Database Model Using PostgreSQL

Created	@October 9, 2023 9:42 AM
Made By	Mariam Tamer

to change the running port from 8000 to any number —>

python manage.py runserver 9000

Template Inheritance

-Components that become **repeated** in the whole page like **Navbar and Footer** are put **only once** in a **base.html file** so you don't have to write them in every page you make ..

-add a new folder called **Layouts** that'll contain your **base.html**

-Your new source will be : **app/templates/appName/Layouts/base.html**

Steps To Create A Database Model

-which dbms ?

-set it up in settings.py

-make sure it’s installed and db is connected

-install postgresql driver

to install postgresql driver :

-make sure your venv is activated

-then type : **pip install poycopg2**

type : **pip freeze** to see what packages are installed

---

next : configure confidentiality :

username, password, dbname, portname, hostname

create user on postgresql:

create user mariam with password ‘000’;

note : you don’t use double quotations “ “ in postgresql unless in 2 cases : alias - arrays

to give user permission

alter user mariam superuser;

to give user permission to create database

alter user mariam createdb;

to display list of roles (aka list all users)

\du

create database

```
create database project;
```

to list all databaases

```
\l
```

---

then add credits in settings.py

```
DATABASES = {
"default": {
"ENGINE": "django.db.backends.postgresql",
"NAME": "project",
"USER": "mariam",
"PASSWORD": "0000",
"HOST": "localhost",
"PORT": "5432",
}
}
```

---

connect to database

```
\c iti
```

-Django connects to DB using ORM : Object Relational Manager.

-Any table in DB is seen as a CLASS.

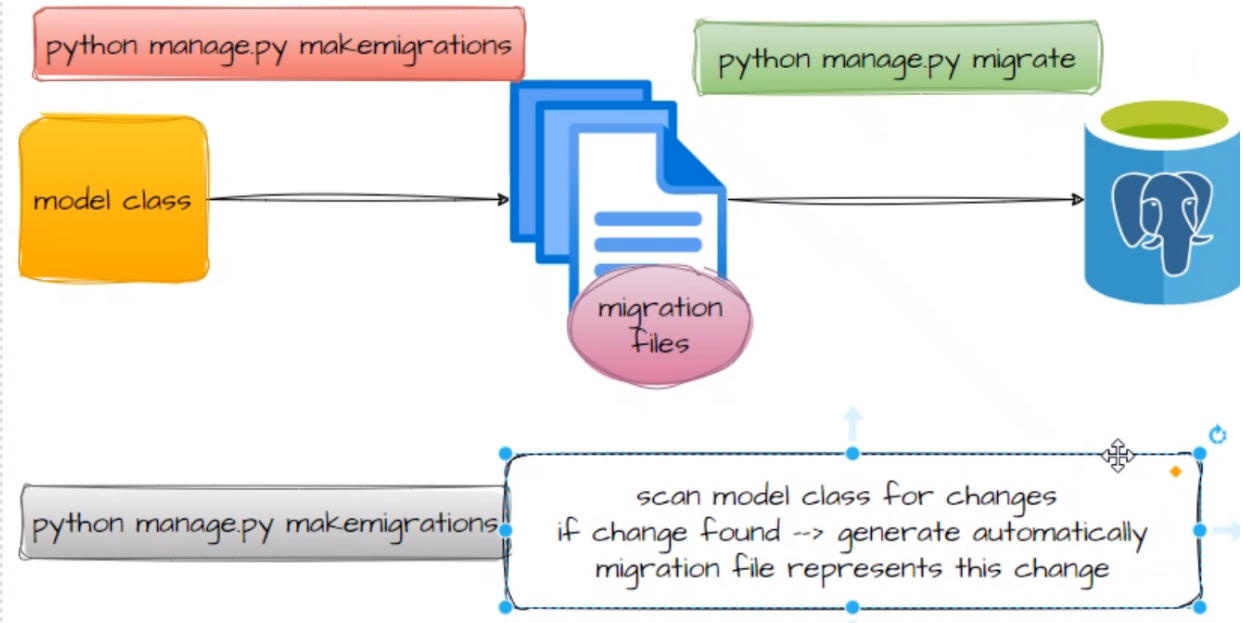
-Any row in DB is seen as an OBJECT.

-bc **class** : **field type + method(Type) type**

-**object** : **represents record**

How to transfer from classes&objects to PostgreSQL in Django so they’re applied in DB?

-Using Migration Files



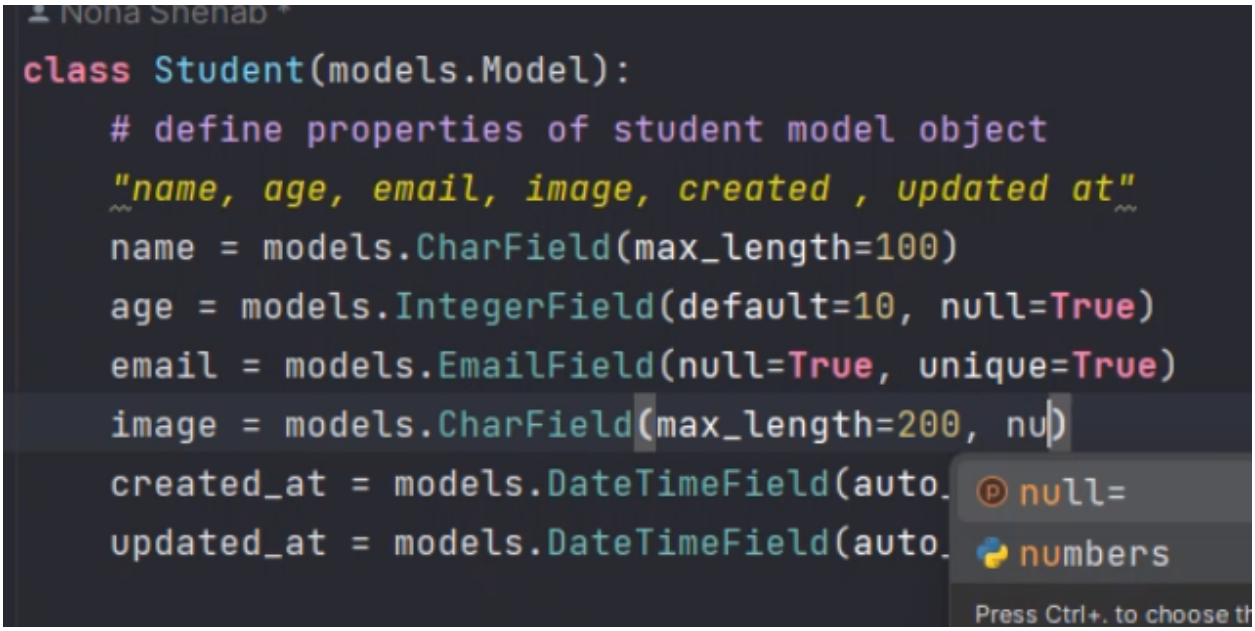
-so to migrate your files, you type :

python3 manage.py makemigrations

python3 manage.py migrate

Creating my own model

-Give attributes to your student table



-Here, **EmailField** is avaialable.

-There's an **auto_now_add=True** property for dates.

-After every edit in your database, type the migration 2 commands

-You can check for your databases in PostgreSQL to see what's added

-In the newly generated migration files, you can find this:

```
    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Student',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True,
                ('name', models.CharField(max_length=100)),
                ('age', models.IntegerField(default=10)),
                ('email', models.EmailField(max_length=254)),
                ('image', models.CharField(max_length=200)),
                ('created_at', models.DateTimeField(auto_now_add=True)),
                ('updated_at', models.DateTimeField(auto_now=True)),
            ],
        ),
    ]
```

to list description of a your models table in javascipt ,(its cols and data types)

\d appName_tableName

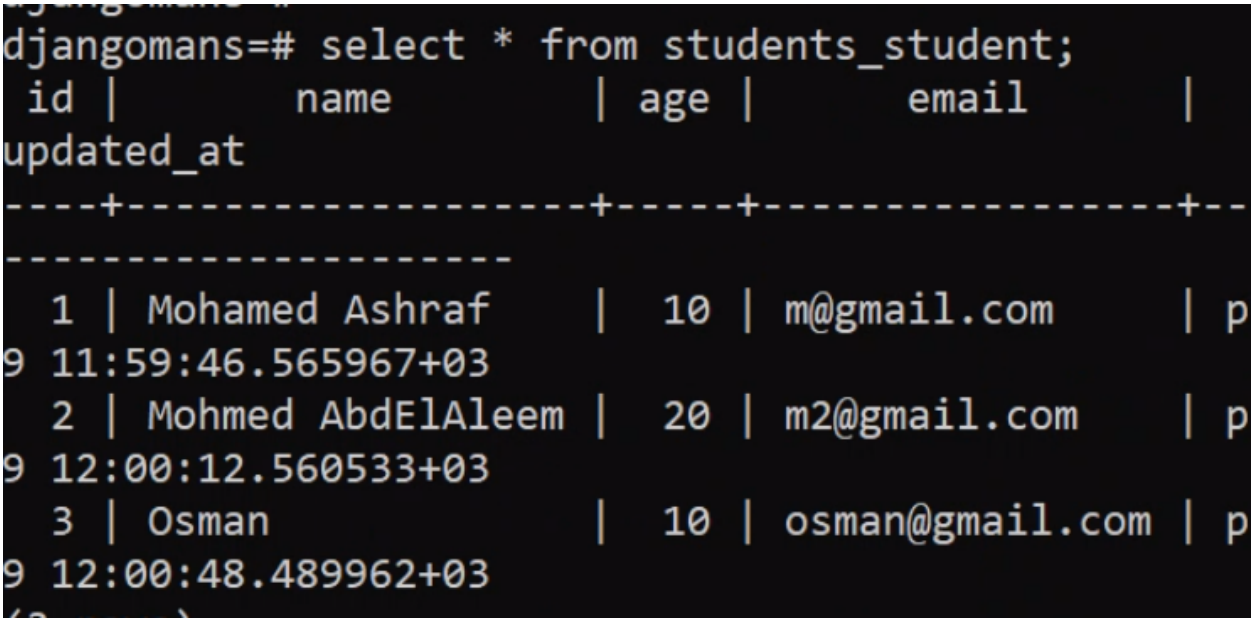
ex : \d project_student

```
djangomans=# \d students_student
                                Table "public.students_student"
   Column   |           Type           | Collation | Nullable |              Default
------------+--------------------------+-----------+----------+-----------------------------------
 id         | bigint                   |           | not null | generated by default as identity
 name       | character varying(100)   |           | not null |
 age        | integer                  |           | not null |
 email      | character varying(254)   |           | not null |
 image      | character varying(200)   |           | not null |
 created_at | timestamp with time zone |           | not null |
 updated_at | timestamp with time zone |           | not null |
Indexes:
    "students_student_pkey" PRIMARY KEY, btree (id)
```

-Any table you generate using **any Framework including models here**, your cols are **NOT Null by default.**

-So if you want a **column to be empty/or not** depending on the user's desire, you add **(Null = True)** property to the column.

To display the table, don't forget to write app_table when selecting :



How Does Interaction Happen?

-Since **model** is the one that **talks to DB,** Django sends the model object so it becomes **saved** in the DB (PostgreSQL here), or Django **requests** an **object** or **set of objects** from the DB

Register To Admin Panel

-In your VSC terminal, type :

python3 manage.py create superuser

Add the model you created to Admin Panel

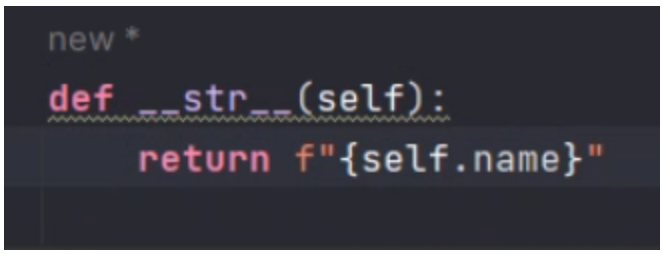
```
from students.models import Student

admin.site.register(Student)
```

To **stringify** your student table objects so they become **normal names** on your admin panel **instead of student.object1 , … —>**

def __str__(self):

return f”{self.name}”



To make your terminal colored :

python manage.py <command> --color <option>

QuerySets

To add new data to your table from a shell in your terminal **for testing** instead of from Admin Panel Site

1) python3 manage.py shell

then write your code like the following :

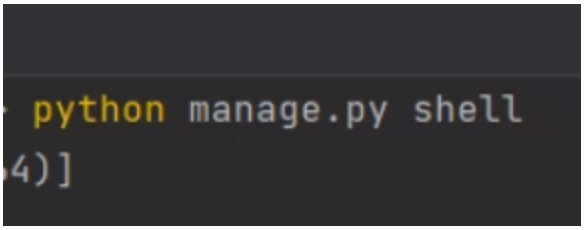
2) import your model table —> from table.models import Table

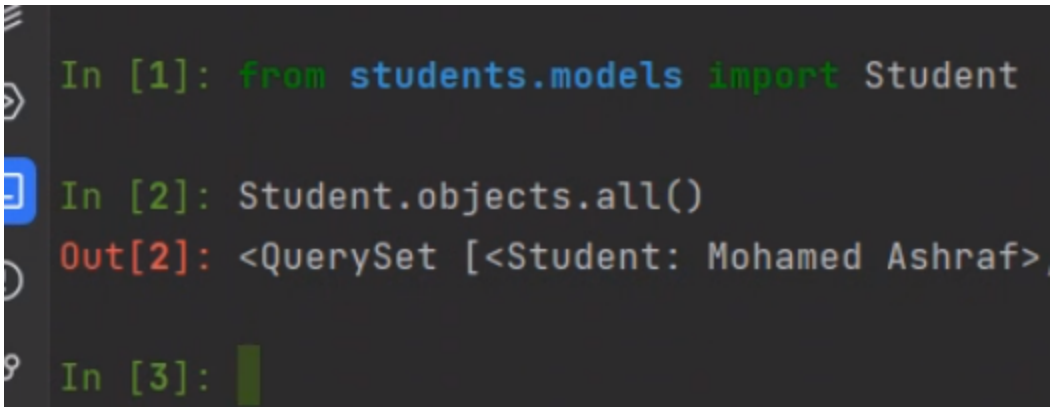
3) to access all data in table —> Student.objects.all()

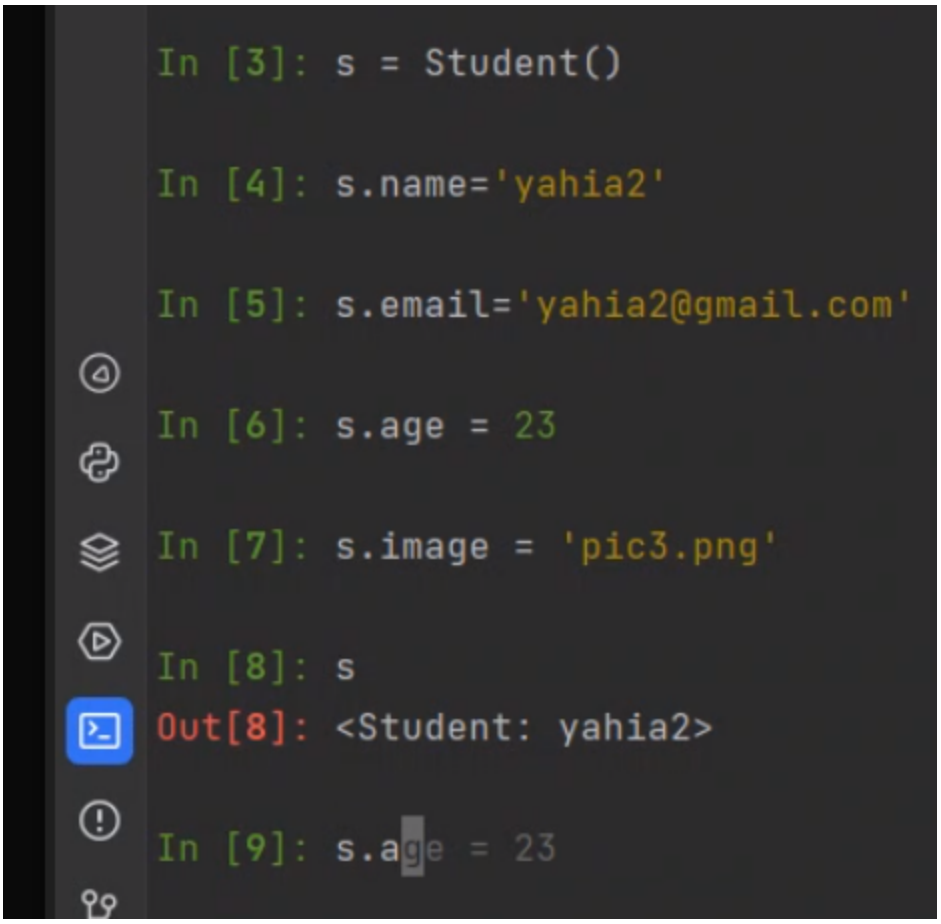
4) create an object from your table so you could add data to it : s = Student()

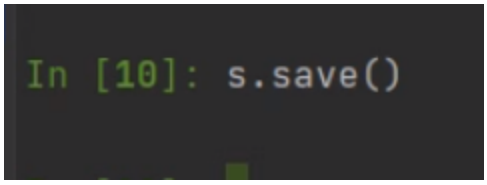
5) then insert the data you want :

6) then s.save() to save all this and check it on your server



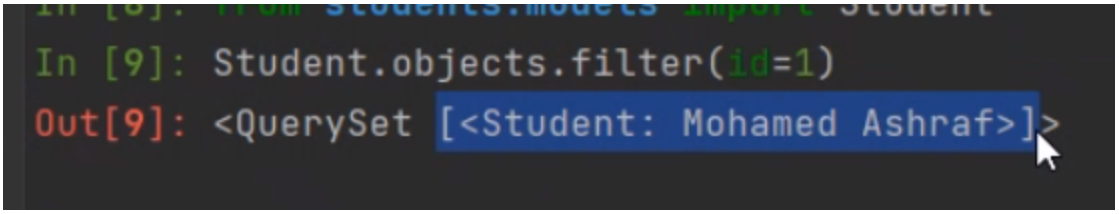






-To get a specific object :

Student.objects.filter(id=1)



-To get specific objects

for ex : get students id > 2

Student.objects.filter(id__gt=2)

id underscore underscore gt

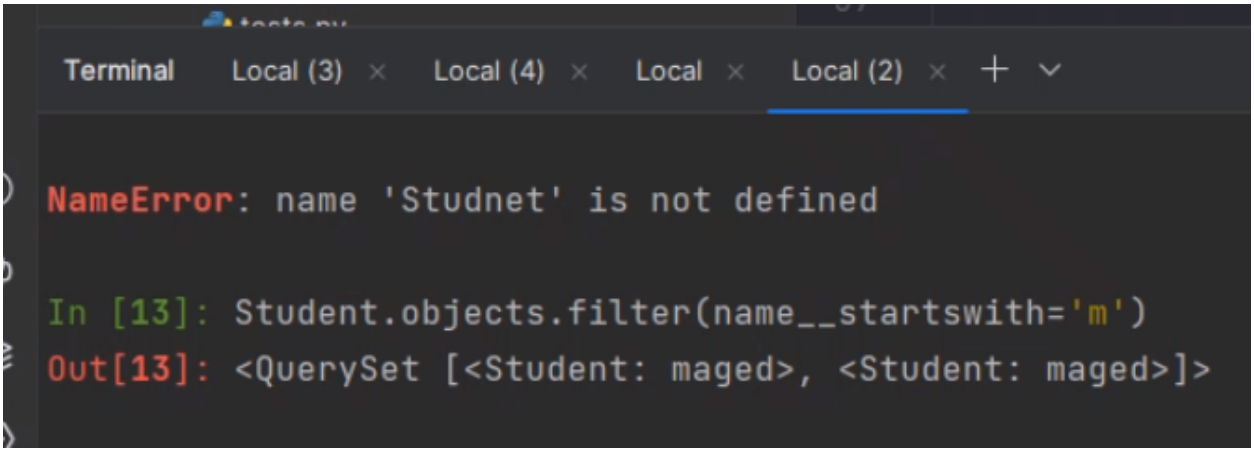
```
In [11]: Student.objects.filter(id__gt=2)
    ...:
Out[11]: <QuerySet [<Student: Osman>, <Student: Yahia>, <Student: yahia2>, <Student: maged>, <Student: maged>]>
```

-to do this :

select * from students where name like ‘m%’;

Student.objects.filter(name__startswith=’m’)

-In shell :



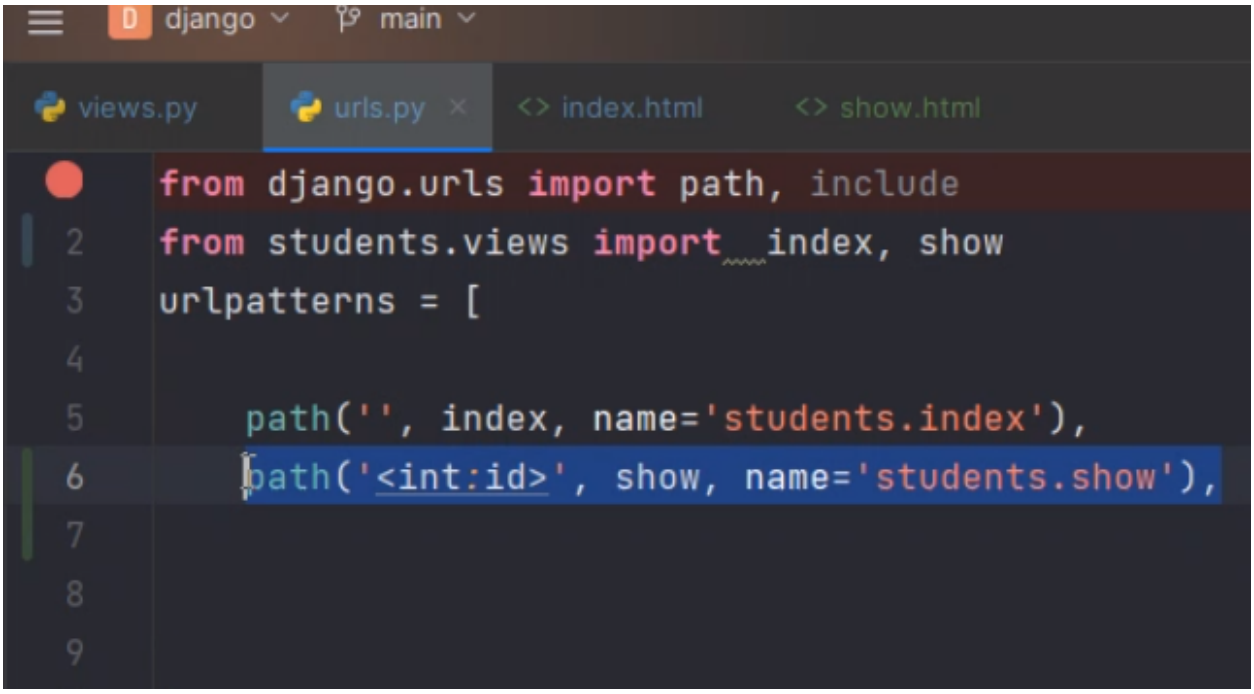
To use them in views.py :

from .models import Student t2rebn

```
14  def show(request, id):
15      student = Student.objects.get(id=id)
16      return render(request,  template_name: 'students/show.html', context={"student":student})
17
18
    new *
19  def delete(request, id ):
20      student = Student.objects.get(id=id)
21      student.delete()
```

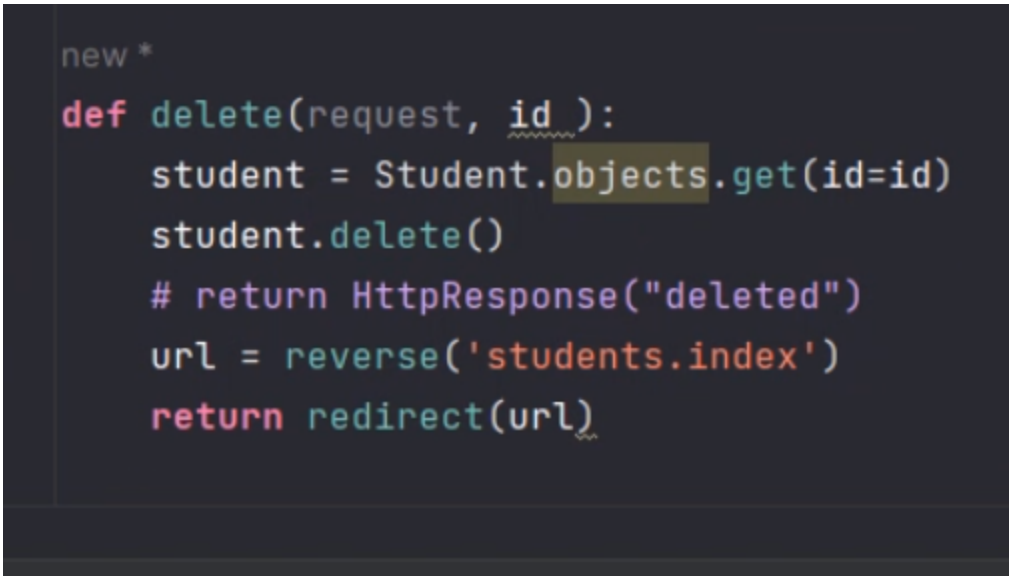
```
18
19
    new *
20  def delete(request, id ):
21      student = Student.objects.get(id=id)
22      student.delete()
23      # return HttpResponse("deleted")
24      return redirect('')
```

And in urls.py :



To redirect to a url back :

use the reverse function



-An advanced (more detailed) example of a DB model :

```
class Product(models.Model):

  x = [
      ('phones', 'phones'),
      ('computers', 'computers'),
      ('gym products' , 'gym products'),
  ]

  name = models.CharField( max_length=30, default='name', verbose_name='product name' )
  content = models.TextField(null=True, blank=True)
  price = models.DecimalField( max_digits=5, decimal_places=2, default=6.3)
  image = models.ImageField(upload_to='photos/%y/%m/%d', default='photos/20/12/2')
  active = models.BooleanField(default=True)
  category = models.CharField(max_length=30,null=True, blank=True, choices=x)

  def __str__(self):
    return self.name   #so they could be called iphone6,.. not product.object

class Meta:
#     verbose_name = ('hamada')
    ordering = ['-price']
```

-**Verbose** i**s the name of the column that appears on server.**

-In **DecimalFIeld**, **max_digit=5** is the **num of unit digits before the dot …..**
decimal_places =3 is the **num of decimal digits after the dot EX : 12200.111**