

Term Project

Caesar Cipher Challenge

Objective

The project was initiated as a challenge to hack a Caesar Cipher encoded message to find the key (k value) and the original plain text message. However, in the spirit of reusability and fulfilling the lab track requirement of the class, this paper presents the solution as a new lab for the class with detail instruction to help future students understand the Cipher concept and programming.

The students will also be introduced “hands-on” to the two most common yet basic hacking methods, brute-force and dictionary attacks, in this project.

Getting Started

1. Concept

Caesar Cipher is a substitution cipher, in which the message has its letters replaced by other letters or group of letters. The substitution is in-place, which means the order of letters in the original message does not change/move around.

2. What you will need

- Python: programming language, beginner friendly, supports all major operating systems. Get it here: <https://www.python.org/downloads/> (version 2.x recommended)
- Text editor: your personal choice, or one of these:
LiClipse / Eclipse (PyDev): <http://www.liclipse.com/download.html>
Notepad++: <http://notepad-plus-plus.org/> (Windows)
- Tutorial:
Official documentation: <https://docs.python.org/2/tutorial/>
Dive Into Python: <http://www.diveintopython.net/toc/index.html>

3. Limitation

- This lab will only affect the English alphabets with 26 letters from A to Z.
- All other characters will be ignored (not encoded/decoded) and remain as-is in the message.

4. Hello World!

- After done installing Python, open your command-prompt or terminal: at the prompt, type *python*, press Enter.

- Now you are in Python. At the prompt, type `print "Hello World!"` and say hello to your first program.
- Play around and see what you can do with Python, maybe some math? Try the tutorial.
- To exit, use `exit()`

```

Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\sphan>python
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>> x=2
>>> y=3
>>> x
2
>>> y
3
>>> x+y
5
>>> x-y
-1
>>> x*y
6
>>> x*(x-y)
-2
>>> exit()

C:\Users\sphan>_

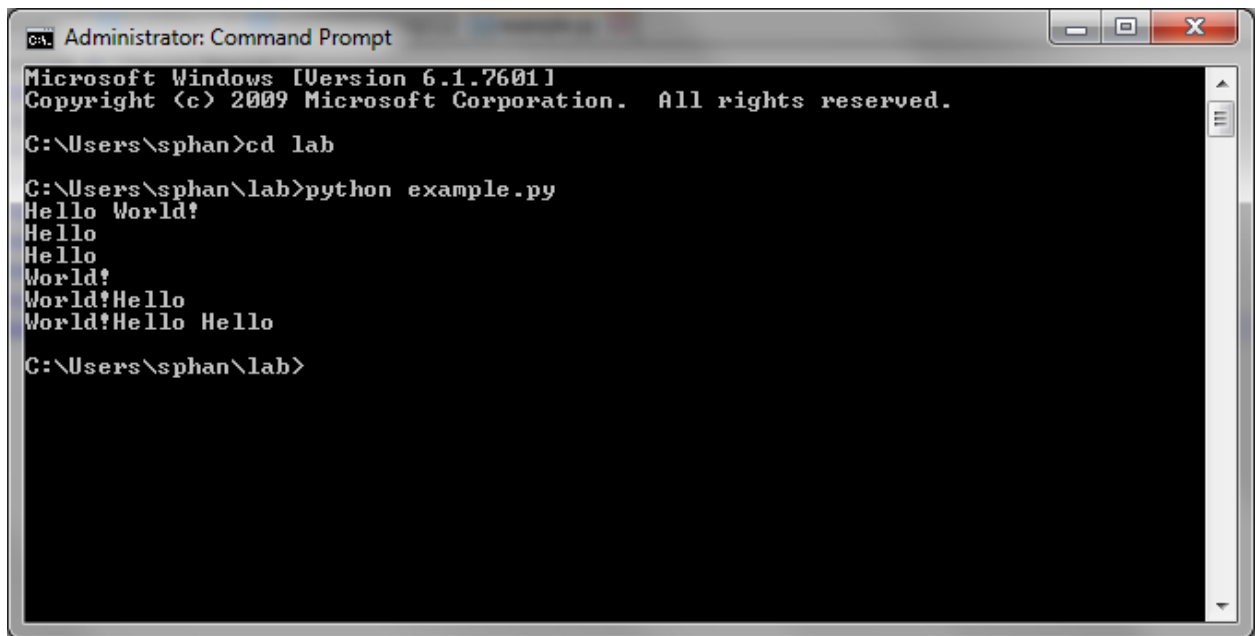
```

- Create a folder name lab, change directory to lab, create a text file with content as follow, and save the file as *example.py*

```

message = "Hello World!"
print message
x = message[0:6]
y = message[6:]
z = message[6:]
print x
print y
print z
print z+y
print z+y+x

```
- Run the file by typing `python example.py` at the terminal prompt, see screenshot.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\sphan>cd lab

C:\Users\sphan\lab>python example.py
Hello World!
Hello
Hello
World!
World!Hello
World!Hello Hello

C:\Users\sphan\lab>
```

Exercises

For the following 3 exercises, download the source files at <https://github.com/sonphanusa/CaesarCipherHacker/>

1. Encoder

The *encoder.py* program, when executed, will prompt users to input the plain text messages, and the key (*k value*) to shift *k* letters that they would like to be encrypted. It will then output the corresponding cipher text.

Open *encoder.py* with your text editor, fill in the blanks for exercise 1, save and run the program by using *python encoder.py*

Example output:

1. Input your message (ignore case & non-alphabet chars): hello world
2. Shift how many letters (key)? k = 2
- 3.
4. RESULTS
5. Plaintext: hello world
6. Ciphertext: jgnnq yqtnf

2. Decoder

The *decoder.py* program, when executed, will prompt users to input the cipher messages, and the key (*k value*). It will then output the corresponding plain text.

Open *decoder.py* with your text editor, fill in the blanks for exercise 2, save and run the program by using *python decoder.py*

Example output:

```
1. Input your cipher (ignore case & non-alphabet chars): jgnnq yqtnf
2. Shifted how many letters (key)? k = 2
3.
4. RESULTS
5. Ciphertext: jgnnq yqtnf
6. Plaintext: hello world
```

3. Hacker

The *hacker.py* program, when executed, will only prompt users to input the cipher messages. It will then output the corresponding plain text for each of the k values (total of 26) in the descending order based on the probability/percentage of the correct guess.

Example output:

```
1. Input your cipher: jgnnq yqtnf
2.
3. RESULTS
4. k = 2 -> message: hello world (100.0%)
5. k = 13 -> message: wtaad ldgas (87.5%)
6. k = 15 -> message: uryyb jbeyq (68.8%)
7. k = 6 -> message: dahhk sknhz (62.5%)
8. k = 9 -> message: axeeh phkew (62.5%)
9. k = 12 -> message: xubbe mehbt (62.5%)
10. k = 19 -> message: qnuux fxaum (62.5%)
11. k = 24 -> message: lipps asvph (62.5%)
12. k = 25 -> message: khoor zruog (62.5%)
13. k = 1 -> message: ifmmp xpsme (56.2%)
14. k = 4 -> message: fcjjm umpjb (56.2%)
15. k = 8 -> message: byffi qilfx (56.2%)
16. k = 5 -> message: ebiil tloia (50.0%)
17. k = 10 -> message: zwddg ogjdv (50.0%)
18. k = 11 -> message: yvccf nficu (50.0%)
19. k = 16 -> message: tqxxa iadxp (50.0%)
20. k = 17 -> message: spwvz hzcwo (50.0%)
21. k = 18 -> message: rovvv gybvn (50.0%)
22. k = 0 -> message: jgnnq yqtnf (43.8%)
23. k = 3 -> message: gdkkn vnqkc (43.8%)
24. k = 7 -> message: czggj rjmgv (43.8%)
25. k = 14 -> message: vszcc kcfzr (43.8%)
26. k = 20 -> message: pmttw ewztl (43.8%)
27. k = 21 -> message: olssv dvysk (43.8%)
28. k = 22 -> message: nkrru cuxrj (43.8%)
29. k = 23 -> message: mjqqt btwqi (43.8%)
```

Open *hacker.py* with you text editor, fill in the blanks for exercise 3, save and run the program by using *python hacker.py*

* Hints:

Plaintext: **abc**defghijklmnopqrstuvwxyz
k=3 → Ciphertext: **def**ghijklmnopqrstuvwxyz**abc**

Look up python functions:

- `raw_input()` for storing user input (what user typed in) as string value
- `int()` for type-casting *string* value to an *integer* (number)

Comments in the code are there for a reason.

Deliverables

1. Your completed source files for 3 exercises.
2. Project report describing your implementation and your sample outputs.
3. Demonstration (during week #15 of the semester, or by appointment).