Quick Overview of

Meta class

in Django Models

part -1

Introduction to Meta class

In Django, Meta class is a inner class of Model class, that is used to change the behaviour of model.

This is how we define Meta class in django models class.

```python
from django.db import models

class Student(models.Model):
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=277)

  class Meta:
    ordering = ["name"]
```

and here we can override the built-in attribute and customised it’s behaviour.

Built-in Attributes in Meta Class

Meta class provides various built-in attributes, that we can override and change any model behaviours

Let’s explore each of them :

abstract

If abstract is True. It means it will be an abstract class, so django will not create a database table for this model. So we can only inherit this model to other model, by default it's False

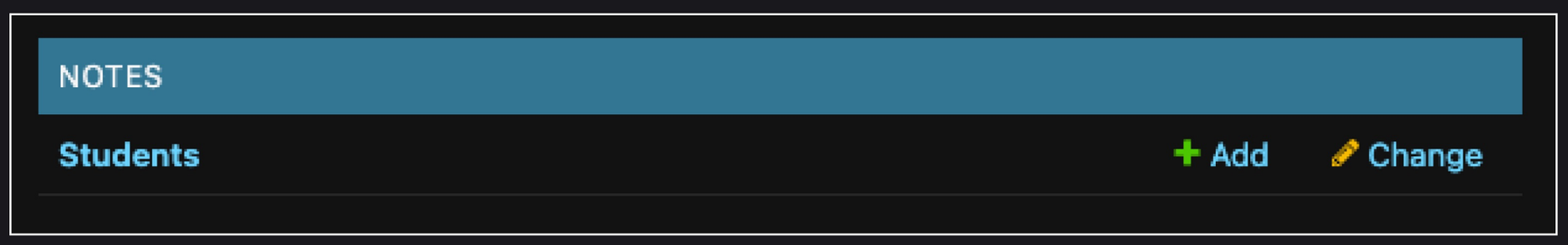
```python
from django.db import models

class BaseModel(models.Model):
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        abstract = True


class Student(BaseModel):
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=277)
```

here we have inherit the BaseModel, so it will automatically inherit, all it's fields.

here we can add abstract True in Meta class, so it will be an abstract class

verbose_name

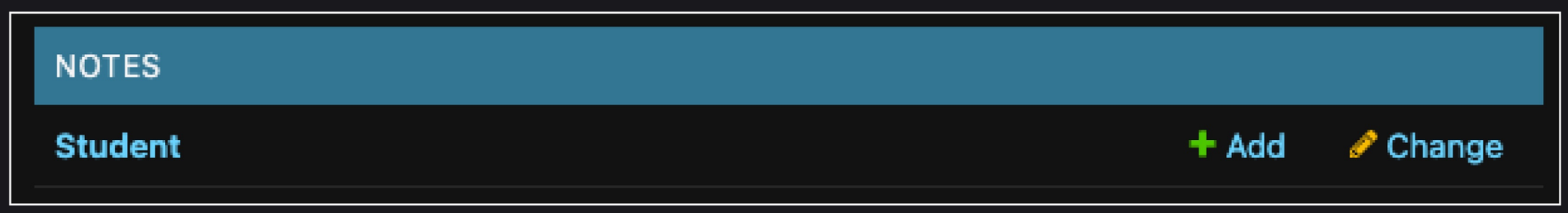
verbose_name attribute provides a human readable name of model. It is used primarily for display purposes in the Django admin interface and other places.




```python
from django.db import models

class Student(BaseModel):
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=277)

  class Meta:
    verbose_name = "Student"
```

verbose_name_plural

verbose_name_plural attribute provides a **plural** name of model. If this isn't given, Django will use verbose_name + "s".




```python
from django.db import models

class Student(BaseModel):
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=277)

  class Meta:
    verbose_name_plural = "Student"
```

ordering

ordering attribute, provides an ordering of an objects, when we retrieve objects from model.

```python
from django.db import models

class Student(BaseModel):
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=277)

  class Meta:
    ordering = ['-created_at']
```

and here we can define the list of fields that we used for ordering, here we specify ‘ -created_at ‘ its means descending order.

#pythonworld

For more django
related content

Subscribe

To My Youtube Channel

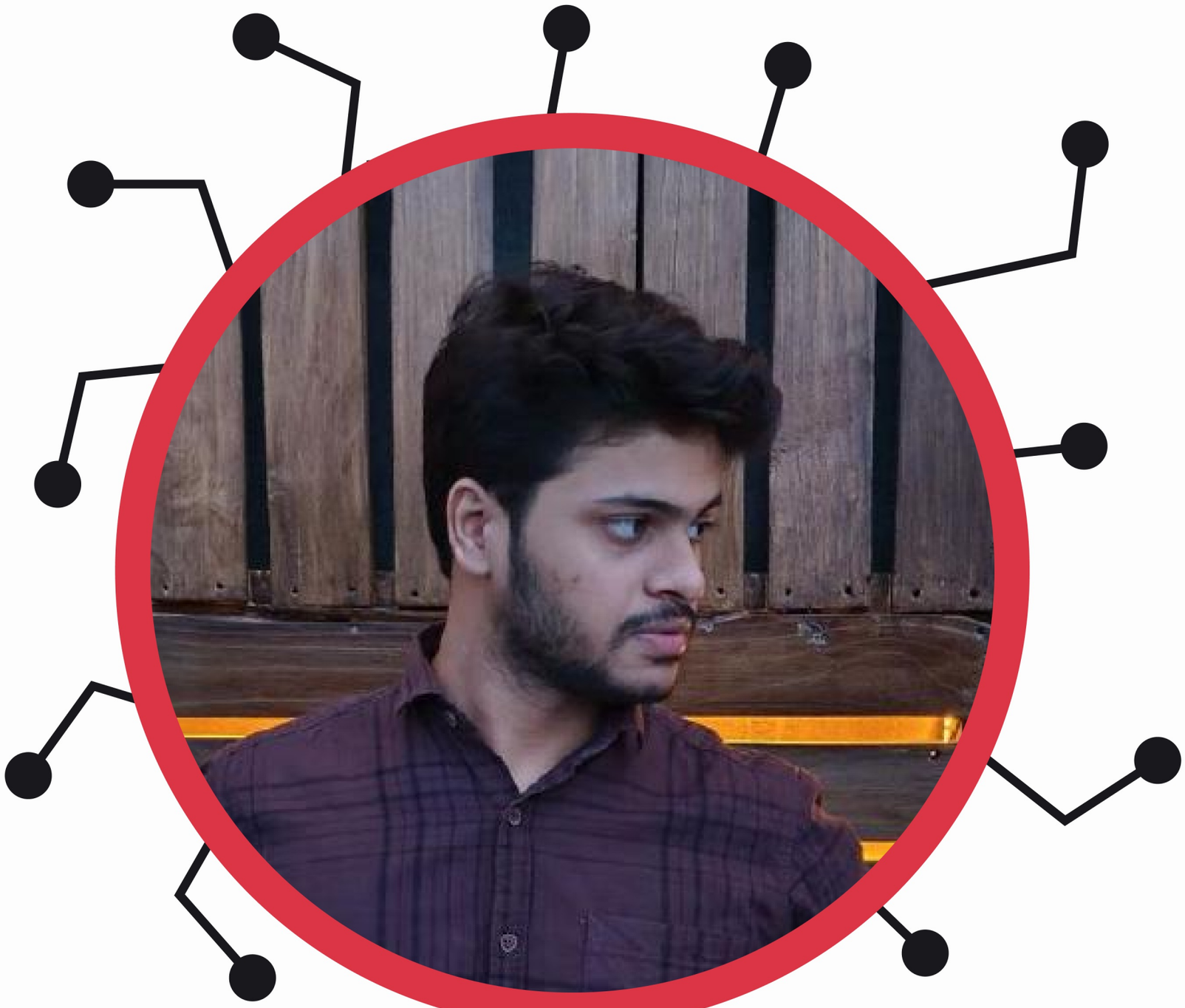
Link in bio

#pythonworld

DID YOU FIND THIS

HELPFUL

Let me know in the comment



Aashish Kumar

Software Engineer

Follow for more ❤️