EEC 688
Secure and Dependable Computing

PROJECT

Son Phan (2367854)

PROJECT

Fdhvdu Flskhu

Fkdoohqjh

lq sbwkrq

PROJECT

Caesar Cipher

Challenge

in Python

Problem

Substitution Ciphers

- Idea: each letter or group of letters is replaced by another letter or group of letters
- Caesar cipher – circularly shift by 3 letters
  - a -> D, b -> E, … z -> C
  - More generally, shift by k letters, k is the key
- Monoalphabetic cipher – map each letter to some other letter
  - A b c d e f … w x y z
  - Q W E R T Y … V B N M <= the key

Challenge

Caesar Cipher Crack: write a program that when…

- Given an encoded message as a Caesar cipher text
- Find:
  - the key (k value)

    &
  - the plain text message

Reward

"I'll give you an A for the course."

- Dr. Zhao

“Challenge accepted.”

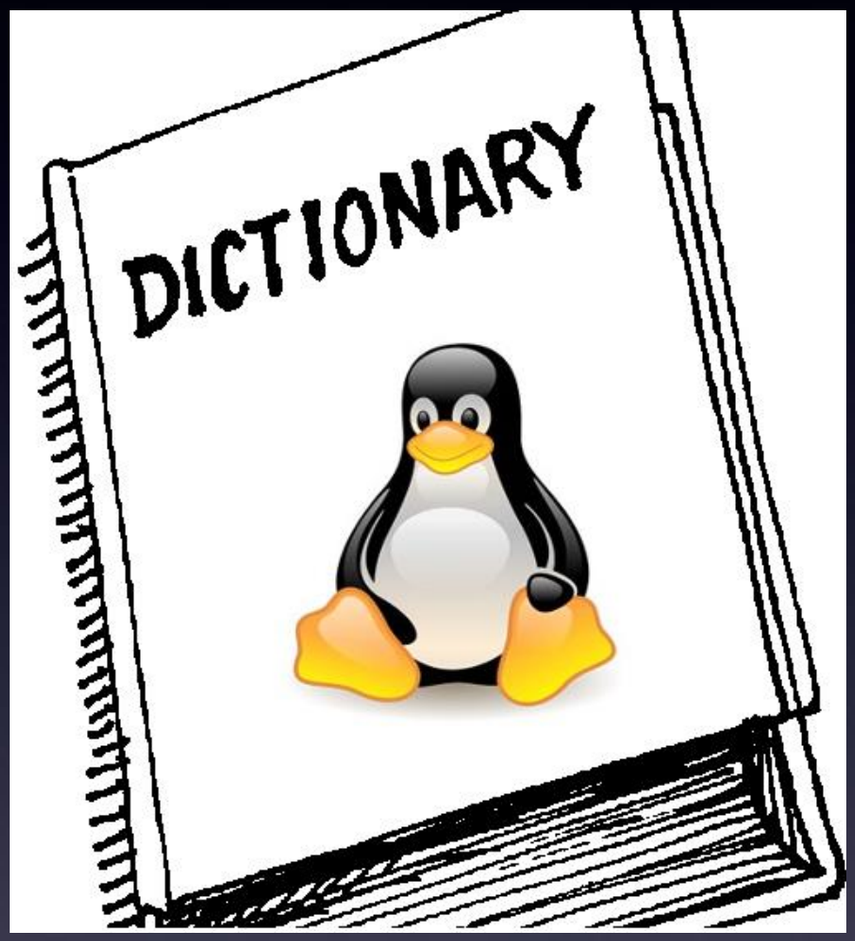
Brain-storm...

khoor zruog

vs.

hello world

*Hint



Features

Encoder:

- Prompt for user-input message and key (k value)
- Encode it and output the cipher text

Decoder:

- Prompt for cipher text and key (k value)
- Decode it and output the plain message

Hacker:

- Prompt for user-input cipher message
- Hack it to output the key and the plain message

Features

Instructor:

- Full solution.
- Include all steps.
- Quickly check for answers.

Features

- Language: Python
- Dictionary (English): Unix
- Interface: command-line

Beautiful is better than ugly

Explicit is better than implicit

Simple is better than complex

Complex is better than complicated

Readability counts

*Source: PEP 20 (The Zen of Python)"

Python

```
print "Hello, World"
```

Java

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

Code Snippets

Encoder

```
# GET USER INPUT
input_ms    = raw_input("Input your message (ignore case & non-alphabet chars): ").lower()
input_key   = int(raw_input("Shift how many letters (key)? k = "))
# ENCODE
cipher_ms   = caesar_encode(input_ms, input_key, alphabet)
```

Decoder

```
# GET USER INPUT
cipher_ms    = raw_input("Input your cipher (ignore case & non-alphabet chars): ").lower()
input_key   = int(raw_input("Shifted how many letters (key)? k = "))
# DECODE
plain_ms    = caesar_decode(cipher_ms, input_key, alphabet)
```

Code Snippets

```
# SETUP VARIABLES
alphabet = "abcdefghijklmnopqrstuvwxyz"
```

```
# CAESAR FUNCTIONS
def translator(text, intab, outtab):
    trantab = maketrans(intab, outtab)
    return text.translate(trantab)

def caesar_encode(plaintext, k, letters):
    subs = letters[k:]+letters[:k]      #substituted letters
    return translator(plaintext, letters, subs)

def caesar_decode(ciphertext, k, letters):
    subs = letters[-k:]+letters[:-k]    #substituted letters
    return translator(ciphertext, letters, subs)
```

Code Snippets

Hacker

```
# SETUP DICTIONARY and VARIABLES
dictionary  = set(open('words.txt','r').read().lower().split())
max_len     = max(map(len, dictionary)) #longest word in the set of words
```

```
# GET USER INPUT
cipher_ms = raw_input("Input your cipher: ").lower()

# INIT VARS
words_found = set() #set of words found, starts empty
words_count = []    #list of words count
guess_texts = []    #list of guessed texts

# BRUTE-FORCE CAESAR GUESSING
# Loop from k=1 to k=length-of-alphabet-string (ie. 26) and decode cipher message
# Slice through decoded message as chunks to match words in dictionary
# Count number of words that matched and append the count to a list/array
# whichever k that led to the most words matched has the highest probability of decoding the original text.
print
print "RESULTS"
for k in xrange(len(alphabet)):
    guesstext = caesar_decode(cipher_ms, k, alphabet) #decode cipher message for each k value
    guess_texts.extend([guesstext]) #store into list

    for i in xrange(len(guesstext)):        #for each possible starting position in the corpus
        chunk = guesstext[i : i+max_len+1]  #chunk that is the size of the longest word

        for j in xrange(1, len(chunk)+1):   #loop to check each possible subchunk
            word = chunk[:j]                #subchunk

            if word in dictionary:          #constant time hash lookup if it's in dictionary
                words_found.add(word)       #add to set of words

    words_count.extend([len(words_found)]) #store into list
    words_found.clear() #clear set ready for the next iteration

max_count = max(words_count) #max_index = words_count.index(max_count)
# List of keys sorted descending by the number of words matched in dictionary
key_list  = sorted(range(len(words_count)), key=lambda k:words_count[k], reverse=True)

for i in xrange(len(key_list)):
    # Calculate the probability of each guess by dividing the number of words count
    # of each k to the highest number of words found
    guessprob = "({0:.1f}%)".format((float(words_count[key_list[i]])/max_count)*100)
    print "k =", key_list[i], "-> message:", guess_texts[key_list[i]], guessprob
```

Code Analysis

Thanks to GitHub

and

Eclipse

LIVE DEMO

https://github.com/sonvphan/

Future

- Fork me on GitHub
https://github.com/sonphanusa/CaesarCipherHacker/

- More dictionaries matching: slang (LOL), emoticons,…

- Wheel of Cipher: game for cipher geeks.

Remarks

- Fun – “I <3 Challenge.”
- Learn – put what we learned in class to good use
- Share – source code, class materials
- Grow – Anyone can code.

Reference

- Python:
  https://www.python.org/

- Tim Peters, The Zen of Python:
  http://legacy.python.org/dev/peps/pep-0020/

- Maketrans()
  http://www.stealthcopter.com/blog/2009/12/python-cryptograph-using-maketrans-for-substitution-and-caesar-ciphers/

- Stackoverflow community:
  http://stackoverflow.com/questions/19338113/how-to-find-possible-english-words-in-long-random-string

Q&A

Thanks!