**TOPIC : Django Signals**

**Q1.** 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.

**A1:**Django signals are executed synchronously by default when signal is sent the receiver are executed.

**Code:**

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

from django.dispatch import receiver

from django.contrib.auth.models import user

def saved\_client (sender,instance,created,\*\*kwargs):

print(f”saved client: {instance.username}”)

client=Client (clientname=’harsha’)

saved.client()

print(“this prints after client is saved and signal is executed “)

**explanation:** the client\_saved function is executed synchronously after the saved.client().

**Q2.** 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.

**A2:** Yes, Django signals run in the same thread as the caller by default.

**Code:**

Import threading

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

from django.dispatch import receiver

from django.contrib.auth.models import user

def saved\_client (sender,instance,created,\*\*kwargs):

print(f”saved client: {instance.username}”)

print(f”current thread : {threading.current\_thread().name}”)

def client\_save():

client=Client(username=’harsha’)

client.save()

client\_user()

**explanation:** the function displays the current thread name ,which will be the same as the thread that called client\_user().it conforms that the signal is processed in the same thread.

**Q3.** 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.

**A3:** It runs the same database transaction as the caller

Code :

from django.db import transaction

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

from django.dispatch import receiver

from django.contrib.auth.models import user

from django.db import models

def client\_saved (sender,instance,created,\*\*kwargs):

print(f”saved client: {instance.username}”)

if created:

raise Exception(“simulating an error”)

try:

with transaction.atomic():

client=Client(username=’harsha’)

client.save()

except Exception as e:

print(f” Excxeption occurred:{e}”)

expalanation: function raises an exception , simulating an error ,this happens within a transaction block .this demonstrates that the signal runs within the same transaction context.

**TOPIC 2: Custom Classes in Python**

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

An instance of the Rectangle class requires length:int and width:int to be initialized.

We can iterate over an instance of the Rectangle class

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>}

**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 dimension in rectangle:

print(dimension)

**explanation:**

the methods is used to initializes the length and width attributes and allows iteration over the rectangle instance and the output will be

length:10

width:5

this gather all the requirements specified for the rectangle class.