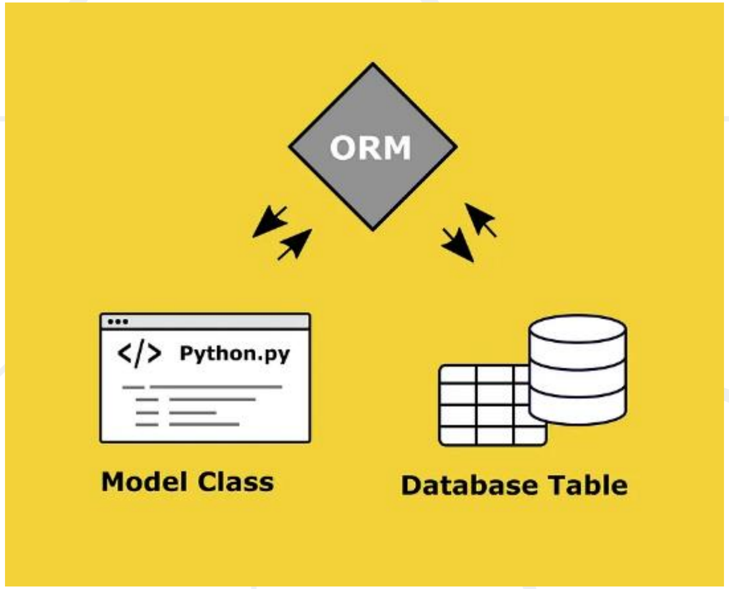
ORM Introduction

Overview, Drivers, Django ORM



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

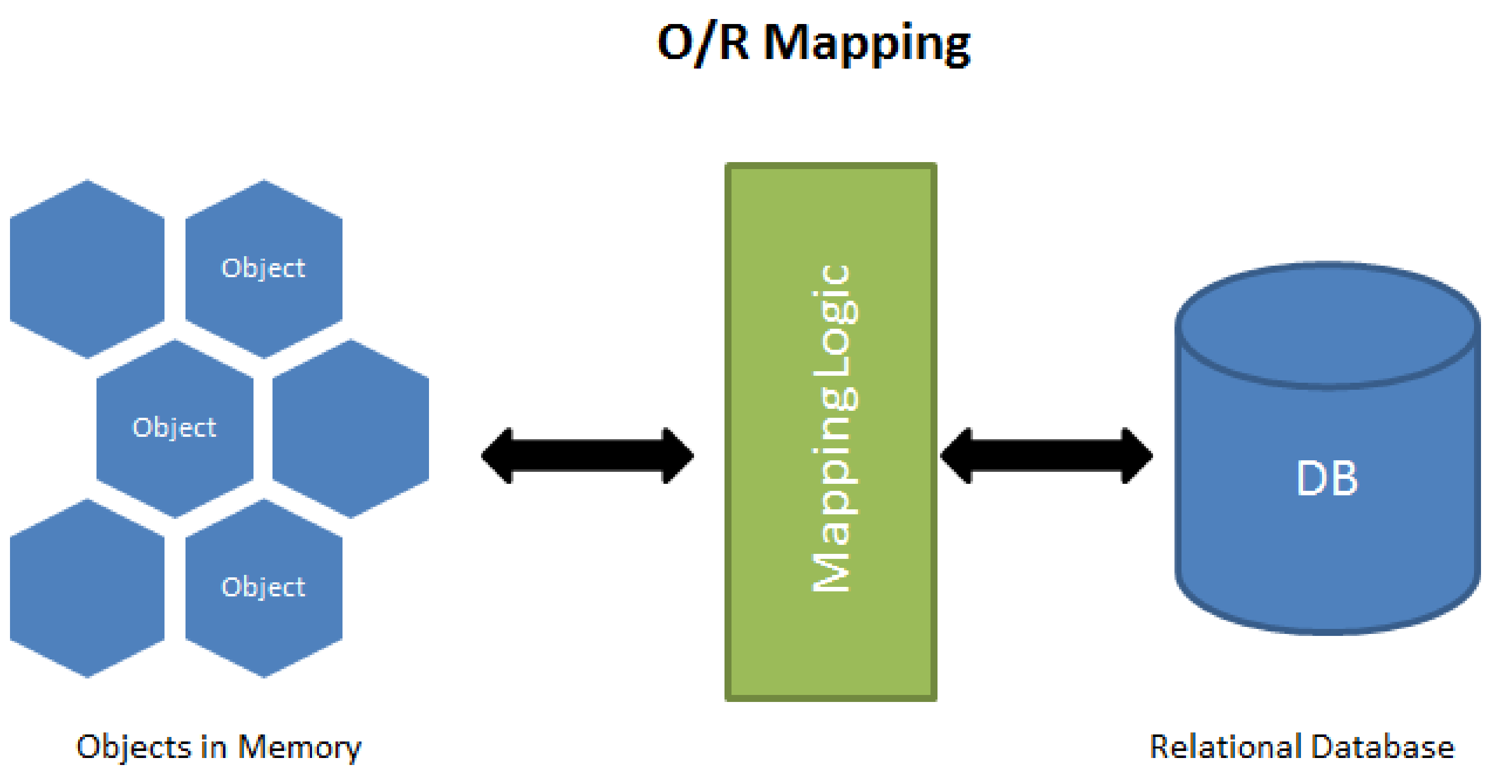
https://about.softuni.bg/

Table of Contents

Questions

sli.do

#python-db

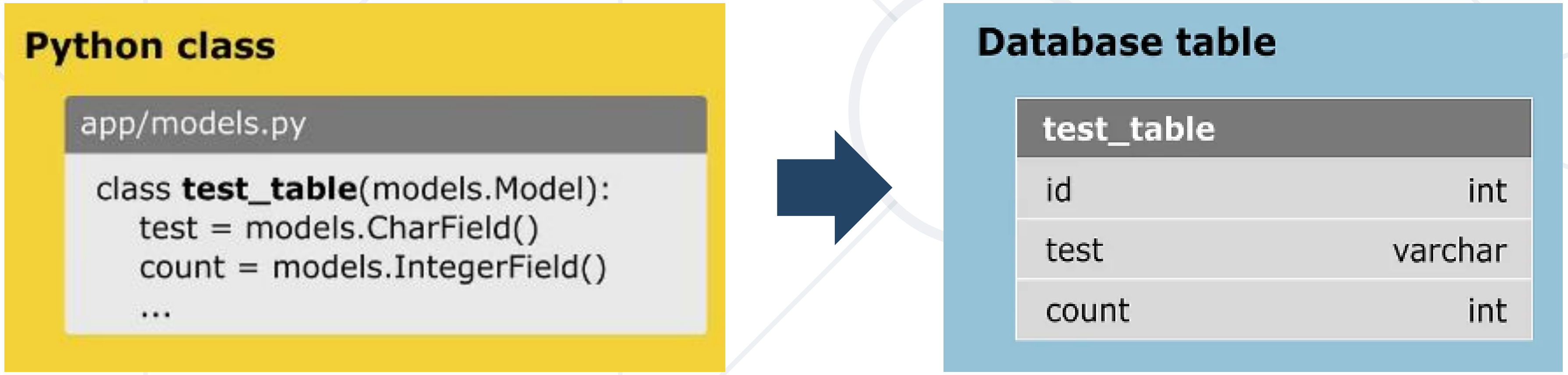


Introduction to ORM

Object-Relational Mapping

What is ORM?

- **Object-Relational Mapping** (ORM) allows manipulating databases using common **classes** and **objects**
  - **Python Classes → Database Tables**



Python class

app/models.py

```
class test_table(models.Model):
    test = models.CharField()
    count = models.IntegerField()
    ...
```

Database table

test_table	
id	int
test	varchar
count	int

Benefits of ORM

- Enables you to write code in your preferred **OOP language**
- Provides **a layer of abstraction** that **hides the complexity** of the database schema and relationships
- Can **automate** some common tasks:
  - Creating, updating, and deleting records
  - Validating data
  - Managing transactions and connections

Drawbacks of ORM

- When it comes to **complex queries** and **aggregations** that require high performance and flexibility
  - Can generate **inefficient** or **suboptimal** SQL queries
    - Poor performance
    - Excessive memory usage
    - Unexpected errors
- Limits **control** and **customization** over the SQL queries and operations
  - Harder to optimize
  - May not support some advanced features or functions

SQL Injection

- **SQL Injection** is a type of an **injection attack**
  - Makes it possible to execute **malicious** SQL statements
  - Attackers can use **SQL Injection vulnerabilities** to bypass application security measures so they could:
    - go around authentication and authorization
    - retrieve the content of the entire database
    - add, modify, and delete records in the database
    - gain unauthorized access to sensitive data
- ORM **reduces** explicit SQL queries and is much **less vulnerable** to SQL injection

SQL Injection Example

- Input fields vulnerable to **SQL Injection**

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']
# SQL query vulnerable to SQLi
SELECT id FROM users WHERE username='" + uname + "' AND
password='" + passwd + "'
# Set the passwd field
password' OR 1=1
# Database server runs the following SQL query
SELECT id FROM users WHERE username='username' AND
password='password' OR 1=1'
```

ORM Advantages and Disadvantages

Advantages

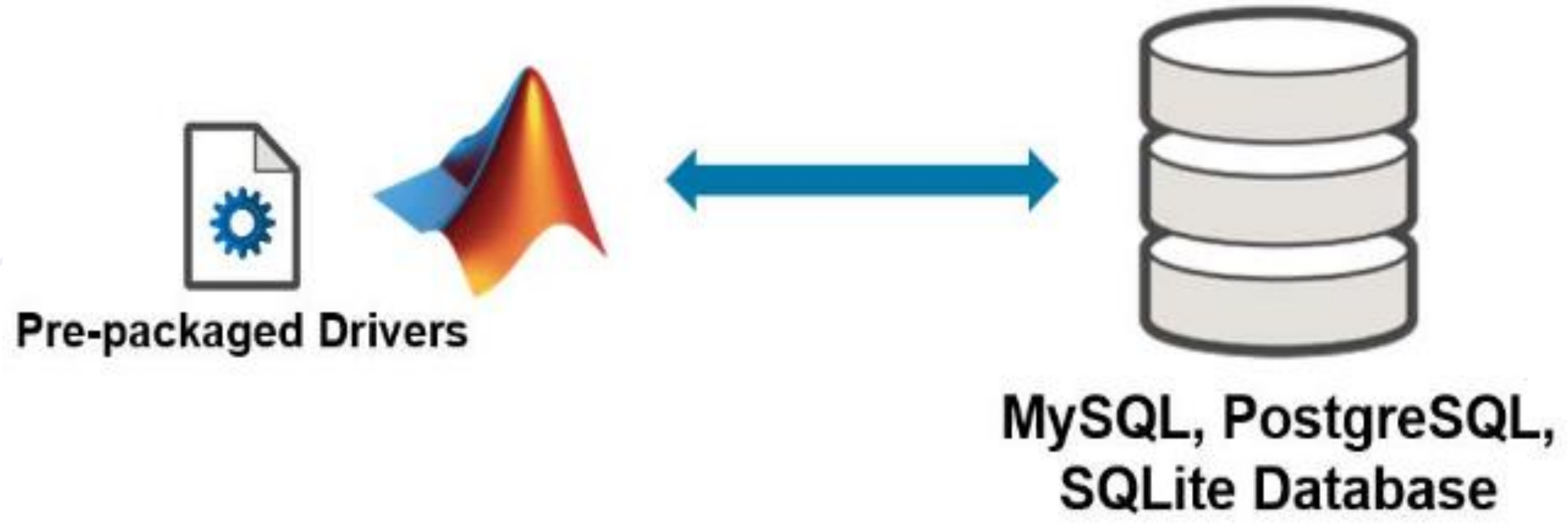
- Developer productivity – **speeds up** development
- **SQL injection** is a lot more difficult
- ORMs work very well with **CRUD**
- Better **readability**

Disadvantages

- **Reduced performance** (N+1 issue or autogenerated SQL)
- **Reduces flexibility** (some operations are hard to implement)

Popular ORM Tools for Python

- **Django**
  - A great tool for building web applications rapidly
- **web2py**
  - An open source full-stack Python framework
- **SQLObject**
  - An ORM that provides an object interface to your database
- **SQLAlchemy**
  - Provides persistence patterns designed for efficient and high-performing database access

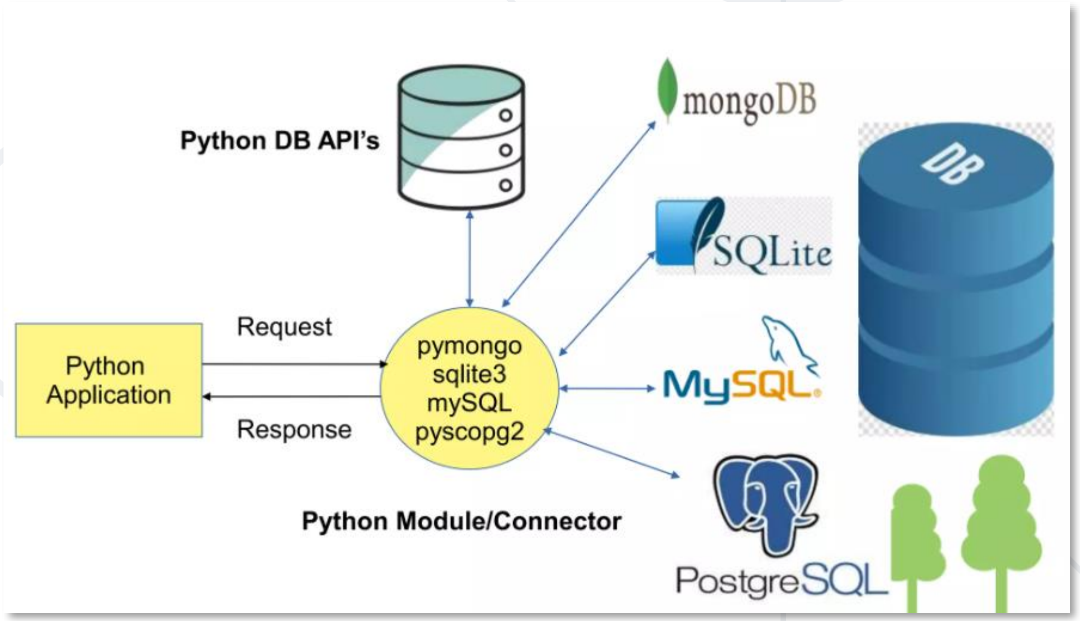


Database Drivers

Psycopg2

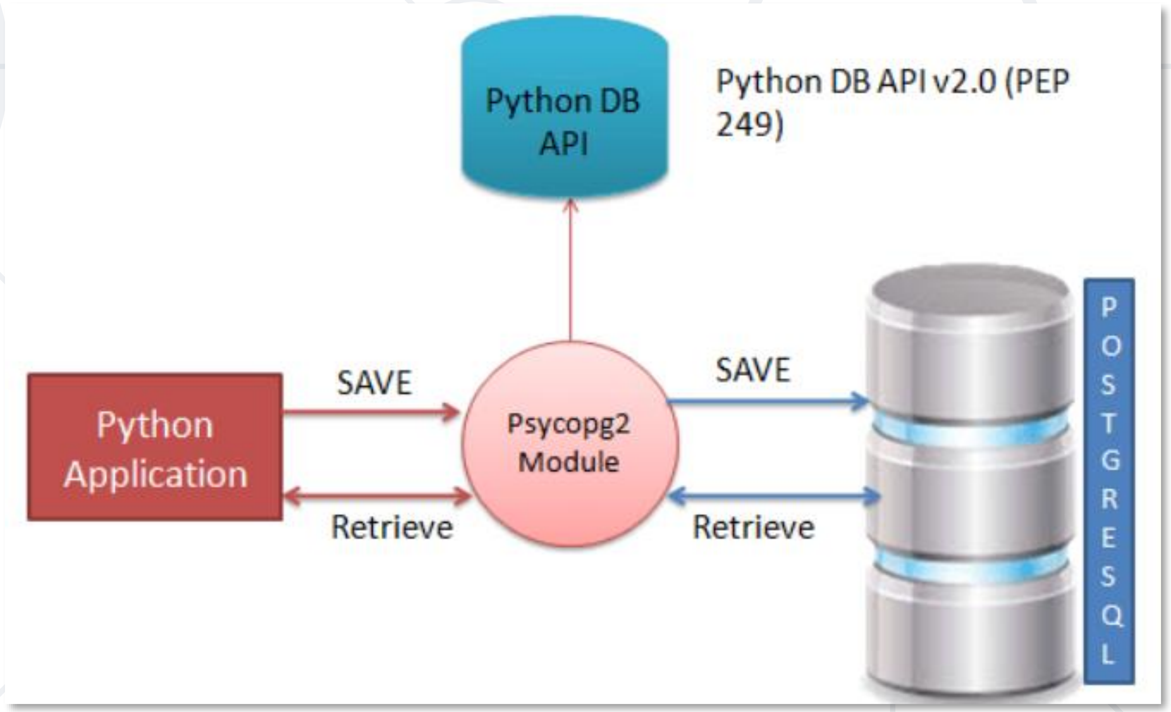
Database Drivers

- **Database Driver** (module/connector) is a computer program that implements a protocol for a database connection
  - Works like an **adapter** that connects a generic interface to a specific database vendor implementation
  - Accesses the physical data through a stand-alone engine
  - Submits SQL statements to and retrieves results from the engine



Psycopg2

- PostgreSQL database **adapter** for Python programming language
- Use Psycopg2 module to:
  - **Connect** to PostgreSQL
  - Perform SQL **queries** and database **operations**
- Psycopg2 is an **external** module



Psycopg2 Advantages

- **Preferred** module for PostgreSQL interaction
- **Trusted** by Python and PostgreSQL frameworks
- Actively **maintained**
- Fully **supports** Python's primary versions, ensuring seamless integration
- **Thread-safe design** built to ensure safe and reliable operation
- Designed to **handle heavy multi-threaded** applications

Install Psycopg2

- You need to install the current version of **Psycopg2** on your machine to use PostgreSQL from Python
- Using the following **pip command**, you can install Psycopg2 on **any operating system**

```
pip install psycopg2
```

Django & Django ORM

Django Project, Django App

What is Framework?

- Platform for **developing software applications**
- Provides a **foundation** on which software developers can **build programs** for a specific platform
- A framework **includes an API**
- May include **code libraries**, a **compiler**, and other programs **used in the software development process**

What is Django Framework?

- **High-level** Python Web Framework
- Ridiculously **fast**
- Reassuringly **secure**
- Exceedingly **scalable**
- **Free** and **Open Source**

Django ORM

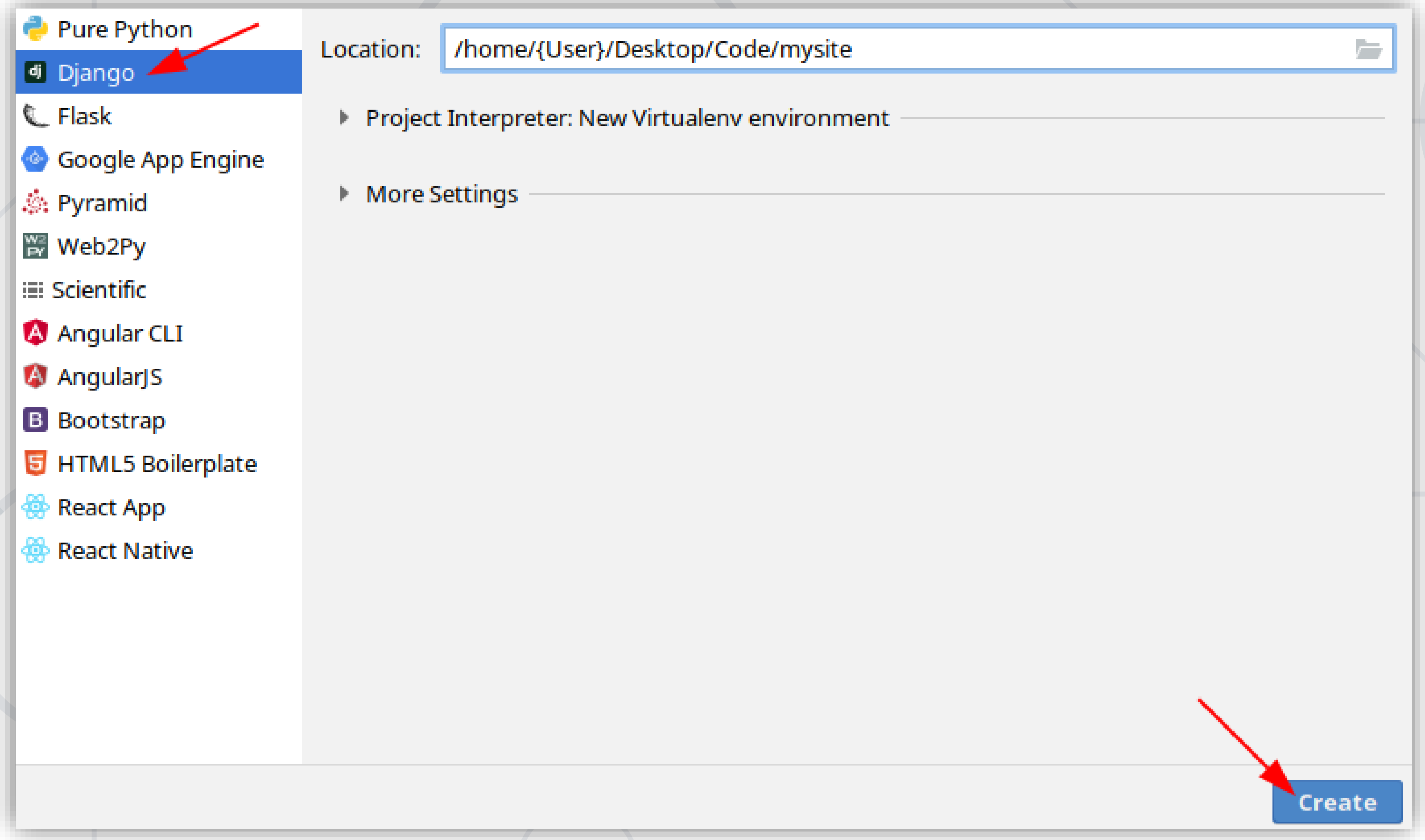
- **Django** is equipped with an **ORM**
- One of the **best ORMs** available in the industry today
- Tightly **coupled** with the **Django framework**
- Very **efficient**
- Ability to handle **medium** to **low complexity** queries and **medium** to **huge** datasets
- **Migrations** are another **useful feature**

Creating a Django Project

Where the Magic Happens

Creating a Django Project

- Open PyCharm Professional -> File -> New Project



Location: /home/{User}/Desktop/Code/mysite

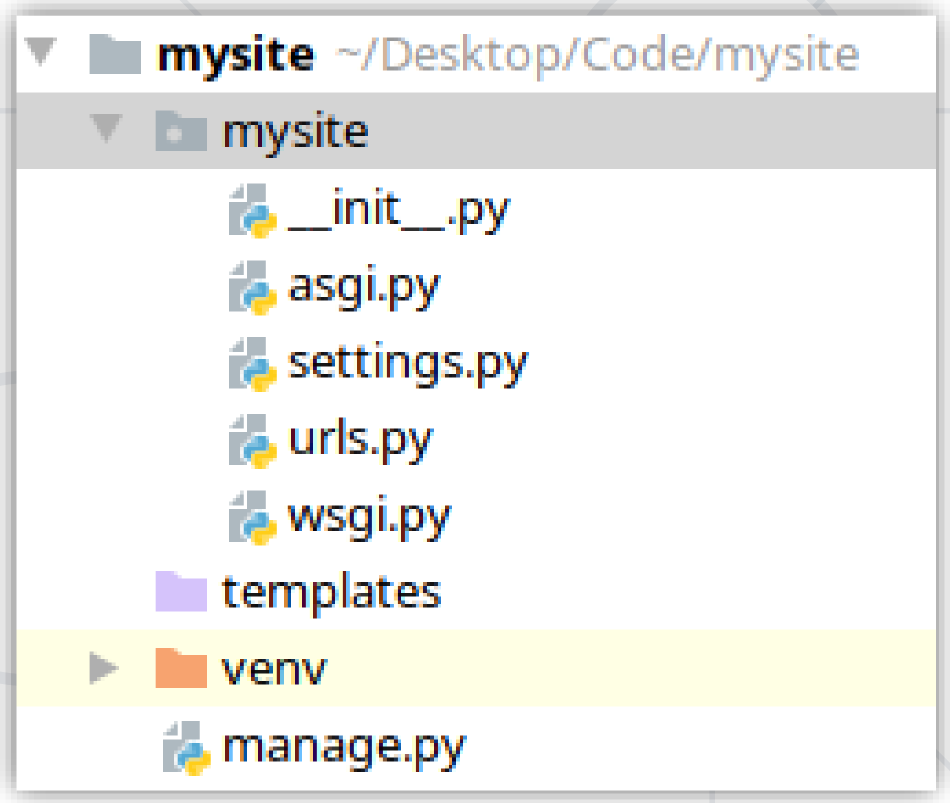
Project Interpreter: New Virtualenv environment

More Settings

Project Structure

Software University

- __init__.py
  - The directory is a Python package
- settings.py
  - The configuration file for the Django Project
- urls.py
  - Contains the list of URLs
- manage.py
  - Tool for executing commands



```
mysite ~/Desktop/Code/mysite
  mysite
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
  venv
  manage.py
```

Running a Django Project (1)

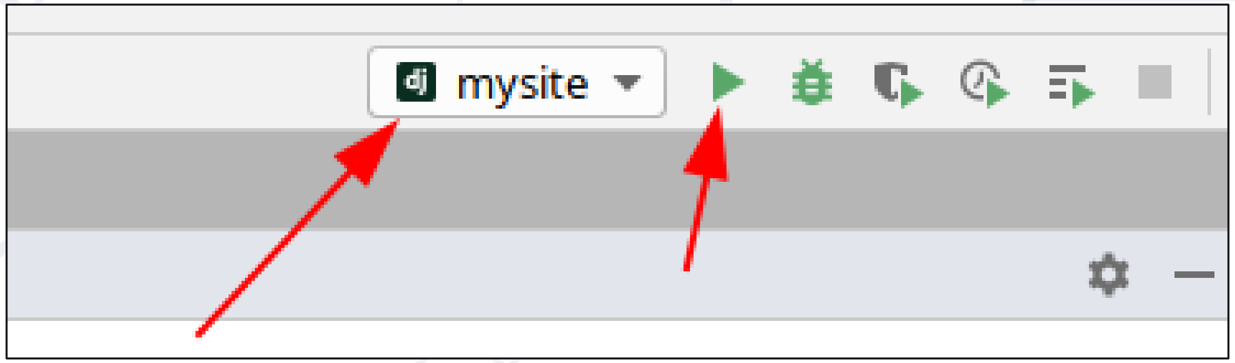
- Using **Terminal command**

```
python manage.py runserver
```

- Using **Keyboard Shortcut** in PyCharm

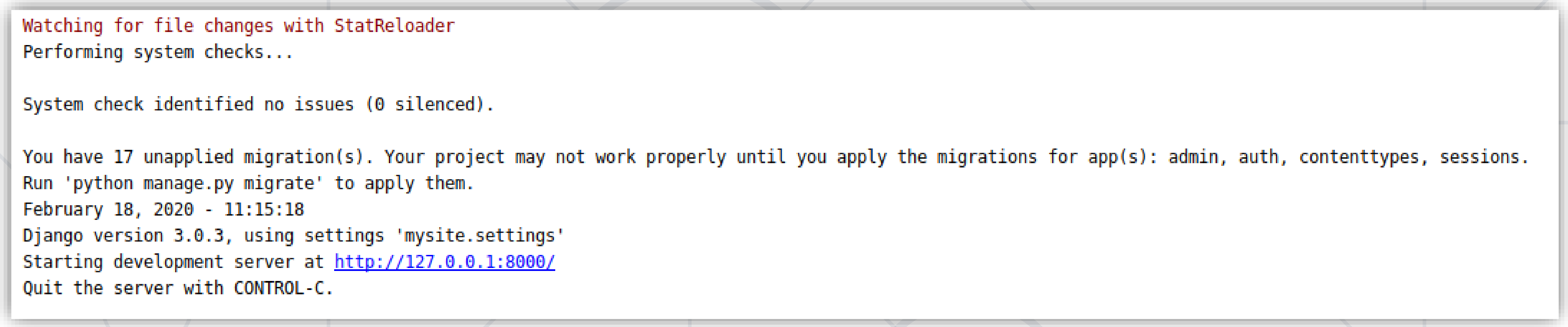
```
Shift + F10
```

- Using PyCharm **Run button**



Running a Django Project (2)

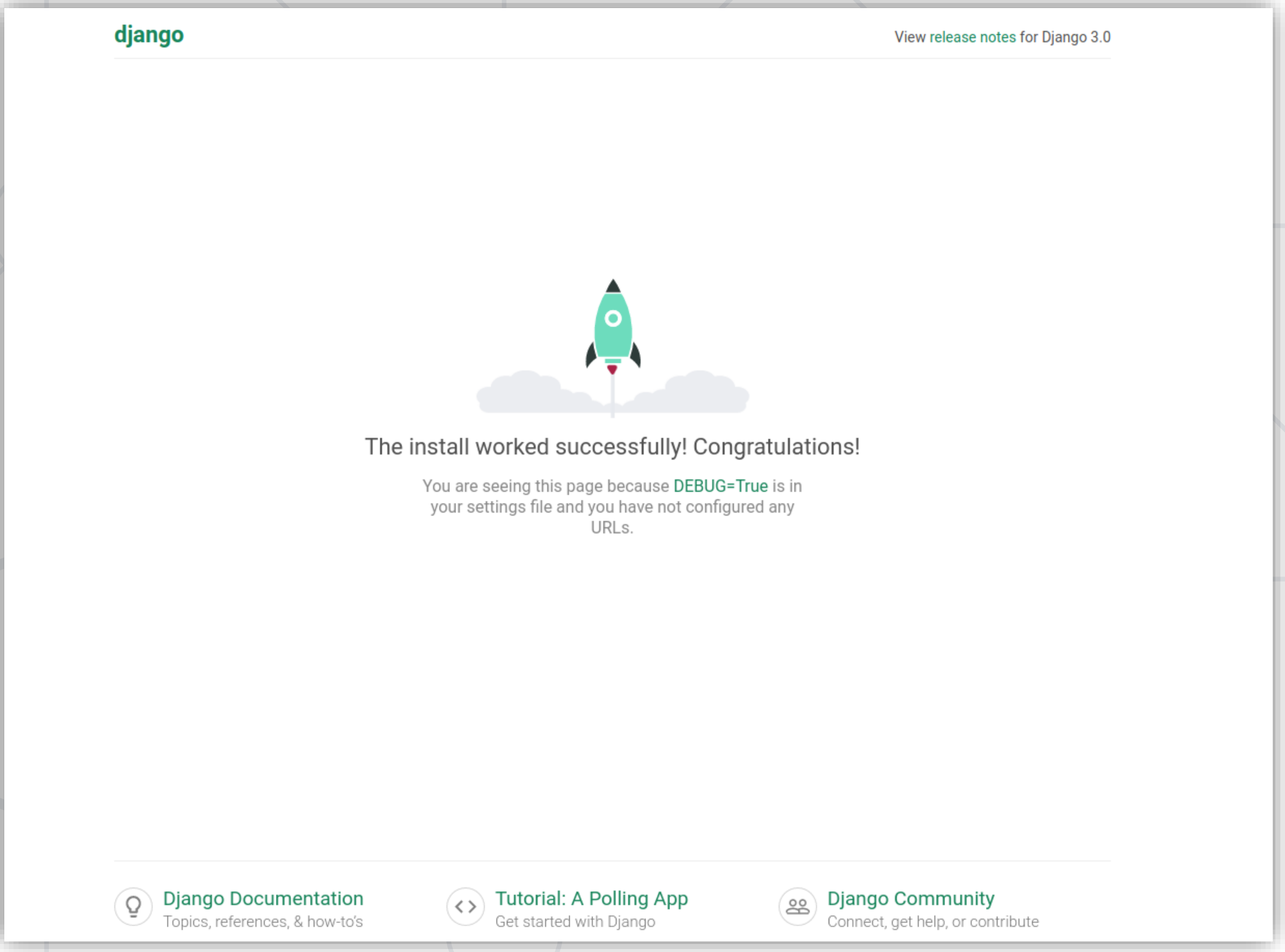
- You'll see the following output on the command line:



```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
February 18, 2020 - 11:15:18
Django version 3.0.3, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

- The **`runserver`** command starts the development server on the internal IP at **port 8000** by default
- Note: this server is intended for use only while in **Development Mode**

Running a Django Project (3)



django

View release notes for Django 3.0

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

Django Application

The Bread and Butter of a Django Project

App vs Project

- Django App:
  - A **Web application that does something** - e.g., a blog system or a small task app
  - One app can be used **in multiple projects**
- Django Project:
  - A **collection of configuration and apps** for a particular website
  - The project can **contain multiple apps**

Creating a Django App

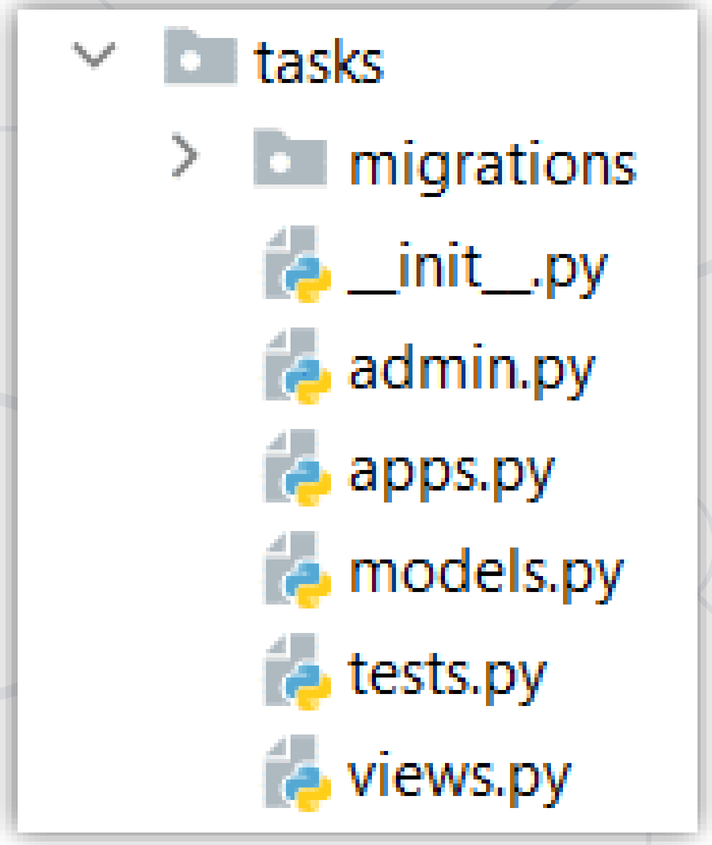
- The app is created in the **same directory** as the **`manage.py`** file
- Use the **terminal command**

```
python manage.py startapp tasks
```

- Django **automatically** generates the **basic directory structure** of an app
- The app can be moved **inside the root directory of your Django project** (one good way of project management)

Directory Structure

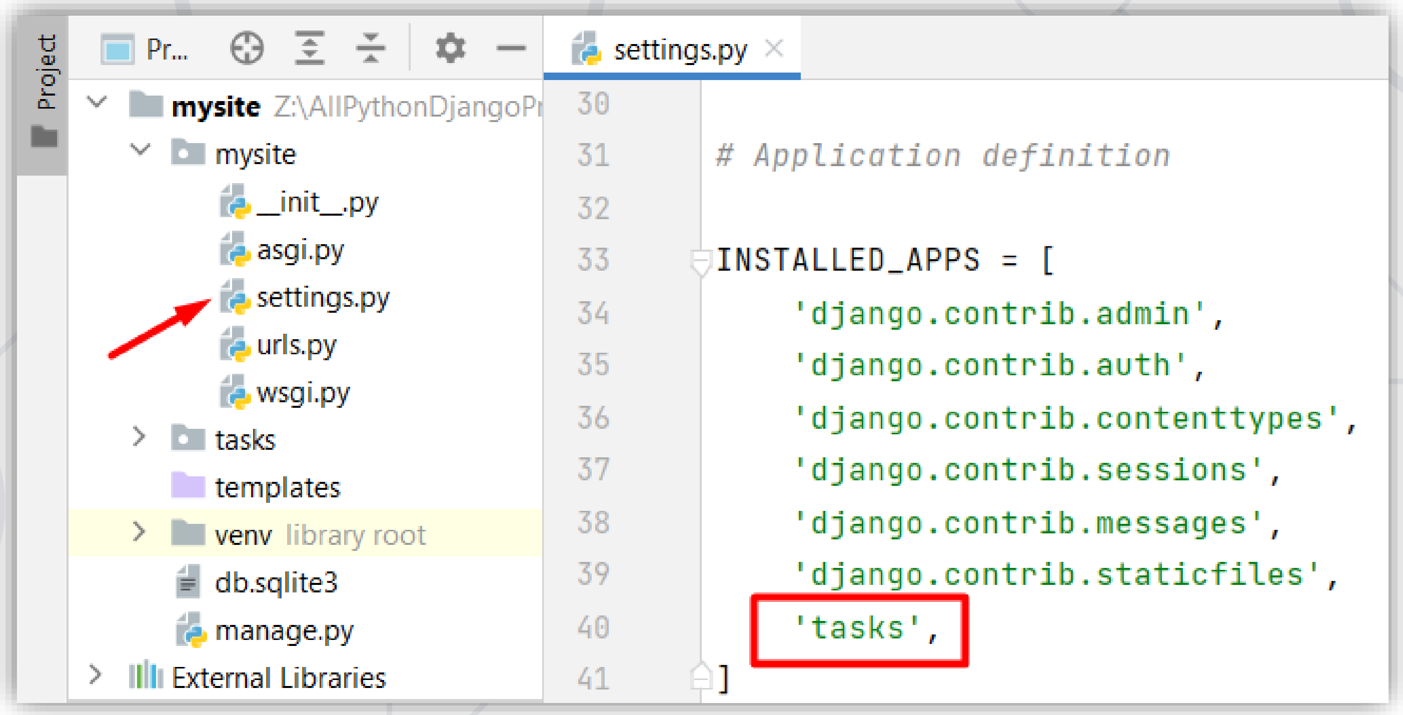
- **admin.py**
  - The admin site module
- **models.py**
  - The models of the app
- **views.py**
  - The views of the app
- **migrations**
  - Command-line utility for propagating changes in models



```
tasks
  migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  views.py
```

Including an App

- To include an app in a project, **add a reference** to the app in the **`INSTALLED_APPS`** setting



```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tasks',
]
```

Ready-to-Use Skeleton

Ready-to-Use Skeleton (1)

- You will be provided with a **ready-to-use skeleton** for the purposes of the ORM Course
  - No need to create a project and apps from scratch
- **Running** the **skeleton**
  - Open it with **Pycharm Professional**
  - Install **dependencies**

```
py -m pip install -r requirements.txt
```

  - Install **psycopg2**

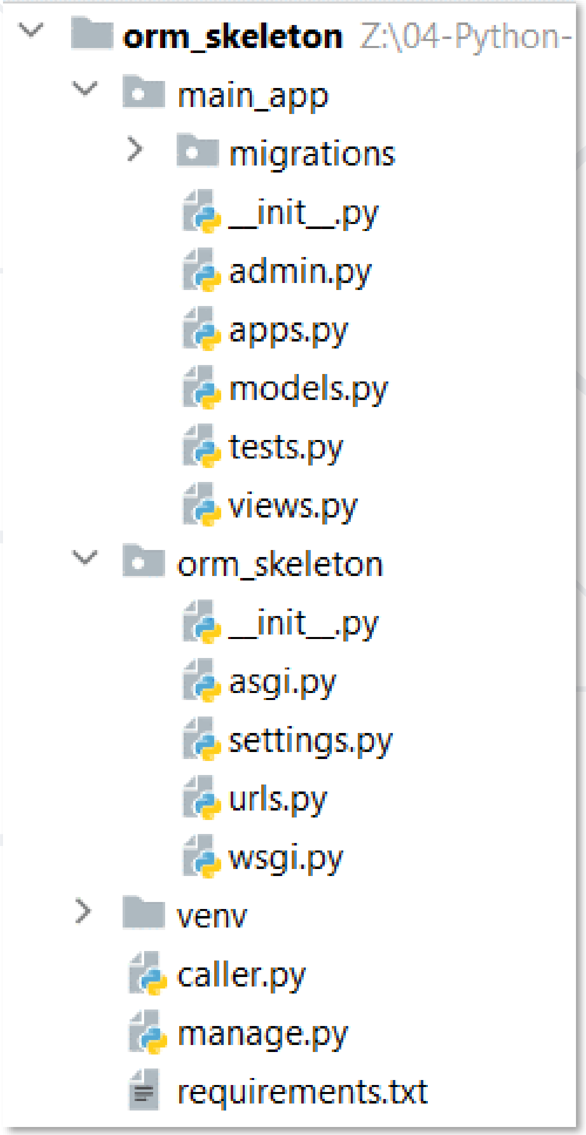
```
py -m pip install psycopg2
```

  - Run the project

```
Shift + F10
```

Ready-to-use Skeleton (2)

- Skeleton structure
  - settings.py
    - database settings
  - main_app
    - migrations folder
    - admin.py, models.py
  - caller.py
    - Python code and Django queries
  - requirements.txt
    - dependencies



```
orm_skeleton Z:\04-Python-
  main_app
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
  orm_skeleton
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  venv
  caller.py
  manage.py
  requirements.txt
```

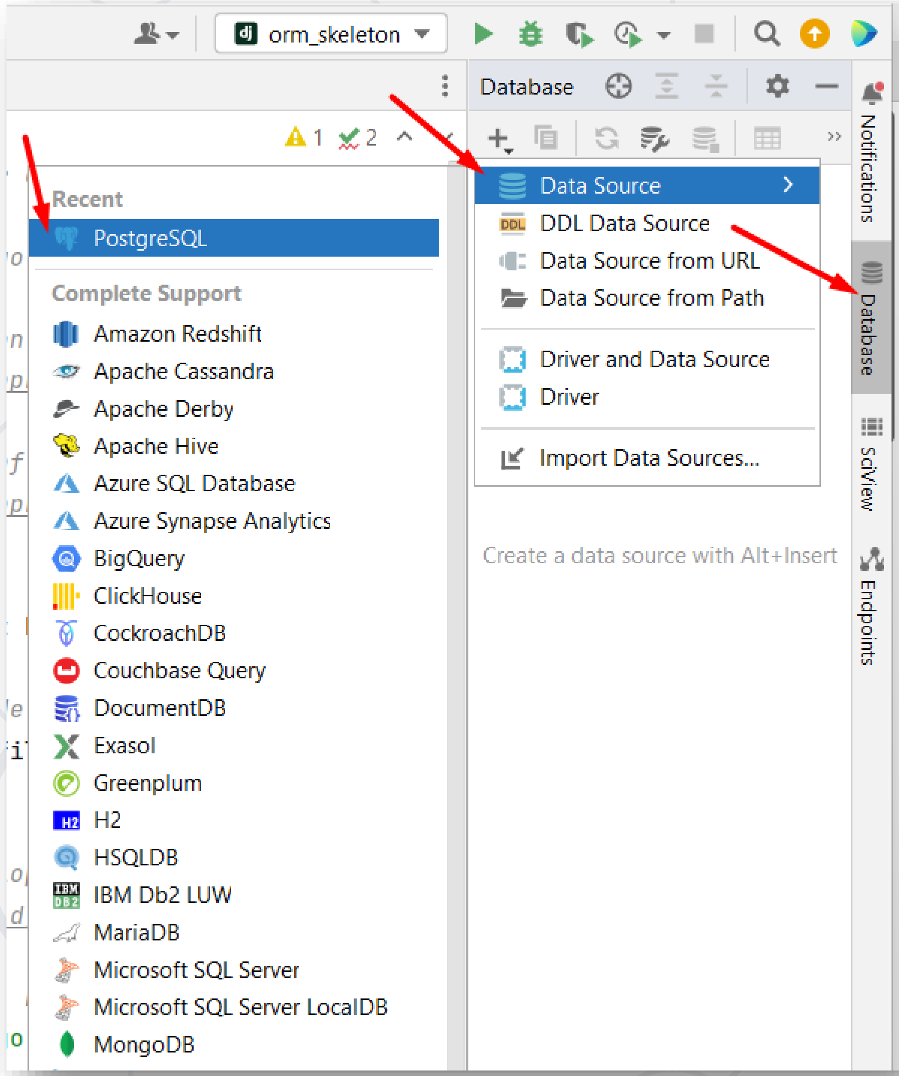
Ready-to-use Skeleton (3)

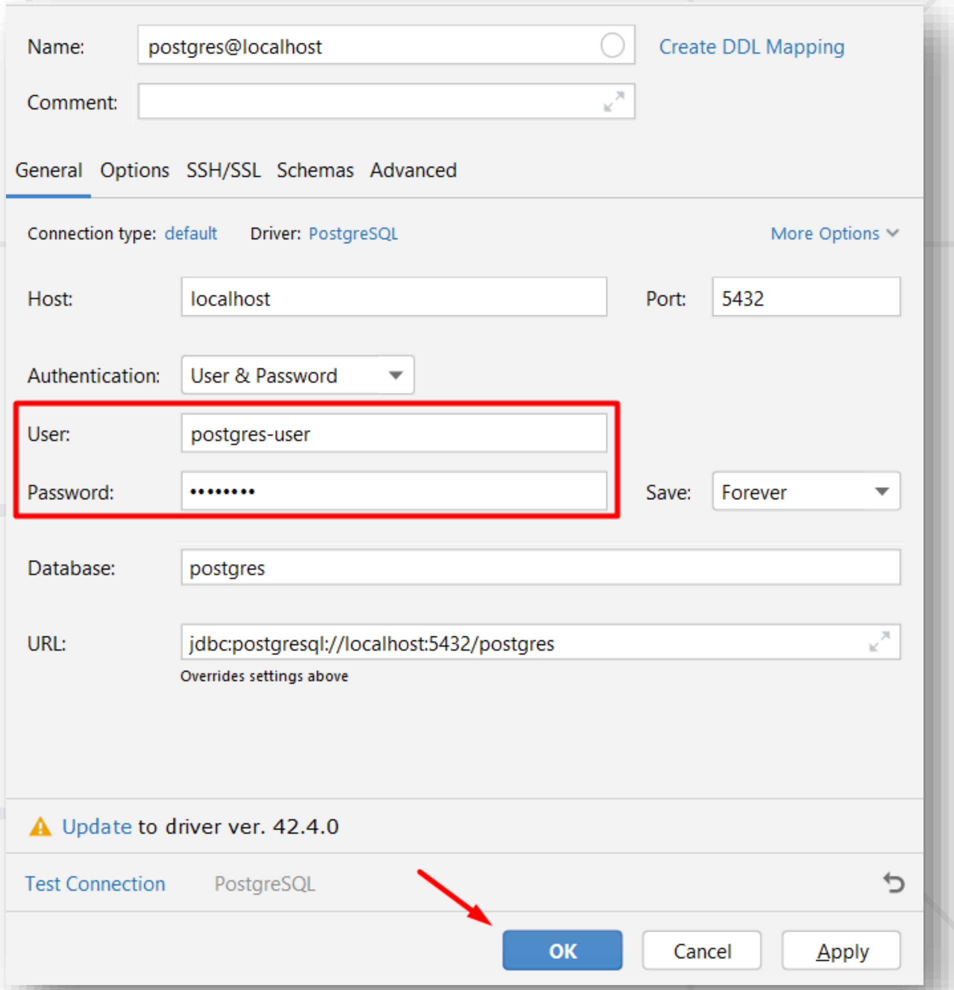
- **Activate Python Virtualenv** for **Windows**
  - Open a **Command Prompt** terminal in the project root directory
  - Execute the following command

```
venv\Scripts\activate
```

Setting up a Database

Connect to PostgreSQL





Name:	postgres@localhost
Comment:	

Create DDL Mapping

General Options SSH/SSL Schemas Advanced

Connection type: default Driver: PostgreSQL More Options

Host:	localhost	Port:	5432
Authentication:	User & Password		
User:	postgres-user		
Password:	••••••••	Save:	Forever
Database:	postgres		
URL:	jdbc:postgresql://localhost:5432/postgres		

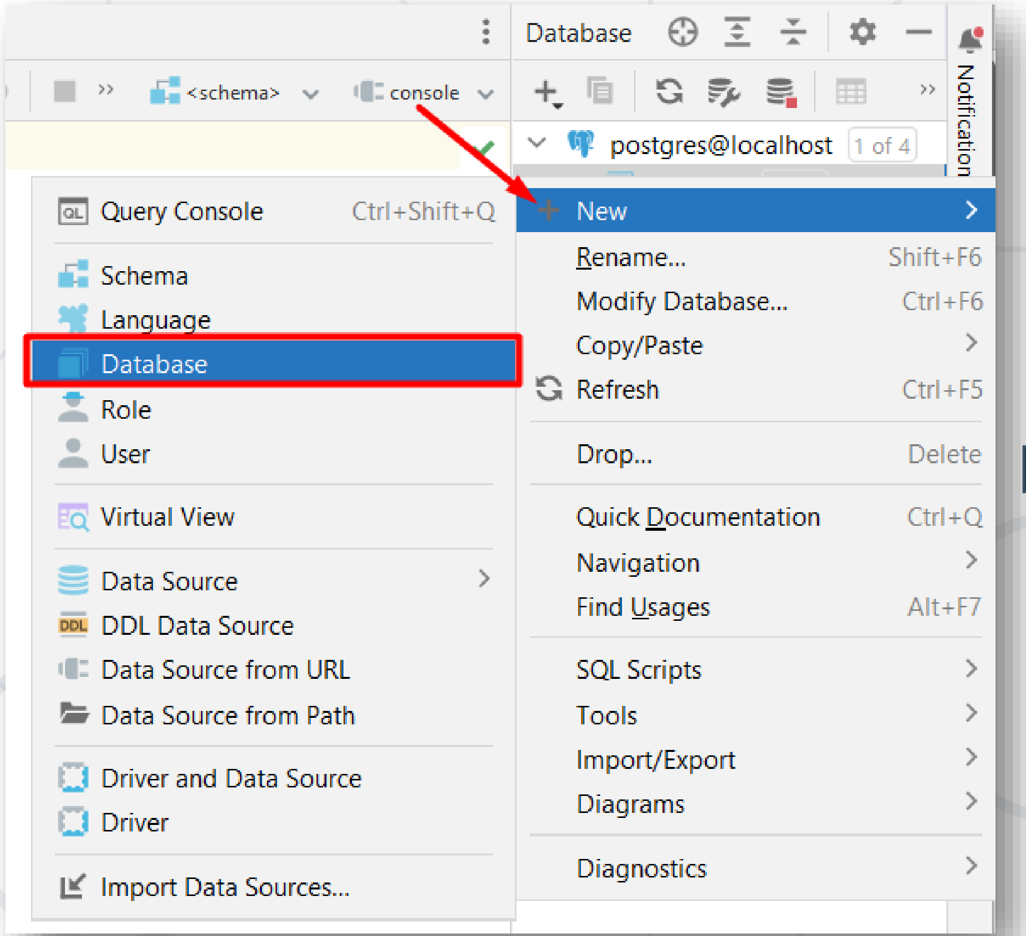
Overrides settings above

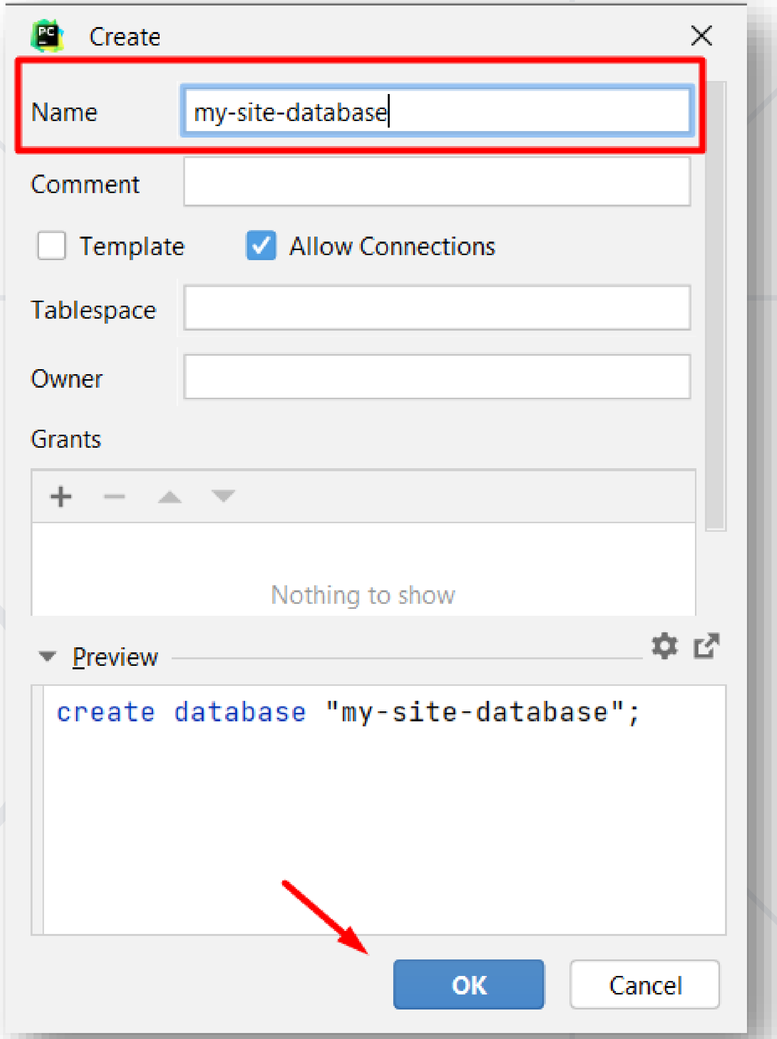
Update to driver ver. 42.4.0

Test Connection PostgreSQL

OK Cancel Apply

Create a Database





Name: my-site-database

[ ] Template [✓ Allow Connections]

```
create database "my-site-database";
```

Set up PostgreSQL

- To configure our project to work with **PostgreSQL**, we need to set it up in the **settings.py** file

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'my-site-database',
        'USER': 'postgres-user',
        'PASSWORD': 'postgres',
        'HOST': '127.0.0.1',
        'PORT': '5432'
    }
}
```

Use PostgreSQL

Name of the database

Database user credentials

Django dbshell

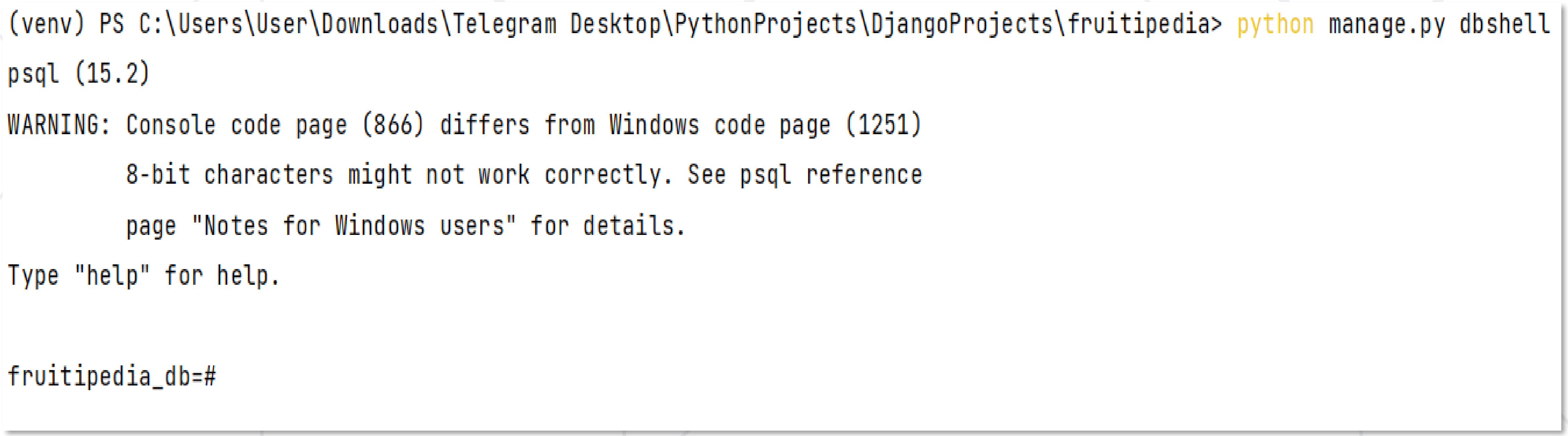
Django dbshell

- An interactive **command-line** interface **shell environment**
- Runs the command-line client for **specified database**, **or the default database**
- A very useful tool for **SQL database debugging** when working on a Django application

Running Django dbshell

```
python manage.py dbshell
```

```
py -m manage dbshell
```



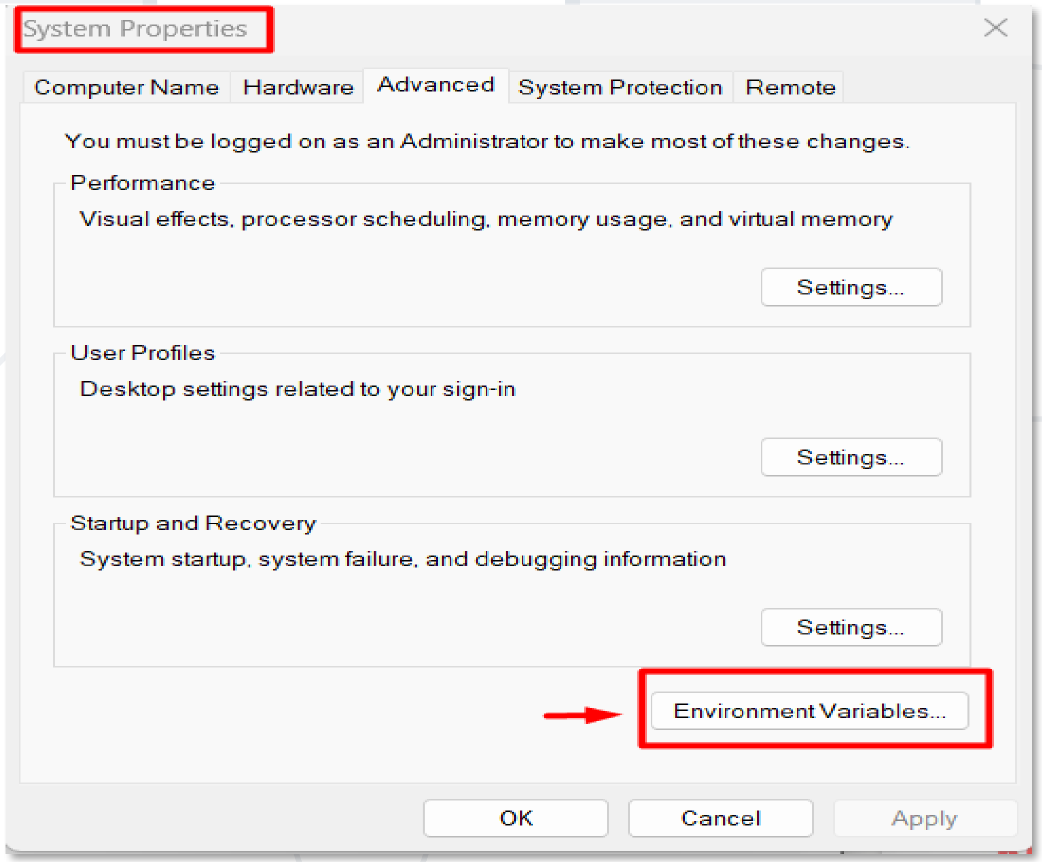
```
(venv) PS C:\Users\User\Downloads\Telegram Desktop\PythonProjects\DjangoProjects\fruitipedia> python manage.py dbshell
psql (15.2)
WARNING: Console code page (866) differs from Windows code page (1251)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

fruitipedia_db=#
```

Running Django dbshell (2) - Error

- In case you receive an **Error** like:
  - "**CommandError:** You appear not to have the 'psql' program installed or on your path"
- Follow the next steps:
  - Find your **PostgreSQL binary path**
    - **C:\Program Files\PostgreSQL\<version>\bin**
    - Where **<version>** is your current PostgreSQL version(e.g., 15)
  - **Add** this **path** to **Windows PATH environment variables**

Add Path to Windows Path Environment (1)

- **C:\Program Files\PostgreSQL\<version>\bin – copy your path!**
- Add it to the **Windows PATH environment variables – View Advanced System Settings**


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

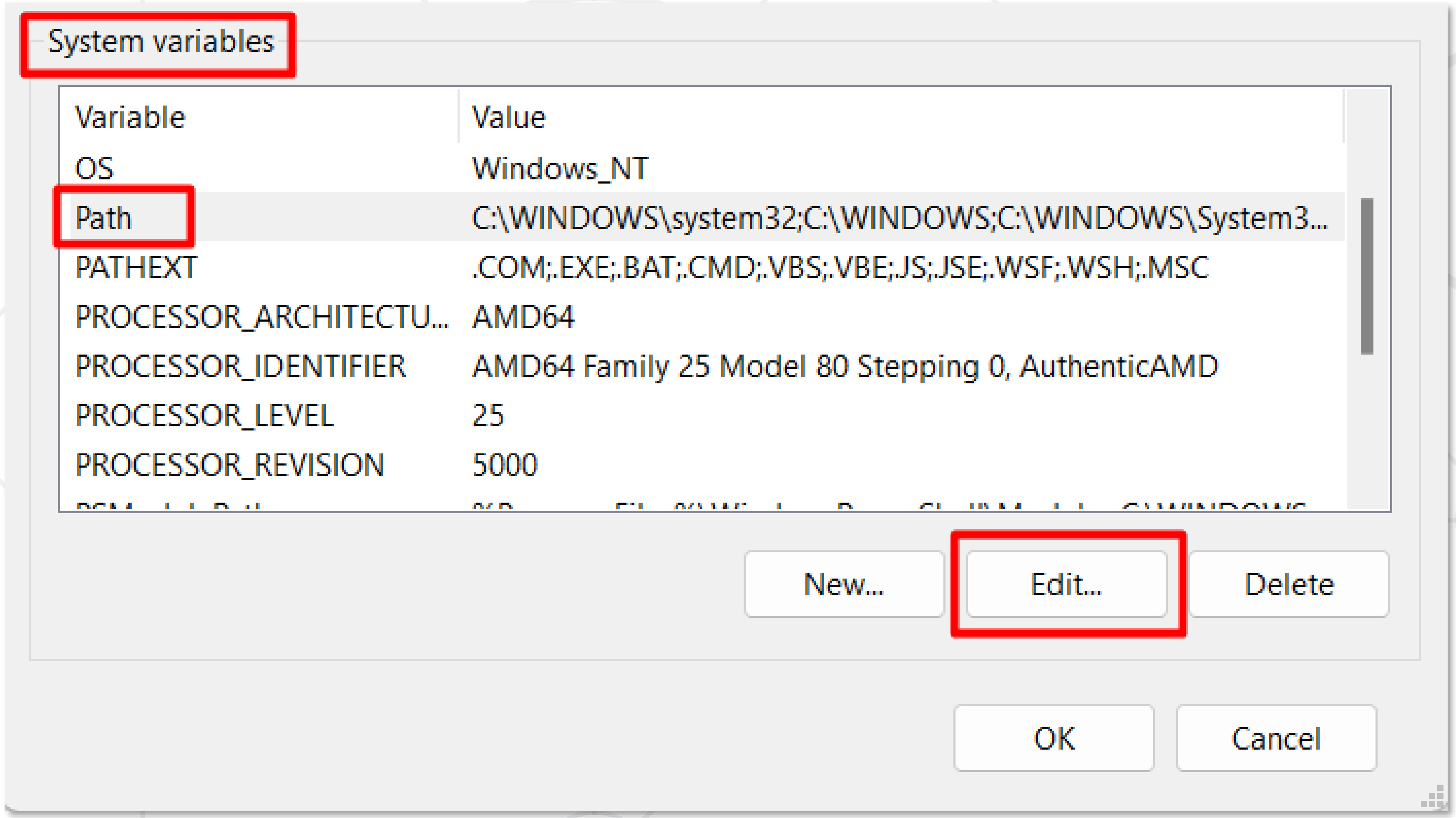
User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

Add Path to Windows Path Environment (2)

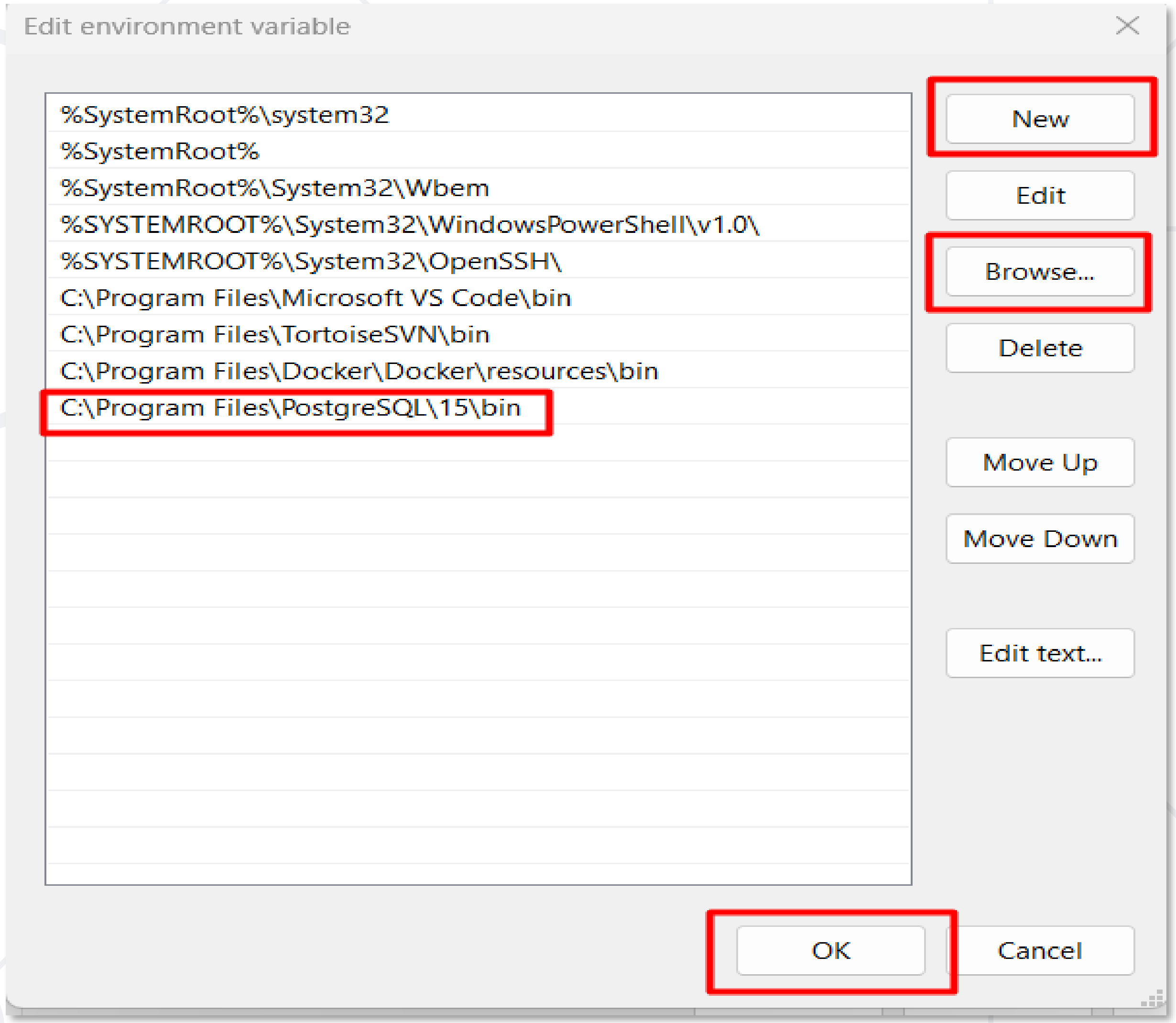


System variables

Variable	Value
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System3...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD
PROCESSOR_LEVEL	25
PROCESSOR_REVISION	5000

Add Path to Windows Path Environment (3)

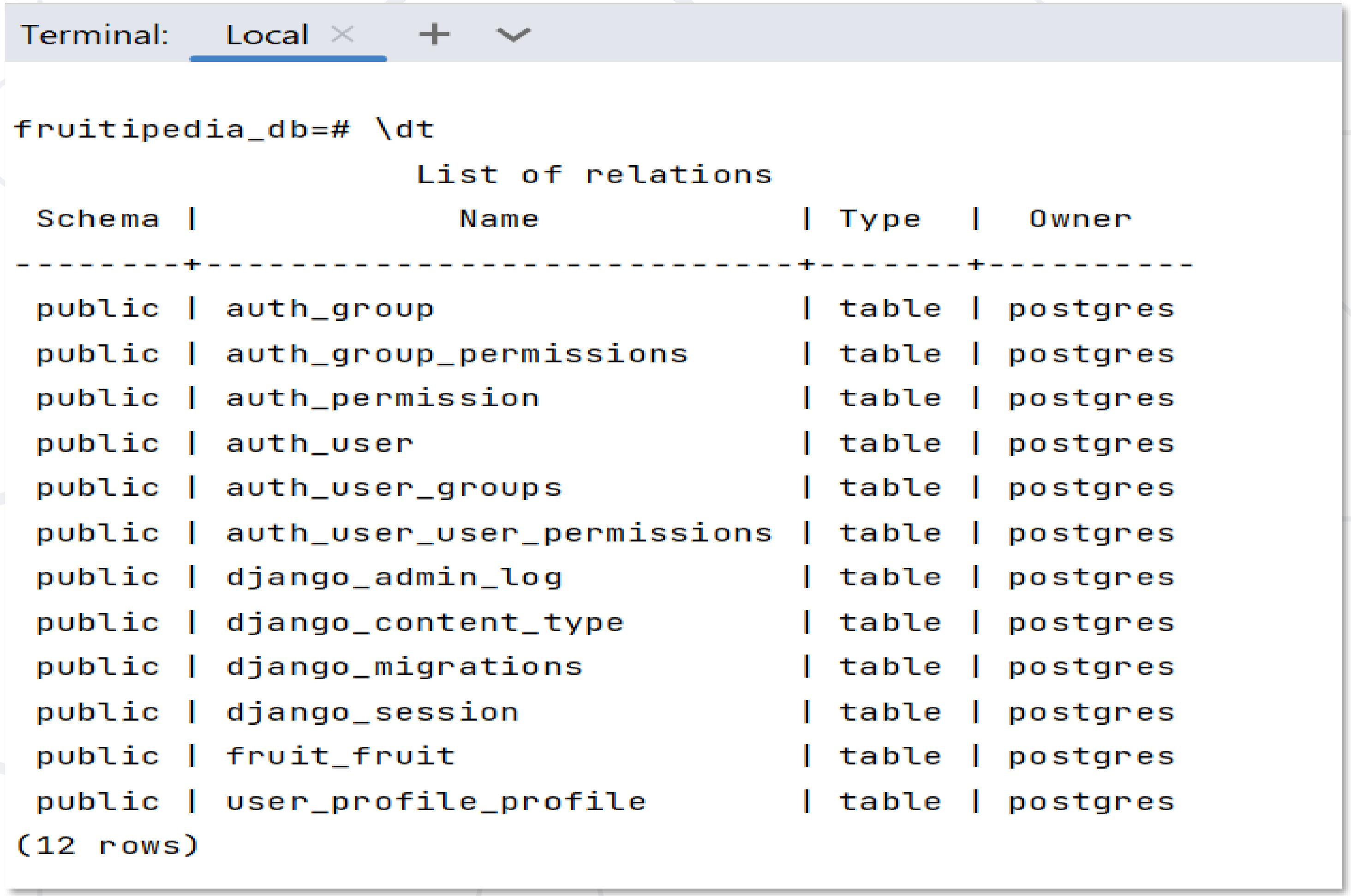
- **Navigate** to your path or **paste** the copied one
- Press **OK** in all windows and **restart** your **IDE**


Edit environment variable

%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Microsoft VS Code\bin
C:\Program Files\TortoiseSVN\bin
C:\Program Files\Docker\Docker\resources\bin
C:\Program Files\PostgreSQL\15\bin

Using Django dbshell (1)

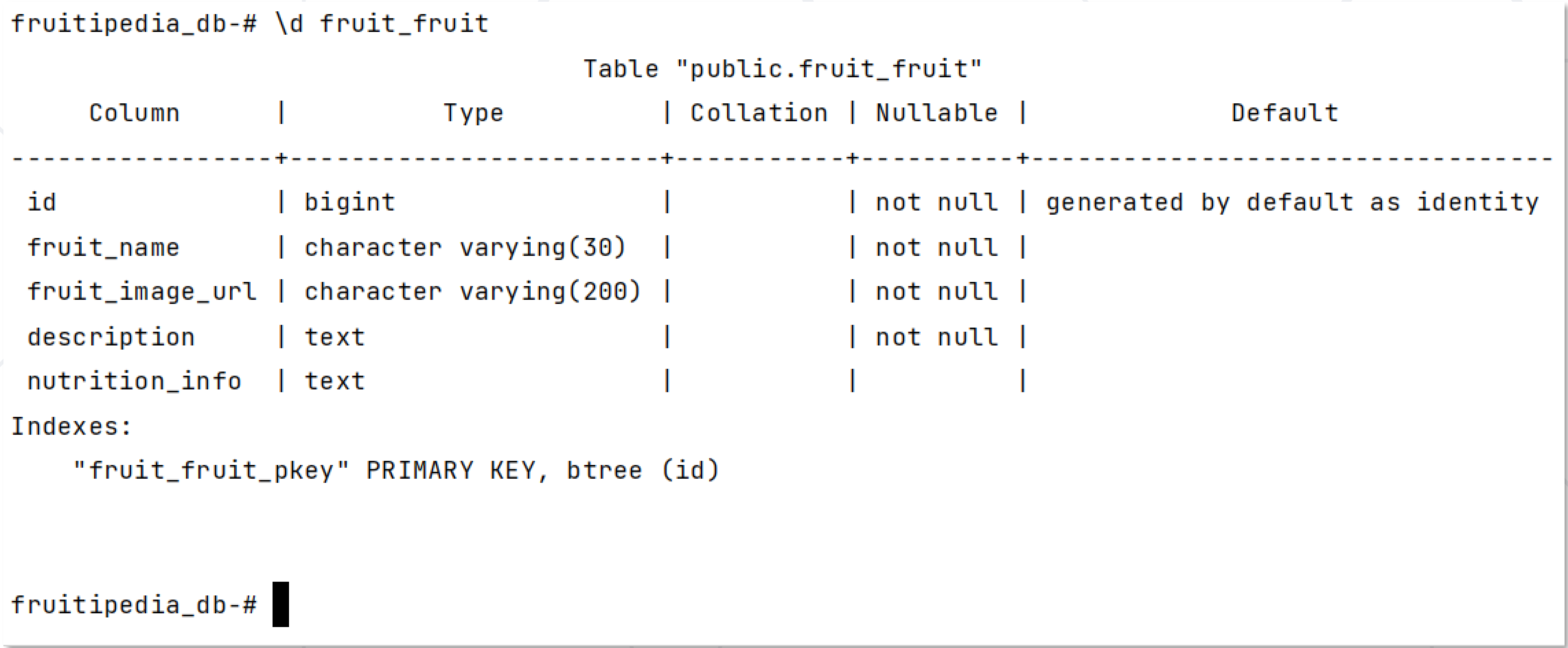
- **\dt** command shows **all** tables in the **current** database



```
fruitipedia_db=# \dt
                        List of relations
 Schema |              Name              | Type  |  Owner
--------+--------------------------------+-------+----------
 public | auth_group                     | table | postgres
 public | auth_group_permissions         | table | postgres
 public | auth_permission                | table | postgres
 public | auth_user                      | table | postgres
 public | auth_user_groups               | table | postgres
 public | auth_user_user_permissions     | table | postgres
 public | django_admin_log               | table | postgres
 public | django_content_type            | table | postgres
 public | django_migrations              | table | postgres
 public | django_session                 | table | postgres
 public | fruit_fruit                    | table | postgres
 public | user_profile_profile           | table | postgres
(12 rows)
```

Using Django dbshell (2)

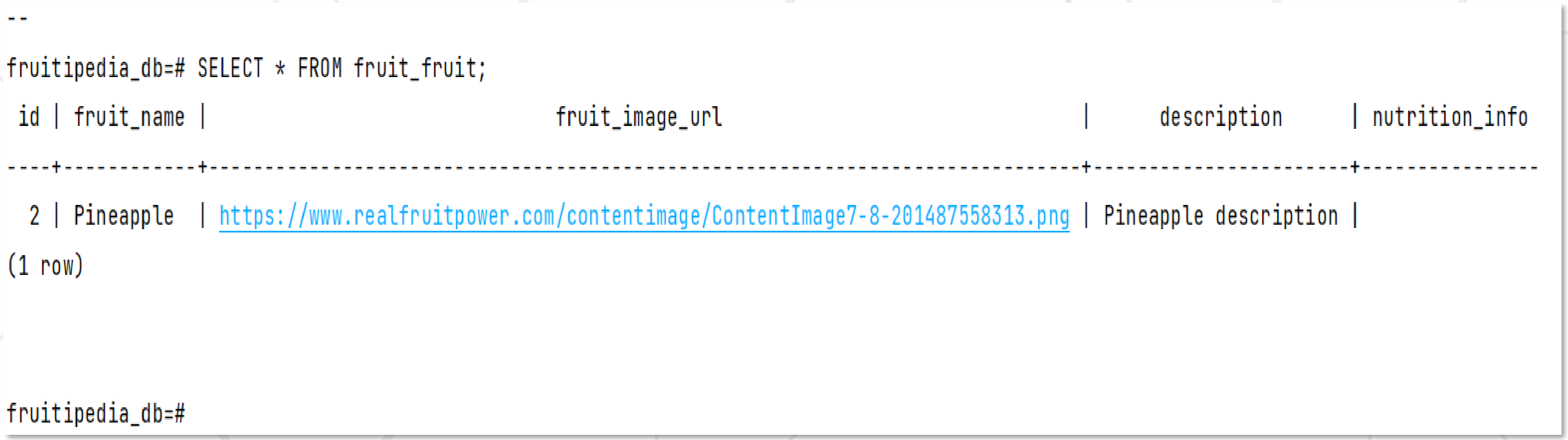
- **\d <table_name>** command shows a **specific** table



```
fruitipedia_db-# \d fruit_fruit
                                          Table "public.fruit_fruit"
     Column      |          Type          | Collation | Nullable |               Default
-----------------+------------------------+-----------+----------+------------------------------------
 id              | bigint                 |           | not null | generated by default as identity
 fruit_name      | character varying(30)  |           | not null |
 fruit_image_url | character varying(200) |           | not null |
 description     | text                   |           | not null |
 nutrition_info  | text                   |           |          |
Indexes:
    "fruit_fruit_pkey" PRIMARY KEY, btree (id)


fruitipedia_db-#
```

Using Django dbshell (3) - Queries

- **SELECT * FROM <table_name>;**



```
--
fruitipedia_db=# SELECT * FROM fruit_fruit;
 id | fruit_name |                              fruit_image_url                               |     description      | nutrition_info
----+------------+---------------------------------------------------------------------------+----------------------+---------------
  2 | Pineapple  | https://www.realfruitpower.com/contentimage/ContentImage7-8-201487558313.png | Pineapple description |
(1 row)


fruitipedia_db=#
```

Practice

Live Demo in Class

Summary

- **ORM**
  - Object-Relational Mapping
  - Python Classes -> DB Tables
- **DB Drivers**
  - Psycopg2
- **Django and Django ORM**
  - Django Project, Django App
- **Django dbshell**

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter International

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License