

# **Questions for Django Trainee at Accuknox**

## [Topic: **Django Signals**](https://docs.djangoproject.com/en/3.2/topics/signals/)

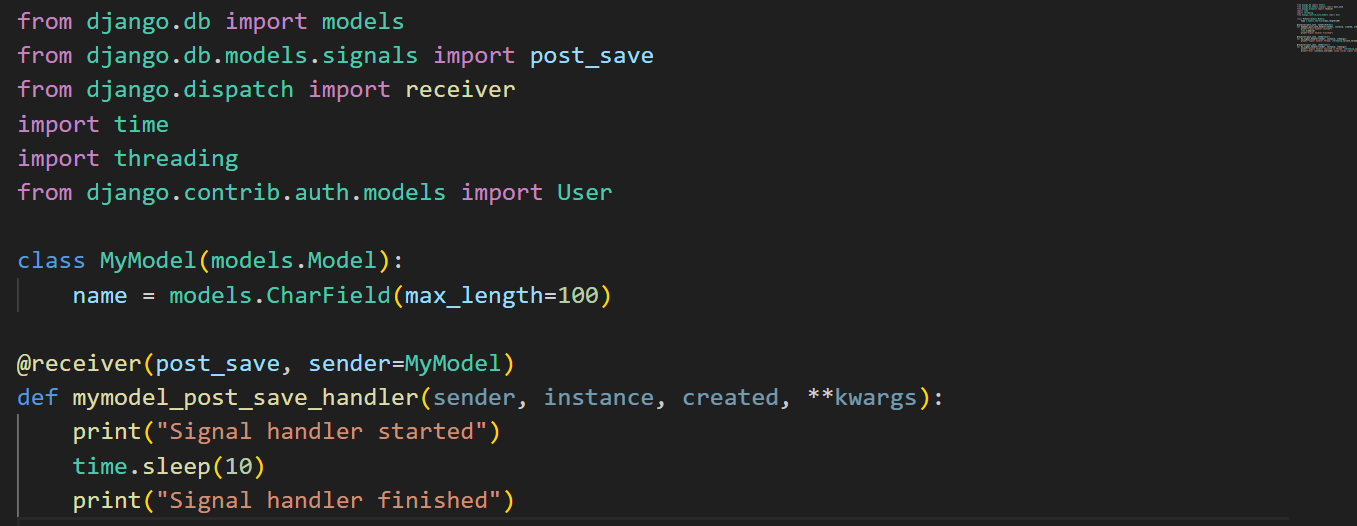
**Question 1**: By default are django signals executed synchronously or asynchronously? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

**Ans:**

Yes, by default, Django signals are executed synchronously. We'll use Django's post\_save signal. Since Django signals run in the same thread, time.sleep(10) will pause everything, including the HTTP response

.

**models.py**



**urls.py**

**A screen shot of a computer code

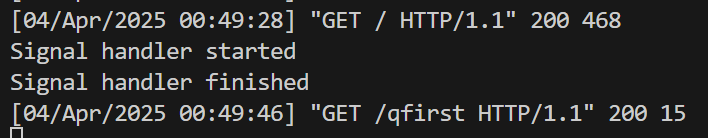
AI-generated content may be incorrect.**

**views.py**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Terminal Output:**

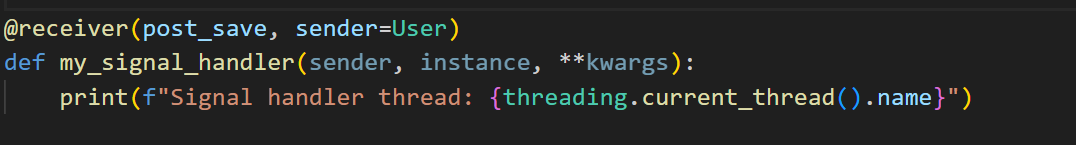
****

**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.

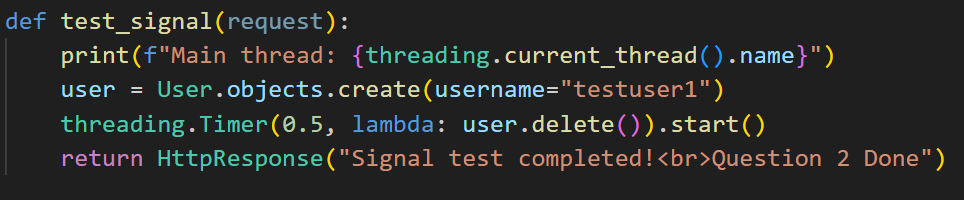
**Ans:**

Yes, Django signals run in the same thread as the caller by default. This means that when a signal is sent, the connected receiver function executes synchronously in the same thread as the signal sender.

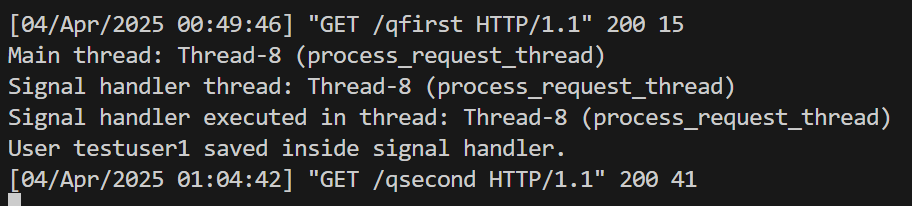
**models.py**

****

**views.py**

****

**Terminal Output:**

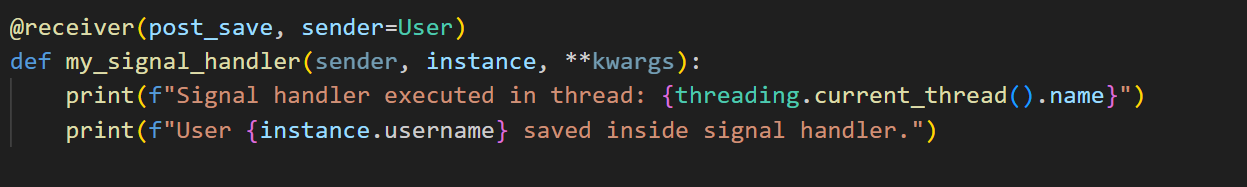
****

**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.

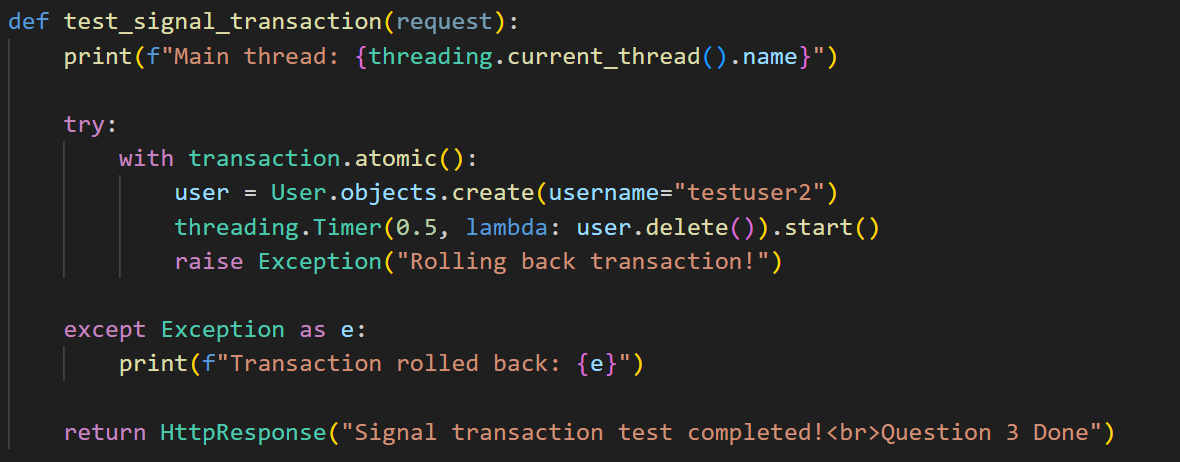
**Ans:**

By default, Django signals do not necessarily run in the same database transaction as the caller. This means that if a database transaction is rolled back, a signal handler might have already executed and performed irreversible actions.

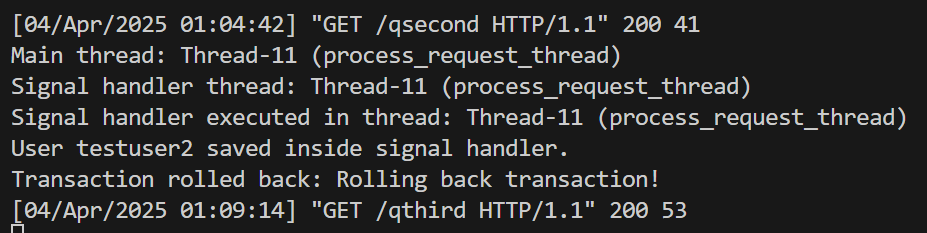
**models.py**



**views.py:**



**Terminal Output:**



## 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>}**

**Ans:**

class Rectangle:

    def \_\_init\_\_(self, l: int, w: int):

        self.length = l

        self.width = w

    def \_\_iter\_\_(self):

        yield {'length': self.length}

        yield {'width': self.width}

myRect = Rectangle(5, 3)

for val in myRect:

    print(val)