How to connect Python to PostgreSQL Database

Currently I am working with PostgreSQL to manage spatial databases. To do calculations on the data in the database some Python modules are being written. Here I briefly explain how to connect Python to a database stored in and managed by the DBMS PostgreSQL. Python provides a module psycopg2, [http://initd.org/psycopg/](http://initd.org/psycopg/) for interaction with PostgreSQL. I am using this module to accomplish my tasks. You may also run others popular modules for PostgreSQL such as PyGreSQL, [http://www.pygresql.org/](http://www.pygresql.org/) and pyPySQL, [http://pypgsql.sourceforge.net/](http://pypgsql.sourceforge.net/) These packages containing modules that provide a Python DB-API 2.0 compliant interface to PostgreSQL databases:

Here are some stages to connect to a database and to retrieve the content:  
**Step 1:** Import the module psycopg2

```python
>>> import psycopg2 as dbapi2
```

**Step 2:** Connect to the database

```python
>>> db = dbapi2.connect (database="sdb_example", user="postgres",
                        password="db_pass")
```

**Step 3:** define a cursor to work with. Note that Python/Psycopg cursors are not cursors as defined by PostgreSQL

```python
>>> cur = db.cursor()
```

**Step 4:** at this stage, you are ready to run SQL statements to retrieve the data. For example, I have a table population_density consisting of population and its classes, as well as the geometry type for the area where the population is calculated.

Population_density table:
I am going to list gid, population density and population class, so I have the following SQL statement:

```
SELECT gid, pop_densit, pop_class FROM population_density;
```

In Python, we use the execute function to run sql SELECT queries:
```
>>> cur.execute ("SELECT gid, pop_densit, pop_class FROM population_density");
```

**Step 5:** when we have executed the query, we need to define a list, for example rows, to place the query results in.
```
>>> rows = cur.fetchall()
```
```
[(2, Decimal('21.7095'), 'low'), (3, Decimal('44.2259'), 'low'), (8, Decimal('48.1209'), 'low'), (9, Decimal('15.2835'), 'low'), (5, Decimal('110.2618'), 'medium'), (6, Decimal('81.8404'), 'medium'), (1, Decimal('158.3395'), 'high'), (4, Decimal('334.5707'), 'high'), (7, Decimal('186.1550'), 'high')]
```

Below are another example and a different style to display the query results:
```
>>> cur.execute ("""SELECT gid, pop_densit, area FROM population_density
WHERE pop_class = 'low' AND area > 100 ORDER BY gid DESC;"""");
```
```
>>> rows = cur.fetchall()
```
>>> for i, row in enumerate(rows):
print "Row", i, "value = ", row

Row 0 value =  (9, Decimal('15.2835'), Decimal('122.6811'))
Row 1 value =  (2, Decimal('21.7095'), Decimal('203.2293'))

If we finish working with a cursor or database, simply type the following statements (not a must):
cur.close()
db.close()

More about psycopg2:
http://wiki.postgresql.org/wiki/Using_psycopg2_with_PostgreSQL
http://initd.org/psycopg/