Strings in Python: Cipher Applications

CS 8: Introduction to Computer Science
Lecture #7

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- **Midterm #1 grades will be available soon!**

- Turn in Homework #3 today
- Homework #4 is assigned and due next Thursday
- Lab #3 is due on Friday

- NEW!! Project #1 is out! Due on **May 12th**

- **Don't forget your TAs' and Instructor's office hours!! ☺**

Use Of **for loops**

- Using for loops in Python is flexible
- We can use them on lists:

  for n in ('joe', 'bob', 'sue'): ...*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="*"):
```

→

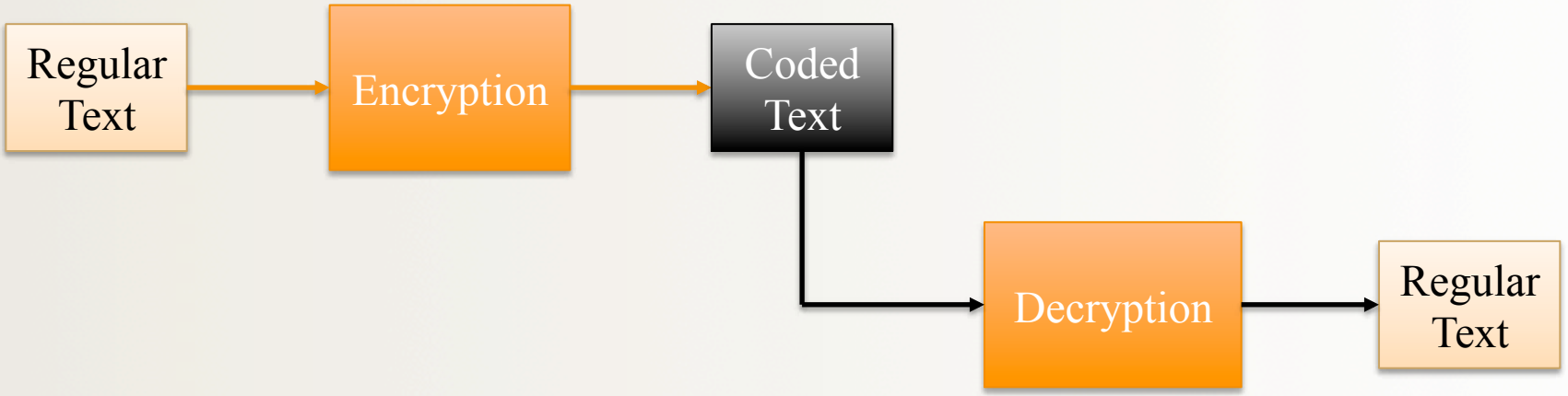
```
0*1*2*
```

Let's Apply This Stuff to Ciphers!

Ciphers!

- String manipulation lends itself well to problems of coding/decoding private/secret messages

- You need encryption and decryption algorithms



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?

- 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?

Encryption for Mirrored Alphabet

- Just reverse order of characters in alphabet

```
def encrypt(message):
    result = ''                                  # start with empty result
    for c in message:
        nc = ord(c)
        nr = ord('a') + ord('z') - nc   # mirror formula
        result = result + chr(nr)  # accumulate one char at a time
    return result
```

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

A Simple Substitution Cipher

- Note that the same function decrypts as well!

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

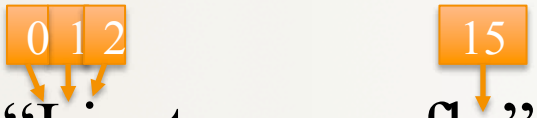
- Let's try it out!

- What happens if I `encrypt("CAT")`?
  - Why?
  - How to fix?

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: “Ijs an l”

Odd: “ utwnafy”

Combined (odd+even): “ utwnafyIjs an l”

Scrambling Even & Odd Positions

```
def scramble2Encrypt(plainText):
    evenChars = ""                          # Initialize some variables
    oddChars = ""
    charCount = 0                           # this will tell me char. position
    for ch in plainText:                    # Go through every character
        if charCount % 2 == 0:
            evenChars = evenChars + ch      # accumulate the even chars
        else:
            oddChars = oddChars + ch        # accumulate the odd chars
        charCount = charCount + 1           # character count goes up by 1
    cipherText = oddChars + evenChars       # combine the odd+even chars
    return cipherText
```

Unscrambling

- The same encryption function won't work in reverse.
- We need a separate decryption function.
  - First cut the encrypted string in half
  - 1st half is the odds, 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
- 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'll need another function: **input( )**
- *Use:*

```
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!

```
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! ")
```

Example of Using Input

```
def sayHello( ):
    name = input("Hi! What's your name? ")
    print("Hello", name, "and welcome to CS8!")
```

YOUR TO-DOs

- Finish reading **Chapter 3 (up to 3.5)**
- Also read **Chapter 4 (4.1 - 4.3)**

- Finish **Homework4** (due **Thursday 5/4**)
- Finish **Lab3** (due **Friday 4/28**)
- Start working on **Project1** (due **Friday 5/12**)

- Laugh more, frown less

</LECTURE>