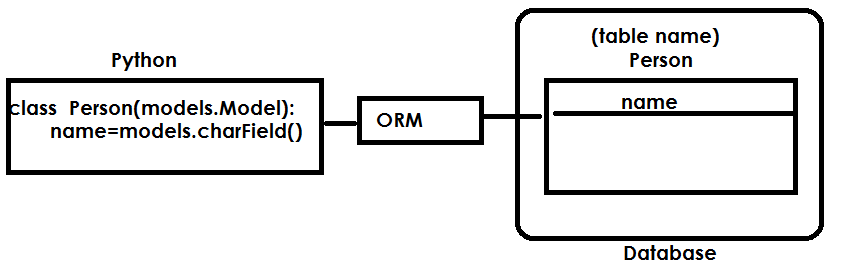
**Syntax:**

**class <class-name>(models.Model):**

variable-name/field-name=datatype

variable-name/field-name=datatype

variable-name/field-name=datatype

****

**model field data types**

**AutoField :** An **[IntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/" \l "django.db.models.IntegerField" \o "django.db.models.IntegerField)** that automatically increments according to available IDs. You usually won’t need to use this directly; a primary key field will automatically be added to your model if you don’t specify otherwise.

**IntegerField**

An integer. Values from **-2147483648** to **2147483647** are safe in all databases supported by Django.

### BigIntegerField

A 64-bit integer, much like an **[IntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/" \l "django.db.models.IntegerField" \o "django.db.models.IntegerField)** except that it is guaranteed to fit numbers from **-9223372036854775808** to **9223372036854775807**.

**FloatField**

A floating-point number represented in Python by a **float** instance.

**CharField**

A string field, for small- to large-sized strings

**DateTimeField**

A date and time, represented in Python by a **datetime.datetime** instance. Takes the same extra arguments as **[DateField](https://docs.djangoproject.com/en/4.0/ref/models/fields/" \l "django.db.models.DateField" \o "django.db.models.DateField)**.

**How to create model?**

Open application folder.

Open models.py

from django.db import models

# Create your models here.

class Emp(models.Model):

empno=models.IntegerField()

ename=models.CharField(max\_length=20)

salary=models.FloatField()

**How to configure database connection?**

Django framework provides a default database called **SQLite.**

This database is installed along with django framework.

**Open project folder/project configuration folder**

Open settings.py module

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

**makemigrations**

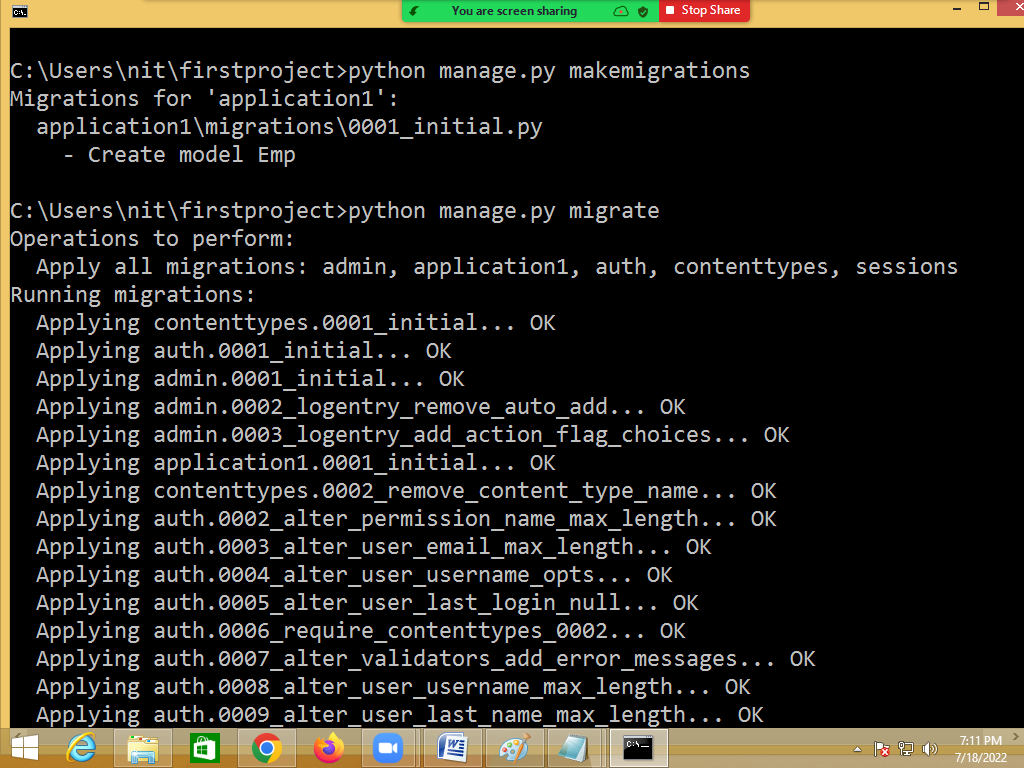
**python manage.py makemigrations <application-name>**

makemigrations command generate equal SQL statements.

**migrate**

This command create tables for each model represented in models.py

**python manage.py migrate**



**Inside database table is created with applicationname\_modelname**

How to perform CRUD operations on model?

1. Inserting
2. Updating
3. Deleting
4. Reading

All these operations are done using methods of model class.

**Create row/record**

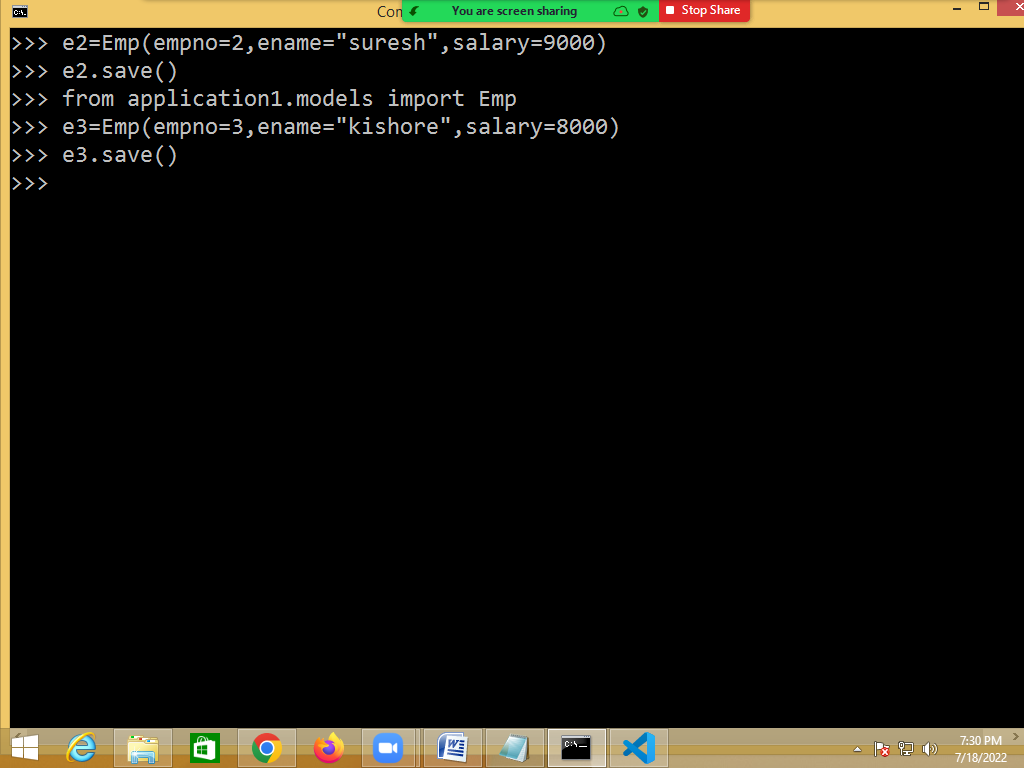
This creation for row or record is done in two ways.

1. Create method
2. Creating model object

Creating row is nothing but creating object of model class.

**Open interactive mode/shell**

**Python manage.py shell**

****