Python Training.....

-- Jeetendra Bhattad

Agenda

- Errors v/s Exceptions

Errors

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

```
>>> while True print 'Hello world'
  File "<stdin>", line 1, in ?
    while True print 'Hello world'
                   ^
SyntaxError: invalid syntax
```

Exceptions

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

In general, when a Python script encounters a situation that it can't cope with, it raises an exception.

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.

Most exceptions are not handled by programs, however, and result in error messages as shown next:

Exceptions – e.g

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero
>>> 4 + x*3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'x' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: cannot concatenate 'str' and 'int' objects`
```

The last line of the error message indicates what happened.

Exceptions come in different types, and the type is printed as part of the message: the types in the example are ZeroDivisionError, NameError and TypeError.

Handling Exceptions

- try-except block
- try-except-finally block
- Try-except-else-finally block : finally must be always at end.
- generic except block for handling all exceptions
- raising exception