Lecture 12

Topics: *Chapter 5. Computing with strings*

- 5.4 String representation and message encoding
- 5.5 String methods
- 5.6 Lists have methods too
- 5.7 From encoding to encryption

5.4 String representation and message encoding

How are strings stored in a computer?

5.4 String representation and message encoding

How are strings stored in a computer?

Each character is translated into a number, and the entire string is stored as a sequence of (binary) numbers in computer memory.

5.4 String representation and message encoding

Computer systems today use industry standard encoding of characters that is understandable by all kinds of computers.

5.4 String representation and message encoding

Computer systems today use industry standard encoding of characters that is understandable by all kinds of computers.

One of these standards is ASCII (American Standard Code for Information Interchange).

ASCII uses numbers through 0 to 127 to represent characters typically found on an (American) computer keyboard, as well as certain special values known as *control codes*.

5.4 String representation and message encoding

Computer systems today use industry standard encoding of characters that is understandable by all kinds of computers.

One of these standards is ASCII
(American Standard Code for Information Interchange).

ASCII uses numbers through 0 to 127 to represent characters typically found on an (American) computer keyboard, as well as certain special values known as *control codes*.

For example,

A-Z are represented by values 65-90
a-z are represented by values 97-122
0-9 are represented by values 48-57

5.4 String representation and message encoding

Computer systems today use industry standard encoding of characters that is understandable by all kinds of computers.

One of these standards is ASCII (American Standard Code for Information Interchange).

In Python:
To get an ASCII code of a character use command

```
ord(<character>)
```

'`ord`' stands for 'ordinal'

Command `chr` goes the other direction.

5.4 String representation and message encoding

ASCII
American Standard Code for Information Interchange

In Python interpreter's interactive window, write:

```
>>> ord('f')
102

>>> ord('5')
53

>>> chr(80)
'P'

>>> chr(90)
'Z'

>>> for i in range(97,123):
        chr(i)
```

5.4 String representation and message encoding

ASCII
American Standard Code for Information Interchange

Example: Let's write a program that will encode our message by the ASCII code.

Here is the *design/algorithm*:

Take the message to encode from the user
For each character in the message
    Print the ASCII code of the character (same line, separated by space)

see program encoding.py

5.4 String representation and message encoding

ASCII

American Standard Code for Information Interchange

Should we write a decoding program?

5.4 String representation and message encoding

ASCII

American Standard Code for Information Interchange

Should we write a decoding program?

Yes, but let's take a look at functions that work on lists….. first

5.5 String methods

On page 148 **Table 5.2** we are given a list of string methods:

Function	Meaning
`s.capitalize()`	Copy of s with only the first character capitalized.
`s.center(width)`	Copy of s centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of sub in s.
`s.find(sub)`	Find the first position where sub occurs in s.
`s.join(list)`	Concatenate list into a string, using s as separator.
`s.ljust(width)`	Like center, but s is left-justified.
`s.lower()`	Copy of s in all lowercase characters.
`s.lstrip()`	Copy of s with leading white space removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of oldsub in s with newsub.
`s.rfind(sub)`	Like find, but returns the rightmost position.
`s.rjust(width)`	Like center, but s is right-justified.
`s.rstrip()`	Copy of s with trailing white space removed.
`s.split()`	Split s into a list of substrings (see text).
`s.title()`	Copy of s with first character of each word capitalized.
`s.upper()`	Copy of s with all characters converted to uppercase.

Table 5.2: Some string methods

5.5 String methods

On page 148 **Table 5.2** we are given a list of string methods:

Function	Meaning
s.capitalize()	Copy of s with only the first character capitalized.
s.center(width)	Copy of s centered in a field of given width.
s.count(sub)	Count the number of occurrences of sub in s.
s.find(sub)	Find the first position where sub occurs in s.
s.join(list)	Concatenate list into a string, using s as separator.
s.ljust(width)	Like center, but s is left-justified.
s.lower()	Copy of s in all lowercase characters.
s.lstrip()	Copy of s with leading white space removed.
s.replace(oldsub,newsub)	Replace all occurrences of oldsub in s with newsub.
s.rfind(sub)	Like find, but returns the rightmost position.
s.rjust(width)	Like center, but s is right-justified.
s.rstrip()	Copy of s with trailing white space removed.
s.split()	Split s into a list of substrings (see text).
s.title()	Copy of s with first character of each word capitalized.
s.upper()	Copy of s with all characters converted to uppercase.

Table 5.2: Some string methods

5.5 String methods

Example: a decoder from ASCII

Design / Algorithm:

Input: the sequence of numbers separated by space, as string

message = ""
for each number in the sequence:
    convert it to the corresponding Unicode character
    add the character to the end of the message

Output: decoded string

see the program decoder.py

5.6 Lists have methods, too

Here are few more methods for Lists:

r.append(x)	appends the element to the end of the list
r.count(x)	returns the number of x's occurrences
r.index(x)	returns smallest index *k* such that r[k] == x
r.insert(i, x)	inserts element x into i^{th} place
r.pop([i])	returns the i^{th} element of the list and removes it from the list
r.remove(x)	removes the first occurrence of element x in the list
r.reverse()	reverses the elements/items of *r* in place
r.sort()	sorts the elements/items of *r* in place (numerical, or alphabetical order, or …)

Note that the majority of these methods are modifying the original list

5.6 Lists have methods, too

Example: Let's write a program that given the date in the format mm/dd/yyyy will be displaying it in the form month's name, year

Example of input: 01/20/1979
The output produced: January 20, 1979

Design / algorithm:

get the date (format mm/dd/yyyy, as string)
split the date into three strings (*month*, *day*, *year*)
find the month in the list of months' names (by index)
output the date in new format

see the program date.py

5.6 Lists have methods, too

Advanced Example: given a phrase (no punctuation symbols, only letters and spaces) sort the letters alphabetically (with capital letters before the lowercase), keeping the spaces in place

Example of input: Mary likes ice cream
The output: Maac ceeei ikl mrrsy

Design / algorithm:
get the input (string *inp*)
split *inp* using space as separator (list *inpL*)
split *inp* into list of letters, ignoring spaces, then sort
create new list and fill it correspondingly with inpL (*result*)
join the list using '' as separator (*resultingString*)

see the program re_ordering.py

5.7 From encoding to encryption

Encoding using industry-standard mappings of characters into numbers is mainly used for storage.

Encryption is used for keeping information secret. It uses the encoding which does not follow simple industry-standard mappings of characters into numbers.

5.7 From encoding to encryption

Encoding using industry-standard mappings of characters into numbers is mainly used for storage.

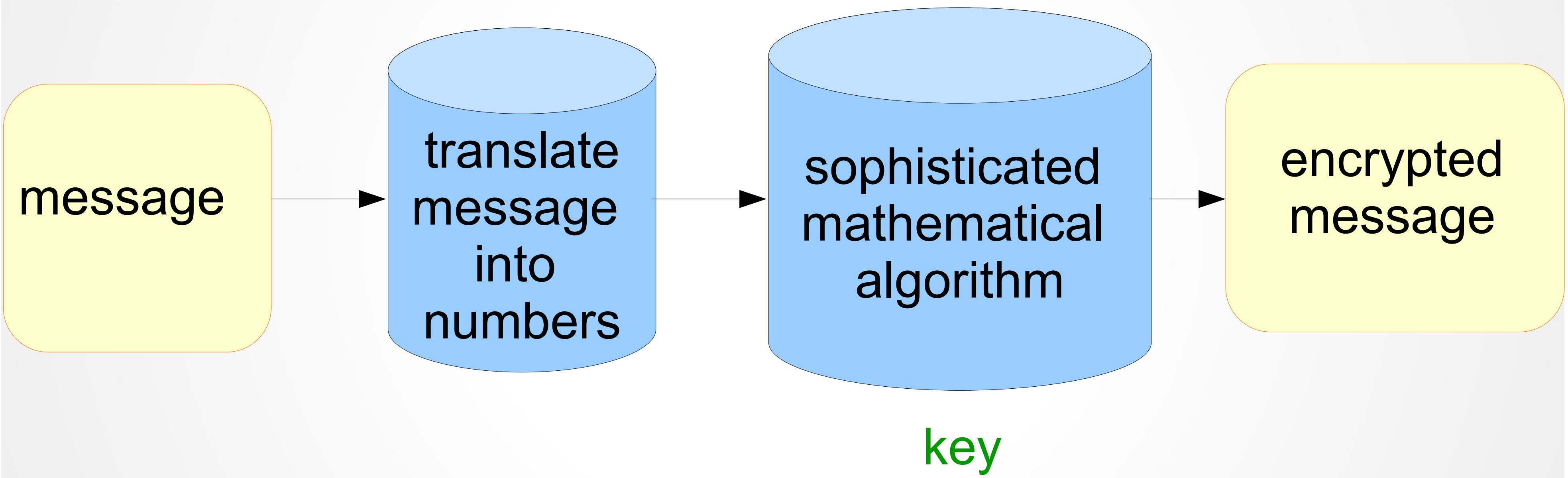
Encryption is used for keeping information secret. It uses the encoding which does not follow simple industry-standard mappings of characters into numbers.

The study of encryption methods is a sub-field of mathematics and computer science, called *cryptography*.

The original message is called *plaintext*, the resulting encrypted message is called *cyphertext*.

5.7 From encoding to encryption

Modern approaches to encryption:



5.7 From encoding to encryption

Caesar cipher

See programming exercise 7 on page 172

The cipher is based on the idea of shifting each letter in the *plaintext message* a fixed number (called the *key*) of positions in the alphabet.

Example: assume key = 2
Then word Sourpuss → Uqwtrwuu
to decode: shift 2 letters left

5.7 From encoding to encryption

Caesar cipher

See programming exercise 7 on page 172

The cipher is based on the idea of shifting each letter in the *plaintext message* a fixed number (called the *key*) of positions in the alphabet.

Example: assume key = 2
Then word Sourpuss → Uqwtrwuu
to decode: shift 2 letters left

Use ASCII codes `chr(ord(ch) + key)`

5.7 From encoding to encryption

Caesar cipher

Design / algorithm:

get the *key* value from the user
get the *message* to translate from the user

```
for char in message:
    newChar = chr(ord(char)+key)
    append the code to the resulting encrypted message
```

display the encrypted message

5.7 From encoding to encryption

Caesar cipher

We designed and implemented the encryption algorithm.

If you want, design and implement the decryption algorithm!