Postgresql and Python Tip Sheet
GEO 826: Geocomputation
Halloween, 2014

This sheet presents hopefully useful information for working with the Postgresql database in Python.

I. Connecting to Postgresql

Postgresql is a free database package with enterprise-level functionality, including lots of spatial ability. See its web page: http://www.postgresql.org/
Like many other applications in the linux environment, you can start an interactive session from the command prompt. Unlike most applications we use, you can start it in any working directory you want, with access to whatever databases and tables you want to work with.

To start it, the default is:

```
psql -d [db name]
```

In lab you only have one database, which is your user account name. That one starts by default, so you simply need to type:

```
psql
```

In a moment you have a prompt. You can list all your tables with \d, which is not too exciting, since you don't have any interesting ones yet (you should see two that have spatial stuff in them). You can issue any valid SQL at this prompt. Remember that SQL ends with the semicolon (;)!

Quit the interpreter with \q

II. Reading and creating table dumps

Creating tables is often a big pain. Fortunately it is pretty easy to read and write text file table dumps that can be used to make tables more portable, to back up databases, etc. From the unix prompt:

```
pg_dump -t [table name] [db name] > mytable.sql
```

would write the table to a new file called mytable.sql. This is a text file with sql commands that can recreate the table. You could zip it, email it to a friend, whatever. It can be used to regenerate the table. From the linux prompt:

```
psql -d [db name] -f mytable.sql
```

To create a single dump file for every table in a database, simply don't include the -t parameter.

III. Using psycopg2

This is a popular Python database interface. See its web page: http://pypi.python.org/pypi/psycopg2/
Essentially you import the module. Then establish a connection to a particular database within Postgresql. Then create a cursor (curser? Only if you have problems!) to that connection. Finally, execute valid SQL commands using the cursor. Retrieve contents of the cursor by calling its fetch methods: fetchall (a list of tuples (records)) or fetchone (retrieves the current record).

```
import psycopg2
db = psycopg2.connect (database="michigan", user="ashton", password="notmypwd")
cursor = db.cursor()
cursor.execute("SELECT datname from pg_database")
data = cursor.fetchall()

for row in data:
   print row
```