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

Ans:The django signals are executed synchronously.This means that when a signal is sent, the signal handlers are executed immediately, in the same thread, and before control returns to the caller.

Code Snippet:

```
import time
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.contrib.auth.models import User

# Signal receiver function
@receiver(post_save, sender=User)
def user_saved_handler(sender, instance, **kwargs):
    print("Signal handler started.")
    time.sleep(5)
    print("Signal handler completed after 5 seconds.")

print("Saving user instance...")
user = User.objects.create(username='test_user')
print("User instance saved.")
```

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.

Ans:The Django signals run in the same thread as the caller. This means that when a signal is triggered, its handler runs in the same thread from which the signal was sent.

Code Snippet:

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

```
from django.contrib.auth.models import User

# Signal receiver function
@receiver(post_save, sender=User)
def user_saved_handler(sender, instance, **kwargs):
    print(f"Signal handler running in thread:
{threading.current_thread().name}")

def create_user():
    print(f"Main code running in thread:
{threading.current_thread().name}")
    user = User.objects.create(username='test_user')
    print("User creation done.")

create_user()
```

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.

Ans:Yes, by default, Django signals run in the same database transaction as the caller. This means that if a signal is triggered during a transaction, the signal handler will be executed within that same transaction, ensuring that any database operations performed inside the signal handler will either be committed or rolled back along with the main transaction.

Code Snippet:

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

# Signal receiver function
@receiver(post_save, sender=User)
def user_saved_handler(sender, instance, **kwargs):
    print(f"Signal handler: In transaction =
{transaction.get_connection().in_atomic_block}")

def create_user_in_transaction():
    with transaction.atomic():
```

```
        print(f"Before saving user: In transaction =
{transaction.get_connection().in_atomic_block}")
        user = User.objects.create(username='test_user')
        print(f"After saving user: In transaction =
{transaction.get_connection().in_atomic_block}")

create_user_in_transaction()
```