# FILE HANDLING in Python

# Problem

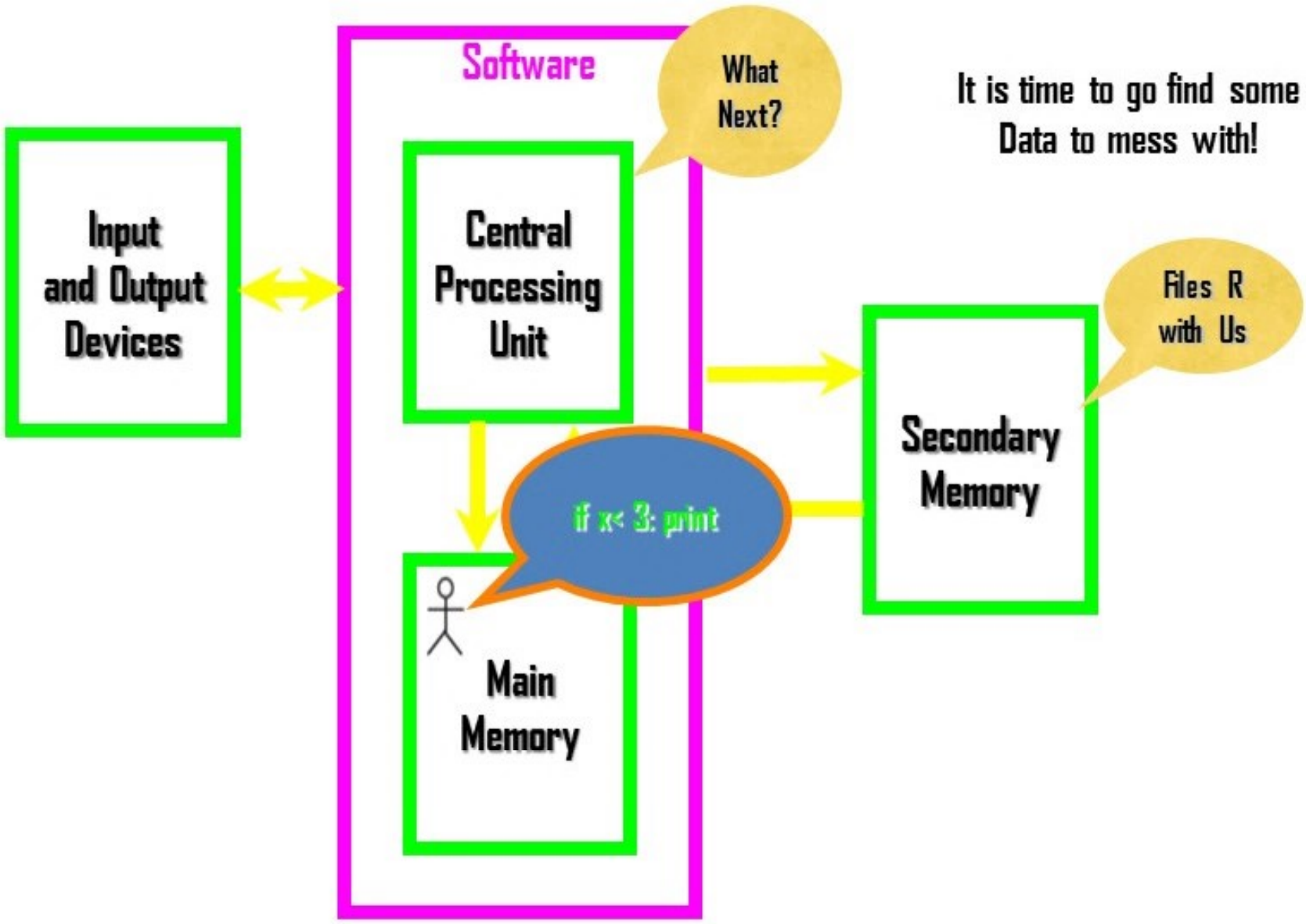
- Consider the following scenario

  -> You are asked to find the number of students who secured **centum in mathematics** in their examination. **A total of 6 lakhs students** appeared for the examinations and their results are available with us.

- The processing is typically supposed to be automatic and running on a computer. As data are most helpful when presented systematically and in fact educational to highlight their practicality.

# Pseudocode

```
OPEN file
REPEAT
    READ each line of file
    SET count = 0
    PARSE the line
    IF maths_mark == 100 THEN
            COMPUTE count as count + 1
    END IF
until end of file is reached
PRINT count
```


Software
What Next?
It is time to go find some Data to mess with!
Input and Output Devices
Central Processing Unit
Files R with Us
Secondary Memory
if x< 3: print
Main Memory

# Input / Output

- Input is any information provided to the program
  - Keyboard input
  - Mouse input
  - File input
  - Sensor input (microphone, camera, photo cell, etc.)
- Output is any information (or effect) that a program produces:
  - sounds, lights, pictures, text, motion, etc.
  - on a screen, in a file, on a disk or tape, etc.

# Kinds of Input and Output

- What kinds of Input and Output have we knew
  - – print (to the console)
  - – input (from the keyboard)
- So far…
  - – Input: keyboard input only
  - – Output: graphical and text output transmitted to the computer screen
- Any other means of I/O?

# Necessity of Files

- Small businesses accumulate various types of data, such as financial information related to revenues and expenses and data about employees, customers and vendors.
- Traditional file organization describes storing data in paper files, within folders and filing cabinets.
- Electronic file organization is a common alternative to paper filing; each system has its benefits and drawbacks.

# What is a file?

- A file is some information or data which stays??? in the computer storage devices.

- We already know about different kinds of file, like music files, video files, text files, etc.

# Introduction to file handling

- **Files** – Huge volume or Collection of data
- **Types** – Binary, Raw, Text, etc.
- **Open** any file before read/write.

# Opening a File

file object = open(file_name [, access_mode]
[, buffering])

## Modes of opening a File:

- **r** – Reading only
- **r+** - Both Read/Write
- **w** – Writing only
- **w+** - Both Read/Write
- **A** – Appending
- **a+** - Appending/Reading

# File Object Attributes

| Attribute | Description |
| --- | --- |
| file.closed | Returns true if file is closed, false otherwise. |
| file.mode | Returns access mode with which file was opened. |
| file.name | Returns name of the file. |

# Example

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
```

# Functions for file Handling

- The read functions contains different methods:
  - **read()** #return one big string
  - **readline()** #return **one line at a time**
  - **readlines()** #returns a **list of lines**

# File Handling functions contd..

- This method writes a sequence of strings to the file.
  - **write ()** #Used to write a **fixed sequence of characters to a file**
  - **writelines()** #writelines can **write a list of strings**.

# Functions contd...

- The append function is used to append to the file instead of overwriting it.
- To append to an existing file, simply open the file in **append mode ("a"):**
- When you're done with a file, use **close()** to close it and free up any system resources taken up by the open file.

**To open a text file, use:**

```
fh = open("hello.txt", "r")
```

**To read a text file, use:**

```
fh = open("hello.txt","r")
print fh.read()
```

**To read one line at a time, use:**

```
fh = open("hello".txt", "r")
print fh.readline()
```

**To read a list of lines use:**

```
fh = open("hello.txt.", "r")
print fh.readlines()
```

**To write to a file, use:**

```
fh = open("hello.txt","w")
write("Hello World")
fh.close()
```

**To write to a file, use:**

```
fh = open("hello.txt", "w")
lines_of_text = ["a line of text", "another line of text", "a third line"]
fh.writelines(lines_of_text)
fh.close()
```

**To append to file, use:**

```
fh = open("Hello.txt", "a")
write("Hello World again")
fh.close
```

**To close a file, use**

```
fh = open("hello.txt", "r")
print fh.read() fh.close()
```

# Playing Randomly in files

- **fileObject.tell()** -> current position within a file
- **fileObject.seek(offset [,from])** -> Move to new file position.
  - Argument offset is a byte count.
  - Optional argument whence defaults to 0 (offset from start of file, offset should be >= 0); other values are 1 (move relative to current position, positive or negative), and 2 (move relative to end of file, usually negative, although many platforms allow seeking beyond the end of a file)

# Example for random seeking

```
fobj = open('/tmp/tempfile', 'w')
fobj.write('0123456789abcdef')
fobj.close()
fobj = open('/tmp/tempfile')
fobj.tell() #tell us the offset position
0L
fobj.seek(5) # Goto 5th byte
fobj.tell()
5L
fobj.read(1) #Read 1 byte
'5'
fobj.seek(-3, 2) # goto 3rd byte from the end
fobj.read() #Read till the end of the file
'def'
```

# Python code to copy a file to another file

```
import sys
if len(sys.argv) < 3:
    print("Wrong parameter")
    print("./copyfile.py file1 file2")
    sys.exit(1)
f1 = open(sys.argv[1])
s = f1.read()
f1.close()
f2 = open(sys.argv[2], 'w')
f2.write(s)
f2.close()
```

# Interacting with OS

- A separate module called OS is available as part of python.
- This module facilitates functions related to Operating System.

- **os.getcwd()** – gets current working directory
- **os.listdir(directory)** – gets children of directory
- **os.chdir(path)** – change current working directory to path
- **os.mkdir(path)** – make directory called path
- **os.remove(path)** – remove file (not directory) called path
- **os.rmdir(path)** – remove directory (not file) called path
- **os.rename(oldpath, newpath)** – rename (or move) oldpath to newpath
- **os.path.isfile(path)**, **os.path.isdir(path)** – Boolean functions indicating if a particular pathname refers to a real file/directory
- **os.system(command)** – execute a terminal command as indicated by the string command

# Tips and Tricks makes it Easier...

- Number of characters in a file is same as the length of its contents.
  - def charcount(filename):
  - return len(open(filename).read())
- Number of words in a file can be found by splitting the contents of the file.
  - def wordcount(filename):
  - return len(open(filename).read().split())
- Number of lines in a file can be found from readlines method.
  - def linecount(filename):
  - return len(open(filename).readlines())

**Maximum Total?**

**Number of students secured 100?**

## PROCEDURE

1. **Open the file** (Excel Sheet) containing data.
2. **Save** the same file as filename.csv (Comma separate value)
3. Open the file using **open( )** function.
4. Read the contents using **read()** functions and append the fields it in a list.
5. **Perform the required operations** on list and display the result.

File Handling

---

```
l = [ ]
for line in open('e:\Workspace\sslc.csv'):
#print line
res = line.split(",")
l.append(int(res[8]))
print "Maximum Total", max(l)

c = 0
for line in open('e:\Workspace\sslc.csv'):
res = line.split(",")
if int(res[5]) == 100:
c = c + 1
print "No. of Hundreds in Maths", c
```

# TRY it...

- Do you know how many CPU(s) are there in your processor? or what is the model name?

Let us write some code which can help us to know these things.

- If you are in Linux, then you can actually view the output of the */scpu* command first. You can actually find the information in a file located at */proc/cpuinfo*.
- Now try to write code which will open the file in read only mode and then read the file line by line and find out the number of CPU(s).

# Sample Problems

- Given a HTML file as input, display the file after removing all the tags.

Procedure (Hint: Process one character at a time)

i. When the SIGNAL variable is GREEN, write the current character to an output file.

ii. Skip the character if SIGNAL is RED.

SIGNAL becomes RED when the current character is "<" (start of tag) and it becomes GREEN when the current character is ">" (end of tag)

# Sample Problems contd..

- In a plain text file, author has written comments inside '(' and ')'. Read this input file and the output file should contain no comments. [Logic: Keep count on opening and closing brackets when needed]
- Write a program to read a file and count the number of occurrences of alphabets. Display the result in sorted order of the occurrences.