Tutor Notes

Part 1) Getting text from a file

- read in from file
- OPTIONAL: write to file

Part 2) reading in several files

- Making a list with the file names you want to use
- Putting a loop around the read-in code. Save all read-in texts in a list.
- Put the training part in a loop that goes through all of the texts.

Part 3) Bigram + Trigram

- In the training part:

  Inside the inner loop want to store:

  ```
  current_pair = text[num] + " " + text[num+1]
  current_word = text[num+1]
  next_word = text[num +1]
  ```

  (change the loop to stop at (num_words-2))
  Repeat the if-else code that is written for the current_word the same way for the current_pair.

- In the generating part:
  Inside the inner loop we want to first check if current_pair in cups, otherwise check if current_word in cups.

  Get the next_word_options accordingly.

  Now, if next_word_options is not empty (or None) choose a random next word, print it, and update current_pair and current_word (in this order!)

Part 1: Reading Texts!

Reading in from files:

```
import random

# Reading a file:
with open("intermediate_winnie_the_pooh.txt", "r") as input_file:
      source_text = input_file.read()


# Training:
cups = {}
split_text = source_text.split()
num_words = len(split_text)

for i in range(num_words-1):
      current_word = split_text[i]
      next_word = split_text[i+1]
      if current_word not in cups:
            cups[current_word] = [next_word]
      else:
            cups[current_word].append(next_word)


# Start the program
print("I am a markov chain generator.")
current_word = input("What word do you want to start with? ")

# Generating:
print(current_word, end=" ")
for i in range(100):
      if current_word in cups:
            next_word_options = cups[current_word]
            next_word = random.choice(next_word_options)
            print(next_word, end=" ")
            current_word = next_word

# OPTIONAL: writing to output file
```

Part 2: Longer Texts!

Reading in from files:

Same code but adding the line:

```
split_text = split_text.replace("?", "")
split_text = split_text.replace(".", "")
split_text = split_text.lower()
...
```

And other punctuation symbols they like to replace.

Part 3: One file is not enough

Reading in several text files: (Note: bold lines are new or edited lines)

```
import random

files = ["intermediate_beatles.txt", "long_disney.txt"]

# Training:
cups = {}
for file in files:
      source_text = ""
      with open("intermediate_winnie_the_pooh.txt", "r")as input_file:
            source_text = input_file.read()

      split_text = source_text.split()
      num_words = len(split_text)

      for i in range(num_words-1):
            current_word = split_text[i]
            next_word = split_text[i+1]
            if current_word not in cups:
                  cups[current_word] = [next_word]
            else:
                  cups[current_word].append(next_word)


# The rest stays the same as before
```

Part 4: A Smarter Generator

Note: Trigrams Only

This version stores the current_pair as the key of the dictionary. (Note: bold lines are new or edited lines)

```
import random

# Reading a file: (or more files if previous part is done)
with open("intermediate_winnie_the_pooh.txt", "r") as input_file:
      source_text = input_file.read()

# Training:
cups = {}
split_text = source_text.split()
num_words = len(split_text)

for num in range(num_words-2):
      current_pair =  split_text[num] + " " + split_text[num+1]
      next_word = split_text[num + 2]

      if current_pair not in cups:
          cups[current_pair] = [next_word]
      else:
          cups[current_pair].append(next_word)


# Start the program
print("I am a markov chain generator.")
current_pair = input("What word_pair do you want to start with? ")

# Generating:
print(current_pair, end=" ")

for i in range(100):
      next_word_options = None
      if current_pair in cups:
            next_word_options = cups[current_pair]
      if next_word_options:  #students can also check for empty
            next_word = random.choice(next_word_options)
            print(next_word, end=" ")
            current_pair = current_pair.split()[1] + " " + next_word
```

Part 4.3:

Note: Trigrams and Bigrams

This version stores the curent_word **and** current_pair as the keys of the dictionary.
(Note: bold lines are new or edited lines)

```
import random

# Reading a file: (or more files if previous part is done)
with open("intermediate_winnie_the_pooh.txt", "r") as input_file:
      source_text = input_file.read()


# Training:
cups = {}
split_text = source_text.split()
num_words = len(split_text)

for num in range(num_words-2):
      current_pair =  split_text[num] + " " + split_text[num+1]
      current_word = split_text[num+1]
      next_word = split_text[num + 2]

      if current_pair not in cups:
          cups[current_pair] = [next_word]
      else:
          cups[current_pair].append(next_word)

      if current_word not in cups:
          cups[current_word] = [next_word]
      else:
          cups[current_word].append(next_word)


# Start the program
print("I am a markov chain generator.")
current_word = input("What word do you want to start with? ")

# Generating:
print(current_word, end=" ")
current_pair = current_word
for i in range(100):
      next_word_options = None
      if current_pair in cups:
            next_word_options = cups[current_pair]
      elif current_word in cups:
            next_word_options = cups[current_word]
```

```
if next_word_options:  #students can also check for empty
      next_word = random.choice(next_word_options)
      print(next_word, end=" ")
      current_pair = current_word + " " + next_word
      current_word = next_word
```