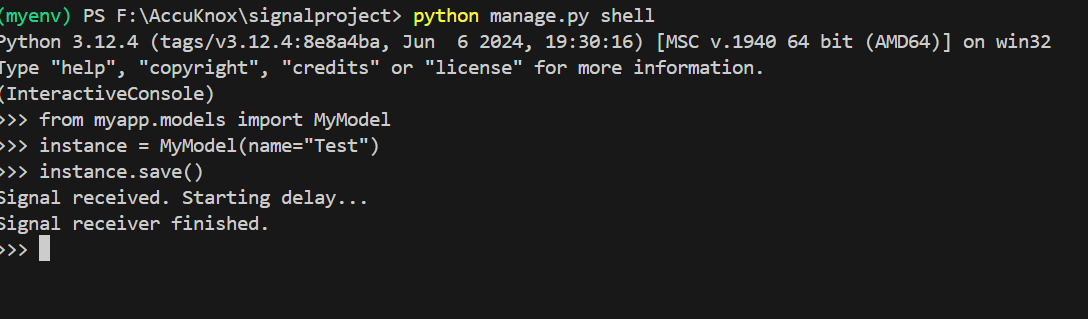
**Code will shared in github.**

Question-1-Solution:-

Django signals are executed synchronously. This means that when a signal is sent, the receiver function connected to that signal will be executed in the same thread before the original function finishes its execution.

To prove this, let's look at an example where we connect a receiver to the post\_save signal and introduce a delay in the receiver function. As the signal runs synchronously by default, the delay in the receiver will block the original function.

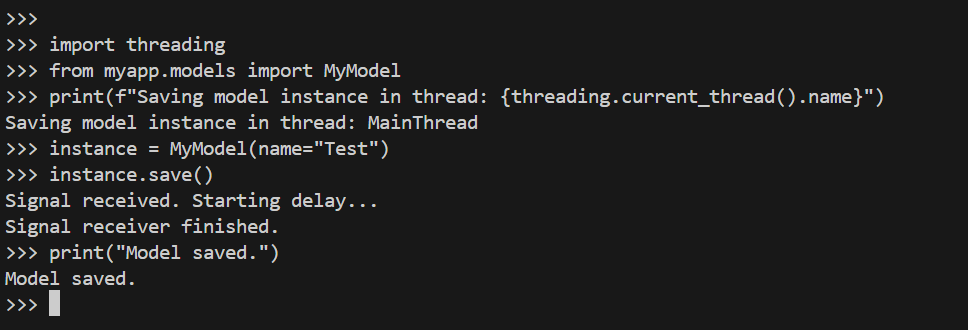
Output-



Question-2-Solution :-

Yes, by default, Django signals run in the same thread as the caller. This means that when a signal is triggered, the signal handler (receiver) executes in the same thread as the function that sent the signal.

Output:-



Question- 3 – Solution :-

Django signals do not run in the same database transaction as the caller unless explicitly managed using transaction.atomic. This means that Django signals can fire before or after a transaction has been fully committed, depending on the signal used (pre\_save, post\_save, etc.). To ensure signals are executed within a transaction, you need to manually wrap the signal handling logic inside a transaction block.

Output :-

