The `psycopg2` Package

Reference:

- Package
- Website
- Docs
- Source Repo
- Usage Example
- `connect()`
- `connection`
- `cursor`
- Transactions Control

The `psycopg2` ("psycho pee gee") package provides a way for Python to interface with PostgreSQL databases.

> Psycopg is the most popular PostgreSQL adapter for the Python programming language. At its core it fully implements the Python DB API 2.0 specifications. Several extensions allow access to many of the features offered by PostgreSQL. - Psycopg website

Run a `psycopg2` application "in development" using a database server on your local machine, and/or "in production" using a remote database server hosted by a provider like Heroku. If you run it in development, you should be able to connect via localhost, whereas if you run it in production, you should be able to connect using the production server's credentials. The professor recommends using Table Plus or some other GUI interface to your PostgreSQL databases, local or remote.

Installation

As a prerequisite: install PostgreSQL on your local machine. If you are on a Mac, use Homebrew: `brew install postgresql` and follow the post-installation instructions. Make sure you can connect to your local PostgreSQL installation via a GUI or command-line interface. If attempting to connect from the command-line, try running `psql` or perhaps `psql -U your_username`, depending on the name of your computer's user and method of PostgreSQL installation. Note the username and password you are using to connect.

After demonstrating your ability to connect to a local PostgreSQL installation, install `psycopg2`, if necessary:

```
pip install psycopg2
```

Usage

Place the following contents inside a new Python script:

```python
import psycopg2

# OPEN DATABASE CONNECTION

# If you installed PostgreSQL via Homebrew on a Mac, there should be both a
database and a user named after your computer's username.
connection = psycopg2.connect(dbname="mjr", user="mjr", password="")

# PERFORM A DATABASE OPERATION

with connection:
    with connection.cursor() as cursor:
        sql_query = "SELECT usename, usecreatedb, usesuper, passwd FROM
pg_user;"

        cursor.execute(sql_query)

        print("usename | usecreatedb | usesuper | passwd")

        for row in cursor.fetchall():
            print(row)

# CLOSE DATABASE CONNECTION

connection.close()
```

Finally, run the Python script to see the results of your SQL query output into the terminal. Oh yea!

Now that you know how to use Python to execute a SQL query, practice using Python to manage databases and tables, then populate them and query them.