(Bottom Half of Screen)

Intro to PL/Python

Josh Williams

What is PL/Python?

- Procedural Language INSIDE PostgreSQL
- Not a replacement for SQL
- Run manually (functions / DO in 9.0)
  - or automatically (triggers)

Why PostgreSQL?

- Powerful, yet very fast.
- Feature rich, but you can't beat the price.
- It's future is clear!

Why Python?

- Easy to read. Easy to learn. Easy to use.
- Powerful, yet very fast.
- Possibly already using it outside the database.
- It's Python!!

Okay, enough crowd pandering

- Step 1: Configure PostgreSQL for Python
- Step 2: Install PL/Python into database(s)
- Step 3: Profit

1. Configure PG for Python

- If using source, `./configure --with-python`
- If using packages, install ”-plpython” or ”-py...”
- In either case you need Python installed (duh)

Is it there?

- SELECT * FROM pg_pltemplate;
- tmplname == plpythonu
  - or plpython2u
  - or plpython3u

Wait, u?

- PL/Python is an untrusted language...
- Only DB superusers can create functions :(
- Functions can communicate outside the DB :)

2. Enable Databases

- Command line: createlang plpythonu *dbname*
- Inside DB: CREATE LANGUAGE plpythonu;
  - Hint: Install in template1
- SELECT * FROM pg_language;

CREATE FUNCTION...

```
CREATE FUNCTION function_name (....)
  AS $dollarquote$ ...python here...
  ... $dollarquote$ LANGUAGE plpythonu;
```

- See ”Intro to PL/pgSQL” for construct details

The `...python here...` bit

- Where the interesting Python-y stuff happens
- Simple processing of function parameters
  - But also... Import modules (stats, XML, ASCII plot?)
  - Interact with the outside world, the inside world

The web IN your database redux

- Need a type to represent composite RSS data
- Need RSS reader module … feedparser!
- CREATE FUNCTION … RETURNS SETOF …

It works!

- But 11 lines? This is Python, we can do better!
- There … Two lines (plus the module import)

Something a little more fancy

- Create two really basic functions:

```
CREATE FUNCTION pydir(text) RETURNS SETOF text...
CREATE FUNCTION isdir(text) RETURNS boolean...
```

- Call SRF using just SELECT pydir('/')

Enter Recursive Queries (8.4+)

- CTE construct: WITH RECURSIVE …
- Initial value(s) UNION recursive query
- Then SELECT … FROM *cte*

But that's usual Python stuff

- PL/Python implicitly includes module ”plpy”
  - Execute SQL statements
  - Generates messages/raises errors back to PG

Last contrived example

- Basic personal finance application
- Add a trigger: Net balance must be positive

Two Step Process

- CREATE FUNCTION … RETURNS trigger
- CREATE TRIGGER accounts_check AFTER
  - INSERT OR UPDATE ON accounts FOR EACH
  - STATEMENT EXECUTE PROCEDURE accheck();

Boooring

- Wait, we're using Python. We can do more!
- How about an early warning email?
  - See package smtplib

That's PL/Python

Any questions?

Intro to PL/Python

http://joshwilliams.name/plpython/ (Now!)

In the mean time: joshwilliams@ij.net