By default, Django signals run in the same database transaction as the caller. This means that if a signal handler modifies the database, those changes are part of the same transaction and can be rolled back if the transaction is rolled back.

- Snippet of Code is as Follows;

signal.py

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

# Signal handler
@receiver(pre_save, sender=User)
def pre_save_user_handler(sender, instance, **kwargs):
    print(f"Signal handler modifying User: {instance.username}")
    instance.username = 'modified_in_signal'

# Create user and trigger signal within a transaction
1 usage
def create_user_with_signal():
    with transaction.atomic():
        user = User.objects.create(username='original_username')
        print(f"User before commit: {User.objects.get(pk=user.pk).username}")
        raise Exception("Rollback to show signal changes are not committed")

# Call the function to execute
create_user_with_signal()
```

1. Signal Handler: Modifies the username of a User before saving.

2. Transaction Management: Creates a User and prints the username, demonstrating the effect of the signal handler within the same transaction.

3. Rollback: The transaction is rolled back, so changes made by the signal handler are not committed.