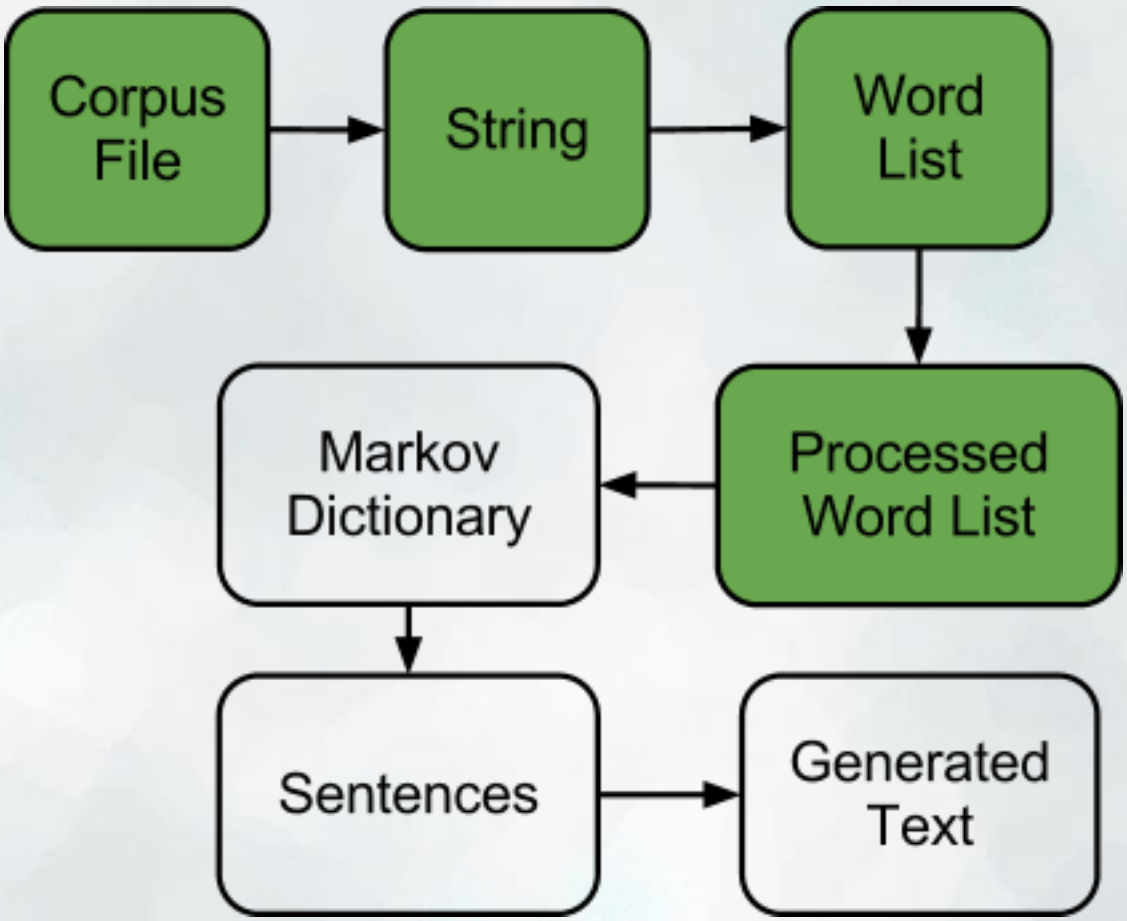
CIS 122

That's the Key

The Big Picture



The Next Step

- We can turn a file into a list of words and periods
- Now we'd like to turn that list into a markov dictionary
  - Associate each word with words that follow it

    fuzzy → [wuzzy, wuzzy, wuzzy, was]

- How do we represent this information?
  - Need a new type

Beyond Lists

- Lists associate values with specific indices
  - ['A', 'B', 'C']
  - The 0th element is 'A'
  - The 2th element is 'C'

- What if we want to associate values with other keys
  - The 42 element
  - The -12 element
  - The 'a' element
  - The 'elephant' element

Dictionaries

- Dictionaries to the rescue!
  - Associate **keys** with **values**
  - Keys can have any (immutable) type
  - Values can have any type

```
fruitColors = { 'apple' : 'red', 'pear' : 'green', 'banana' : 'yellow' }

>> fruitColors[ 'apple' ]
'red'
```

Dictionaries

dictionary = { key1 : value1, key2 : value2, key3 : value3, ... }

key1 → value1
key2 → value2
key3 → value3

Dictionaries

- Dictionaries act a lot like lists
- We can access specific elements
  - But we access them with keys, not indices
  - fruitColors['apple']
- We can modify values
  - fruitColors['apple'] = 'green'
- Keys cannot be modified
  - If you want a different key, make a new one
  - fruitColors['grape'] = 'purple'

Dictionaries

- Let's write a function to give the number of days in a month
  - daysInMonth('January') → 31
  - daysInMonth('February') → 28

- One approach would be to use a ton of if statements

```
def daysInMonth(month):
 if month == 'January':
 return 31
 elif month == 'February':
 return 28
```

- How could we use dictionaries to simplify our code?

Dictionaries

- Store number of days per month in a dictionary
  - Then look up the month we're interested in

```
def daysInMonth(month):
monthDict = {'January' : 31, 'February' : 28, ... }
return monthDict[month]
```

Dictionaries

- We can also build up dictionaries from scratch

```
shoeSize = { }

shoeSize[ 'Bob' ] = 10

shoeSize[ 'Betty' ] = 7

shoeSize[ 'Bertha' ] = 8
```

Have I Seen this Key Before?

- We can only look up keys already in our dictionary

```
>>> coinValue = { 'penny' : 1, 'nickel' : 5, 'dime' :10 }
>>> coinValue[ 'quarter' ]
<ERROR>
```

- How do we tell if a key is present?
  - Use the in keyword

```
>>> 'penny' in coinValue
True

>>> 'quarter' in coinValue
False
```

Have I Seen this Key Before?

- The in keyword works on any kind of sequence

```
5 in [1, 2, 3, 4, 5]
True

6 in [1,2,3,4,5]
False

'a' in 'lighthouse'
False

'light' in 'lighthouse'
True
```

Markov Time

- Let's use a Python dictionary to represent a Markov Dictionary
- What would our keys be?
- What would our values be?

Markov Time

- Let's write a function makeMarkovDict(wordList)
  - Takes a processed word list as input
  - Return a Markov Dictionary
    - Keys are words in list
    - Values are lists of words following that key
- Where do we start?