Bootcamp
Data Engineering

Day00
psycopg2 documentation

Psycopg2 basics

Psycopg is a very popular PostgreSQL database adapter for the Python programming language. Its full documentation can be seen **here**.

The function `connect()` creates a new database session and returns a new connection instance.

```
import psycopg2

def get_connection():
conn = psycopg2.connect(
database="appstore_games",
host="localhost",
user="postgres_user",
password="12345"
)
return (conn)
```

Cursors allows Python code to execute PostgreSQL command in a database session.

```
curr = conn.cursor()
```

Tables can be created with the cursor.

```
curr.execute("""CREATE TABLE members (
id serial PRIMARY KEY,
firstname varchar(32),
lastname varchar(32),
birthdate date
)
""")
```

It's also possible to remove a table.

```
curr.execute("DROP TABLE members")
```

To make changes persistent in the database, we need to commit (queries are called transactions). Finally, we can close the connection.

```
conn.commit()
conn.close()
```

This gives the following full code.

```
import psycopg2

def get_connection():
conn = psycopg2.connect(
database="appstore_games",
host="localhost",
user="postgres_user",
password="12345"
)
return (conn)

if __name__ == "__main__":
conn = get_connection()
curr = conn.cursor()
curr.execute("""CREATE TABLE members (
id serial PRIMARY KEY,
firstname varchar(32),
lastname varchar(32),
birthdate date
)
""")
conn.commit()
conn.close()
```

Inserting data

Data can be inserted into a table with the following syntax.

```
curr.execute("""
INSERT INTO members(firstname, lastname, birthdate) VALUES
('Eric', 'Clapton', '1945-13-30'),
('Joe', 'Bonamassa', '1977-05-08')
""")
```

Delete data

Data can also be deleted.

```
curr.execute("""DELETE FROM members
WHERE lastname LIKE 'Clapton'
""")
```

Useful functions

get connections

```
def get_connection():
conn = psycopg2.connect(
database="appstore_games",
host="localhost",
user="postgres_user",
password="12345"
)
return (conn)
```

Showing table content

We must use the `fetchall` function to gather all the result in a list of tuples.

```
def display_table(table: str):
conn = set_connection()
curr = conn.cursor()
curr.execute("""SELECT * FROM %(table)s
LIMIT 10""", {"table": AsIs(table)})
response = curr.fetchall()
for row in response:
print(row)
conn.close()
```

Create a table

```
def create_table():
conn = get_connection()
curr = conn.cursor()
curr.execute("""CREATE TABLE test(
FirstName varchar PRIMARY KEY,
LastName varchar,
Age int
);""")
conn.commit()
conn.close()
```

Drop table

```
def delete_table(table: str):
conn = set_connection()
curr = conn.cursor()
curr.execute("DROP TABLE IF EXISTS %(table)s;", {"table": AsIs(table)})
conn.commit()
conn.close()
```

Inserting data into a table

```
def populate_table():
conn = get_connection()
curr = conn.cursor()
curr.execute("""INSERT INTO test
(FirstName,
LastName,
Age) VALUES
(%s, %s, %s)""",
('Michelle',
'Dupont',
'33'))
conn.commit()
conn.close()
```