Topic: **Django Signals**

Question 3:

By default do django signals run in the same database transaction as the caller? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

Answer:

Django signals do indeed operate in the same database transaction as the caller by default. In particular, signal that are triggered by operation are processed within the same transaction context as the signals themselves, such as post_save, pre_save, post_delete, and pre_ delete.

The caller's database transaction may be impacted by a signal handler error, which would reverse the entire operation. On the other hand, the signal's activities are also rolled back if the caller's transaction fails.

We may construct a straightforward example where an exception is triggered inside the signal handler, resulting in the caller's database transaction rolling back, to demonstrate this point definitively. To verify when the transaction has successfully committed, we'll utilize Django's transaction.on_commit().

Program:
Following Code Demonstrate's Django Signals Running in the Same Database Transaction:
#models.py

```
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.db import transaction

class MyModel(models.Model):
    name = models.CharField(max_length=100)
@receiver(post_save, sender=MyModel)
def my_signal_handler(sender, instance, **kwargs):
    print("Signal handler called.")
      raise Exception("Something went wrong in the signal handler!")
```

#views.py

```
from django.shortcuts import HttpResponse
from django.db import transaction
from .models import MyModel

def create_model_instance(request):
    try:
        with transaction.atomic():
            instance = MyModel.objects.create(name="Test Instance")
            print("Model instance created successfully.")
            transaction.on_commit(lambda: print("Transaction successfully
committed."))
        return HttpResponse("Instance created")
    except Exception as e:
        print(f"Error occurred: {e}")
        return HttpResponse("Failed to create instance")
```

Output:

```
Model instance created successfully.
Signal handler called.
Error occurred: Something went wrong in the signal handler!
```