

# **Questions for Django Trainee at Accuknox**

## [Topic: **Django Signals**](https://docs.djangoproject.com/en/3.2/topics/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: Yes the django signals are executed synchronously ie when a signal is sent and the connected receivers are executed immediately.

The code snippet is:

class MyModel(models.Model):

name = models.CharField(max\_length=100)

Def post\_save\_reciever(sender, instance, created, \*\*kwargs):

print(f”Started processing for: {instance.name}”)

time.sleep(2)

print(f”Finished processing for: {instance.name}”)

def create\_model\_instance(name):

instance = MyModel(name):

instance.save()

**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 runs in the same thread as the caller. This means the signals handlers execute in the same thread where the signal was sent.

Code snippet:

class MyModel(models.Model): name = models.CharField(max\_length=100)

def post\_save\_receiver(sender, instance, created, \*\*kwargs): print(f"Signal received in thread: {threading.current\_thread().name}")

def create\_model\_instance(name): instance = MyModel(name=name) instance.save()

**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: By default, Django signals do run in the same database transaction as the caller. If the transaction is rolled back, the changes made by the signal handlers will also be rolled back.

Code snippet:

class MyModel(models.Model): name = models.CharField(max\_length=100)

def post\_save\_receiver(sender, instance, created, \*\*kwargs): print(f"Processing {instance.name}")

def create\_model\_instance(name):

with transaction.atomic():

instance = MyModel(name=name)

instance.save()

raise Exception("Simulating an error")

## Topic: 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:

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}

rect = Rectangle(10, 5)

for dimension in rect:

print(dimension)