Web Application Deployment: Heroku

Robert M. Dondero, Ph.D
Princeton University

Objectives

- This lecture will cover:
  - How to deploy a web app and database to Heroku
  - Using PostgreSQL
  - Database connection pooling

Agenda

- **Heroku**
- Defining the app
- Deploying the app (demo)
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Running the App (demo)

Heroku

Orion Henry

Adam Wiggins

James Lindenbaum

Heroku

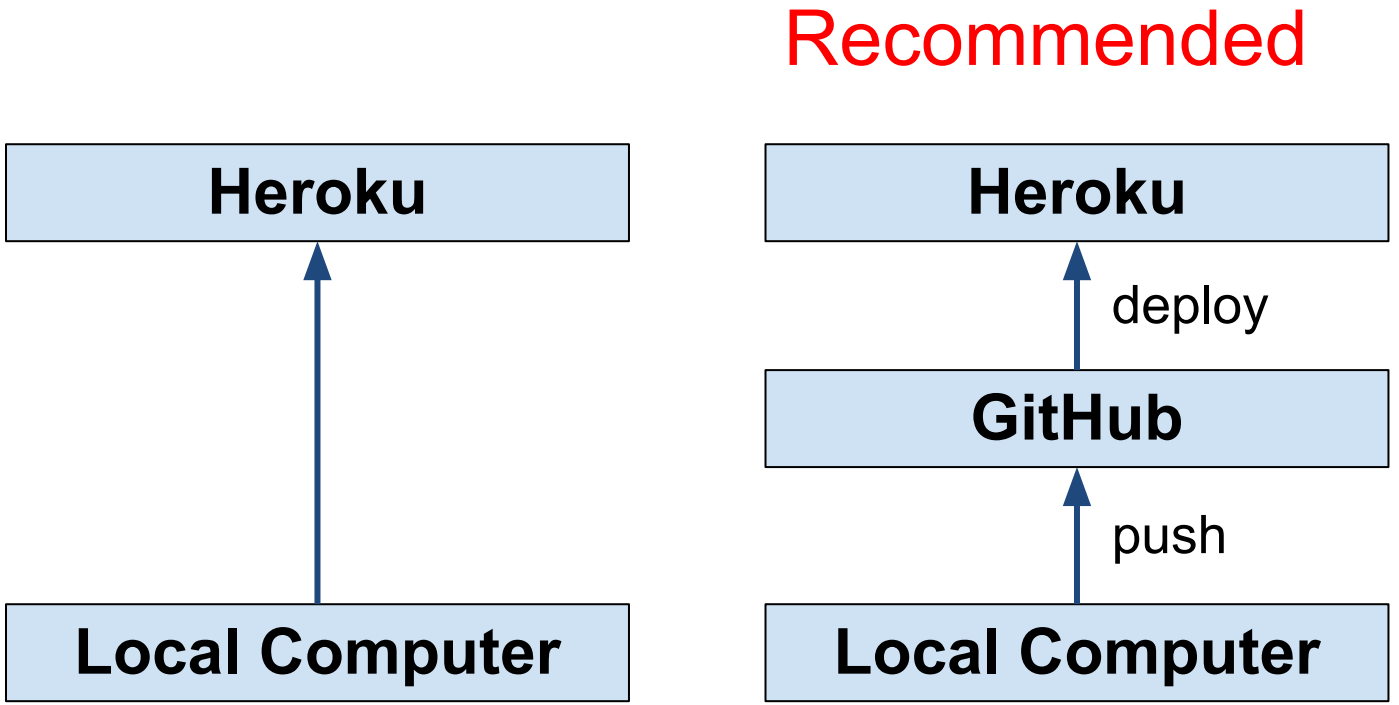
- Render
  - Can create **app** or **DB** first
- Heroku
  - Must create **app** first
  - Then create **DB** specifically for the app

Agenda

- Heroku
- **Defining the app**
- Deploying the app (demo)
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Running the App (demo)

Defining the App

- Deployment



Defining the App

- You must change names as appropriate…

Defining the App

- After performing setup steps in *Git and GitHub Primer* document...

- Create a GitHub repo
  - Browse to https://github.com
    - Click the *New* button
      - For *Repository name* enter `pennyheroku`
      - Click the *Private* radio button
      - Check the *Add a README file* check box
      - Click the *Create Repository* button

Defining the App

- Clone the GitHub repo to my local computer

```
$ git clone https://github.com/rdondero/pennyheroku.git
```

  - Creates `pennyheroku` dir on local computer

Defining the App

- Create these files in pennyheroku dir:
  - From the PennyFlaskJinja app
    - runserver.py (optionally)
    - penny.sql
    - penny.sqlite (optionally)
    - header.html, footer.html, index.html, searchform.html, searchresults.html
    - penny.py
  - Described later in this lecture
    - database.py

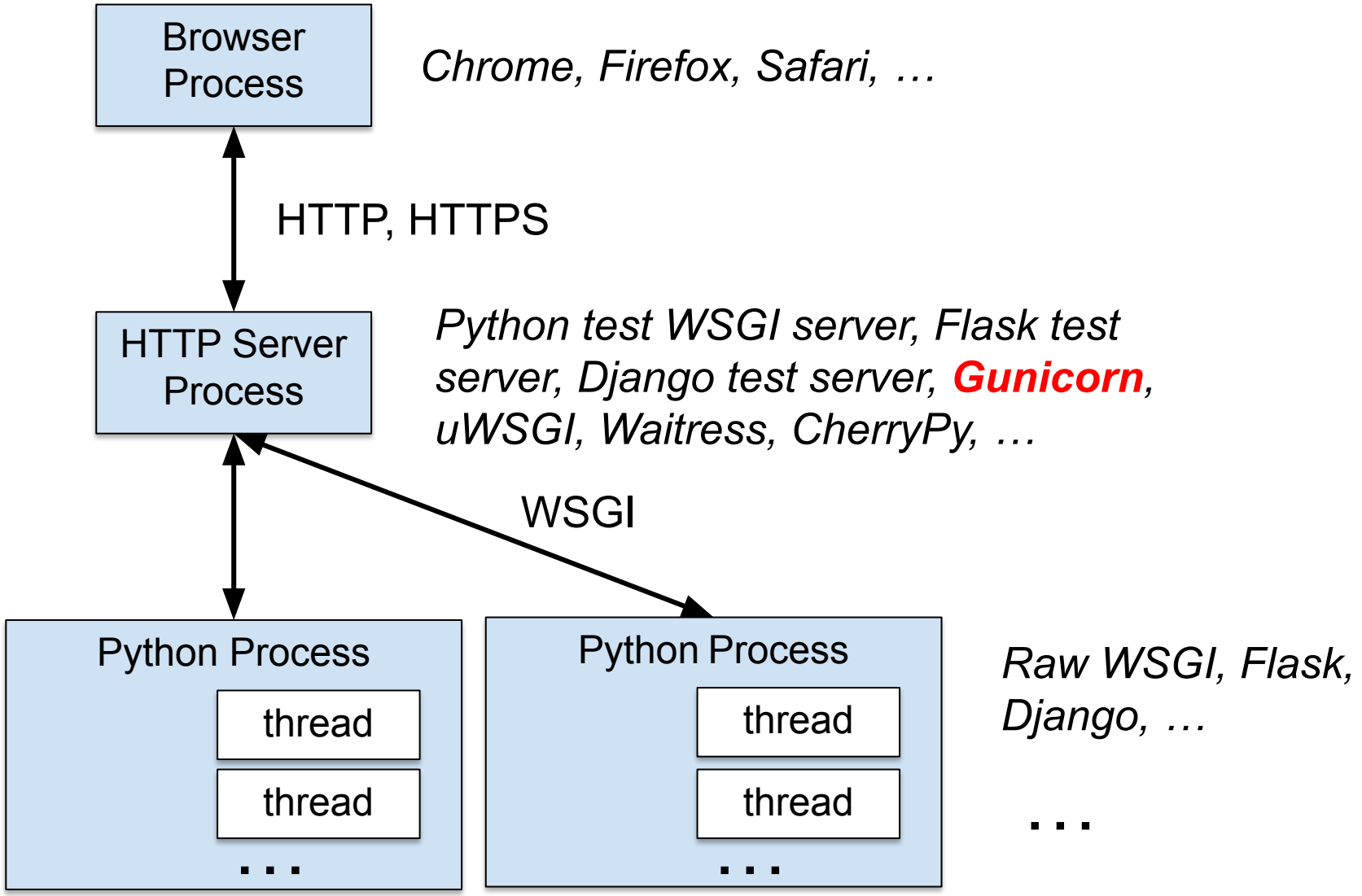
Defining the App

- Also create in pennyheroku dir:
  - **requirements.txt**

```
Flask
psycopg
psycopg-pool
SQLAlchemy
python-dotenv
gunicorn
```

  - Tells Heroku what additional modules the app uses
  - Create manually, or issue command
    `python -m pip freeze > requirements.txt`

Aside: Gunicorn



Defining the App

- Also create in pennyheroku dir:
  - **runtime.txt**

```
python-3.12.3
```

    - Tells Heroku which version of Python it should use

Defining the App

- Also create in pennyheroku dir:
  - **Procfile**

```
web: gunicorn penny:app
```

  - Tells Heroku the app's process type
    - It's a web app
  - Tells Heroku the command to start the process
    - Gunicorn (https://gunicorn.org/)

Defining the App

- Stage the app to the local git repo, commit the app to the local git repo, and push it to the GitHub repo

```
$ cd pennyheroku
$ git add .
$ git commit -m "initial load"
$ git push origin main
```

Agenda

- Heroku
- Defining the app
- **Deploying the app (demo)**
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Running the App (demo)

Deploying the App

- Create a Heroku account
  - Browse to http://signup.heroku.com
    - Follow the instructions
      - Need not provide a credit card number
  - Link your Heroku account to the *GitHub Student Developer Pack*
    - Procedure unknown

Deploying the App

- “Install Heroku” on GitHub
  - Inform GitHub that Heroku is permitted to access the repo
  - Browse to https://github.com/apps/heroku/installations/new
    - Click *rdondero*

Deploying the App

- “Install Heroku” on GitHub (cont.)
  - In the resulting Heroku page
    - Click *Only select repositories*
    - Click *Select repositories*
    - Choose *rdondero/pennyheroku*
    - Click *Save*

Deploying the App

- Log into Heroku
  - Browse to https://id.heroku.com/login
    - For *Email address* enter `rdondero@cs.princeton.edu`
    - For *Password* enter *`yourpassword`*

Deploying the App

- (If necessary) Browse to Heroku dashboard
  - Browse to https://dashboard.heroku.com/apps
  - Click *Dashboard*

Deploying the App

- Create the app
  - In resulting page:
    - Click *New* → *Create new app* button
  - In resulting page:
    - For *App name* enter `pennyheroku`
    - For *Location* choose `United States`
    - Click *Create app* button

Deploying the App

- Configure the app
  - In resulting page:
    - In the *Deployment* method area click *GitHub: Connect to GitHub*
    - In the *Connect to GitHub* area, for *Search for repository to connect to* enter `rdondero` and `pennyheroku`
    - Click on the *Search* button
    - Click on the *Connect* button for rdondero/pennyheroku

Deploying the App

- Configure the app (cont.)
  - In the *Manual deploys* area
    - For *Choose a branch to deploy* select `main`
    - Click the *Deploy Branch* button
    - Observe the build log

Agenda

- Heroku
- Defining the app
- Deploying the app (demo)
- **Creating the DB (demo)**
- Using the DB (demo)
- Using the DB: Python
- Running the App (demo)

Creating the DB

- In the *pennyheroku* page
  - Click on the *Resources* tab
- In the resulting *Resources* page
  - Click on the *Explore Add-ons on Elements Marketplace* link
- In the resulting *Heroku Add-ons* page
  - Click on *Heroku Postgres*

Creating the DB

- In the resulting *Heroku Postgres* page
  - Under *Plans & Pricing* choose `Essential 0`
  - Click on the *Install Add-on* button
- In the resulting *Online Order Form* page
  - As *App to provision* to enter and click on `pennyheroku`
  - Click on *Submit Order Form*
  - Wait a minute or two for Heroku to process the order

Creating the DB

- In the resulting *pennyheroku* page
  - Click on *Settings* tab
  - Click on *Reveal Config Vars*
    - Note value of DATABASE_URL
    - **Save it someplace**
    - Let's call it ***YourHerokuDbUrl***

Creating the DB

- Note:
  - Heroku reserves the right to change *YourHerokuDbUrl* at any time
    - Changes DATABASE_URL in Heroku app, but…
    - Breaks external apps

Agenda

- Heroku
- Defining the app
- Deploying the app (demo)
- Creating the DB (demo)
- **Using the DB (demo)**
- Using the DB: Python
- Running the App (demo)

Using the DB

- From a command-line
  - Install `psql`
    - Without installing PostgreSQL server
    - See https://www.risingwave.dev/docs/current/install-psql-without-postgresql/

Using the DB

- From a command-line (cont.)
  - Install `psql` (Mac)
    - First install homebrew; then:

```
$ brew update
$ brew install libp
$ brew link --force libpq
$
```

Using the DB

- From a command-line (cont.)
  - Install `psql` (MS Windows)
  - Download the installer at https://www.postgresql.org/download/windows/
  - Run the installer
    - Select *Command Line Tools* and uncheck other options during installation

Using the DB

- From a command-line (cont.)
  - Install `psql` (Linux)

```
$ sudo apt update
$ sudo apt install postgresql-client
$
```

Using the DB

Some `psql` statements

sqlite3 Statement	psql Statement
`.help`	`\h`
`.quit`	`\q`
`.tables`	`\d`
`.schema` *table*	`\d` *table*
`.read` *file*	`\i` *file*

Using the DB

```
$ cat penny.sql
DROP TABLE IF EXISTS books;

CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);

INSERT INTO books (isbn, author, title)
   VALUES ('123', 'Kernighan','The Practice of Programming');
INSERT INTO books (isbn, author, title)
   VALUES ('234', 'Kernighan','The C Programming Language');
INSERT INTO books (isbn, author, title)
   VALUES ('345', 'Sedgewick','Algorithms in C');
$
```

Using the DB

```
$ psql YourHerokuDbUrl
yourdbname=> \i penny.sql
DROP TABLE
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
yourdbname=> SELECT * FROM books;
 isbn |   author    |          title
------+-------------+-------------------------------
 123  | Kernighan | The Practice of Programming
 234  | Kernighan | The C Programming Language
 345  | Sedgewick | Algorithms in C
(3 rows)

yourdbname=> \q
$
```

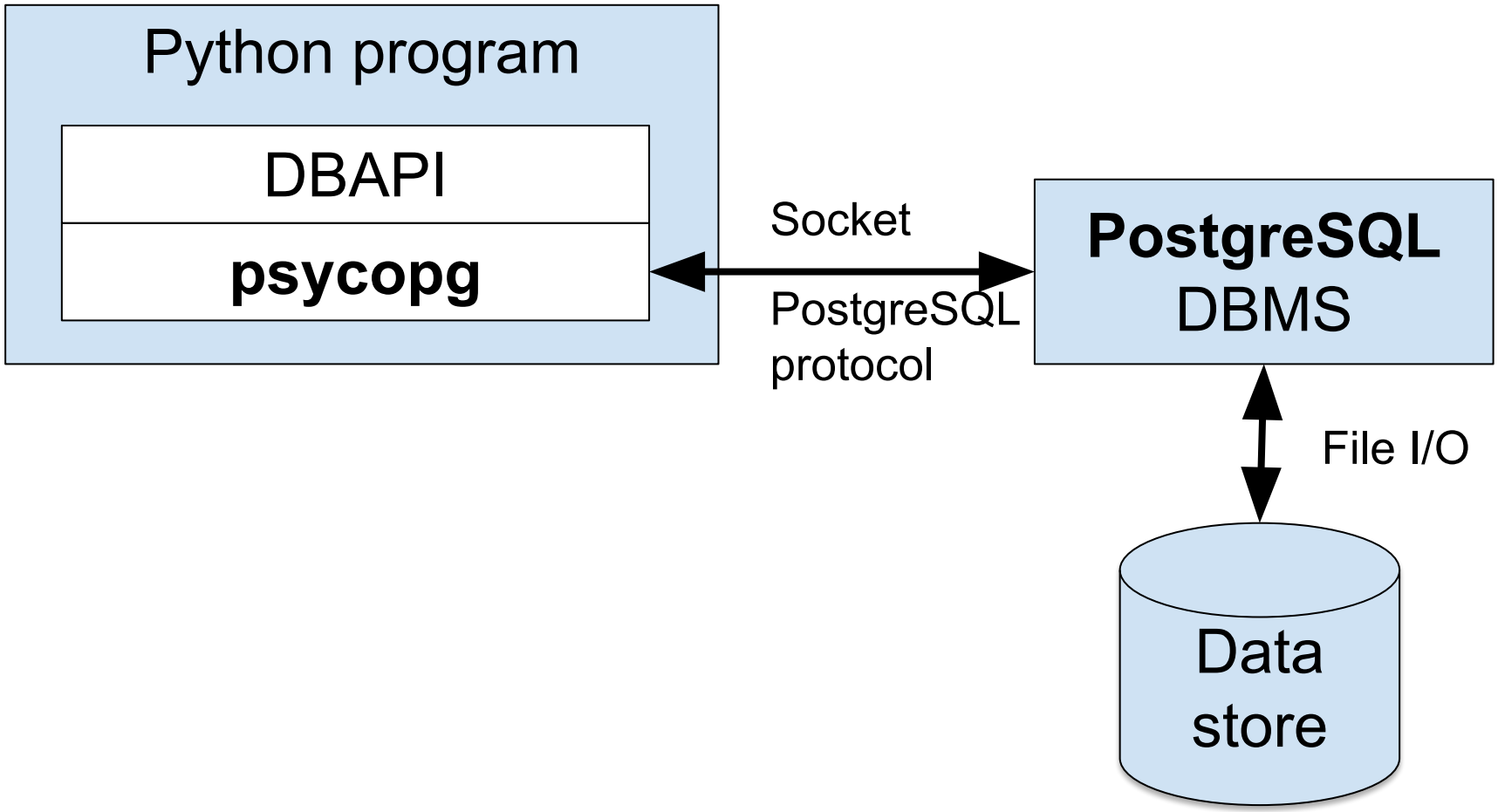
Using the DB

- Graphical PostgreSQL clients:
  - pgAdmin4
  - DBeaver
  - Postico (Mac)
  - …
- List of graphical clients: https://wiki.postgresql.org/wiki/PostgreSQL_Clients

Agenda

- Heroku
- Defining the app
- Deploying the app (demo)
- Creating the DB (demo)
- Using the DB (demo)
- **Using the DB: Python**
- Running the App (demo)

Using the DB: Python



Using the DB: Python

- Installing the `psycopg` driver:

```
$ activate333
$ python -m pip install psycopg
$
```

Using the DB: Python

- See **pennyheroku/databasesqlite.py**
  - Baseline version
  - Uses SQLite

```
$ python databasesqlite.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- See **pennyheroku/databasepostgres1.py**
  - Uses PostgreSQL
  - Uses an environment variable

```
$ export DATABASE_URL=YourHerokuDbUrl
$ python databasepostgres.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Problem**:
  - Creating PostgreSQL DB connections is *very* slow
- **Attempted solution:**
  - One global DB connection

Using the DB: Python

- See **pennyheroku/databasepostgres2bad.py**

```
$ export DATABASE_URL=YourHerokuDbUrl
$ python databasepostgres2bad.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Problem**:
  - Race condition!!!
- **Solution:**
  - ***DB connection pooling***
    - Maintain a “pool” of open DB connections that threads can use

Using the DB: Python

- See **pennyheroku/databasepostgres3.py**

```
$ export DATABASE_URL=YourHerokuDbUrl
$ python databasepostgres3.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Alternative solution**:
  - ***SQLAlchemy***

Using the DB: Python

- ***SQLAlchemy***
  - An ***Object Relational Mapper (ORM)*** for Python
    - Maps each DB table to a Python class
    - Maps each DB row to a Python object
  - Works for many popular relational DBMSs
    - SQLite, PostgreSQL, MySQL and MariaDB, Oracle, Microsoft SQL Server
  - See optional lecture for more thorough description

Using the DB: Python

- Installing SQLAlchemy

```
$ activate333
$ python -m pip install SQLAlchemy
$
```

Using the DB: Python

- See **pennyheroku/database.py**

```
$ export DATABASE_URL=YourHerokuDbUrl
$ python database.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- See **pennyheroku/database.py** (cont.)
  - Bonus:

```
$ export DATABASE_URL="sqlite:///penny.sqlite"
$ python database.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- SQLAlchemy in COS 333 projects?
  - Pros:
    - Hides SQL
    - Portable across many relational DBMSs
    - Automatically uses prepared statements
    - Automatically provides DB connection pooling
    - Allows use of ***Alembic*** (database migration)
    - Harder to make a mistake

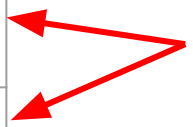
Using the DB: Python

- SQLAlchemy in COS 333 projects?
  - Cons:
    - Hides SQL!!!
    - Must learn SQLAlchemy syntax
    - More complex than SQL
    - Less important than SQL
      - Python only
      - Cannot be used via CLIs (`sqlite3`, `psql`)
    - Poor documentation

Using the DB: Python

For PostgreSQL DB on local computer:

DB SW	Optimization Technique	Time * (sec)	Thread safe?
psycopg	(None)	236.9	Yes
psycopg	One global connection	3.2	No
psycopg	DB connection pooling	6.7	Yes
SQLAlchemy	DB connection pooling	12.2	Yes



Use one of these in your project

* Time to call `get_books()` 10000 times

Aside: PgBouncer

- PostgreSQL connection pool shared by **threads:**
  - Use `psycopg_pool.ConnectionPool`
  - Use SQLAlchemy
- PostgreSQL connection pool shared by **processes**
  - Use ***PgBouncer***
    - Render DB: optional; difficult
    - Neon DB: optional; easy

Agenda

- Heroku
- Defining the app
- Deploying the app (demo)
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- **Running the App** (demo)

Running the App

- Browse to the app!
  - Browse to https://pennyheroku-??????.herokuapp.com/
  - Shortcut: Click on the *Open app* button
  - 

Lecture Summary

- In this lecture we covered:
  - How to deploy a database and web app to Heroku
  - Using PostgreSQL
  - Database connection pooling
- See also:
  - **Optional lecture**: SQLAlchemy
  - **Optional lecture**: PostgreSQL
  - **Optional lecture:** Web Application Deployment: Render