Errors

* Errors and Exception Handling.

EOL error ⇒ (i.e) end of line error.

eg:- print 'Hello.

o/p = EOL error. (forgot to put (') at the end of line).

* Exceptions :- specific description of types of errors.

- 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. (i.e) you don't need to break the entire code, you can have your rest of the code run.

• eg:- (i) 2 + `5'

o/p = type error unsupported operand type for + : 'int' and 'str'.

- * Try and Except (keyword)

(ii) try:
    2 + `5'
except TypeError:
    print "There was a type error!"

o/p = There was a type error!

no need to write the type of exception, the code will still be executed the same way

- Thus using try & except format rest of the code can be run even if there is an error (i.e) it is not unconditionally fatal.

- here the code tries to do 2+'s' excepts that there is a type error and print - There was a type error! thus it does not show the error and stop or breaks the code.
- The code which can cause an exception to occur is put in the try block and the handling of the exception is implemented in the exception block.

syntax:

```
try: You do your operations here----
     ------
     ------
except exception I:
    If there is exception I, then execute
    this block.
except exception II:
    If there is exception II, then execute
    this block.
    ------
    ------
else:
    If there is no exception then execute
    this block.
```

- Finally :- allows us to place code that will always be run even if there are exceptions!
The finally : block of code will always be run regardless if there was an exception in the try code block.

syntax

```
try:
    code block here
    .....
    .....
    Due to any exception, this code may be skipped!
Finally:
    This code would always be executed.
```

Eg :-

```
def askint():
    try:
        val = int(raw_input('Please enter an integer: '))
    except:
        print 'looks like you did not enter an integer!'
    finally:
        print 'Finally block executed!'
    print val
```

(i) askint()
o/p = Please enter an integer : 3
o/p = Finally block executed!
3

(ii) ask int()

o/p = Please enter an integer : FIVE

o/p = looks like you did not enter an integer!

Finally block executed!