Raising Exceptions in Python

To throw an exception from an operation implementation, you simply instantiate the exception, initialize it, and throw it. For example:

The mapping for exceptions generates a constructor that accepts values for data members, so we can simplify this example by changing our raise statement to the following:

```
Python

class FileI(Filesystem.File):
    # ...

def write(self, text, current=None):
    # Try to write the file contents here...
    # Assume we are out of space...
    if error:
        raise Filesystem.GenericError("file too large")
```

If you throw an arbitrary Python run-time exception, the Ice run time catches the exception and then returns an UnknownException to the client. Similarly, if you throw an "impossible" user exception (a user exception that is not listed in the exception specification of the operation), the client receives an UnknownUserException.

If you throw a run-time exception, such as MemoryLimitException, the client receives an UnknownLocalException. For that reason, you should never throw system exceptions from operation implementations. If you do, all the client will see is an UnknownLocalException, which does not tell the client anything useful.



Three run-time exceptions are treated specially and not changed to <code>UnknownLocalException</code> when returned to the client: <code>ObjectNotExistException</code>, <code>OperationNotExistException</code>, and <code>FacetNotExistException</code>.

See Also

- Run-Time Exceptions
- Server-Side Python Mapping for Interfaces
- Python Mapping for Exceptions
- Facets and Versioning