

Strings in Python: Cipher Applications

CS 8: Introduction to Computer Science, Winter 2018
Lecture #9

Ziad Matni
Dept. of Computer Science, UCSB

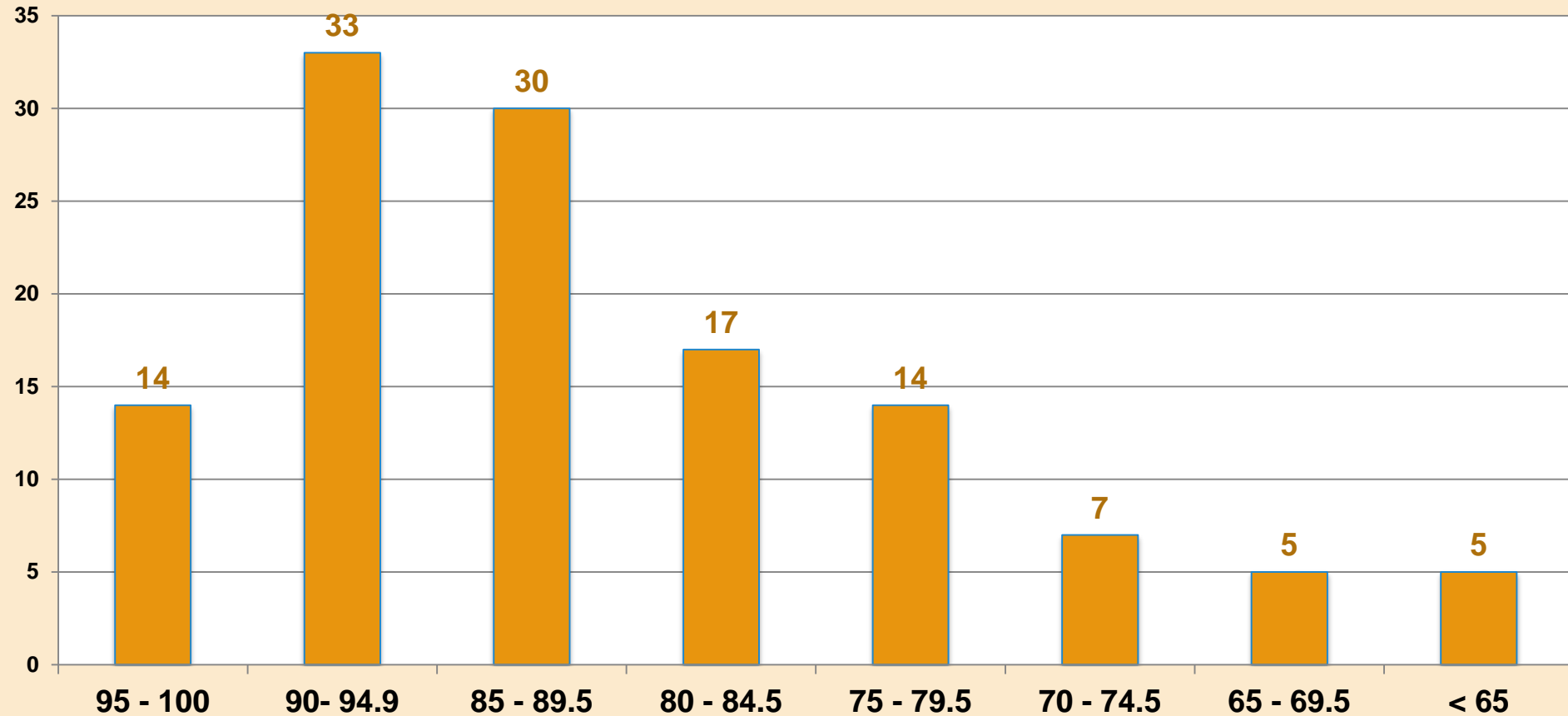
Administrative

- Homework #4 is due today
- Homework #5 is out and **DUE on MONDAY (2/26)**
- Project #1 is due on Friday (2/23)
- Lab #3 is due by midnight today on submit.cs
- Midterm grades are up
 - To review your exam results, visit your lab TA's office hours

Grade Distribution for Midterm Exam

CS 8, Wi 18 (Matni)

Average = 85%, Median = 87%



Lecture Outline

Finishing up Ch. 3 in the book

- String manipulations using for-loops
 - Cipher applications and examples
- Algorithmic approaches to programming

Use Of for loops

- Using for loops in Python is flexible

- We can use them on lists:

```
for n in (4, 3, 9, -22): ...etc...
```

```
for s in ("Bob", "Michelle", "Joe", "Amal"): ...etc...
```

- We can use them on a “range” of numbers:

```
for n in range(0, 14, 2): ...etc...
```

Use Of **for loop** To Go Thru A String

- We can also use them to *go through* a string one letter (i.e. character) at a time:

```
myString = "Hello!"  
for ch in myString:  
    print (ch)
```

Will give me:

H
e
l
l
o
!

Variations on the `print()` Function

- By default, `print()` issues a 'newline' character at the end
 - That's why successive `print()`s are done on separate lines
- You can optionally do this differently with the `end=` operator inside of `print()`.

EXAMPLES:

```
for n in range(0, 3):  
    print(n)
```

→

0
1
2

vs.

```
for n in range(0,3):  
    print(n, end=" "):
```

→

0 1 2

vs.

```
for n in range(0,3):  
    print(n, end="*"):
```

→

0*1*2*

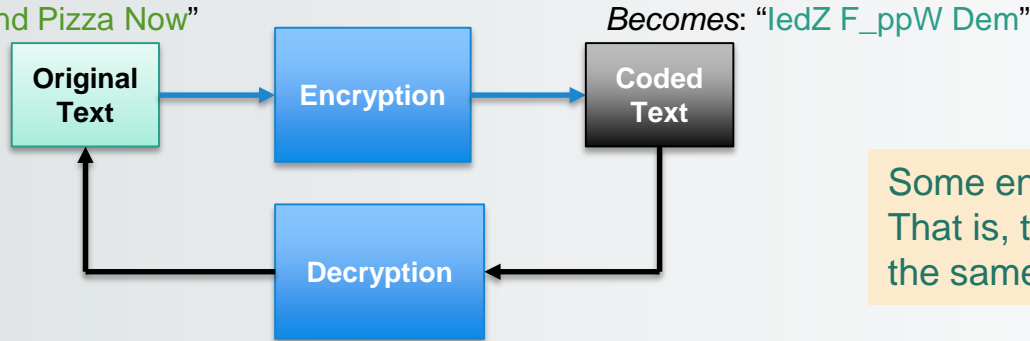
Let's Apply This Stuff to Ciphers!

Ciphers!



- String manipulation lends itself well to problems of coding/decoding private or secret messages
- To do so, you need encryption and decryption algorithms

Example: “Send Pizza Now”



Some encryptions are symmetric:
That is, the **encryption algorithm** is
the same as the **decryption algorithm**

Examples

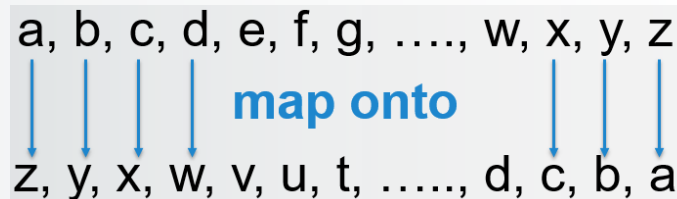
- Make every letter the letter after it
 - Letter ‘a’ becomes ‘b’, ‘b’ becomes ‘c’, etc...
 - So that “**hello**” becomes “**ifmmp**” (Encryption)
 - How would you decrypt this? Is this a symmetric encryption scheme?
- Mirrored Alphabet (or “the first shall be the last”)
 - The letters a, b, c, d, ... w, x, y, z map onto
z, y, x, w, ... d, c, b, a
 - So that “**bye**” becomes “**ybv**”
 - How would you decrypt this? Is this a symmetric encryption scheme?

Mirrored Alphabet Cipher

- Let's examine the thinking behind this:

a, b, c, d, e, f, g, , w, x, y, z
↓ ↓ ↓ ↓ map onto ↓ ↓ ↓
z, y, x, w, v, u, t, , d, c, b, a

Our Algorithm



1. Given a string (message) with N number of letters
2. Go thru every letter in order to examine it (*how?*)
3. Apply “mapping formula” to each letter
(*don't know what that “formula” is yet, but that's ok...*)
4. Once formula is applied,
“gather up the new letters” into a NEW string (*how?*)
5. Return that NEW string as the encoded message

Encryption for Mirrored Alphabet

- Just reverse order of characters in alphabet

```
def encrypt(message):    # message is a string type
    result = ''          # start with an empty result
    for c in message:    # go thru every letter in message
        # let's apply the "mirror" formula:
        nc = ord(c)
        nr = ord('a') + ord('z') - nc
        # then accumulate the encoded chars, one at a time
        result = result + chr(nr)
    # Now we're done with the for-loop, so let's return when we got:
    return result
```

Example run:

```
>>> encrypt("abcdefghijklmnopqrstuvwxyz")
'zyxwvutsrqponmlkjihgfedcba'
```

A Simple Substitution Cipher

- Note that the same function `decrypts` as well!

```
>>> encrypt('zyxwvutsrqponmlkjihgfedcba')  
'abcdefghijklmnopqrstuvwxyz'
```

- What happens if I try `encrypt("CAT")`?
 - Why?
 - There's a question on Homework #5 on this...

Scrambling Even & Odd Positions

From textbook, 3.4, pg 94

- Extract even and odd parts (i.e. positions of letters) of the message and combine them

- Example:

Original: "I just wanna fly"
Even: "ljs an l"
Odd: " utwnafy"
Combined (odd+even): " utwnafyljs an l"

0 1 2

15

Scrambling Even & Odd Positions

```
def scramble2Encrypt(plainText):          # plainText is a string type
    # Initialize these 3 variables
    evenChars = ""
    oddChars = ""
    charCount = 0          # charCount is supposed to tell me char. position
    # Go through every character in the string
    for ch in plainText:
        if charCount % 2 == 0:
            evenChars = evenChars + ch    # accumulate the even chars in one string
        else:
            oddChars = oddChars + ch      # accumulate the odd chars in another
        charCount = charCount + 1        # (still in the loop) character count goes up by 1
    # Done with the for-loop!
    cipherText = oddChars + evenChars    # combine the odd+even char strings into one
    return cipherText
```


Unscrambling

- The same encryption function **won't** work in reverse.
 - That is, it's not a symmetrical encryption
- We need a separate decryption function.
 - First, cut the encrypted string in half
 - 1st half is the original's odd chars, 2nd half is the evens
 - Now I have 2 sub-strings and I can re-construct the original
 - Take one from the evens, then one from the odds, and repeat until done
- See Section 3.4.2 in textbook for full decryption function in Listing 3.3 (page 98)

Asking for Input from the User

- We know how to output to the display: Good ole **print()** function!
- What if we want to get an input from the keyboard? We've used another function for that: **input()**
- *Example:*

```
numb = input("Gimme a number! ")
```

```
name = input("Gimme a name!!!! ")
```

NOTE: You don't *have to* specify what kind of input you're getting, but you can, if you want to!

(The default, if you don't specify, is *string* type)

```
num1 = int(input("Gimme a whole number! "))
```

```
num2 = float(input("Gimme a number with a decimal point! "))
```

```
num3 = complex(input("Gimme a complex number! "))
```

```
myStr = str(input("Gimme a string! ")) # kind of unnecessary?
```

YOUR TO-DOs

- ❑ Do **Homework5** (due **Monday 2/26**)
- ❑ Turn in **Lab3** **tonight** on submit.cs
- ❑ Turn in **Project1** on **Friday** on submit.cs

- ❑ 2 words: Transcendental Meditation

</LECTURE>