Yes, Django signals run in the same thread as the caller. This means that the signal handler executes within the same thread that triggered the signal, ensuring that the thread's context and resources are shared.

- Snippet of Code is as follows;

signal.py

```python
import threading
from django.db.models.signals import pre_save
from django.dispatch import receiver
from django.contrib.auth.models import User

@receiver(pre_save, sender=User)
def pre_save_user_handler(sender, instance, **kwargs):
    print(f"Signal handler running in thread: {threading.get_ident()}")


# Print the current thread ID before triggering the signal
print(f"Code running in thread: {threading.get_ident()}")

'''
Output of the following code will be;

Signal handler running in thread: 139851784163584
Code running in thread: 139851784163584

'''
```

1. Thread Identification: The threading.get_ident() function prints the current thread's identifier.

2. Same Thread Confirmation: By comparing the thread ID before and during the signal handling, you confirm that both are executed in the same thread, demonstrating that Django signals run in the same thread as the caller.