Python and Databases

Drivers

We saw that PostgreSQL, like many databases, operates in a client-server architecture.

Often, the client is created and run inside a piece of software, such as your program written in Python.

To do that, you need Python **drivers** for your database. These are packages that act as a client for the database, connecting, sending, and retrieving data.

Pyscopg2

The most popular **driver** for PostgreSQL in python is a package called `psycopg2`.

To connect to your database, you will most likely need to install this package. It can be installed with either the `psycopg2` package or the `psycopg2-binary` package (the former expects local C libraries to connect to Postgres, the later comes with all the necessary libraries already).

DB-API

The makers of Python new that connecting to databases was a thing that would be needed.

So they created a uniform API that all drivers could implement.

This has allowed many other packages to be written that work with a variety of SQL databases (not just PostgreSQL). Often we use those packages to connect to the database.

Pyscopg2 - Direct Use

```
with psycopg2.connect(dbname='test', user='postgres') as conn:
    with conn.cursor() as cur:

        cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)",
            (100, "abc'def"))

        cur.execute("SELECT * FROM test;")

        for record in cur:
            print(record)

        # Make the changes to the database persistent
        conn.commit()
```

Using Dataset

https://dataset.readthedocs.io/en/latest/

```
import dataset

db = dataset.connect('postgres://postgres@localhost:5432/postgres')
test_table = db['test']

db.insert({'num': 10, 'data': 'foo'})

recs = db.query("SELECT * FROM test;")

for rec in recs:
    print(rec)
```

Bulk Inserting with Pyscopg2

```python
from psycopg2.extras import execute_batch

dat = [(2, 'foo'), (5, 'bar')]

with psycopg2.connect(dbname='test', user='postgres') as conn:
    with conn.cursor() as cur:
        query = "INSERT INTO test(num, data) VALUES (%s, %s)"
        execute_batch(cur, dat)
```

Bulk Inserting with Dataset

```python
import dataset

db = dataset.connect('postgres://postgres@localhost:5432/postgres')

test_table = db['test']

data = [{'num': 10, 'data': 'foo'}, {'num': 5, 'data': 'bar'}]

test_table.insert_many(data)
```