5.1 Connecting to MySQL Using Connector/Python

The `connect()` constructor creates a connection to the MySQL server and returns a `MySQLConnection` object.

The following example shows how to connect to the MySQL server:

```
import mysql.connector

cnx = mysql.connector.connect(user='scott', password='password',
                              host='127.0.0.1',
                              database='employees')
cnx.close()
```

Section 7.1, “Connector/Python Connection Arguments” describes the permitted connection arguments.

It is also possible to create connection objects using the connection.MySQLConnection() class:

```
from mysql.connector import (connection)

cnx = connection.MySQLConnection(user='scott', password='password',
                                 host='127.0.0.1',
                                 database='employees')
cnx.close()
```

Both forms (either using the `connect()` constructor or the class directly) are valid and functionally equal, but using `connect()` is preferred and used by most examples in this manual.

To handle connection errors, use the `try` statement and catch all errors using the errors.Error exception:

```
import mysql.connector
from mysql.connector import errorcode

try:
  cnx = mysql.connector.connect(user='scott',
                                database='employ')
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password")
  elif err.errno == errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
  else:
    print(err)
else:
  cnx.close()
```

Defining connection arguments in a dictionary and using the `**` operator is another option:

```
import mysql.connector

config = {
  'user': 'scott',
  'password': 'password',
  'host': '127.0.0.1',
  'database': 'employees',
  'raise_on_warnings': True
}

cnx = mysql.connector.connect(**config)

```

```
13
    cnx.close()
```

Using the Connector/Python Python or C Extension

Connector/Python offers two implementations: a pure Python interface and a C extension that uses the MySQL C client library (see Chapter 8, *The Connector/Python C Extension*). This can be configured at runtime using the `use_pure` connection argument. It defaults to `False` as of MySQL 8, meaning the C extension is used. If the C extension is not available on the system then `use_pure` defaults to `True`. Setting `use_pure=False` causes the connection to use the C Extension if your Connector/Python installation includes it, while `use_pure=True` to `False` means the Python implementation is used if available.

> **Note**
>
> The `use_pure` option and C extension were added in Connector/Python 2.1.1.

The following example shows how to set `use_pure` to False.

```
1  import mysql.connector
2
3  cnx = mysql.connector.connect(user='scott', password='password',
4                                host='127.0.0.1',
5                                database='employees',
6                                use_pure=False)
7  cnx.close()
```

It is also possible to use the C Extension directly by importing the `_mysql_connector` module rather than the `mysql.connector` module. For more information, see Section 8.2, "The _mysql_connector C Extension Module".