# Ciphers

## Breaking the Shift Cipher

## Based off Chapter 7 of *Hacking Secret Ciphers with Python*

Lecture notes of Alexander Wood
CSCI 360 Cryptography and Cryptanalysis
awood@jjay.cuny.edu

John Jay College of Criminal Justice

These slides are based off of Chapter 7 of *Hacking Secret Ciphers With Python* by Al Sweigart, available here: `http://inventwithpython.com/hacking/chapter7.html`

# The Brute Force Technique

The brute force technique is what we call literally just trying every key in the cryptosystem's keyspace until we find the right one.

# The Brute Force Technique

*What is the keyspace of the shift cipher?*

# The Brute Force Technique

*What is the keyspace of the shift cipher?*

The shift cipher has a keyspace of only 26 (counting the ‘zero shift’), meaning that we only have to try 25 to hack the shift cipher!

# Kerckoff's Principle

Recall *Kerckhoff's Principle*, which states that even an adversary who has all of the information about how your cryptosystem operates, but does not have the private keys, will not be able to break your cryptosystem.

Does the shift cipher satisfy this principle?

# Kerckhoff's Principle

The shift cipher does *not* satisfy Kerckhoff's principle because it would be easy enough to decipher, even by hand.

# Implementing the Brute Force Attack

Implementing the brute force attack is remarkably straightforward.

1. Try the first key.
2. If this does not work, move on to the next key.
3. Repeat until you have found the correct key.

We would like to automate this task so that we don't have to do all of this by hand.

# Sample Run

```
Key #0: GUVF VF ZL FRPERG ZRFFNTR.
Key #1: FTUE UE YK EQODQF YQEEMSQ.
Key #2: ESTD TD XJ DPNCPE XPDDLRP.
Key #3: DRSC SC WI COMBOD WOCCKQO.
Key #4: CQRB RB VH BNLANC VNBBJPN.
Key #5: BPQA QA UG AMKZMB UMAAIOM.
Key #6: AOPZ PZ TF ZLJYLA TLZZHNL.
Key #7: ZNOY OY SE YKIXKZ SKYYGMK.
Key #8: YMNX NX RD XJHWJY RJXXFLJ.
Key #9: XLMW MW QC WIGVIX QIWWEKI.
Key #10: WKLV LV PB VHFUHW PHVVDJH.
Key #11: VJKU KU OA UGETGV OGUUCIG.
Key #12: UIJT JT NZ TFDSFU NFTTBHF.
Key #13: THIS IS MY SECRET MESSAGE.
Key #14: SGHR HR LX RDBQDS LDRRZFD.
Key #15: RFGQ GQ KW QCAPCR KCQQYEC.
Key #16: QEFP FP JV PBZOBQ JBPPXDB.
Key #17: PDEO EO IU OAYNAP IAOOWCA.
Key #18: OCDN DN HT NZXMZO HZNNVBZ.
Key #19: NBCM CM GS MYWLYN GYMMUAY.
Key #20: MABL BL FR LXVKXM FXLLTZX.
Key #21: LZAK AK EQ KWUJWL EWKKSYW.
Key #22: KYZJ ZJ DP JVTIVK DVJJRXV.
Key #23: JXYI YI CO IUSHUJ CUIIQWU.
Key #24: IWXH XH BN HTRGTI BTHHPVT.
Key #25: HVWG WG AM GSQFSH ASGGOUS.
```

# Python Review: for loops

We've seen a `for` loop which iterates over each letter in a string. We also can create a `for` loop which iterates over the return values from a call to `range()`. The `range()` call takes an integer argument and returns a sequence of numbers.

```
>>> for number in range(5):
print(number)



0
1
2
3
4
```

# Python Review: for loops

We can use the `range()` function to specify that we would like to repeat an action a certain number of times.

```
>>> for i in range(4):
...     print('Hello')
...
Hello
Hello
Hello
Hello
>>>
```

# The range type

If you want to get specific, according to the Python 3 documentation, The `range` type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in `for` loops.

# The range type

The `range` constructor can take more than one argument.

```
range([start], bound, [step])
```

# The range type: Example

```
>>> for i in range(3,7):
print(i)


3
4
5
6
```

# The range type: Example

```
>>> for i in range(1, 12, 2):
print(i)


1
3
5
7
9
11
```

# Coding Exercise

Write code which breaks ciphertexts encrypted using the shift cipher via the brute force method.

# Test The Code

Use your code to break the following ciphertexts.

1. R UXEN VH TRCCH
2. FR DBMMR EHOXL FX
3. CXPNCQNA FN'AN BX QJYYH
4. OBR OZKOMG QOFSTFSS
5. PDKQCD IU DAWZ DWO OQOLEYEKJO
6. FTMF U WQQB GZPQD YK TMF
7. AR ITMF YUSTF TMBBQZ
8. DA D NCMVIF OJ OCZ NDUZ JA V MVO
9. ZFBI. J'N QSFUUZ TVSF NZ DBU XPVME FBU NF

For further reading see Chapter 7 of *Hacking Secret Ciphers With Python* by Al Sweigart, available here:

`http://inventwithpython.com/hacking/chapter7.html`