🧩 What is a `.env` file?

A **`.env` file** (environment file) is a simple **text file that stores configuration variables** for your project — like database credentials, secret keys, API tokens, and environment-specific settings — **outside the codebase**.

This helps you:

- Avoid hardcoding sensitive information in your code.
- Maintain different configurations for **development, staging, and production**.
- Make your codebase **cleaner and more secure**.

⚙️ Example of a `.env` file

```
# .env
DEBUG=True
SECRET_KEY=django-insecure-q1234asdfsdf
DATABASE_NAME=mydb
DATABASE_USER=myuser
DATABASE_PASSWORD=mypassword
DATABASE_HOST=localhost
DATABASE_PORT=5432
```

⚠️ Always add `.env` to `.gitignore` to prevent secrets from being pushed to GitHub.

🧠 What is `django-environ`?

`django-environ` is a Python package that helps Django projects:

- **Read** variables from `.env` files.

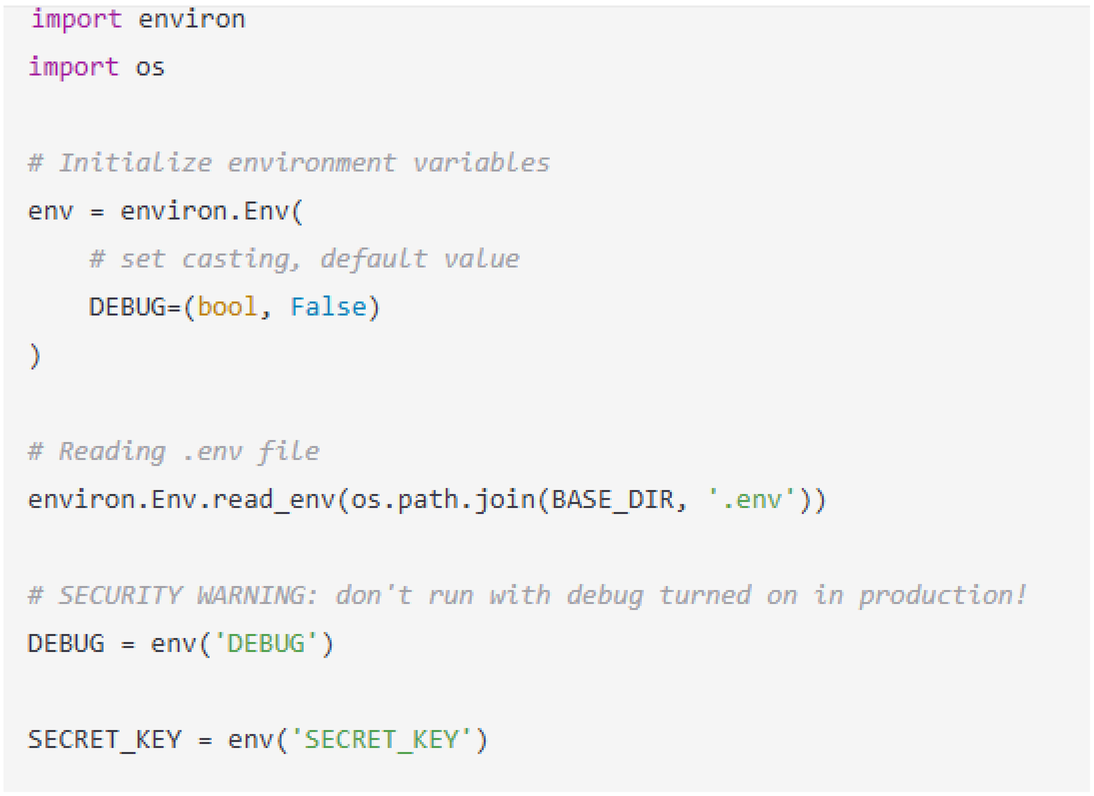
- **Parse** them automatically into the right types (booleans, lists, URLs, etc).
- **Integrate seamlessly** with Django’s [`settings.py`](settings.py).
- Installation: pip install django-environ

It builds on the idea of **12-factor apps**, which promote keeping config in the environment, not in code.

🧰 Setup Step-by-Step

Step 1: Create your .env file. At your project root (same level as manage.py):

Step 2: Modify your settings.py



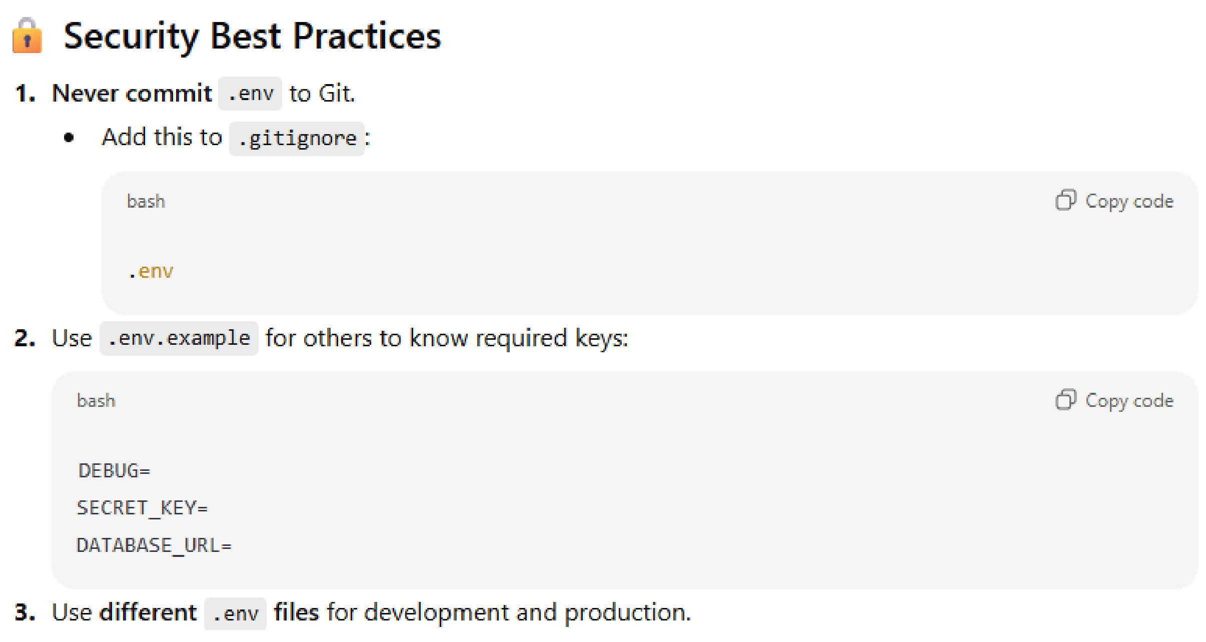
```python
import environ
import os

# Initialize environment variables
env = environ.Env(
    # set casting, default value
    DEBUG=(bool, False)
)

# Reading .env file
environ.Env.read_env(os.path.join(BASE_DIR, '.env'))

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = env('DEBUG')

SECRET_KEY = env('SECRET_KEY')
```



🔒 Security Best Practices

1. **Never commit** `.env` to Git.
   - Add this to `.gitignore`:

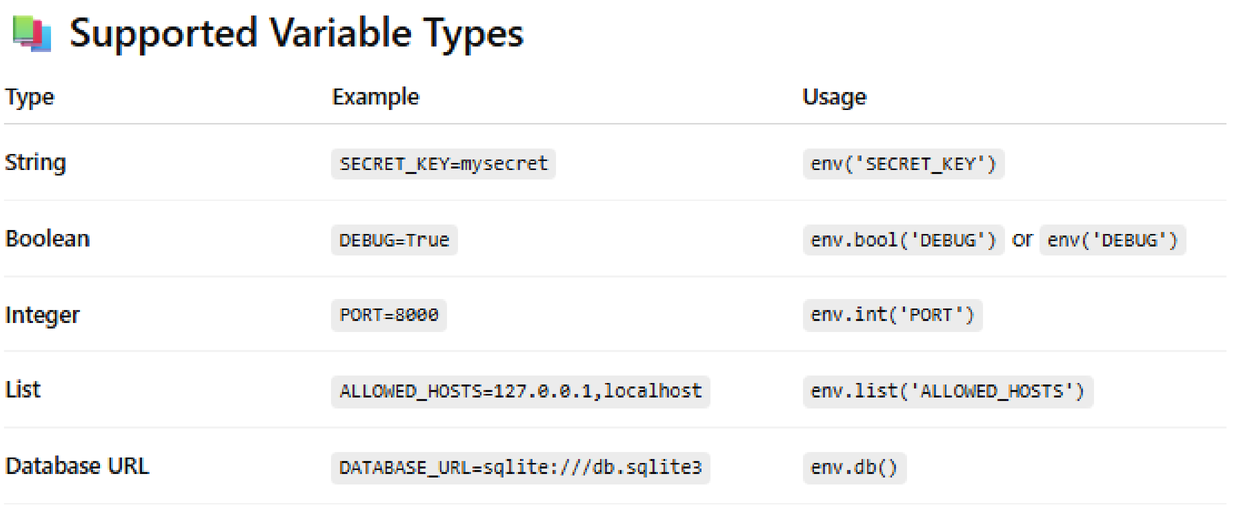
```bash
.env
```

2. Use `.env.example` for others to know required keys:

```bash
DEBUG=
SECRET_KEY=
DATABASE_URL=
```

3. Use **different** `.env` **files** for development and production.

Type Conversions and Supported types:



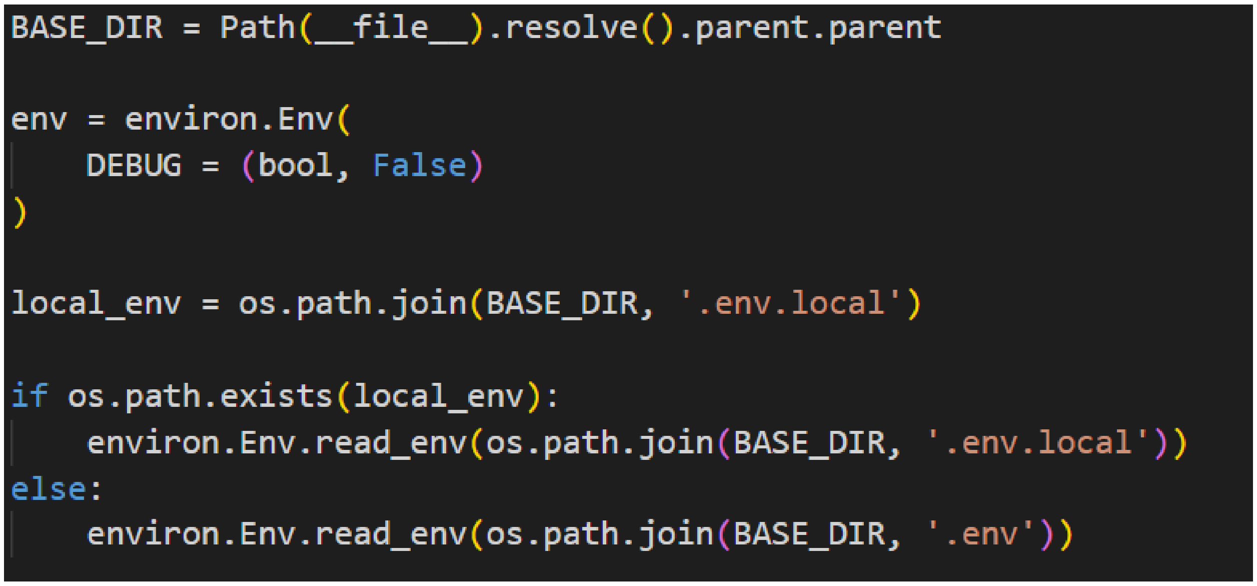
📚 Supported Variable Types

Type	Example	Usage
String	`SECRET_KEY=mysecret`	`env('SECRET_KEY')`
Boolean	`DEBUG=True`	`env.bool('DEBUG')` or `env('DEBUG')`
Integer	`PORT=8000`	`env.int('PORT')`
List	`ALLOWED_HOSTS=127.0.0.1,localhost`	`env.list('ALLOWED_HOSTS')`
Database URL	`DATABASE_URL=sqlite:///db.sqlite3`	`env.db()`

1. **DATABASE_URL format for MYSQL :**
   a. **DATABASE_URL=mysql://dbuser:dbpassword@localhost:3306/mydatabase**
   b. Use URL-encode special characters if your above values contain special characters like @, / etc.

Handling Multiple Env files:

1. Add .env.local at the same level as .env



```python
BASE_DIR = Path(__file__).resolve().parent.parent

env = environ.Env(
    DEBUG = (bool, False)
)

local_env = os.path.join(BASE_DIR, '.env.local')

if os.path.exists(local_env):
    environ.Env.read_env(os.path.join(BASE_DIR, '.env.local'))
else:
    environ.Env.read_env(os.path.join(BASE_DIR, '.env'))
```

🪄 Tip for Production

In production (like on Heroku or Render), you don't even need `.env`.
You can set environment variables directly on the server, and `django-environ` will pick them up automatically.