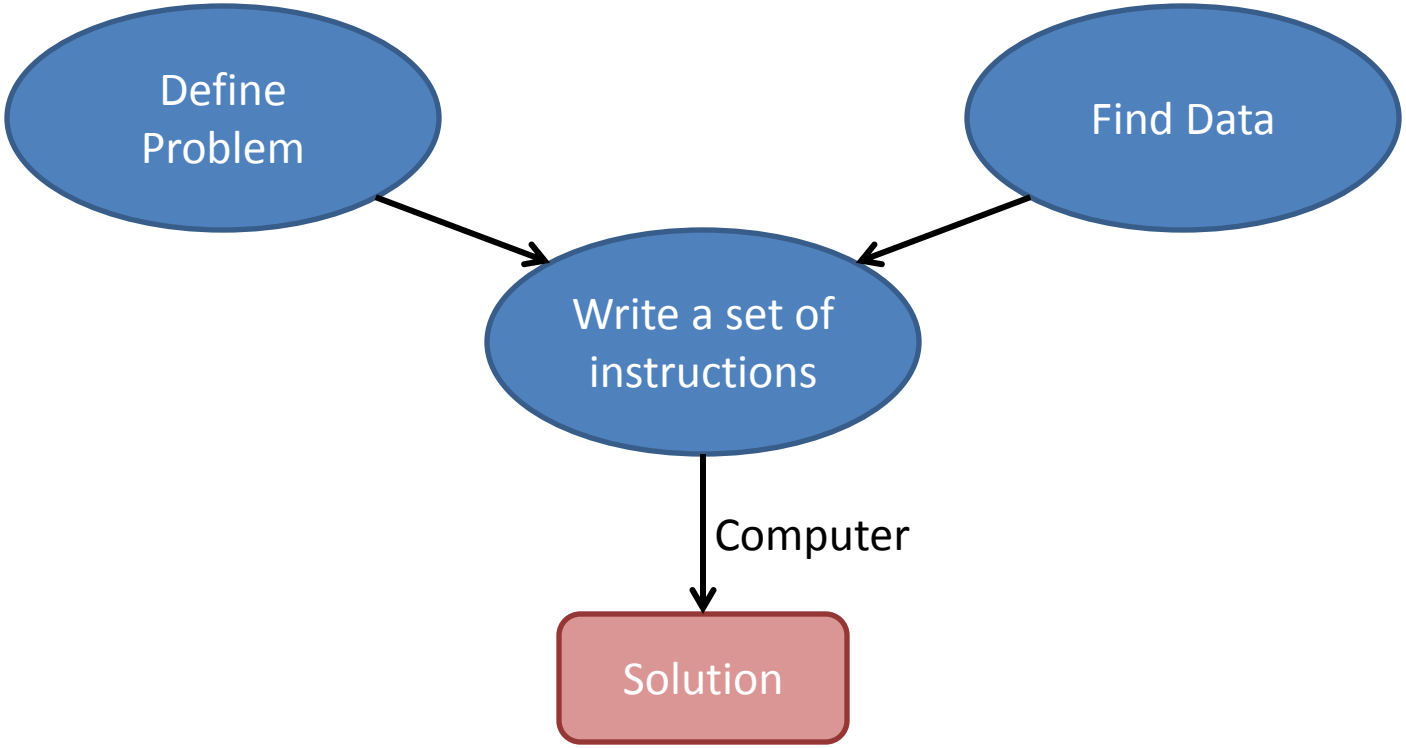
Textual Analysis & Introduction to Python

Feb. 23, 2012

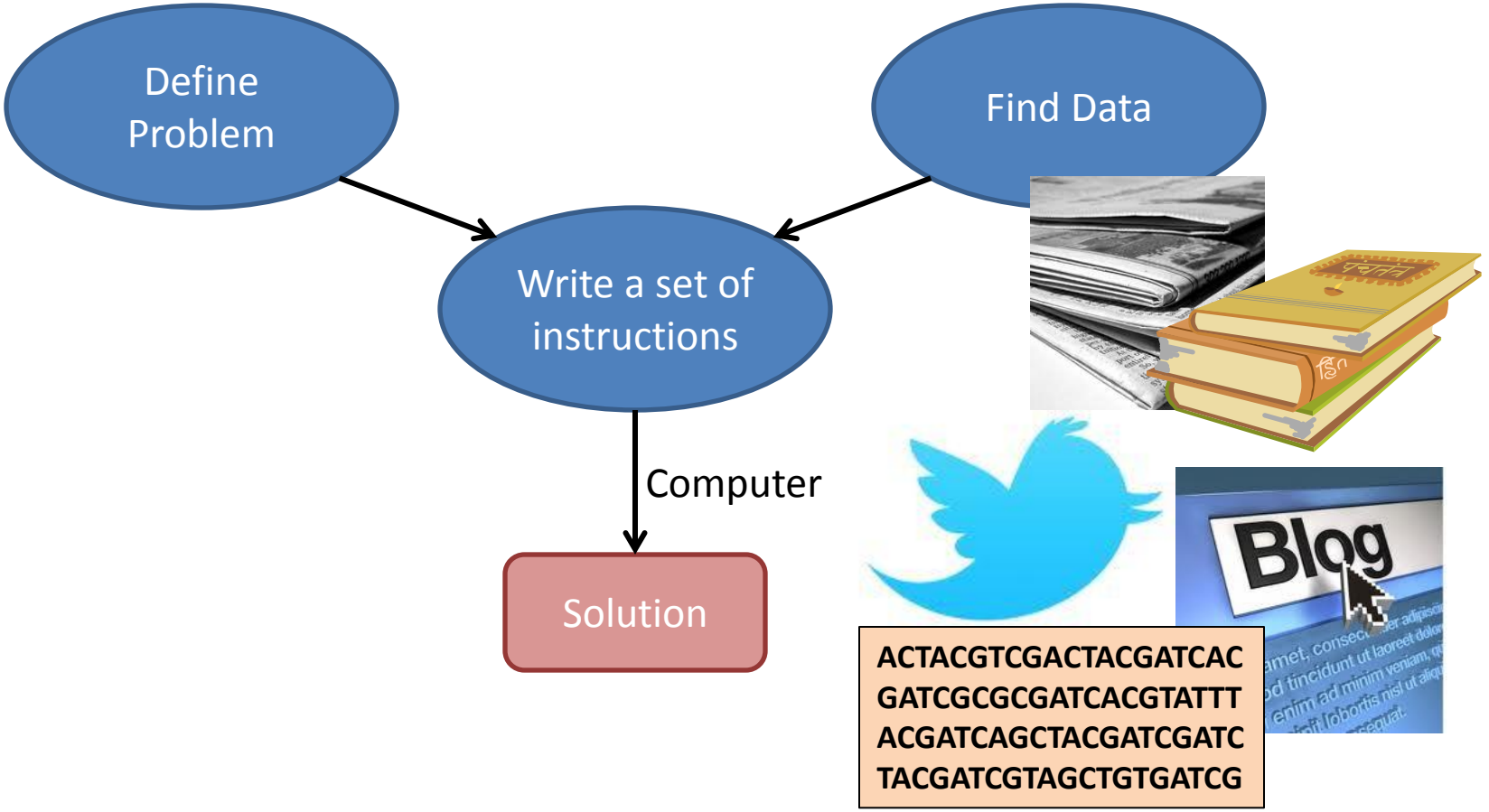
Before we leave Excel…

- Pick and choose ways to increase Degree of Difficulty.
- Display your data in different ways (graph, table, etc.)
- Remember to answer your claim
  - My hypothesis is correct/incorrect because …
  - X% of the time, my hypothesis was correct…
- Be sure to explain what obstacles you encountered
- Look at the rubric!

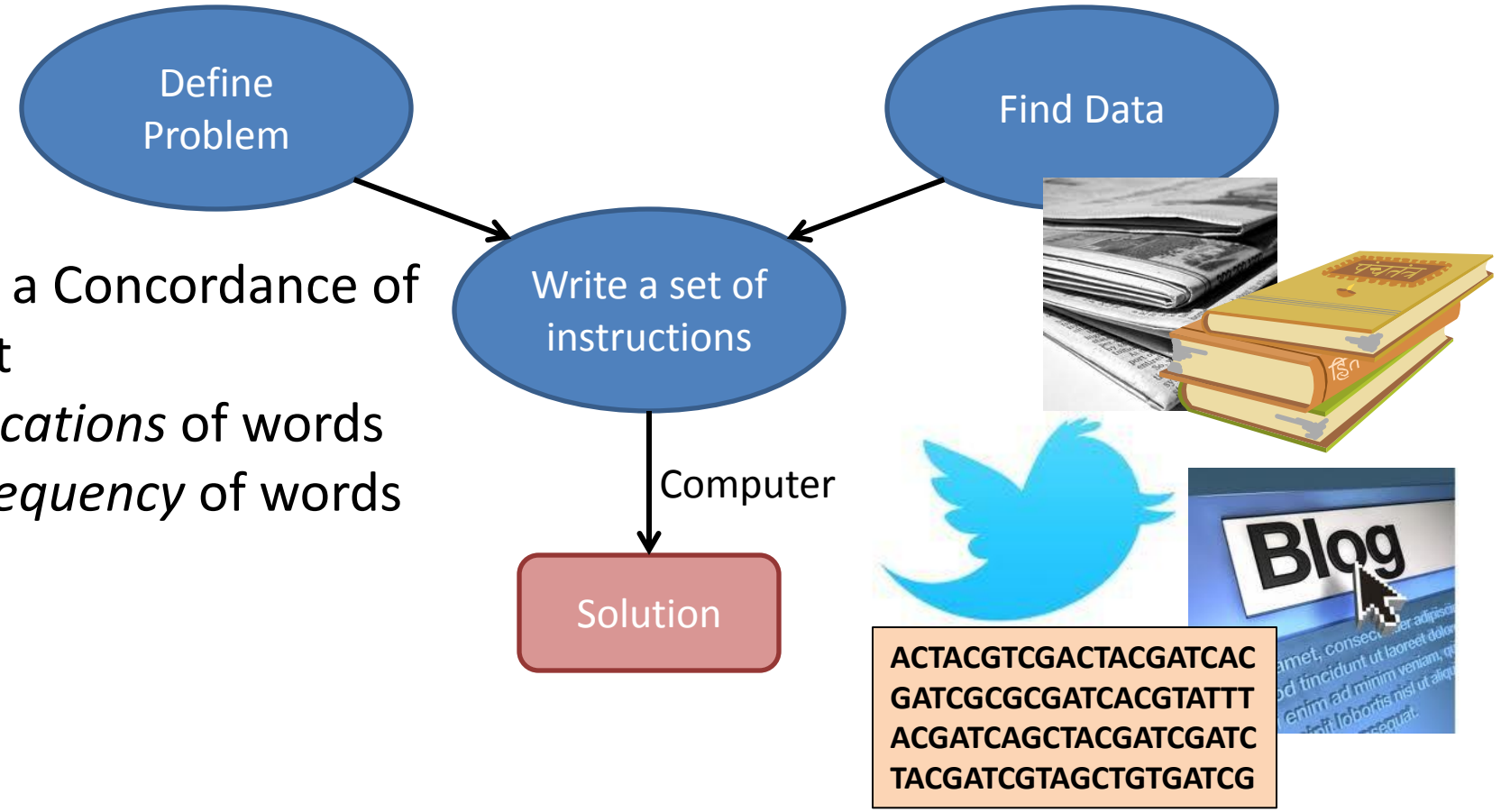
Textual Analysis



Textual Analysis



Textual Analysis



Build a Concordance of a text

- *Locations* of words
- *Frequency* of words

Concordances

Alphabetical index of all words in a text

Word	Page Numbers
Apple	4,7,10,27
Banana	77,110,130
Carrot	50,101
Date	9
…	…

Concordances

- Before computers, was a *huge* pain.
- What texts might have had concordances?

http://en.wikipedia.org/wiki/Concordance_(publishing)

Concordances

- Before computers, was a *huge* pain.
- What texts might have had concordances?
  - The Bible
  - The Quran
  - The Vedas
  - Shakespeare

http://en.wikipedia.org/wiki/Concordance_(publishing)

Concordances

- Before computers, was a *huge* pain.
- What texts might have had concordances?
  - The Bible
  - The Quran
  - The Vedas
  - Shakespeare

Not a "New" Problem:
First Bible Concordance
completed in 1230

http://en.wikipedia.org/wiki/Concordance_(publishing)

Concordances

- How long would the King James Bible take us?
  - 783,137 words

http://agards-bible-timeline.com/q10_bible-facts.html

Concordances

- How long would the King James Bible take us?
  - 783,137 words

800,000 * (3 min. to look up word and put page #) = 2,400,000 minutes
= 40,000 hours
= 1,667 days
= 4.5 years

http://agards-bible-timeline.com/q10_bible-facts.html

Concordances

- How long would the King James Bible take us?
  - 783,137 words

800,000 * (3 min. to look up word and put page #) = 2,400,000 minutes
= 40,000 hours
= 1,667 days
= 4.5 years

Takes 70 hours to read the King James Bible aloud

http://agards-bible-timeline.com/q10_bible-facts.html

Strong's Concordance

- Concordance of the King James Bible
- Published in 1890 by James Strong

ANT
Prv 6: 6 Go to the **a**, thou sluggard; consider her H5244

ANTICHRIST
1Jn 2:18 as ye have heard that **a** shall come, even G500
22 is **a**, that denieth the Father and the Son. G500
4: 3 this is that *spirit* of **a**, whereof ye have G500
2Jn 7 in the flesh. This is a deceiver and an **a**. G500

ANTICHRISTS
1Jn 2:18 are there many **a**; whereby we know that G500

ANTIOCH
Act 6: 5 Parmenas, and Nicolas a proselyte of **A**: G491
11:19 and Cyprus, and **A**, preaching the word G490
20 they were come to **A**, spake unto the G490
22 Barnabas, that he should go as far as **A**. G490
26 brought him unto **A**. And it came to pass, G490
26 disciples were called Christians first in **A**. G490
27 came prophets from Jerusalem unto **A**. G490
13: 1 church that was at **A** certain prophets G490
14 they came to **A** in Pisidia, and went G490
14:19 *certain* Jews from **A** and Iconium, who G490
21 again to Lystra, and *to* Iconium, and **A**, G490
26 And thence sailed to **A**, from whence G490
15:22 their own company to **A** with Paul and G490

Jas 1:21 Wherefore lay **a** all filthiness and G659

APELLES
Ro 16:10 Salute **A** approved in Christ. Salute them G559

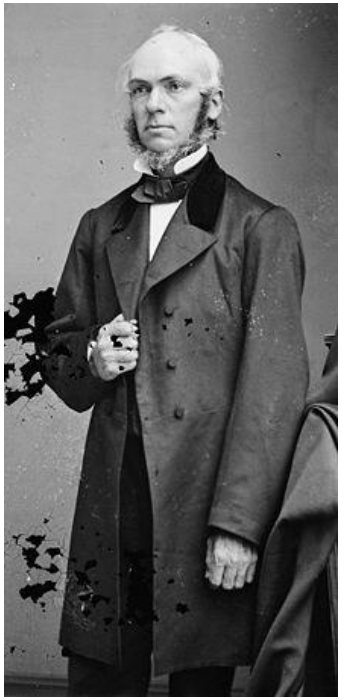
APES
1Ki 10:22 and silver, ivory, and **a**, and peacocks. H6971
2Ch 9:21 and silver, ivory, and **a**, and peacocks. H6971

APHARSACHITES
Ezr 5: 6 companions the **A**, which *were* on this H671
6: 6 companions the **A**, which *are* beyond the H671

APHARSATHCHITES
Ezr 4: 9 the Dinaites, the **A**, the Tarpelites, the H671

APHARSITES
Ezr 4: 9 the Tarpelites, the **A**, the Archevites, the H670

APHEK
Jos 12:18 The king of **A**, one; the king of Lasharon, H663
13: 4 unto **A**, to the borders of the Amorites: H663
19:30 Ummah also, and **A**, and Rehob: twenty H663
1Sa 4: 1 and the Philistines pitched in **A**. H663
29: 1 all their armies to **A**: and the Israelites H663
1Ki 20:26 and went up to **A**, to fight against Israel. H663



Wikipedia

http://www.christianbook.com/reader/?item_no=563788

From Concordance to Word Frequency

Suppose our text has 1000 words total.

Word	Page Numbers	# of Occurrences	Word Frequency
Apple	4,7,10,27	4	4/1000
Banana	77,110,130	3	3/1000
Carrot	50,101	2	2/1000
Date	9	1	1/1000
…	…	…	…

Google Ngrams

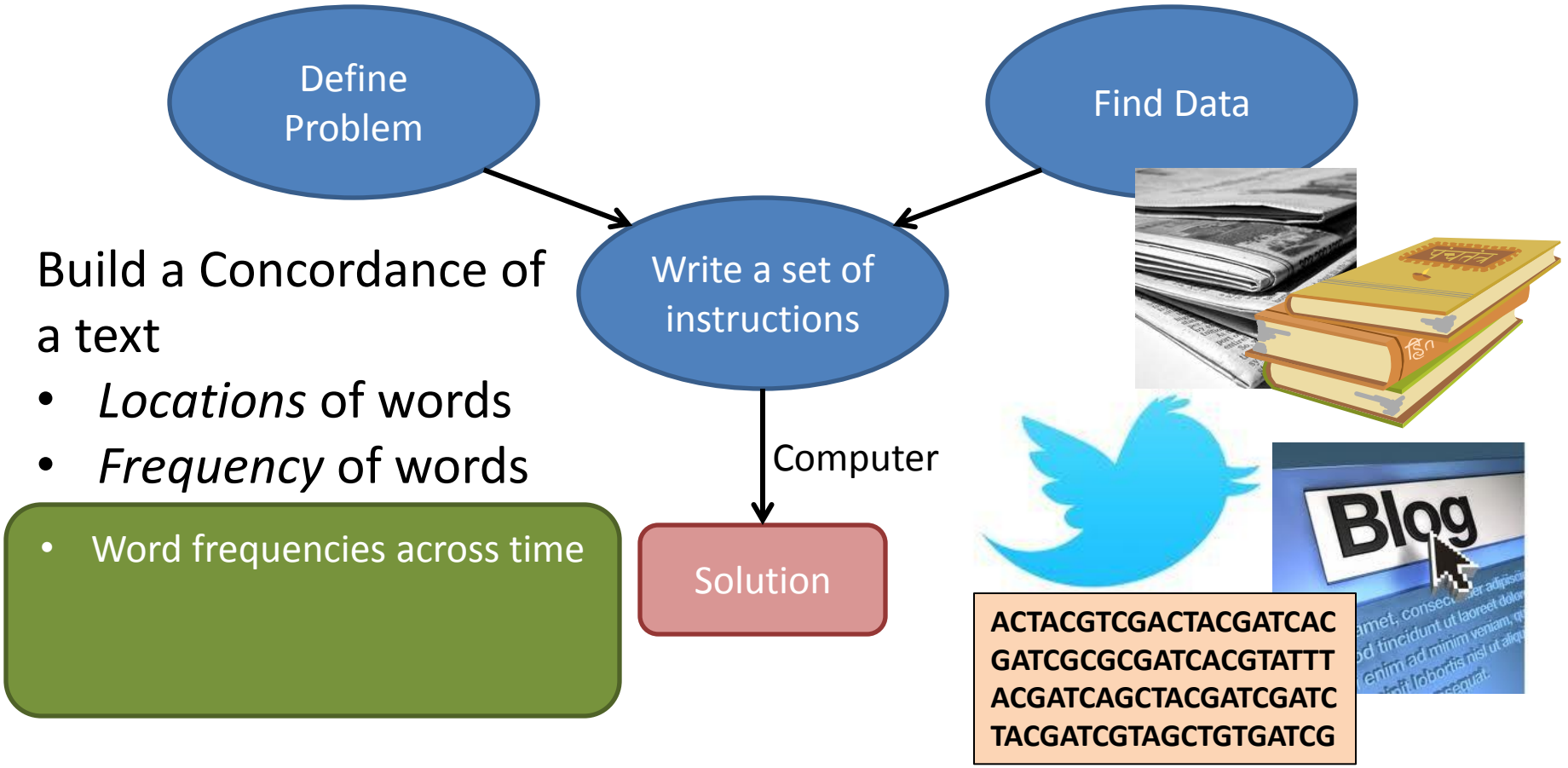
- Click on the “Ngrams” url on the class website
- ngram: a set of *n* words
  - “hello” is a 1-gram
  - “hello there” is a 2-gram

Google Ngrams

- Click on the “Ngrams” url on the class website
- ngram: a set of *n* words
  - “hello” is a 1-gram
  - “hello there” is a 2-gram

- Click on “About Google Books Ngram Viewer” for more information
- Question: what is the data source here?

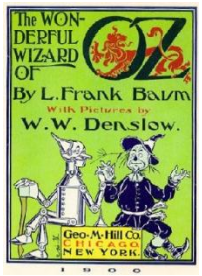
Textual Analysis

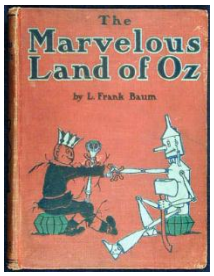


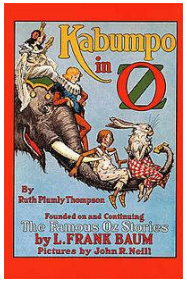
The Wizard of OZ

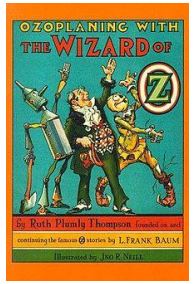
- About 40 Books, written by 7 different authors

#1 ... #14 #15 #16 ... #33









Lyman Frank Baum

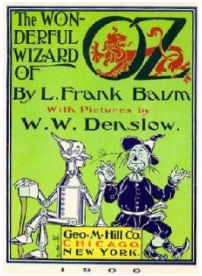
Ruth Plumly Thompson

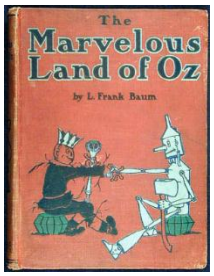
http://www.ssc.wisc.edu/~zzeng/soc357/OZ.pdf

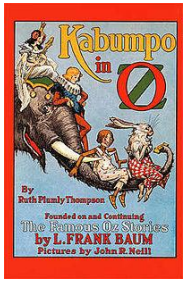
The Wizard of OZ

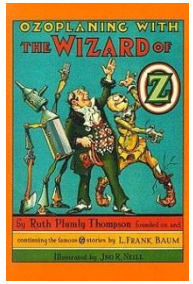
- About 40 Books, written by 7 different authors

#1 ... #14 #15 #16 ... #33









Lyman Frank Baum
(1856-1919)

Ruth Plumly Thompson

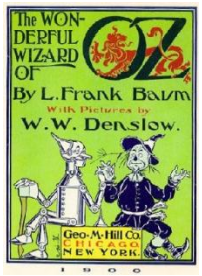
Published in 1921

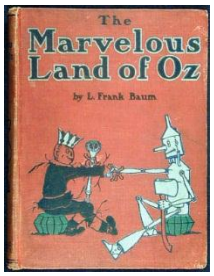
http://www.ssc.wisc.edu/~zzeng/soc357/OZ.pdf

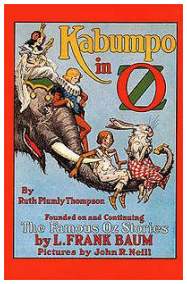
The Wizard of OZ

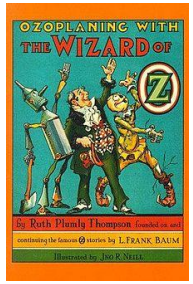
- About 40 Books, written by 7 different authors

#1 ... #14 #15 #16 ... #33









Lyman Frank Baum
(1856-1919)

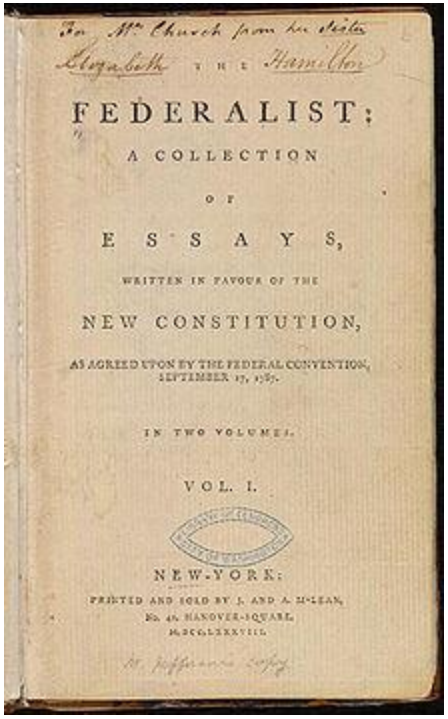
Ruth Plumly Thompson

Published in 1921

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

The Federalist Papers

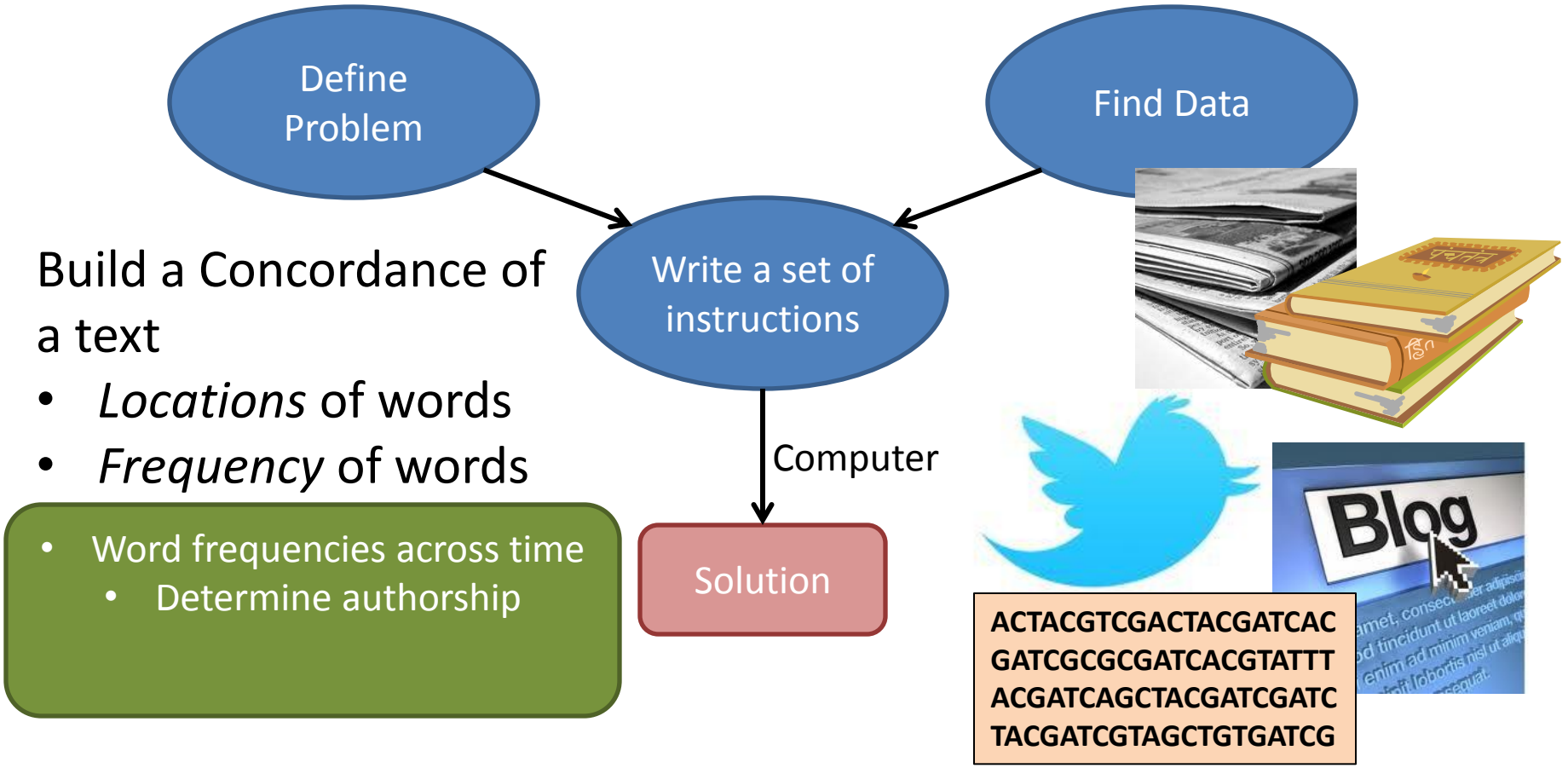
- 85 articles written in 1787 to promote the ratification of the US Constitution
- In 1944, Douglass Adair guessed authorship
  - Alexander Hamilton (51)
  - James Madison (26)
  - John Jay (5)
  - 3 were a collaboration
- Corroborated in 1964 by a computer analysis



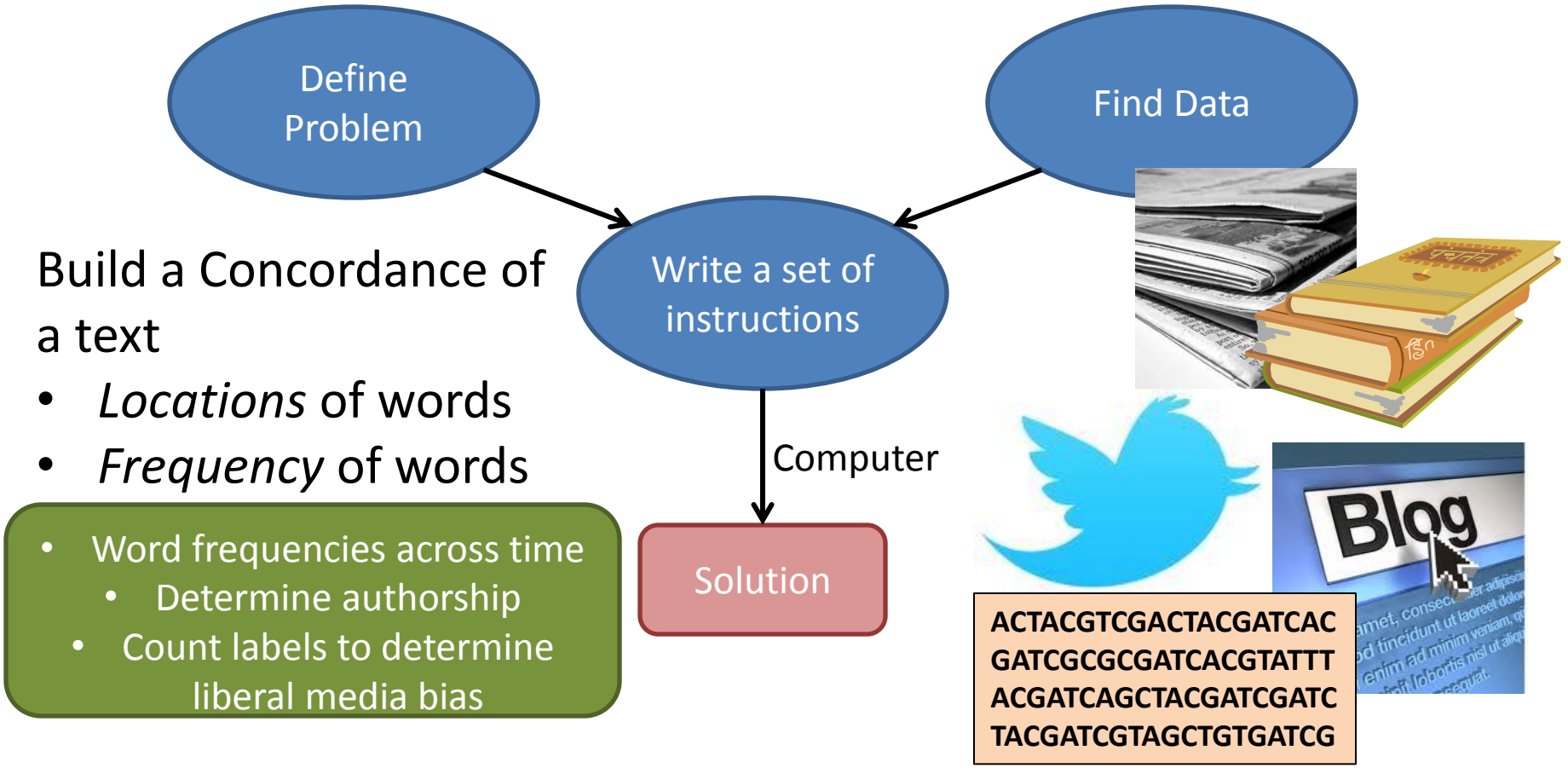
Wikipedia

http://pages.cs.wisc.edu/~gfung/federalist.pdf

Textual Analysis

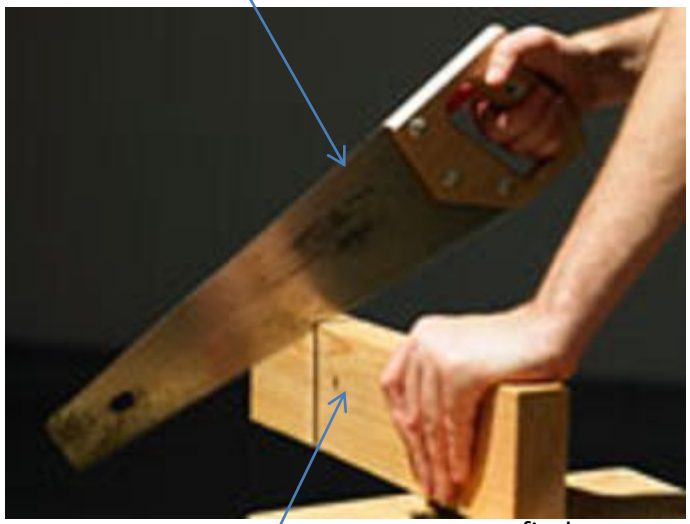


Textual Analysis



How are we going to analyze texts?

Excel

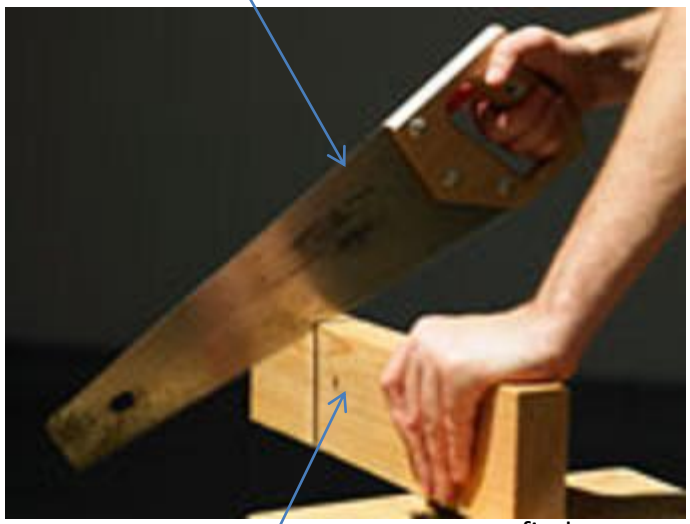


firehow.com

Numerical Data

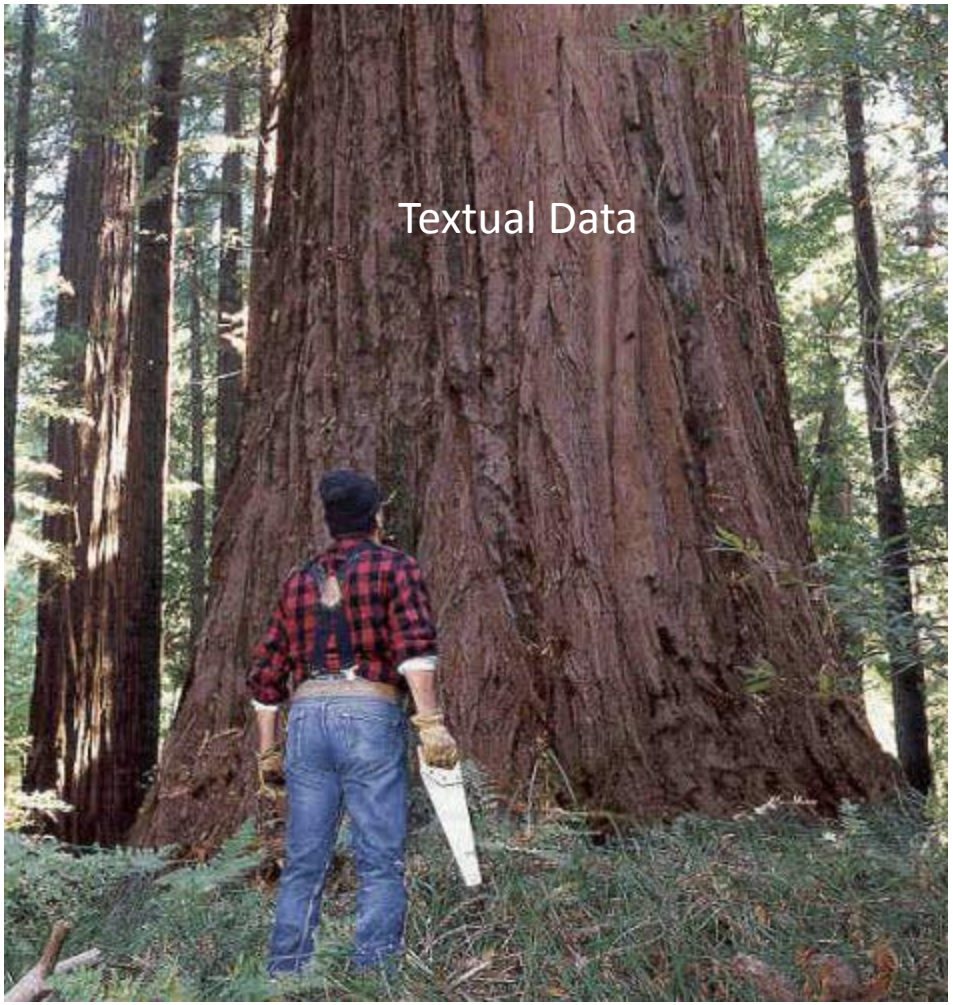
How are we going to analyze texts?

Excel



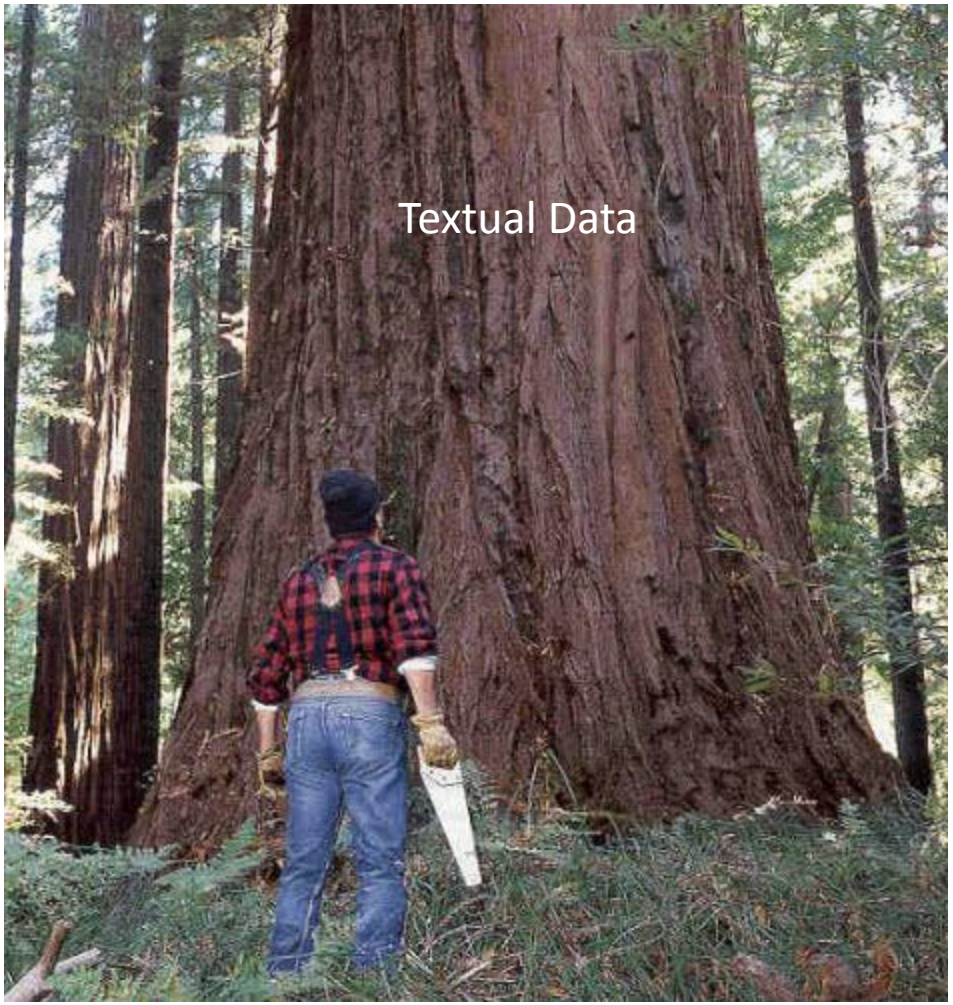
firehow.com

Numerical Data



How are we going to analyze texts?

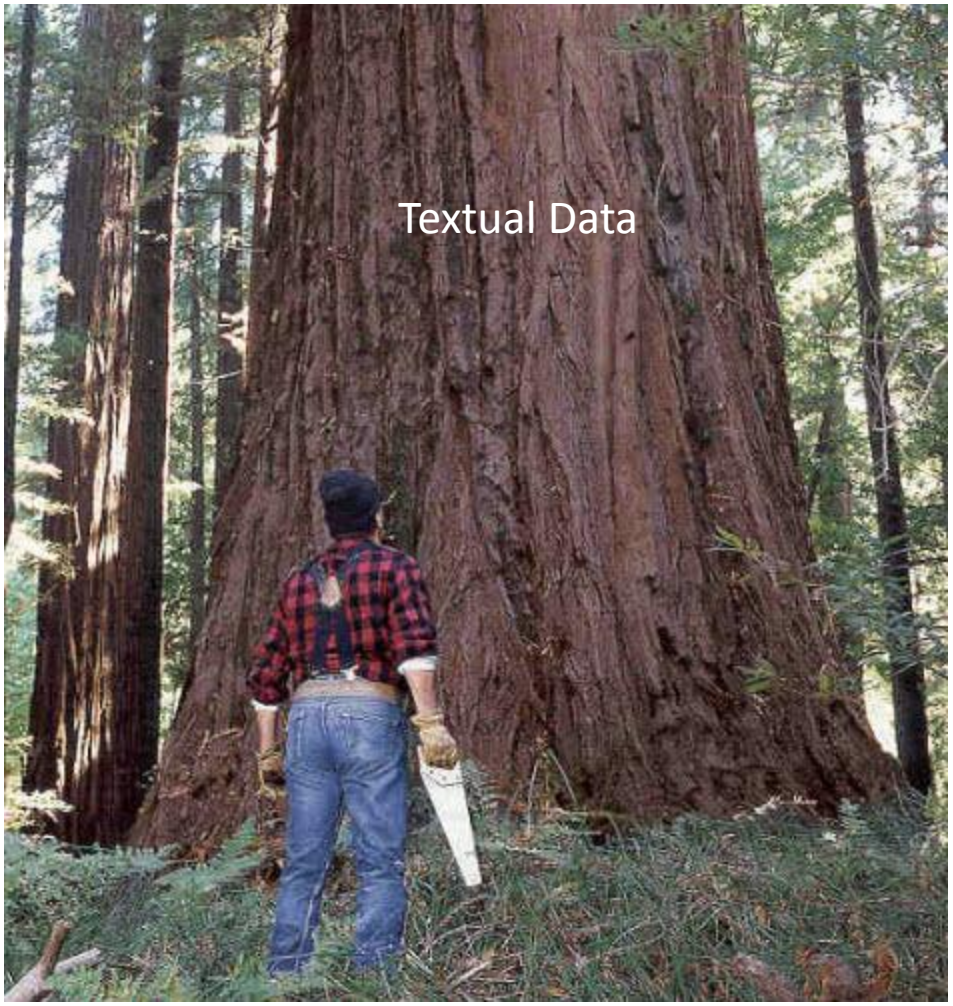
Makita Cordless Chain Saw, $270



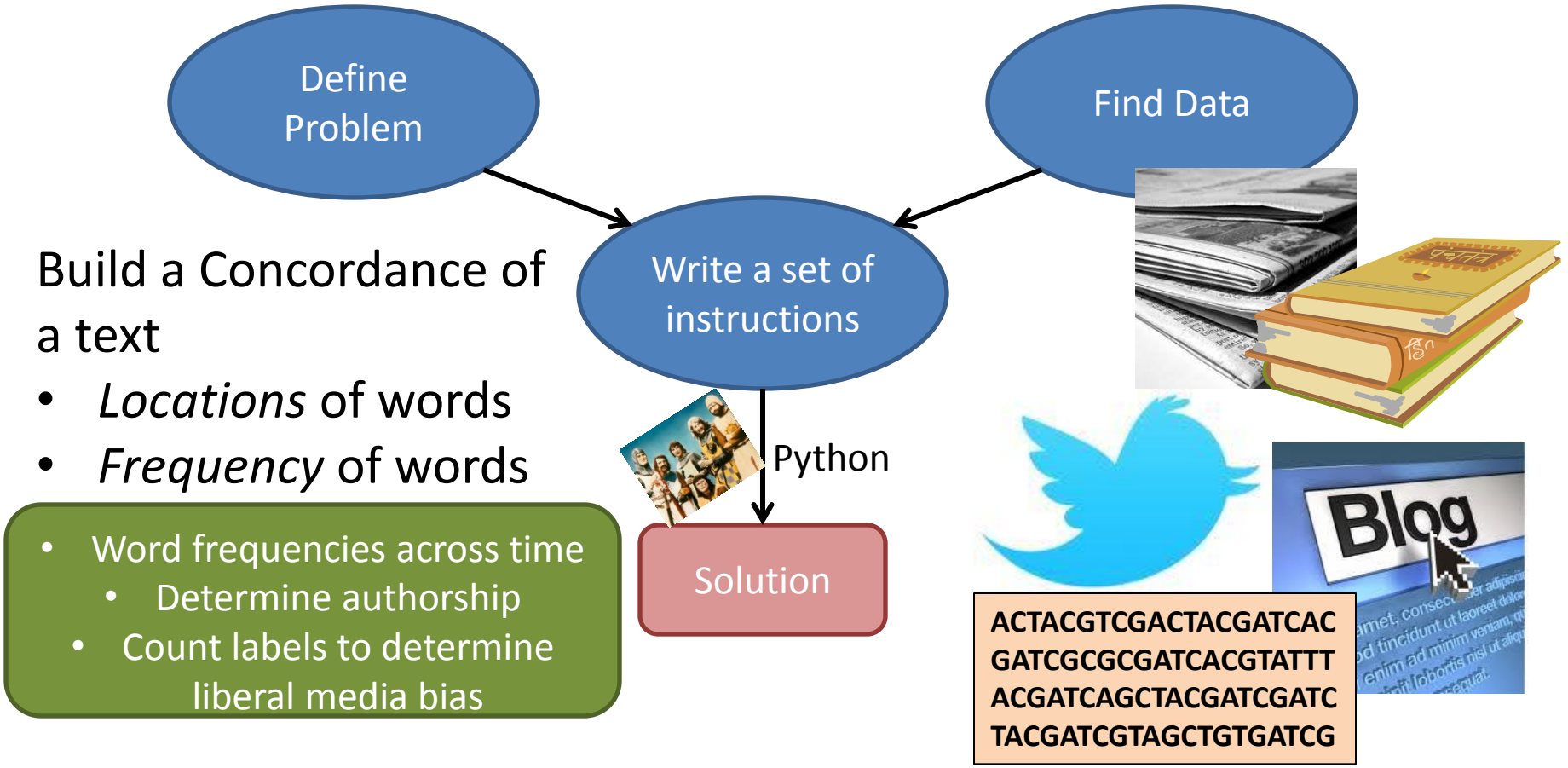
How are we going to analyze texts?

Python: A Programming Language
Free!

9poundhammer.blogspot.com

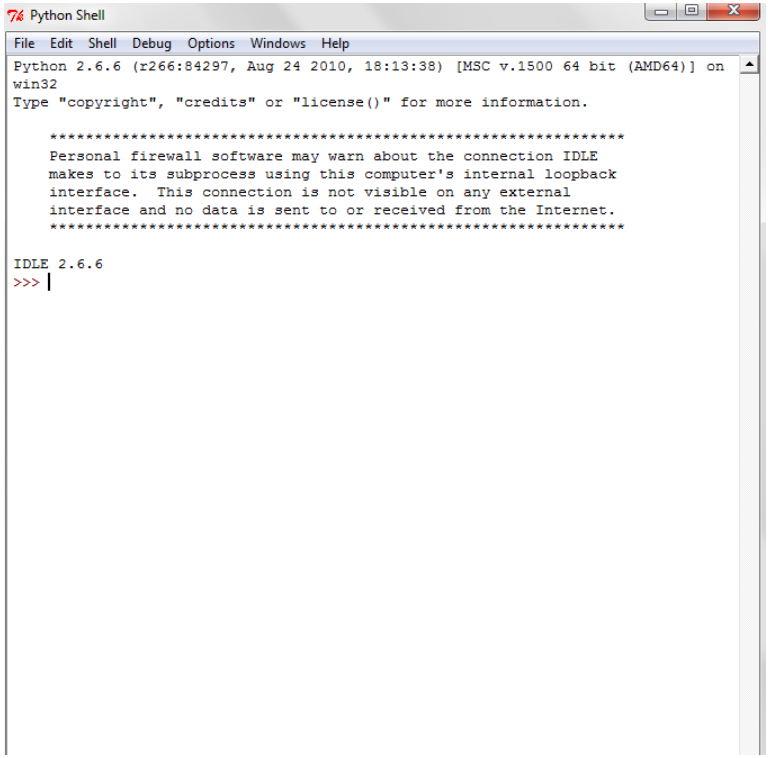


Textual Analysis



Break

- **If you're on a laptop:** install Python 2.6.6 (url)
- **Everyone: Open IDLE (Python GUI)**



Introduction to Python

- **Expressions** are *inputs* that Python evaluates
  - Expressions return an *output*
  - Like using a **calculator**

1. **Expressions**
2. Assignments
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Expressions** are *inputs* that Python evaluates
  - Expressions return an *output*
  - Like using a **calculator**

Type the expressions below

after '>>>' and hit Enter

```
>>> 4+2
6
>>> 4-2
2
>>> 4*2
8
>>> 4/2
2
```

1. **Expressions**
2. Assignments
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

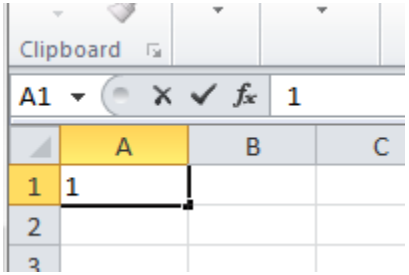
Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.

1. Expressions
2. **Assignments**
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.
  - We've done this kind of thing in Excel

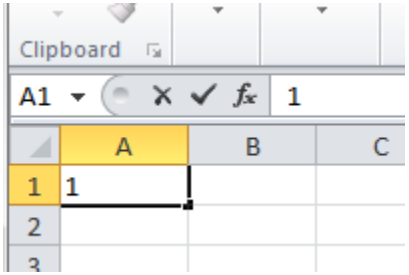


We have *assigned* the number 1 to cell A1.

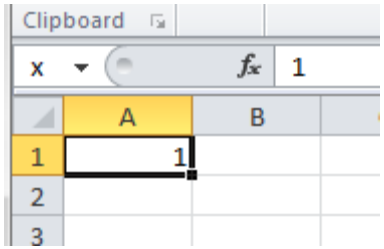
1. Expressions
2. **Assignments**
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.
  - We've done this kind of thing in Excel



We have *assigned* the number 1 to cell A1.



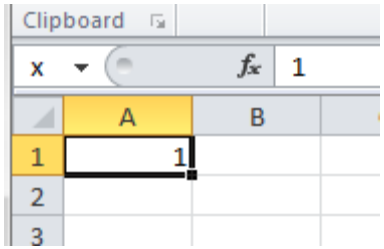
Let's rename cell A1 to x.

1. Expressions
2. **Assignments**
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.
  - We've done this kind of thing in Excel

```
>>> x = 1
```

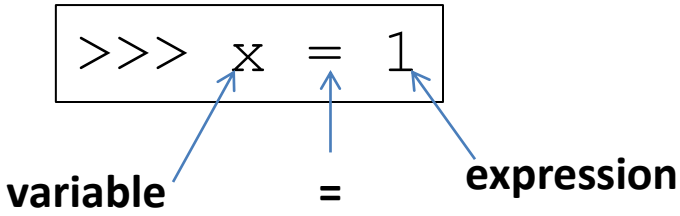


Let's rename cell A1 to x.

1. Expressions
2. **Assignments**
   a) **Variables**
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.
  - We've done this kind of thing in Excel



Memory	
Variable Name	Value
x	1

1. Expressions
2. **Assignments**
   a) **Variables**
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Assignments** do not have an output, they are *stored in memory*.
  - We've done this kind of thing in Excel

```
>>> x = 1
```

  - We can now use x in expressions!

```
>>> x+1
2
>>> (x+2)*3
9
```

Memory	
Variable Name	Value
x	1

1. Expressions
2. **Assignments**
   a) **Variables**
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- You can name your variables anything

```
>>> anna = 100
>>> myNumber = 12345
>>> noninteger = 4.75
```

- Well, *almost* anything
  - No spaces, operators, punctuation, number in the first position
- Variables usually start with a lowercase letter and, if useful, describe something about the value.

1. Expressions
2. **Assignments**
   a) **Variables**
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- Try this:

```
>>> 3/2
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- Try this:

```
>>> 3/2
```

- There are *two* types of numbers in Python. The `type()` function is useful.

```
>>> type(3/2)
<type 'int'>
>>> type(1.5)
<type 'float'>
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- Try this:

```
>>> 3/2
```

- There are *two* types of numbers in Python. The `type()` function is useful.

```
>>> type(3/2)
<type 'int'>
>>> type(1.5)
<type 'float'>
```

- Floats are *decimals*.

```
>>> 3.0/2.0
1.5
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- Try this:

```
>>> 3/2
```

- There are *two* types of numbers in Python. The `type()` function is useful.

```
>>> type(3/2)
<type 'int'>
>>> type(1.5)
<type 'float'>
```

- Floats are *decimals*.

```
>>> 3.0/2.0
1.5
```

General Rule: Expressions for a particular type will *output* that same type!

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Strings** are sequences of characters, surrounded by single quotes.

```
>>> 'hi'
'hi'
>>> myString = 'hi there'
>>> myString
'hi there'
```

1. **Expressions**
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Strings** are sequences of characters, surrounded by single quotes.

```
>>> 'hi'
'hi'
>>> myString = 'hi there'
>>> myString
'hi there'
```

- The + operator concatenates

General Rule: Expressions for a particular type will *output* that same type!

1. **Expressions**
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Strings** are sequences of characters, surrounded by single quotes.

```
>>> 'hi'
'hi'
>>> myString = 'hi there'
>>> myString
'hi there'
```

- The + operator concatenates

```
>>> endString = ' class!'
>>> myString + endString
'hi there class!'
>>> newString = myString + endString
>>> newString
'hi there class!'
```

1. **Expressions**
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Lists** are an ordered collection of items

```
>>> [5,10,15]
[5, 10, 15]
>>> myList = [5,10,15]
>>> myList
[5, 10, 15]
>>> stringList = ['hi','there','class']
>>> stringList
['hi', 'there', 'class']
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Lists** are an ordered collection of items

```
>>> [5,10,15]
[5, 10, 15]
>>> myList = [5,10,15]
>>> myList
[5, 10, 15]
>>> stringList = ['hi','there','class']
>>> stringList
['hi', 'there', 'class']
```

- The + operator concatenates

```
>>> myList + stringList
[5, 10, 15, 'hi', 'there', 'class']
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- To get an element from a list, use the expression `>>> myList[i]` where `i` is the index.
- **List indices start at 0!**

```
>>> myList[0]
5
>>> myList[1]
10
>>> myList[2]
15
```

- What does `>>> myList[1] = 4` do?

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- To get a **range** of elements from a list, use the expression `>>> myList[i:j]` where `i` is the start index (inclusive) and `j` is the end index (exclusive).

```
>>> myList
[5, 4, 15]
>>> myList[0:2]
[5, 4]
>>> myList[1:3]
[4, 15]
>>> newList = [2,5,29,1,9,59,3]
>>> newList
[2, 5, 29, 1, 9, 59, 3]
>>> newList[2:6]
[29, 1, 9, 59]
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- **Indexing** and **ranges** also work on Strings.

```
>>> myString
'hi there'
>>> myString[0]
'h'
>>> myString[5]
'e'
>>> myString[6]
'r'
>>> myString[0:6]
'hi the'
```

1. Expressions
2. Assignments
   a) Variables
3. **Types**
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Introduction to Python

- Remember what assignments do

Memory	
Variable Name	Value
x	1
anna	100
myNumber	12345
noninteger	4.75
myString	'hi there'
endString	' class!'
myList	[5,4,15]
stringList	['hi','there','class']
newList	[2,5,29,1,9,59,3]

1. Expressions
2. **Assignments**
   a) Variables
3. Types
   a) Integers
   b) Floats
   c) Strings
   d) Lists

Class Review

1. Expressions
   - Evaluate *input* and returns some *output* (calculator)
2. Assignments: <variable> = <expression>
   - Store the value of the expression in the variable instead of outputting the value.
   - There is *always* an equals sign in an assignment
   - Variables can be named many things
3. Types
   - Integers vs. Floats (Decimals)
   - Strings in single quotes
   - Lists are sets of other types
   - We can index into Strings & Lists
     - **Indexed starting at 0!**

General Rule: Expressions for a particular type will *output* that same type!