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.

Ans.

By default, Django signals run synchronously. This implies that the signal's sender will pause execution until all connected receivers have completed their tasks.

```
from django.db.models.signals import post_save

from django.dispatch import receiver

from django.contrib.auth.models import User


@receiver(post_save, sender=User)

def on_user_save(sender, instance, **kwargs):

    print(f"User '{instance.username}' was saved. Signal executed synchronously.")

new_user = User(username='Joseph')

new_user.save()
```

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.

Django signals are executed in the same thread as the caller. This behavior can be verified by comparing the thread IDs in both the main thread and the signal handler.

```
from django.db.models.signals import post_save

from django.dispatch import receiver

from django.contrib.auth.models import User

import threading


@receiver(post_save, sender=User)

def on_user_save(sender, instance, **kwargs):

    print("Signal received in thread:", threading.get_ident())

new_user = User(username='Joseph')

new_user.save()

print("Main thread:", threading.get_ident())
```

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.

Django signals are executed within the same database transaction as the caller. This guarantees that all operations performed within the signal handler are included in the same atomic transaction as the caller's database operation.

```
from django.db.models.signals import post_save

from django.dispatch import receiver

from django.contrib.auth.models import User

from django.db import transaction


@receiver(post_save, sender=User)

def user_saved(sender, instance, **kwargs):

    with transaction.atomic():

        print("Signal executed within the same transaction")

new_user = User(username='Joseph')

new_user.save()
```