Errors and Exceptions:

Syntax Errors: Syntax errors, also known as parsing errors, are perhaps the most common kind of complaint you get while you are still learning Python.

Exceptions

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called exceptions and are not unconditionally fatal. It is possible to write programs that handle selected exceptions. Look at the following example, which asks the user for input until a valid integer has been entered, but allows the user to interrupt the program (using Control-C or whatever the operating system supports).

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print "Oops!  That was no valid number.  Try again..."
```

The try statement works as follows.

First, the try clause (the statement(s) between the try and except keywords) is executed.If no exception occurs, the except clause is skipped and execution of the try statement is finished.If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then if its type matches the exception named after the except keyword, the except clause is executed, and then execution continues after the try statement.If an exception occurs which does not match the exception named in the except clause, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops with a message as shown above.

File handling in python.

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a file object.

The open Function

Before you can read or write a file, you have to open it using Python's built-in open() function. This function creates a file object, which

would be utilized to call other support methods associated with it.

Syntax

file object = open(file_name [, access_mode])
Here are parameter details:
file_name: The file_name argument is a string value that contains the name of the file that you want to access.

access_mode: The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).

r : Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.

rb : Opens a file for reading only in binary format.
The file pointer is placed at the beginning of the file.
This is the default mode.

r+ : Opens a file for both reading and writing. The file pointer placed at the beginning of the file.

rb+ : Opens a file for both reading and writing in binary format.
The file pointer placed at the beginning of the file.

w : Opens a file for writing only. Overwrites the file if the file exists.
If the file does not exist, creates a new file for writing.

wb : Opens a file for writing only in binary format. Overwrites the file if the file exists.
If the file does not exist, creates a new file for writing

a : Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.