Python - Error Types

The most common reason of an error in a Python program is when a certain statement is not in accordance with the prescribed usage. Such an error is called a syntax error. The Python interpreter immediately reports it, usually along with the reason.

```
>>> print "hello"
SyntaxError: Missing parentheses in call to 'print'. Did you
mean print("hello")?
```

In Python 3.x, print is a built-in function and requires parentheses. The statement above violates this usage and hence syntax error is displayed.

Many times though, a program results in an error after it is run even if it doesn't have any syntax error. Such an error is a runtime error, called an exception. A number of built-in exceptions are defined in the Python library. Let's see some common error types.

IndexError is thrown when trying to access an item at an invalid index.

```
>>> L1=[1,2,3]
>>> L1[3]
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module>
L1[3]
IndexError: list index out of range
```

ModuleNotFoundError is thrown when a module could not be found.

```
>>> import notamodule
Traceback (most recent call last):
File "<pyshell#10>", line 1, in <module>
import notamodule
ModuleNotFoundError: No module named 'notamodule'
```

KeyError is thrown when a key is not found.

```
>>> D1={'1':"aa", '2':"bb", '3':"cc"}
>>> D1['4']
Traceback (most recent call last):
File "<pyshell#15>", line 1, in <module>
D1['4']
KeyError: '4'
```

ImportError is thrown when a specified function can not be found.

```
>>> from math import cube
Traceback (most recent call last):
File "<pyshell#16>", line 1, in <module>
from math import cube
ImportError: cannot import name 'cube'
```

StopIteration is thrown when the `next()` function goes beyond the iterator items.

```
>>> it=iter([1,2,3])
>>> next(it)
1
>>> next(it)
2
>>> next(it)
3
>>> next(it)
Traceback (most recent call last):
File "<pyshell#23>", line 1, in <module>
next(it)
StopIteration
```

TypeError is thrown when an operation or function is applied to an object of an inappropriate type.

```
>>> '2'+2
Traceback (most recent call last):
File "<pyshell#23>", line 1, in <module>
'2'+2
TypeError: must be str, not int
```

ValueError is thrown when a function's argument is of an inappropriate type.

```
>>> int('xyz')
Traceback (most recent call last):
File "<pyshell#14>", line 1, in <module>
int('xyz')
ValueError: invalid literal for int() with base 10: 'xyz'
```

NameError is thrown when an object could not be found.

```
>>> age
Traceback (most recent call last):
File "<pyshell#6>", line 1, in <module>
age
NameError: name 'age' is not defined
```

ZeroDivisionError is thrown when the second operator in the division is zero.

```
>>> x=100/0
Traceback (most recent call last):
File "<pyshell#8>", line 1, in <module>
x=100/0
ZeroDivisionError: division by zero
```

KeyboardInterrupt is thrown when the user hits the interrupt key (normally Control-C) during the execution of the program.

```
>>> name=input('enter your name')
enter your name^c
Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
name=input('enter your name')
KeyboardInterrupt
```

The following table lists important built-in exceptions in Python.

Exception	Description
AssertionError	Raised when the assert statement fails.
AttributeError	Raised on the attribute assignment or reference fails.
EOFError	Raised when the input() function hits the end-of-file condition.
FloatingPointError	Raised when a floating point operation fails.
GeneratorExit	Raised when a generator's close() method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when the index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
KeyboardInterrupt	Raised when the user hits the interrupt key (Ctrl+c or delete).
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in the local or global scope.
NotImplementedError	Raised by abstract methods.
OSError	Raised when a system operation causes a system-related error.
OverflowError	Raised when the result of an arithmetic operation is too large to be represented.
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.
RuntimeError	Raised when an error does not fall under any other category.
StopIteration	Raised by the next() function to indicate that there is no further item to be returned by the iterator.
SyntaxError	Raised by the parser when a syntax error is encountered.
IndentationError	Raised when there is an incorrect indentation.
TabError	Raised when the indentation consists of inconsistent tabs and spaces.
SystemError	Raised when the interpreter detects internal error.
SystemExit	Raised by the sys.exit() function.
TypeError	Raised when a function or operation is applied to an object of an incorrect type.

UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
UnicodeEncodeError	Raised when a Unicode-related error occurs during encoding.
UnicodeDecodeError	Raised when a Unicode-related error occurs during decoding.
UnicodeTranslateError	Raised when a Unicode-related error occurs during translation.
ValueError	Raised when a function gets an argument of correct type but improper value.
ZeroDivisionError	Raised when the second operand of a division or module operation is zero.