**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.

**Answer 1 :** By default, Django signals are executed **synchronously**. This means that when a signal is sent, the receiver functions are executed in the same thread and must complete before the next line of code is executed.

To demonstrate this, let's create a basic Django setup where we send a signal and observe its synchronous nature.

def user\_saved(sender, instance, \*\*kwargs):

print("Signal received. Starting long task...")

time.sleep(5)

# Simulates a long-running task

print("Long task completed.") # views.py or Django shell from django.contrib.auth.models import User print("Creating user...")

user = User.objects.create\_user(username='testuser', password='testpass')

print("User creation done. Continuing...")

**When you run the code, the following output would be observed:**

Creating user...

Signal received. Starting long task...

(Long task runs for 5 seconds)

Long task completed.

User creation done. Continuing...

**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.

**Answer 2:** Yes, by default, Django signals run in the same thread as the caller. This means the signal handler (receiver function) is executed within the same thread that triggers the signal.

Here's a code snippet to demonstrate this behavior:

def user\_saved(sender, instance, \*\*kwargs):

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

print(f"Before saving user, running in thread: {threading.current\_thread().name}")

user = User.objects.create\_user(username='testuser', password='testpass')

print("User created.")

**When you run the code, the following output would be observed:**

Before saving user, running in thread: MainThread

Signal received in thread: MainThread

User created.

**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 3:** By default, Django signals **do not** run in the same database transaction as the caller. Signals like post\_save are triggered after the database transaction is committed, but they are not inherently part of the same transaction. If you want signals to run within the same transaction, you would need to use the transaction.on\_commit() hook.