Topic: **Django 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.

Answer:

Django signals are processed synchronously by default. This indicates that when a signal is sent, the signal handler (or receiver) is carried out in the same thread as the signal-triggering process or request-response cycle.

We can show a synchronous delay in the signal receiver to support this. We shall demonstrate that the handler sleeps for a short while after the signal is triggered, delaying the method that transmits the signal from finishing.

Following Code proves that Django signals are synchronous by default:

Program:

 # models.py

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

class MyModel(models.Model):
    name = models.CharField(max_length=100)

@receiver(post_save, sender=MyModel)
def my_signal_handler(sender, instance, **kwargs):
    print("Signal received, starting delay...")
    time.sleep(5)  # Simulate a long-running task
    print("Signal handler finished after delay.")
```

 # views.py

```
from django.shortcuts import HttpResponse
from .models import MyModel

def create_model_instance(request):
    print("Creating MyModel instance...")
    instance = MyModel.objects.create(name="Test Instance")
    print("MyModel instance created, returning response...")
    return HttpResponse("Instance created")
```

Output Expected:

1. When we access the /create/ URL, the view creates a MyModel instance, which triggers the post_save signal.
2. We will observe in the console that the signal handler causes a 5-second delay before the response is sent back.

Output:

```
Creating MyModel instance...
Signal received, starting delay...
Signal handler finished after delay.
MyModel instance created, returning response…
```