

§7.0-7.6: Applications: Caesar cipher

15 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

- *Quiz04 today (ch5-6)*

Review of §6.5-6.10

- **Library** modules:
 - Public **interface** (header) vs.
 - Private **implementation**
 - Car: **owner's** manual vs. **shop** manual
- Defining an **abstract data type**
- **Accessor** (set/get) functions
- Using (**import**) our library

Quiz 04 (ch5-6): 10 minutes, 20 points

- Contrast **aliasing** a list with **copying** a list.
 - Write **Python** code to demonstrate the difference
- Contrast a **header (DEF)** file with an **implementation (IMP)** file.
- Why should we use **accessor** (set/get) functions in an ADT?
- Write a Python function **matrix(n_rows, n_cols)** to create and return a **2D list** with the given number of rows/cols.

Quiz 04 (ch5-6): answers #1

- Contrast **aliasing** a list with **copying** a list.
 - **Aliasing**: another name for the same list; modifying elements of the alias also modifies elements of the original list
 - **Copying**: a separate data structure with the same contents

```
origList = [ 1, 2, 3 ]
```

```
aliasList = origList
```

```
copyList = origList[:]
```

```
aliasList[0] = 5
```

```
copyList[1] = 7
```

```
origList[0]      # is 5
```

Quiz 04 (ch5-6): answers #2-3

- Contrast a **header** (**DEF**) file with an **implementation** (**IMP**) file.
 - Header: public interface, doesn't define bodies of functions
 - Implementation: contains bodies of the functions
- Why should we use **accessor** (set/get) functions in an ADT?
 - Hide implementation details from user
 - Maintain the “illusion” of the ADT
 - Ease future upgrades of internal implementation

Quiz 04 (ch5-6): answers #4

- Write a Python function `matrix(n_rows, n_cols)` to create and return a **2D list** with the given number of rows/cols.

```
def create_matrix(n_rows, n_cols):  
    matrix = range(n_rows)  
    for row in range(n_rows):  
        matrix[row] = range(n_cols)  
    return matrix
```

What's on for today (§7.0-7.6)

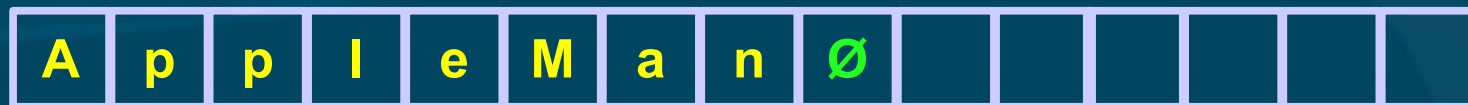
- **Strings**: manipulating text
 - Null-terminated strings
- Application: **cryptology** (substitution cipher)
 - Creating a **library** for cryptology
 - **Public** interface
 - Library-internal **helper** functions

Null-termination in strings

- In Python, strings are a basic **type** (immutable seq)
- But in M2/C, **strings** are fixed-len arrays of CHAR:

```
VAR myName : ARRAY [0..14] OF CHAR;
```
- But the array is not always completely **filled**:

```
myName := "AppleMan";
```
- How to know where the string **ends**?
- Strings are **null-terminated**:
 - The null character CHR(0) is added to the end
 - Anything past the termination char is ignored



Cryptography example

- Cæsar substitution cipher:
 - **Key:** e.g., QAZXSWEDCVFRTGBNHYUJMKIOLP
 - **Cleartext:** input text to encrypt
 - **Ciphertext:** output encrypted text
 - Encoding: replace each **letter** in source with corresponding letter from code key
 - Decoding: same, using the decode key
- **ROT13** was an example of a substitution cipher
 - Key: NOPQRSTUVWXYZABCDEFGHIJKLM

Write a Substitution cipher library

- What public interface do we want for the library?

```
def encode (src, key):
```

```
    """Encode the source string using the given codestring.  
    Returns the encoded string.  
    pre: src must be a string;  
    key must be a permutation of the 26 letters."""
```

```
def decode (src, key):
```

```
    """Decode the source string using the given codestring.  
    Returns the decoded string.  
    pre: src must be a string;  
    key must be a permutation of the 26 letters."""
```

Internal helper functions

- In the implementation it is handy to have some helper functions for **internal** use:

```
def isalpha (ch):
```

```
    """Return true if ch is a letter."""
```

```
def alpha_pos (ch):
```

```
    """Return index of a letter in the range 0 .. 25"""
```

```
def decode_key (enckey):
```

```
    """Create a decode key from an encoding key"""
```

- How to implement these?
 - isalpha() is built-in: ch.isalpha()

Implementing Substitution library

- Main function to **encode** strings:

```
def encode(src, key):  
    """Encode the source string using the given codestring.  
    Returns the encoded string.  
    pre: src must be a string;  
    key must be a permutation of the 26 letters.  
    """  
    dst = ""  
    for ch in src:  
        if ch.isalpha():  
            dst += key[alpha_pos(ch)]  
        else:  
            dst += ch  
    return dst
```

Implementing decode()

- Decoding is just encoding using a reverse key:

```
def decode (src, key):
```

```
    """Decode the source string using the given codestring.  
    Returns the decoded string.
```

```
    pre: src must be a string;
```

```
    key must be a permutation of the 26 letters.
```

```
    """
```

```
    return encode(src, decode_key(key))
```

- Library: <http://twu.seanho.com/python/substitution.py>
- Testbed: <http://twu.seanho.com/python/caesartest.py>

TODO items

- Lab05 due Wed: ch6 # 33 / 35
- HW05 due Fri:
 - ch6 # 25 (hint does not apply in Python)
 - ch6 # 28 (write a Python program to do this)
- 140 Final / 141 midterm next week
 - Wed 24Oct 14:35-15:50 (part 1)
 - Thu 25Oct 13:10-14:15 (part 2)