pyodbc

Open Database Connectivity (ODBC) is a standard API for accessing databases. Most production databases have an ODBC driver available that you can install to access their database with.

One of the most popular methods of connecting via ODBC with Python is the pyodbc package. According to its page on the Python Packaging Index, you can use it on Windows or Linux. The pyodbc package implements the DB API 2.0 specification. You can install pyodbc with pip:

```
pip install pyodbc
```

Let's look at a fairly generic way to connect to SQL Server with pyodbc and select some data like we did in the adodbapi section:

```
import pyodbc

driver = 'DRIVER={SQL Server}'
server = 'SERVER=localhost'
port = 'PORT=1433'
db = 'DATABASE=testdb'
user = 'UID=me'
pw = 'PWD=pass'
conn_str = ';'.join([driver, server, port, db, user, pw])

conn = pyodbc.connect(conn_str)
cursor = conn.cursor()

cursor.execute('select * from table_name')
row = cursor.fetchone()
rest_of_rows = cursor.fetchall()
```

In this code, we create a very long connection string. It has many parts. The driver, server port number, database name, user and password. You would probably want to save most of this information into some kind of configuration file so you wouldn't have to enter it each time. The username and password should never be hard coded, although you will find that it

and password should never be hard coded, although you will find that it happens from time to time in the real world.

Once we have our connection string, we attempt to connect to the database via the **connection** function call. If it connects successfully, then we now have a connection object which we can use to create a cursor object from. Now that we have a cursor, we can query the database and run any other commands that we need to depending on what our database permissions are. In this example, we run a SELECT * on our database to extract all rows. Then we demonstrate that you can just grab one row at a time or pull them all via **fetchone** and **fetchall** respectively. There is also a **fetchmany** function that you can use to specify how many rows you want returned.

If you have a database that works with ODBC, you should give this package a try. Note that Microsoft databases are not the only databases that support this connection method.