POSTGRESQL TUTORIAL

PostgreSQL Python: Connect to PostgreSQL Database Server

Summary: in this tutorial, you will learn how to connect to the PostgreSQL server in Python using the `psycopg2` package.

Neon Postgres – Fully Automated, Generous Free Tier, Instant DBs, Autoscaling, Git-like Branching - Start Free

Sponsored

Creating a virtual environment

First, open the Command Prompt on Windows or Terminal on Unix-like systems.

Second, create a new directory to store the project files such as `suppliers` :

```
mkdir suppliers
```

Third, create a new virtual environment called `venv` using the built-in `venv` module:

```
python -m venv venv
```

Finally, activate the virtual environment on Windows:

```
venv/scripts/activate
```

on Unix-like systems:

```
source venv/bin/activate
```

Installing the psycopg2 module

First, install the `psycopg2` package using the following `pip` command:

```
pip install psycopg2
```

Second, create the `requirements.txt` file:

```
pip freeze > requirements.txt
```

Creating a new database

First, connect to the PostgreSQL server using the `psql` client tool:

```
psql -U postgres
```

Second, create a new database called `suppliers` :

```
CREATE DATABASE suppliers;
```

Third, exit the `psql` :

```
exit
```

Connecting to the PostgreSQL database from Python

First, create a configuration file called `database.ini` in the project directory to store database connection parameters:

```
[postgresql]
host=localhost
database=suppliers
user=YourUsername
password=YourPassword
```

In the `database.ini` file, you need to replace the YourUsername and YourPassword with the real ones.

Second, create a new file called `config.py` in the project directory and define a function called `load_config()` that reads configuration data from the `database.ini` file:

```
from configparser import ConfigParser

def load_config(filename='database.ini', section='postgresql'):
    parser = ConfigParser()
    parser.read(filename)

    # get section, default to postgresql
    config = {}
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            config[param[0]] = param[1]
    else:
        raise Exception('Section {0} not found in the {1} file'.format(section, filen

    return config

if __name__ == '__main__':
    config = load_config()
    print(config)
```

The config.py module uses the built-in `configparser` module to read data from the database.ini file.

By using the `database.ini` , you can change the PostgreSQL connection parameters when moving the code to the production environment.

Notice that if you git source control, you need to add the `database.ini` to the `.gitignore` file to not commit sensitive information to a public repository like GitHub.

Third, create a new file called `connect.py` that uses the `config.py` module to read the configuration and connect to the PostgreSQL:

```
import psycopg2
from config import load_config

def connect(config):
    """ Connect to the PostgreSQL database server """
    try:
        # connecting to the PostgreSQL server
        with psycopg2.connect(**config) as conn:
            print('Connected to the PostgreSQL server.')
            return conn
    except (psycopg2.DatabaseError, Exception) as error:
        print(error)


if __name__ == '__main__':
    config = load_config()
    connect(config)
```

To connect to the `suppliers` database, you use the `connect()` function of the `psycopg2` module.

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

To call the `connect()` function, you specify the PostgreSQL database parameters as a connection string and pass it to the function like this:

```
conn = psycopg2.connect("dbname=suppliers user=YourUsername password=YourPassword")
```

Alternatively, you can use keyword arguments:

```
conn = psycopg2.connect(
    host="localhost",
    database="suppliers",
    user="YourUsername",
    password="YourPassword"
)
```

The following is the list of the connection parameters:

- `database` : the name of the database that you want to connect.
- `user` : the username used to authenticate.
- `password` : password used to authenticate.
- `host` : database server address e.g., localhost or an IP address.
- `port` : the port number that defaults to 5432 if it is not provided.

Since we use the `config.py` module, we can pass the configuration to the `connect()` function and unpack it using the `**` operator:

```
with psycopg2.connect(**config) as conn:
```

The `with` statement automatically closes the connection object so you don't have to call the `close()` method explicitly.

Executing the connect.py module

To execute the `connect.py` file, you use the following command:

```
python connect.py
```

Output:

```
Connected to the PostgreSQL server.
```

The output indicates that you have successfully connected to the PostgreSQL server.

Download the project source code

Summary

- Use the `psycopg2` package to connect to the PostgreSQL server from Python.
- Call the `connect()` function of the `psycopg2` module to connect to the PostgreSQL server.