5.5 STANDARD INPUT, OUTPUT AND ERROR STREAMS

If someone asks you to give input to a program interactively or by typing, you know what device you need for it – *the keyboard*. Similarly, if someone says that the output is to be displayed, you know which device it will be displayed on – *the monitor*. So, we can safely say that the *Keyboard* is the **standard input device** and the *monitor* is **standard output device**. Similarly, any error if occurs is also displayed on the monitor. So, *the monitor* is also **standard error device**. That is,

- standard input device (**stdin**) – reads from the keyboard
- standard output device (**stdout**) – prints to the display and can be redirected as standard input.
- standard error device (**stderr**) – Same as *stdout* but normally only for errors. Having error output separately allows the user to divert regular output to a file and still be able to read error messages.

Do you know internally how these devices are implemented in Python? These standard devices are implemented as files called *standard streams*. In Python, you can use these standard streams files by using **sys** module. After importing, you can use these standard streams (**stdin**, **stdout** and **stderr**) in the same way as you use other files.

Interesting : Standard Input, Output Devices as Files

If you import **sys** module in your program then, **sys.stdin.read( )** would let you read from keyboard. This is because the keyboard is the *standard input device* linked to **sys.stdin**. Similarly **sys.stdout.write( )** would let you write on the standard output device, *the monitor*. **sys.stdin** and **sys.stdout** are standard input and standard output devices respectively, treated as files.

Thus **sys.stdin** and **sys.stdout** are like files which are opened by the Python when you start Python. The **sys.stdin** is always opened in read mode and **sys.stdout** is always opened in write mode. Following code fragment shows you interesting use of these. It prints the contents of a file on monitor without using **print** statement :

These statements would write on file/device associated with sys.stdout, which is the monitor

```
import sys
fh = open(r"E:\poem.txt")
line1 = fh.readline()
line2 = fh.readline()
sys.stdout.write(line1)
sys.stdout.write(line2)
sys.stderr.write("No errors occurred\n")
```

Output produced is :

```
>>> ==========================
>>>
        WHY ?
We work, we try to be better
No errors occurred
>>>
```

*See, **stderr** also displayed its text on monitor.*

Info Box 5.3

Python Data Files : with Statement

Python's **with** statement for files is very handy when you have two related operations which you'd like to execute as a pair, with a block of code in between. The syntax for using with statement is :

```
with open(<filename>, <filemode>) as <filehandle> :
    <file manipulation statements>
```

The classic example is opening a file, manipulating the file, then closing it :

```
with open('output.txt', 'w') as f:
    f.write('Hi there!')
```

The above **with** statement **will automatically close the file** after the nested block of code. The advantage of using a ***with statement*** is that it is guaranteed to close the file no matter how the nested block exits. **Even if an exception (a runtime error) occurs before the end of the block, the with statement will handle it and close the file.**

Absolute and Relative Paths

Full name of a file or directory or folder consists of path\primaryname.extension.

Path is a sequence of directory names which give you the hierarchy to access a particular directory or file name. Let us consider the following directory structure :

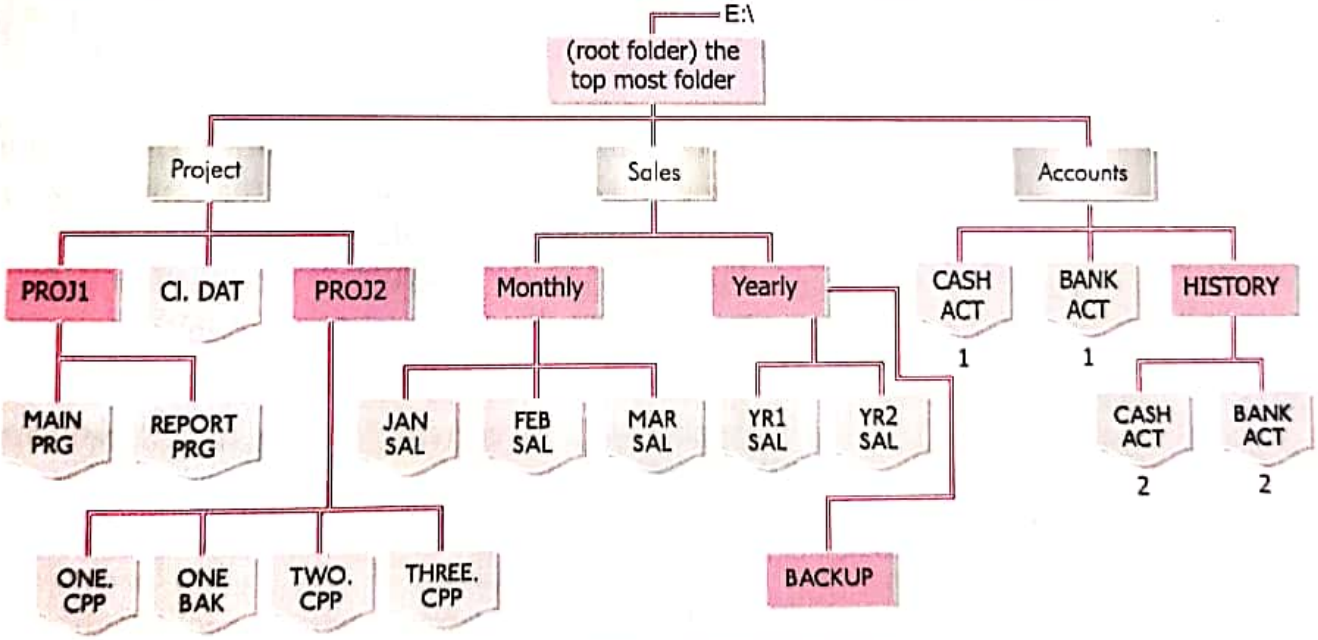

Figure 5.7 A Sample Directory Structure.

Now the *full name* of directories PROJECT, SALES and ACCOUNTS will be

`E:\PROJECT, E:\SALES` and `E:\ACCOUNTS` respectively.

So format of path can be given as

```
Drive-letter:\directory [\directory...]
```

where **first \ (backslash) refers to root directory** and other ('\'s) separate a directory name from the previous one.

Now the directory BACKUP'S path will be :

E:\SALES\YEARLY\BACKUP

As to reach BACKUP the sequence one has to follow is : under drive E, under root directory (first \), under SALES subdirectory of root, under YEARLY subdirectory of SALES, there lies BACKUP directory.

Similarly full name of ONE.VBP file under PROJ2 subdirectory will be :

E:\ PROJECT\PROJ2\ONE.VBP

↓

refers to root directory

PATHNAME

The full name of a file or a directory is also called pathname

Now see there are *two* files with the same name CASH.ACT, one under ACCOUNTS directory and another under HISTORY directory but according to Windows rule no two files can have same names (*path names*). Both these files have same names but their path names differ, therefore, these can exist on system. Therefore, CASH.ACT$_1$'s path will be

E:\ACCOUNTS\CASH.ACT

and CASH.ACT$_2$'s path will be

E:\ACCOUNTS\HISTORY\CASH.ACT

Above mentioned *path names* are ***Absolute Pathnames*** as they mention the *paths* from the top most level of the directory structure.

NOTE

The absolute paths are from the topmost level of the directory structure. The relative paths are relative to **current working directory** denoted as a **dot(.)** while its **parent directory** is denoted with two dots(..).

Relative pathnames mention the paths relative to current working directory. Let us assume that current working directory now is, say, PROJ2. The symbols . *(one dot)* and .. *(two dots)* can be used now in relative paths and pathnames. The symbol . can be used in place of current directory and .. denotes the parent directory.

So, with PROJ2 as current working folder, pathname of TWO.DOC will be :

TWO.DOC in current folder

.\TWO.DOC (PROJ2 being working Folder)

which means under current working folder, there is file TWO.PAS. Similarly, path name for file CL.DAT will be

..\CL.DAT (PROJ2 being working Folder)

CL.DAT in parent folder

which means under parent folder of current folder, there lies a file CL.DAT. Similarly, path name for REPORT.PRG will be

..\PROJ1\REPORT.PRG (PROJ2 being working Folder)

that is under parent folder's subfolder PROJ1, there lies a file REPORT.PRG.

Check Point 5.1

1. In which of the following file modes, the existing data of file will not be lost ?
   (a) 'rb' (b) ab (c) w
   (d) w + b (e) 'a + b' (f) wb
   (g) wb+ (h) w+ (i) r+
2. What would be the data type of variable data in following statements ?
   (a) data = f.read( )
   (b) data = f.read(10)
   (c) data = f.readline( )
   (d) data = f.readlines( )
3. How are following statements different ?
   (a) f.readline( )
   (b) f.readline( ).rstrip( )
   (c) f.readline( ).strip( )
   (d) f.readline.rstrip('\n')