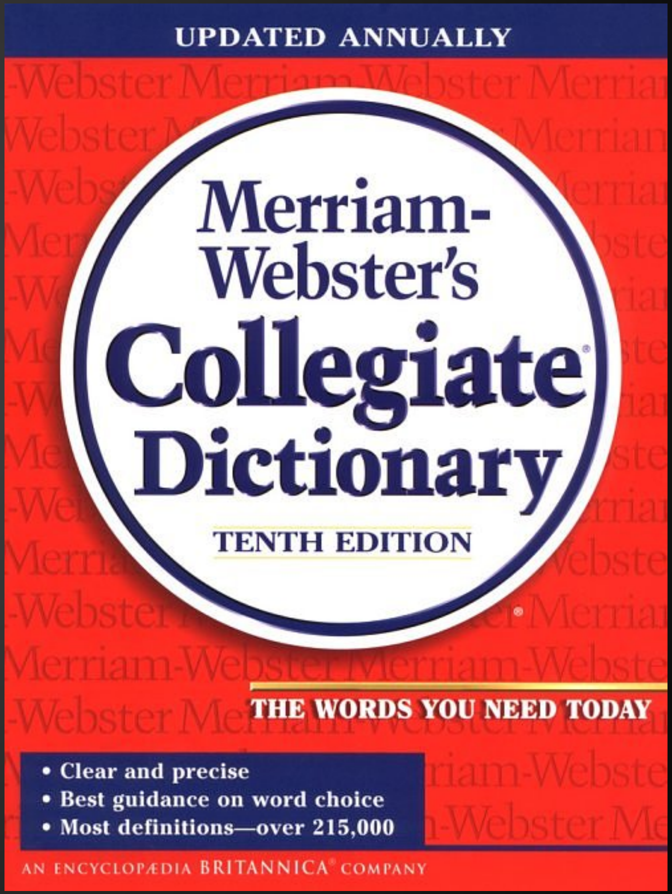
Dictionaries

and some File I/O

Another data structure!

- Lists were our first “data structure”, dictionaries are our second
- In our dictionary we’ll store “key-value pairs” where we can retrieve any value by referring to its key
- In a real dictionary, “key” is the word and “value” is the definition



Dictionary syntax

```
inventory = {‘bananas’: 23, ‘apples’: 47}
inventory[‘papayas’] = 0       # add a new kv-pair
inventory[‘papayas’] += 10   # update value
print inventory[‘bananas’]   # access value
print inventory[‘apples’]      # access value
print inventory[‘papayas’]   # access value
```

File I/O

- I/O stands for input/output
- This lets Python read and write text files on your local machine
- Useful if we want to load external data
  - downloaded books, csv files, etc.
- Or to store data we've computed for later!

File I/O Syntax

```
file_obj = open(‘some_text_file.txt’)  # open file
giant_string = file_obj.read()             # read file
file_obj.close()                            # close file
print giant_string                    # print file contents!

file_to_write = open(‘other_file.txt’, ‘w’)
file_to_write.write(“I’m a banana!”)
file_to_write.close()
```

Case study: Word frequency analysis

- We want to answer some questions about the English language
- How common is the word “happy”?
- What’s the most commonly used word?
- What percent of all words are more than four letters long?

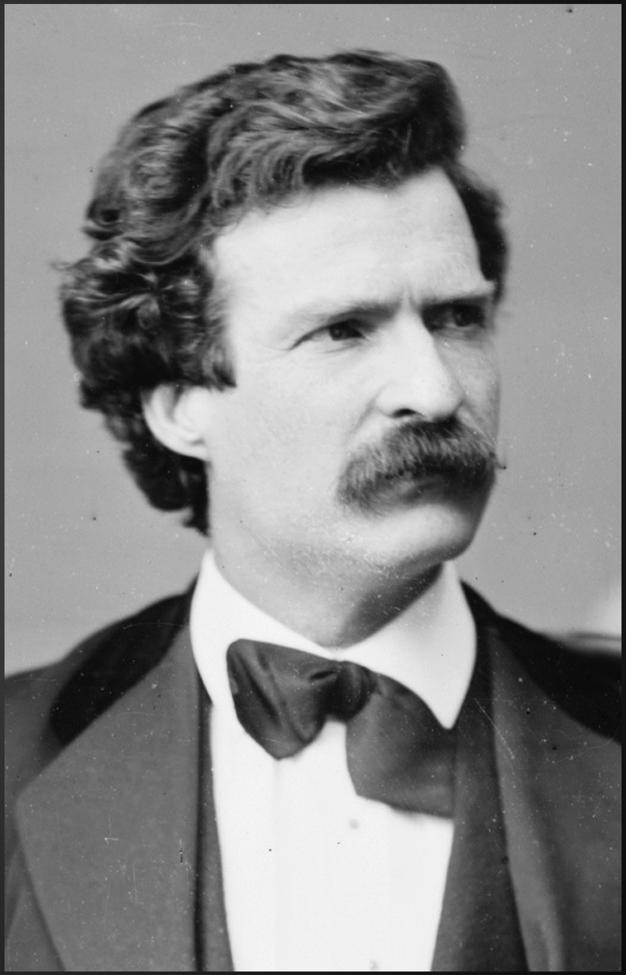
How can we do this using Python?

Steps

1. Find some text data (we’ll use The Adventures of Tom Sawyer)
2. Load the text into a Python program
3. Split the text into a list of words
4. Find the number of times each word appears
5. Use word list and word counts to answer questions
6. Store data to use later
    a. Use the stored data later

Step 1: Find text data

- After an intense Googling session, I found http://www.gutenberg.org/ for free books!
- Many classics were translated from French (e.g. Alexandre Dumas, Jules Verne, etc)
- Mark Twain is a true American hero



Step 2: Load the text into Python

```
file_obj = open(‘tom_sawyer.txt’)
giant_string = file_obj.read()
file_obj.close()
print giant_string
```

Step 3: Split the text

- We want to analyze on a word-by-word basis
- Therefore, we have to convert our giant string into a list of all the words
- You’ve already done this with string.split

Step 3: Split the text

```
file_obj = open(‘tom_sawyer.txt’)
giant_string = file_obj.read()
file_obj.close()
word_list = giant_string.split()
print word_list
```

Step 4: Find count for each word

```
file_obj = open(‘tom_sawyer.txt’)
giant_string = file_obj.read()
file_obj.close()
word_list = giant_string.split()
word_counts = {}                # initialize a new dict
for word in word_list:
    if word not in word_counts:
        # add word to dict if we haven’t seen it yet
        word_counts[word] = 0
    word_counts[word] += 1   # add 1 to count
print word_counts
```

Step 5: Analyze data

```
print “How many times does ‘happy’ appear?”
print word_counts[‘happy’]
print “How many total words are there?”
print len(word_list)
print “What percent of words are ‘happy’?”
print 100.0 * word_counts[‘happy’] / len(word_list)
# 100.0 so division doesn’t round off the decimal!
```

Step 5: Analyze data

How do we find the most commonly used word?

- Iterate through each kv-pair in our dictionary, one by one
  - key -> word, value -> word count
- If the value (word count) of our current word is greater than our maximum so far, then
  - Update our maximum to be the current word count
  - Update our most commonly seen word to be the current word

Step 5: Analyze data

```
print “What’s the most commonly used word?”
max_count = 0
max_word = ‘’
for word in word_counts:
    count = word_counts[word]
    if count > max_count:
        max_count = count
        max_word = word
print max_word, max_count
```

Step 5: Analyze data

```
print “How many words are more than four letters?”
long_word_count = 0
for word in word_list:
    if len(word) > 4:
        long_word_count += 1
print “How many words have more than 4 letters?”
print long_word_count
print “What percent are more than 4 letters?”
print 100.0 * long_word_count / len(word_list)
```

Step 6: Store data to use later

- If we want to run another analysis, we should just use the word counts we already computed instead of the original Mark Twain novel
- So we will store these counts to a file
- This is more File I/O!

Interlude: JSON

- We have a dictionary, and we want to save it to a file
- So first we have to “serialize” the data, or convert it to a string
- JSON: JavaScript Object Notation

```
import json
some_dict = {‘hurshal’: 23, ‘patel’: 47}
dict_string = json.dumps(some_dict)
# dict_string is a string that looks like a dictionary!
```

Step 6: Store data to use later

```
import json
store_file = open(‘word_counts.json’, ‘w’)
text_to_store = json.dumps(word_counts)
store_file.write(text_to_store)
store_file.close()
```

Step 6a: Load saved data!

```
import json
word_counts_file = open(‘word_counts.json’)
word_counts_str = word_counts_file.read()
word_counts_file.close()
# word_counts_str is a string that looks like a dict
word_counts = json.loads(word_counts_str)
# word_counts is actually a dictionary!
```

Questions? Comments? Concerns?