Repetition & Reinforcement

Andey Robins & Dr Borowczak

November 1-3, 2022

Outline

Outline

- Schedule
- Dictionaries Reinforcement
- Example Problems
- Applications for Dictionaries
- Game Modification

Class Schedule

Week	Tuesday	Thursday	Assignments
Oct 31	Reinforcement	Reinforcement	Quest Redo
Nov 7	No Class	No Class	Quest D
Nov 14	Classes	Classes	Lab
Nov 21	Fall Break	Fall Break	
Nov 28	Reinforcement	Reinforcement	Lab
Dec 5	No Class	No Class	Final Quest Retakes
Dec 13	Final Quest	Comprehensive	10:15 AM

Dictionaries

Syntax

Declaration: `dictionary = {}`

Access: `dictionary[key]`

Membership: `if "string" in dictionary:`

Syntax

Looping:

```
for key, val in dictionary:
  print(key)
  print(val)
```

Syntax

Removal: `del dictionary[key]`

What does it do?

```
dictionary = {
  'key1': 1,
  'key2': 2,
  'key3': 3
}

del dictionary['key4']
```

Syntax

Smart Removal:

```
if key in dictionary:
  del dictionary[key]
```

Example Problems

Caesar Cipher Decoder

Given some text encrypted with the Caesar cipher, can we decode it?

Aside: The Caesar Cipher

How do I send secret messages?

How do I send this message in secret to Alicia?

“What time is the Wicys meeting?”

Encryption

Encryption is a methodology for obscuring the true meaning of data. The message from the previous slide might look like this: “Zkdw wlph lv wkh Zlfbv phhwlqj?”

Caesar Cipher

An illustration of the Caesar cipher

Attack

A potential attack vector on the Caesar cipher: frequency analysis

Caesar Cipher Decoder

Given some text encrypted with the Caesar cipher, can we decode it?

```
cipher_text = """
L pxvw qrw ihdu.
Ihdu lv wkh plqg-nloohu.
Ihdu lv wkh olwwoh-ghdwk wkdw eulqjv wrwdo
  reolwhudwlrq.
L zloo idfh pb ihdu.
L zloo shuplw lw wr sdvv ryhu ph dqg wkurxjk ph.
Dqg zkhq lw kdv jrqh sdvw, L zloo wxuq wkh lqqhu
  hbh wr vhh lwv sdwk.
Zkhuh wkh ihdu kdv jrqh wkhuh zloo eh qrwklqj.
  Rqob L zloo uhpdlq.
"""
```

Outline

1. Count the frequency of each letter

1. Count the frequency of each letter
2. Find the most frequent letter

1. Count the frequency of each letter
2. Find the most frequent letter
3. Calculate a shift with the assumption that ‘e’ is the most common letter

1. Count the frequency of each letter
2. Find the most frequent letter
3. Calculate a shift with the assumption that ‘e’ is the most common letter
4. Shift the cipher text letters back

1. Count the frequency of each letter
2. Find the most frequent letter
3. Calculate a shift with the assumption that ‘e’ is the most common letter
4. Shift the cipher text letters back
5. Output the decrypted message.

Frequency Count

```
letter_freq = {}
for letter in cipher_text:
  if letter in letter_freq:
    letter_freq[letter] += 1
  else:
    letter_freq[letter] = 1
```

```
letter_freq = {}
for letter in cipher_text:
  if letter.isalpha():      # addition
    if letter in letter_freq:
      letter_freq[letter] += 1
    else:
      letter_freq[letter] = 1
```

Find the Most Frequent

```
most_letter = ""
most_letter_count = 0
for letter, count in letter_freq.items():
  if count > most_letter_count:
    most_letter = letter
    most_letter_count = count
```

Calculate Shift

Difference between the highest letter and the letter 'e'. `ord()` is the function to do this.

```
shift = ord(highest_letter) - ord('e')
```

Shift Message Back

```
plain_text = ""
for letter in cipher_text:
    plain_letter_ord = ord(letter) - shift
    plain_text += chr(plain_letter_ord)
```

```
plain_text = ""
for letter in cipher_text:
  if letter.isalpha():
    plain_letter_ord = ord(letter) - shift
    plain_text += chr(plain_letter_ord)
  else:
    plain_text += letter
```

Decryption Complete

See the code in Codio for this code in action.

Caesar Cipher Counterexample

What is the following word?

“Mdcc”

Jazz - 3

Limits

What are the limits of the Caesar cipher?

1. Short cipher text
2. Non-letter characters
3. Frequency analysis
4. Languages

The Caesar Cipher Continued

Left Off

```
plain_text = ""
for letter in cipher_text:
  if letter.isalpha():
    plain_letter_ord = ord(letter) - shift
    plain_text += chr(plain_letter_ord)
  else:
    plain_text += letter
```

```
plain_text = ""
for letter in cipher_text:
  if letter.isalpha():
    plain_text += wrap_around_shift(letter, shift) # desir
  else:
    plain_text += letter
```

Wrap around shifting

1. Convert a letter to a number in the alphabet
2. Shift that number by the shift value
3. Ensure that is a valid number in the alphabet (i.e 0 <= n <= 25)
4. Convert the shifted value back to a letter in ASCII

Brief Aside: How do computers store letters?

Binary

Everything in a computer is just a 1 or a 0. We often combine these into sets of 1s and 0s called *bytes*. A byte has 8 *bits*. By assigning different meanings to the numbers between 0 and 255 (

$$2^8 - 1$$

), we can associate different *semantic* values with those numbers.

ASCII

The Ascii table

```
>>> ord('a')
97

>>> ord('b')
98

>>> ord('!')
33
```

```
>>> # The shift between e and h
>>> ord('h') - ord('e')
3
```

```
shift = ord(highest_letter) - ord('e')
...
plain_letter_ord = ord(letter) - shift
plain_text += chr(plain_letter_ord)
```

Handling Edge Cases

```
shift = ord('I') - ord('e')
shift == -28
```

- This number is negative
- This number is greater than 26 (the letters in the alphabet)
- Capitals and lower case aren't next to each other
- We would get random characters if we try to shift outside the alphabet

Handle Negative

How could we handle this negative number?

What way should we make it positive?

```
shift = -28
while shift < 0:
  shift += 26
```

Modular Arithmetic

How many integers are there?

How many numbers are there?

Are there more numbers than integers?

What would it mean to do math if there weren't infinite integers?

(Addition? Multiplication? Number bases?)

Enter Modular Arithmetic

Modular Arithmetic is a form of mathematics that works with a finite number set.

What is 11:00 am plus 50 minutes?

An analog clock

43 minutes plus 1 hour 17 minutes is 2 hours.

or

$43 + 117 = 200$ *(with some modular magic)*

What does this have to do with the alphabet?

What is $s + x$?

p

```
>>> (ord('s') - 97) + (ord('x') - 97)
41
>>> chr((41 % 26) + 97)
'p'
```

Using the modulus operator (remember **%**), we can calculate what the remainder of adding numbers is!

Wrap around shifting

1. Convert a letter to a number in the alphabet
2. Shift that number by the shift value
3. Ensure that is a valid number in the alphabet (i.e 0 <= n <= 25)
4. Convert the shifted value back to a letter in ASCII

Letter to Alpha Number

```
def ord_to_letter_num(letter):

  return letter_num
```

```
def ord_to_letter_num(letter):
  letter_ord = ord(letter)

  return letter_num
```

```
def ord_to_letter_num(letter):
  letter_ord = ord(letter)
  letter_num = letter_ord - 97
  return letter_num
```

Wrap Around Shift

```
def round_shift(letter, shift):

  return chr(plain_num + 97)
```

```
def round_shift(letter, shift):
  cipher_num = ord_to_letter_num(letter)

  return chr(plain_num + 97)
```

```
def round_shift(letter, shift):
  cipher_num = ord_to_letter_num(letter)
  plain_num = (cipher_num - shift) % 26
  return chr(plain_num + 97)
```

Call this function

```
plain_text = ""
for letter in cipher_text:
  if letter.isalpha():
    plain_text += wrap_around_shift(letter, shift) # desir
  else:
    plain_text += letter
```

Result

```
c must not fear.
zear is the mind-killer.
zear is the little-death that brings total obliteration.
c will face my fear.
c will permit it to pass over me and through me.
und when it has gone past, c will turn the inner eye to see
qhere the fear has gone there will be nothing. inly c will
```

Handling Capital letters

Live code demo

A Brief Aside: Unicode

An-nyeong an informal Korean greeting