**Question 1: Are Django signals executed synchronously or asynchronously by default?**

**Answer:**  
By default, Django signals are executed **synchronously**. This means that when a signal is sent, all connected signal handlers are executed immediately before the code moves forward.

**Code :**   
  
from django.dispatch import Signal

import time

# Define a custom signal

my\_signal = Signal()

# Signal handler

def slow\_handler(sender, \*\*kwargs):

print("Handler started")

time.sleep(2) # Simulate slow execution

print("Handler finished")

# Connect the signal

my\_signal.connect(slow\_handler)

# Emit the signal

print("Emitting signal")

my\_signal.send(sender=None)

print("Signal emission complete")

### Question 2: Do Django signals run in the same thread as the caller?

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

Code :  
from django.dispatch import Signal

import threading

# Define a signal

my\_signal = Signal()

# Signal handler

def handler(sender, \*\*kwargs):

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

# Connect the signal

my\_signal.connect(handler)

# Emit the signalprint(f"Main thread:   
{threading.current\_thread().name}")  
my\_signal.send(sender=None)

**Question 3: Do Django signals run in the same database transaction as the caller by default?**

**Answer:**

By default, Django signals run in the same database transaction as the caller, meaning changes made in a signal handler are part of the same transaction.

**Code :**

from django.db import transaction

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

from django.contrib.auth.models import User

from django.dispatch import receiver

@receiver(post\_save, sender=User)

def signal\_handler(sender, instance, created, \*\*kwargs):

print("Signal handler called")

if created:

raise Exception("Simulating error in signal handler")

try:

with transaction.atomic():

print("Creating user")

User.objects.create(username="testuser")

except Exception as e:

print(f"Exception occurred: {e}")

# Check if the user was created

print("Users in database:", User.objects.count())

Custom Classes in Python

class Rectangle:

def \_\_init\_\_(self, length: int, width: int):

self.length = length

self.width = width

def \_\_iter\_\_(self):

# Create a generator that yields the desired format

yield {'length': self.length}

yield {'width': self.width}

# Example usage:

rect = Rectangle(10, 5)

for attribute in rect:

print(attribute)  
  
  
  
This is my submission.  
I have basic knowledge of python, don’t have knowledge of Django.  
  
The solution that I provided above are copied but I learned about them, that what they are and how they are used.  
I will surely work and learn more of this topic.  
  
Thank you for the opportunity.