Digital Career Institute

Python Course - Databases - 3.3 Usage in Python

Digital Career Institute

Goal of the Submodule

By the end of this sub-module, the learners should be able to understand:

- *Connecting to database.*
- *CREATE, READ, UPDATE, DELETE (CRUD) operations.*
- *Pool (cache) connections to database.*
- *Conceptualize database connections in Django.*

Topics

- **Postgres in Python.**
  - Postgres and Python.
- **Introducing `Psycopg2`**
  - Installing **`Psycopg2`.**
  - Use Psycopg2 API to access PostgreSQL databases.
- **Perform data CRUD through Python.**
- **Mapping between Python and Postgres types.**
  - Type conversion from Postgres to Python.
  - Constant and numeric conversion.
- **`Pycopg2` Exceptions.**
- **Connection pooling.**
  - What is Connection pool?
  - Psycopg2's Connection pooling classes.
  - Methods to manage PostgreSQL connection Pool.
- **Django and Postgres**
  - Conceptualizing Django Models and Migrations.

Digital Career Institute

Glossary

Term	Definition
PY DB API Spec v2.0	API design specification to encourage and maintain the similarity between the Python database modules to access databases.
Error	An error is an action that is incorrect or inaccurate.
Built-in method	A ready made functions to be used in Python.

Postgres in Python

Digital Career Institute

Databases in Python

Python has support for working with **databases** via a simple **interface**.

Modules included with Python modules for **SQLite** and **Berkeley DB**.

Modules for **MySQL**, **PostgreSQL**, **FirebirdSQL** and others are available as **third-party modules**.

These modules are based on the [PEP 249 -- Python DB API 2.0]

Native support

Third-party support

Postgres in Python

To communicate with PostgreSQL using Python you need **a Postgres adapter/connector** *(two words for same thing).*

Postgres Python connectors enable Python to access Postgres databases.

The connectors use an API which is compliant with the **PEP249 specification.**

Common Connectors

Connector	License	Platforms	Python versions
Psycopg	LGPL	any (pure Python)	2.4-3.2
PyGreSQL	BSD	any (pure Python)	2.3-2.6
ocpgdb	BSD	Unix based systems	2.3-2.6
py-postgresql	BSD	any (pure Python)	3.0+
bpgsql	LGPL	any (pure Python)	2.3-2.6
pg8000	BSD	any (pure Python)	2.5+ / 3.0+

Introducing Psycopg2

- **Psycopg2** is the most popular and stable Postgres adapter for the Python programming language.
- At its core, it fully implements the Python **DB API 2.0 specifications**.
- **Psycopg2** is Open Source under the terms of the GNU Lesser General Public License, allowing use of both free and proprietary software.

Psycopg2 is:

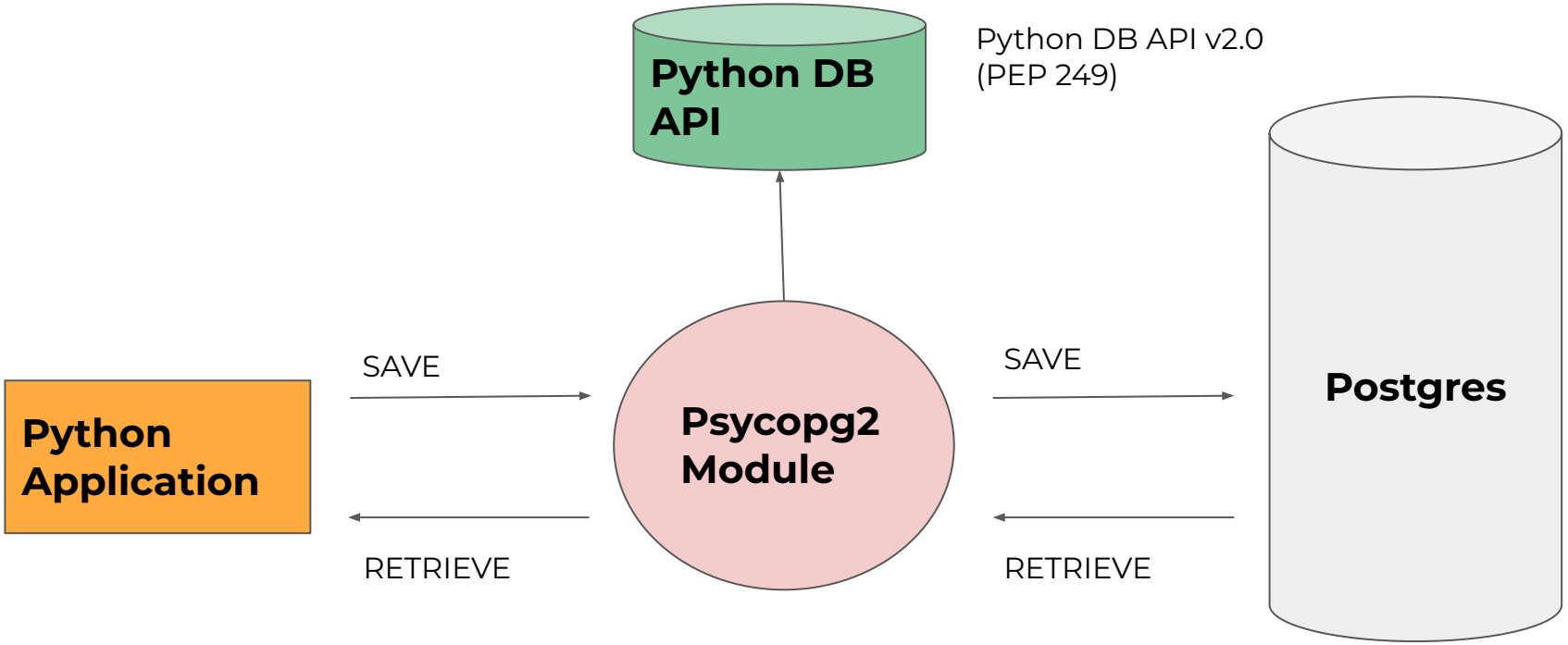
1. *Used in most of Python frameworks.*
2. *Actively maintained.*
3. *Able to take SQL queries.*

Pyscopg2 Specifics

Psycopg is written mostly in C and wraps the libpq library with the result of being both fast and secure.

- Supports Python versions from 2.5 to 3.4.
- Supports Postgres versions from 7.4 to 9.4.
- Thread-safe: threads can use different connections or share the same connection.
- Asynchronous nonblocking I/O also integrated with coroutine-based libraries.
- Adaptation of many Python objects to database type: tuples to records, lists to an array dictionaries to hstore, flexible JSON support.
- Extendible with new adapters to convert Python objects to SQL syntax and type casters to convert PostgreSQL types back into Python objects.

Psycopg2 and Python Overview



Installing Psycopg2

To install Psycopg2, just open your terminal and run:

```
$ pip3 install psycopg2
```

The current psycopg2 module supports:

- Python version 2.
- PostgreSQL server versions from 7.4 to 13.3,
- PostgreSQL client library version from 9.1.

Psycopg2 - Connecting to Database

You can create new connections using the `connect()` function.

You need to know the following detail of the PostgreSQL server to perform the connection.

- **Username** *(username for the PostgreSQL db).*
- **Password.**
- **Host Name** *(This is the server name or IP address on which PostgreSQL is running).*
- **Database Name.**

Psycopg2 - Connect to Database

Example

```
import psycopg2

# Connect to an existing database
 connection =   psycopg2.connect(user="dci",
                                 password="dci@21",
                                 host="127.0.0.1",
                                 port="5432",
                                 database="dci_test")
```

Psycopg2 - Creating a Table

In Psycopg2, the Python command to open a cursor to perform database operations:

```
cur = conn.cursor()
```

We can create as many cursors as we want from a single connection object. Cursors created from the same connection are not isolated, i.e., any changes done to the database by a cursor are immediately visible by the other cursors.

Psycopg2 - Creating a Table

And then pass the CREATE TABLE query to the `cursor.execute()`.

```
cur = conn.cursor()

cur.execute("CREATE TABLE test(id serial PRIMARY KEY, sname
CHAR(50), roll_num integer);")
conn.commit()
conn.close()
```

The `execute()` method could take any SQL query.

Recap

```
import psycopg2

# Connect to an existing database
 connection =   psycopg2.connect(user="dci",
                                 password="dci@21",
                                 host="127.0.0.1",
                                 port="5432",
                                 database="dci_test")
# Create cursor
cur = conn.cursor()

# Create table
cur.execute("CREATE TABLE test(id serial PRIMARY KEY, sname CHAR(50), roll_num
integer);")
conn.commit()
conn.close()
```

CRUD Operations (in Psycopg2)

CRUD

CRUD = **C**reate, **R**ead, **U**pdate **D**elete.

4 basic operation of persistent storage.

Persistent storage = *any data storage that retains data after power to that device is shut off.*

CRUD	SQL
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

CRUD

- *Create*, or add new entries.
- *Read*, retrieve, search, or view existing entries.
- *Update*, or edit existing entries.
- *Delete*, deactivate, or remove existing entries.

Well written CRUD operations form the core of any persistent web application.

CRUD Example

```
#Example CREATE
cur.execute("INSERT INTO test (id, sname, roll_num) \
      VALUES (10, 'Sara', 3)");

#Example READ
cur.execute("SELECT id, sname, roll_num from test")
rows = cur.fetchall();

#Example UPDATE
cur.execute("UPDATE test set roll_num = 3 where ID=10")

#Example DELETE
cur.execute("DELETE from test where ID=10;")
```

CRUD in HTTP

The acronym CRUD also appears in discussion of RESTful APIs. Each letter in the acronym may be mapped to a Hypertext Transfer Protocol (HTTP) method:

CRUD	HTTP
Create	POST
Read	GET
Update	PUT
Delete	DELETE

Mapping between Python and Postgres Types

Python and Postgres Types

There is default mapping specified to convert Python types into PostgreSQL equivalent, and vice versa.

Whenever you execute a PostgreSQL query using Python table[1 in the next slide] is used by psycopg2 to return the result in the form of Python objects.

[1] https://www.postgresql.org/docs/9.4/plpython-data.html

Python and Postgres Types

Python	PostgreSQL
None	NULL
bool	bool
float	real or double
int	smallint integer bigint
Decimal	numeric
str	varchar text
date	date
time	time timetz

Python and Postgres Types

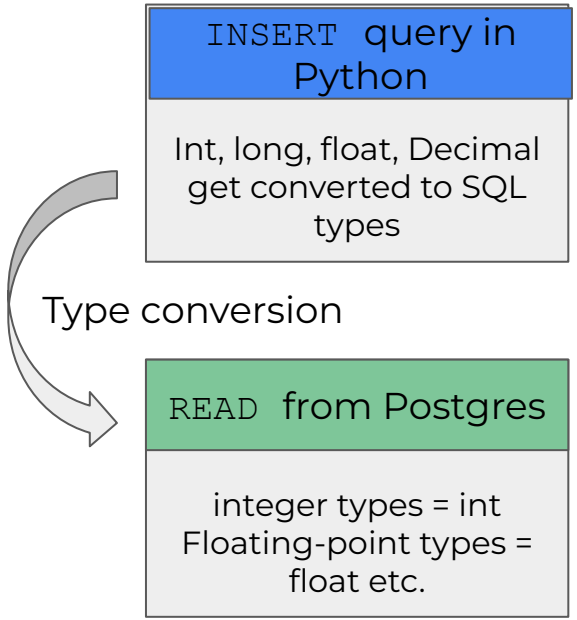
datetime	timestamp timestamptz
timedelta	interval
list	ARRAY
tuple	Composite types IN syntax
dict	hstore

Constant and Numeric Conversions

When you try to insert Python `None` and `boolean` values into Postgres, the respective value gets converted into the proper SQL literals. The same is true with Python numeric types. Numeric types get converted into equivalent PostgreSQL types.

⚠️

Example



Psycopg2 Exceptions

Psycopg2 `Exceptions` Module

- Psycopg2 comes with a few built-in exceptions to help debug programs [1].

- Two most commonly occurring exceptions in psycopg2 are `OperationalError` and `ProgrammingError` exception classes.

[1] https://www.postgresql.org/docs/9.2/errcodes-appendix.html

Psycopg2 `Exceptions` Module

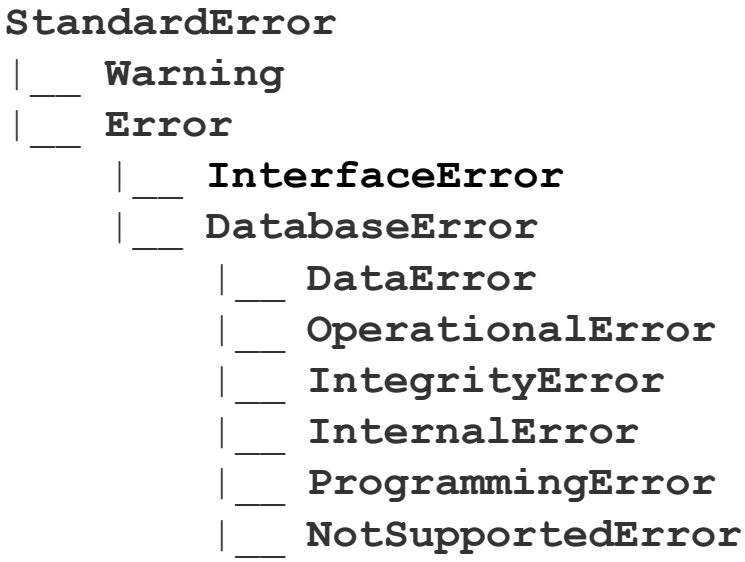
An **`OperationalError`** typically occurs when the parameters passed to the **`connect()`** method are incorrect, or if the server runs out of memory, or if a piece of datum cannot be found, etc.

A **`ProgrammingError`** happens when there is a syntax error in the SQL statement string passed to the **`execute()`** method, or if a SQL statement is executed to delete a non-existent table, or an attempt is made to create a table that already exists.

Other Psycopg2 `Exceptions`

Exception Class	Function
`InterfaceError`	Raised for errors that are related to the database interface rather than the database itself.
`DatabaseError`	Raised for errors that are related to the database.
`DataError`	Raised for errors that are due to problems with the processed data like division by zero, numeric value out of range, etc.
`OperationalError`	Raised for errors that are related to the database's operation and not necessarily under the control of the programmer.
`IntegrityError`	Raised when the relational integrity of the database is affected.
`InternalError`	Raised when the database encounters an internal error.
`ProgrammingError`	Raised for programming errors.
`NotSupportedError`	Raised in case a method or db API was used which is not supported by the database.

Psycopg2 `Exceptions` Hierarchy



At the core of the lesson

- Postgres can be used with Python using adapters.
- Psycopg2 is the most common and stable adapter.
- Psycopg2 allows to write SQL queries right in the Python code.
- CRUD operations are the backbone of a persistent application.