## ***Django Assignment***

Deepak Sahu

## 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 synchronous by default.

Code -

import time

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.dispatch import receiver

@receiver(post\_save, sender=User)

def slow\_signal\_handler(sender, instance, created, \*\*kwargs):

print("Signal handler started.")

time.sleep(5)

print("Signal handler finished.")

if \_name\_ == "\_main\_":

print("Creating a new user.")

user = User.objects.create(username="test\_user")

print("User creation finished.")

**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.

**Answer** - Yes django signals run in the same thread as the caller.

Code -   
  
import threading

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.dispatch import receiver

@receiver(post\_save, sender=User)

def signal\_handler(sender, instance, created, \*\*kwargs):

# Print the thread ID in the signal handler

print(f"Signal handler running in thread: {threading.current\_thread().ident}")

if \_name\_ == "\_main\_":

# Print the thread ID in the caller

print(f"Code triggering signal running in thread: {threading.current\_thread().ident}")

user = User.objects.create(username="test\_user")

**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.

**Answer** - Yes, Django signals run in the same database transaction as the caller by default.

Code -

from django.db import transaction

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.dispatch import receiver

from django.db import connection

@receiver(post\_save, sender=User)

def signal\_handler(sender, instance, created, \*\*kwargs):

print("Signal handler: Trying to create another user.")

User.objects.create(username="signal\_user")

print("Signal handler: User created.")

if \_name\_ == "\_main\_":

try:

with transaction.atomic():

print("Main transaction: Creating a new user.")

user = User.objects.create(username="main\_user")

print("Main transaction: User created.")

raise Exception("Forcing a rollback.")

except Exception as e:

print(f"Exception caught: {e}")

print("Users in the database:", User.objects.all())

## Custom Classes in Python

**Description:** You are tasked with creating a Rectangle class with the following requirements:

1. An instance of the Rectangle class requires length:int and width:int to be initialized.
2. We can iterate over an instance of the Rectangle class
3. When an instance of the Rectangle class is iterated over, we first get its length in the format: **{'length': <VALUE\_OF\_LENGTH>}** followed by the width **{width: <VALUE\_OF\_WIDTH>}**

**Answer -**

In the below code i have written the custom class Rectangle with length and width as an integer.

We have initialized length and width then with the help of \_iter\_() we can iterate through the Rectangle class object and use yield to get length then width in the dictionary format.

**Code** -

class Rectangle:

def \_init\_(self, length: int, width: int):

self.length = length

self.width = width

def \_iter\_(self):

yield {"length": self.length}

yield {"width": self.width}

rectangle = Rectangle(10, 5)

for dim in rectangle:

print(dim)