**Questions for Django Trainee at Accuknox**

[Topic: **Django Signals**](https://docs.djangoproject.com/en/3.2/topics/signals/)

**Question 1**: By default are django signals executed synchronously or asynchronously? 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.

**Question 2**: Do django signals run in the same thread 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.

**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.

QUESTION 1:

By default, Django signals are executed synchronously. This means that the signal handler runs in the same process and thread as the signal sender, and the sender waits for the handler to complete before continuing execution.

**# models.py**

from django.db import models

from django.db.models.signals import post\_save

from django.dispatch import receiver

import time

class MyModel(models.Model):

name = models.CharField(max\_length=100)

# Signal handler

@receiver(post\_save, sender=MyModel)

def my\_handler(sender, instance, \*\*kwargs):

print("Signal handler started.")

time.sleep(5) # Simulate a long-running task

print("Signal handler finished.")

# Trigger the signal by saving an instance

instance = MyModel(name="Test")

instance.save()

print("Save method completed.")

QUESTION 2:

Yes, Django signals run in the same thread as the caller by default.

# models.py

from django.db import models

from django.db.models.signals import post\_save

from django.dispatch import receiver

import threading

class MyModel(models.Model):

name = models.CharField(max\_length=100)

# Signal handler

@receiver(post\_save, sender=MyModel)

def my\_handler(sender, instance, \*\*kwargs):

print(f"Signal handler thread: {threading.current\_thread().name}")

# Trigger the signal by saving an instance

instance = MyModel(name="Test")

print(f"Caller thread: {threading.current\_thread().name}")

instance.save()

QUESTION 3:

Yes, Django signals run in the same database transaction as the caller by default, especially when signals are triggered by database operations like save() or delete().

# models.py

from django.db import models, transaction

from django.db.models.signals import post\_save

from django.dispatch import receiver

class MyModel(models.Model):

name = models.CharField(max\_length=100)

# Signal handler

@receiver(post\_save, sender=MyModel)

def my\_handler(sender, instance, \*\*kwargs):

print(f"Signal handler in transaction: {transaction.get\_connection().in\_atomic\_block}")

# Trigger the signal within a transaction block

with transaction.atomic():

instance = MyModel(name="Test")

instance.save()

print(f"Caller in transaction: {transaction.get\_connection().in\_atomic\_block}")