

I. 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:

```
>>> while 1 print 'Hello world'  
File "<stdin>", line 1, in ?  
while 1 print 'Hello world'  
      ^
```

SyntaxError: invalid syntax

II. Exceptions

```
>>> 10 * (1/0)
```

Traceback (most recent call last):

File "<stdin>", line 1, in ?

ZeroDivisionError: integer division or modulo

```
>>> 4 + spam*3
```

Traceback (most recent call last):

File "<stdin>", line 1, in ?

NameError: spam

```
>>> '2' + 2
```

Traceback (most recent call last):

File "<stdin>", line 1, in ?

TypeError: illegal argument type for built-in operation

Python Library Exceptions:

<http://www.python.org/doc/current/lib/module-exceptions.html>

III. Run-time Error Checking

Example 1:

```
def division(x,y):  
    try: return x/y  
    except ZeroDivisionError:  
        print "Can't divide by zero."
```

```
>>> division(4,5.8)  
0.68965517241379315  
>>> division(4,0)  
Can't divide by zero.
```

Example 2:

```
while 1:  
    try:  
        x = int(raw_input("Please enter a number: "))  
        break  
    except ValueError:  
        print "Oops! That was no valid number. Try again..."
```

Example 3:

```
except (RuntimeError, TypeError, NameError):  
    pass
```

Example 4:

```
try:  
    f = open('myfile.txt')  
    s = f.readline()  
    i = int(string.strip(s))  
except IOError, (errno, strerror):  
    print "I/O error(%s): %s" % (errno, strerror)  
except ValueError:  
    print "Could not convert data to an integer."  
except:  
    print "Unexpected error:", sys.exc_info()[0]  
    raise
```

IV. More on Classes: Inheritance and Multiple Inheritance

A language feature would not be worthy of the name "class" without supporting inheritance. The syntax for a derived class definition looks as follows:

```
class DerivedClassName (BaseClassName) :  
    <statement-1>  
    .  
    .  
    .  
    <statement-N>
```

Inheritance means exactly that: The new class "inherits" the properties (data and methods) from the old class just like you *wish* you inherited that money and mansion from your rich uncle.

Example:

```
class SeqClass:
    def __init__(self):
        self.data = []
    def store_data(self, sequence):
        self.data.append(sequence)
    def display(self):
        print self.data
class GetSeq(SeqClass):
    def get_data(self,filename):
        fn=open(filename, 'r')
        for line in fn.readlines():
            self.store_data(line)
>>> getseq = GetSeq()
>>> getseq.get_data('tmp.py')
>>> getseq.display()
['AGGCGTCTCCTGAGTC\n',
'AUUGUCUGUAAUCUGUGUGUCUGUAUGC\n',
'SODIEEIALDJASXOEKFPWIEJII\n']
```