ES 112
Sorting
File Handling

Computing

IIT Gandhinagar, India

Oct, 2015

Sorting

Problem1 :
`Input`: A list of numbers.
`Goal`: To sort the list.

$$\text{Big Problem} \longrightarrow \text{Small Problem}$$

Sorting

Problem1 :
`Input`: A list of numbers.
`Goal`: To sort the list.

Big Problem $\longrightarrow$ Small Problem

Problem2 :
`Input`: A list `lst` where `lst[0:(n-1)]` is sorted but `lst[n-1]` is not in its correct place.
`Goal`: Put the number `lst[n-1]` in its correct place.

Solution: Problem 2

```
1  lst=[3,5,9,10,11,8]
2  n=len(lst)
3  x=lst[n-1]
4  #-------------------
5  i=n-2
6  while i>=0 and x<lst[i]:
7      lst[i+1]=lst[i]
8      i-=1
9
10 lst[i+1]=x
11 #-------------------
12 print lst
13 # [3, 5, 8, 9, 10, 11]
```

Solution: Problem1

```
lst=[26,3,2,37,2,1,45,30]
N=len(lst)
for n in range(2,N+1):
    x=lst[n-1]
    i=n-2
    while i>=0 and x<lst[i]:
        lst[i+1]=lst[i]
        i-=1
    lst[i+1]=x

print lst
```

Insertion Sort

Time and Space Complexity

- How much space and time does an algorithm take?

Time and Space Complexity

- How much space and time does an algorithm take?
- Measured in terms of the input size.

Time and Space Complexity

- How much space and time does an algorithm take?
- Measured in terms of the input size.
- Space required for the above sorting algorithm $\sim n$ (linear).

Time and Space Complexity

- How much space and time does an algorithm take?
- Measured in terms of the input size.
- Space required for the above sorting algorithm $\sim n$ (linear).
- Time required for the above sorting algorithm $\sim n^2$ (quadratic).

Time and Space Complexity

- How much space and time does an algorithm take?
- Measured in terms of the input size.
- Space required for the above sorting algorithm $\sim n$ (linear).
- Time required for the above sorting algorithm $\sim n^2$ (quadratic).
- There are better sorting algorithms.

Exercise

Write a program to sort a list in reverse order.

File Handling

File handling consists of mainly the following tasks:

- Opening a file.

File Handling

File handling consists of mainly the following tasks:

- Opening a file.
- Reading from a file, writing to a file, modifying a file.

File Handling

File handling consists of mainly the following tasks:

- Opening a file.
- Reading from a file, writing to a file, modifying a file.
- Closing a file.

File Sample

Name	Roll	Midsem	FinalSem
Zarski	12	75	82
Herbrand	14	68	75
Henkin	16	60	75

File Name: `Input.txt`

Opening a File

- We use the function `open()` to open a file.
- Syntax: `open(FileName, mode)`
- `FileName` is a string that gives the name (sometimes along with the path) of the file.
- `mode` tell how to open the file.
- Some common modes are `''r''`,`''w''`,`''r+''`, `''a''` (*read, write, read/write,append* respectively)

Example

```
1 f=open("Input.txt","r")
2
3 print f.read()
```

- The file `Input.txt` is opened in read mode. The program can only read from it.
- The file should already exist, otherwise it will show some run-time error.
- `read()` function reads the entire file as a string.
- `read(n)` function reads n characters from the file.
- File pointer: It points to the character from where next character is to be read. Initially it points to the 1st character in the file. After reading N character it points to the $N + 1$st character.
- When it reaches the end of the file any further attempt to read from the file will return the empty string ‘‘’’.

Reading Lines

- `readline()` reads one line at a time.
- When the program has read everything any further attempt to read from the file will return the empty string ‘‘’’.
- If the program has already read a portion of the file, the next read starts from the next character in the remaining portion.
- `close()` function closes a file.

```
1 f=open("Input.txt","r")
2
3 print f.read(13)
4 print f.readline()
5 print f.readline()
6
7 f.close()
```

Total Marks Calculation

Task: Compute the total marks obtained by each student.

```
# Cmpute the Total marks
f=open("Input.txt","r")

header = f.readline() # Does not contain actual info about marks
print header.split(),"\n\n\n"
#split() makes a list with individual words
print header

for line in f:
    lst=line.split()
    n=len(line)-1
    total=int(lst[2])+int(lst[3]) #int() converts a string to integer
    newline=line[:n]+"      "
    print newline, total

f.close()
```

Write The Full Information in a File

Task: Compute the total marks obtained by each student and write to a file

```
# Write data to File
f=open("Input.txt","r")
of=open("Output.txt","w")

header = f.readline() # Does not contain actual info about marks

of.write(header)

for line in f:
    lst=line.split()
    n=len(line)-1
    total=int(lst[2])+int(lst[3]) #int() converts TO integer
    newline=line[:n]+"      "+str(total)+"\n" #str() converts TO string
    of.write(newline)

f.close()
of.close()
```

- Warning: Opening an existing file in write mode will replace the file.
- `of.write(str)` writes the string `str` in the file pointed by the file object `of`.

Formatted Output

```
1 s="There are %d sheds in old campus" % (6)
2 print s
3 # 'There are 6 sheds in old campus'
4 print "%s got %d in midsem" % ("Zarski",65)
5 # Zarski got 65 in midsem
6 print "Pi is not %f" % 3.1415
7 # Pi is not 3.141500
8 print "Pi is not %.2f" % 3.1415
9 # Pi is not 3.14
```

Formatted Output

```
1 f=open("FormatOp.txt","w")
2
3 f.write("Replace all %d's by %d's"%(3,4))
4
5 f.close()
```

More File Handling Functions

- `f.tell()` tells from where in the file the next character is going to be read.
- With `f.seek(offset,fromWhere)` we can change this position. (See: http://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files)

Exercise

Read an existing file and write the content of the file in a new file with all the lower case letters replaced by uppercase letters.