Text Comparison With Iteration
Code snippets 2-9
Text Comparison
Text Comparison

• Can you identify the authors by comparing the style of writing?
Text Comparison

• How do you discuss analytical differences between different texts?

• You need computable metrics that have a grounded meaning
Unique Words Ratio

- Fraction of words that occur only a single time
- Measures richness of vocabulary
- Shakespeare’s plays include 28,829 different words, 12,493 which occur only a single time
- 1 in 70 words in plays written by Shakespeare are single occurrence words!
Word lengths ratio

- Ratios between short words to medium-length words, and long words to medium-length words
- Measures difficulty of vocabulary

I do not like them, Sam-I-am.
I do not like green eggs and ham.
Readability
Emotional Sentiment

- Words are assigned an emotional sentiment score
- Whether a text has a positive or negative sentiment is simply a summation over the words' individual sentiment scores
- Quick demo
Iteration with for loops
for loop

for (2) in (1):

(3)

1. Input iterable: a list, set, dictionary, range, etc

2. The temporary iteration variable: give a descriptive type that matches the type of the individual item in the operable

3. The iteration computation: occurs once for each item in the input iterable

Don’t forget the colon!
Counting with for loops

numbers = range(10)
count = 0
for n in numbers:
    count += n
print(count)
squares = [i**2 for i in range(10)]

squares = []
for i in range(10):
    squares.append(i**2)
Putting items in a list

```
with open('file.txt') as f:
    words = f.read().split()

a_words = []
for word in words:
    if word[0] == 'a':
        a_words.append(word)

print(a_words)
```
Creating a dictionary

```python
with open('file.txt') as f:
    words = f.read().split()

first_letters = {}

for word in words:
    first_letters[word] = word[0]

print(short_words)
```
Creating a dictionary of lists

```python
with open('file.txt') as f:
    words = f.read().split()

words_by_letter = {}

for word in words:
    first_char = word[0]
    if first_char not in words_by_letter:
        words_by_letter[first_char] = []
    words_by_letter[first_char].append(word)

print(words_by_letter)
```
Iterating through a dictionary

for key, value in dictionary.items():
    print("{}: {}").format(key, value)

Temporary variable for key
Temporary variable for value
Don’t forget items()!
Loop
Iteration Computation
Iterating through a dictionary

counter = Counter(words)
for word, count in counter.items():
    print("{}: {}".format(word, count))
Iterating through a dictionary

for letter, words in words_by_letter.items():
    print("{{} {}".format(letter, words))
While loops

while (1):
    (2)

1. Condition that evaluates to True or False. If True, code in while loop will execute, otherwise it will skip it.

2. The iteration computation: occurs each time the condition is True.
While loops

users_choice = request_choice()

while users_choice == 'invalid':
    users_choice = request_choice()

# Do something with users_choice

Initialize condition variable
Loop
Update condition
Use Output Variables
While loops

users_choice = request_choice()

while check_if_invalid(users_choice):
    users_choice = request_choice()

# Do something with users_choice
While loops

```
match_result = play_match()

while match_result == 'draw':
    match_result = play_match()

print("You " + match_result)
```