PROGRAMMING

Lecture 16

Hanbat National University
Dept. of Computer Engineering
Changbeom Choi

OUTLINE

Comparison functions

Listing out palindromes

Vegetable and fruit store

COMPARISON FUNCTIONS

PROBLEM 1: SORTING NAMES

Consider a list of names, L:

N = [“Chris Terman”, “Tom Brady”, “Eric Grimson”,
    “Joseph Shin”, “Changbum Choi”]

In each of the names in the list N, the **first name appears before the last name**. We want to sort the list in the **alphabetical order** of the **last name**. The **first names** are to be used only as the **tie-breaker** if the last names are the same. Modify the bubble sort so that we also use it to sort the list N.

Using a function as a parameter

```
def cmp_numbers(x1, x2):
    return x1 > x2
def bubble_sort(L, func):
    for i in range ( len(L) - 1):
        for j in range(len(L) - 1 - i):
            i1 = (len(L)-1) - (j + 1)
            i2 = (len(L)-1) -  j
            if func(L[i1], L[i2]):
                L[i1], L[i2] = L[i2], L[i1]
    return L
L = [4, 3, 9, 7, 8]
bubnble_sort(L, cmp_numbers)
```

```
if L[i1] > L[i2}
```

How to write a function for comparing names:

1. Split each name into two parts: the first name and the last name. You may use the function, **split** in the standard Python module, **string** to split a name**:**

```
import string
name = string.split(name, “ ”)
```

where name[0] and name[1[are the first and last names, respectively.

2. Compare the last names. If they are tied, then compare the first names

3. Return the comparison result.

```
def cmp_names(name1, name2):
    Fill in this block.





……………………………

……………………………
bubble_sort(N, cmp_names)
```

LISTING OUT PALINDROMES

PROBLEM 2: PLAYING WITH PALINDROMES

In problem 2-2 in week 7, you implemented an **iterative version** of a program for finding all palindromes in the file **words.txt**. In this problem, **change** that program to its **recurve version** to do the same task and print out a summary report of palindromes in the increasing order of their lengths as shown in the next slide. Your task is to complete the program for the summary report by filling in the missing part. You should type in the code given in the lecture note before filling in the missing part.

(Continued)

Length	Frequency	Palindromes
2	oo	xx, xx, …., xx
3	oo	xxx, xxx, …………… …., xxx
		…………………………………………
4	oo	xxxx, xxxx, ………………, xxxx

…………………………………………………………………………

* Length: at most **two digits**
** Frequency: at most **two digits**
*** Palindromes occupy **one or more lines** for each length.

Sample output

```
Length Frequency    Palindroms

  2       1         aa
  3       45        aba aga aha ala ama ana ava awa bib
                    bob bub dad did dud eke eme ere eve
                    ewe eye gag gig hah huh mem mim mom
                    mum nun oho pap pep pip pop pup sis
                    sos tat tit tot tut vav waw wow yay
  4       11        anna boob deed keek kook noon otto
                    peep poop sees toot
  5       21        alula civic deked deled dewed kaiak
                    kayak level madam minim radar refer
                    rotor sagas semes seres sexes shahs
                    solos stets tenet
  6       6         denned hallah marram redder selles
                    terret
  7       7         deified halalah reifier repaper reviver
                     rotator sememes
```

Pseudo code

1. Select all palindromes from words.txt to create a list of palindromes.
2. Sort the list in the increasing order of word lengths.
3. **Print out a summary report as shown in the previous slide.**

```
def main():
    pal_list = create_list()
    pal_list.sort()
    print_summary(pal_list)

main()
```

```
def palin(word):
    if len(word) <= 1:
        return True
    else:
        return (word[0] == word[-1]) and palin(word[1:-1])

def create_list():
    f = open("words.txt", "r")
    words = []
    for word in f:
        word = word.strip()
        if palin(word):
            words.append((len(word),word))
    f.close()
    return words
```

Step 3: Print out a summary report.

input: A sorted list of palindromes

Output : A summary report (see the previous slide)

For each length of words:

- Create the list of palindromes
- Count their frequency
- Print length, frequency, and palindromes

```
def print_summary(palind):
    length = 0
    print "  Length Frequency     Palindroms"
    flag = True
    for lw, word in palind:
        if length != lw:
            if flag == True:          Why this?
                flag = False
            else:
                display(lst, length)      Print the palindromes of length
            lst = [word]
            length = len(word)
        else:
            lst.append(word)
    display(lst, length)       Why this?
```

```
def new_line(count):
    return count >= 35
```

```
def display(lst, length):
```

Fill in this block to complete the function.

This function has two parameters:

lst : a list of palindromes of length, **length**

length: the length of palindromes in **lst**

Use the above function **new-line** to check if there are too many palindromes of the same length to fit in one line.

VEGETABLE AND FRUIT STORE

PROBLEMS 3: RUNNING A STORE

Suppose that your neighbor opened a vegetable and fruit store. In order to provide a better service to customers, he has decided to use a computer program for his business. He offered you a summer internship at his store to develop this program, and you have happily accepted his offer. This program should handle three kinds of transactions: **selling items**, **listing the current stock**, and **reporting all sales done during the day**. Thus, the main menu of the program would look like the one shown in the next slide.

(Continued)

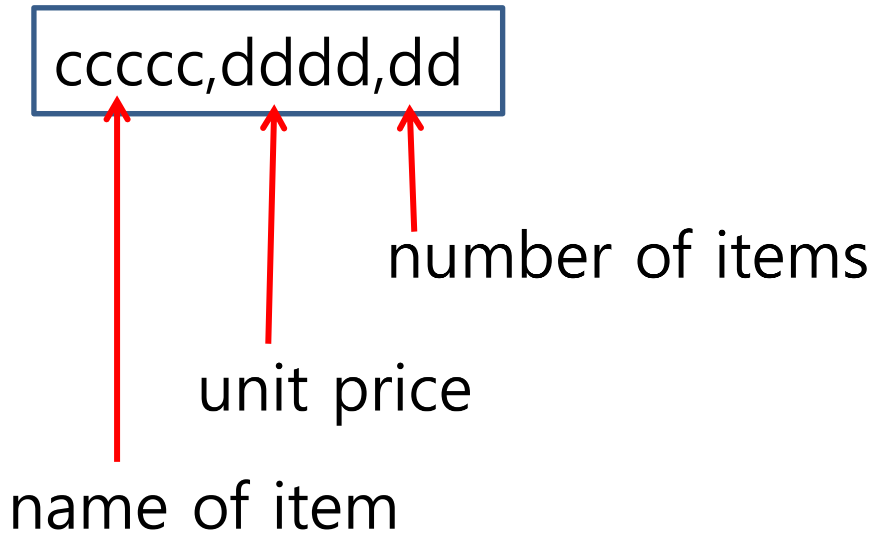
Main menu:

```
What would you like to do?
   S: Sell item
   P: Print stock
   R: Report sales
   E: Exit
Enter your choice (S, P, R, or E)>>
```

(Continued)

The stock item is maintained in a file, stock.txt in the following format:

```
apple, 900,190
orange,1300,90
pineapple,550,13
carrot,600,60
cucumber,900,30
egg plant, 1100,20
zuccini,1300,10
garlic,300,70
```



The file stock.txt should be created by yourself.

(Continued)

When the store is open, the file **stock.txt** will be **loaded** into a list **stock_list**, which is required to be sorted in the alphabetical order of item names.

```
    stock_list = load_stock(stock.txt)
    stock_list.sort()
```

For every transaction of selling an item(S), the program should update the **stock_list** for this item:

```
  stock quantity  = stock quantity – selling quantity
```

and add the transaction to the list, **sales_hist**. At the end of the day, the **stock_list** should be written back to the file, **stock.txt**.

(Continued)

Two list, **stock_list** and **sales_hist** are used to prepare **stock(P)** and **sales(S) reports**, respectively.

When the user enters "E", the program should be terminated after writing **stock_list** back to **stock.txt.**

Your task is to **understand and type in a hard copy of program** for solving this problem so that the program works correctly. You should also **add comments** so that your program is readable.

(Continued)

Main program

```
stock_list = load_stock(“stock_txt”)
sales_hist = []
while True:
    s = show_menu()
    if s = “E”:
        break
    elif s==“S”:
        sell( stock_list, sales_hist)
    elif s ==“P”:
        print_stock(stock_list)
    elif s == “R”
        print_sales(sales_hist)
    else:
        input_error(s)
store_stock(stock_list)
```

PROBLEM 4: USING A DICTIONARY

Change your program so as to load **stock.txt** into a **dictionary** instead of a **list**.