Django Trainee - AccuKnox: Django Signals Answers

# Topic: Django Signals

## Question 1: Are Django signals executed synchronously or asynchronously by default?

By default, Django signals are executed synchronously. This means they run in the same execution thread as the caller.

### Proof with Code:

```python
import time
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.contrib.auth.models import User

@receiver(post_save, sender=User)
def slow_signal_handler(sender, instance, **kwargs):
    print("Signal handler started")
    time.sleep(5)  # Simulate delay
    print("Signal handler finished")

# Create a user and measure execution time
start_time = time.time()
user = User.objects.create(username="testuser")
end_time = time.time()

print(f"Execution time: {end_time - start_time} seconds")
```
#### Expected Output:
```
Signal handler started
(Sleeps for 5 seconds)
Signal handler finished
Execution time: 5.x seconds
```
This proves that Django signals run synchronously by default.

## Question 2: Do Django signals run in the same thread as the caller?

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

### Proof with Code:

```python
import threading
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.contrib.auth.models import User

@receiver(post_save, sender=User)
def check_thread(sender, instance, **kwargs):
    print(f"Signal handler running in thread: {threading.current_thread().name}")

# Check the thread before triggering the signal
print(f"Main execution running in thread: {threading.current_thread().name}")

# Create a user and trigger the signal
user = User.objects.create(username="testuser")
```
#### Expected Output:
```
Main execution running in thread: MainThread
Signal handler running in thread: MainThread
```
Since both run in the same thread, Django signals execute in the same thread as the caller.

## Question 3: Do Django signals run in the same database transaction as the caller?

Yes, Django signals run in the same database transaction by default.

### Proof with Code:

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

@receiver(post_save, sender=User)
def rollback_test(sender, instance, **kwargs):
```

~~~~
    print("Signal received, raising an exception!")
    raise ValueError("Rolling back transaction")

try:
    with transaction.atomic():
        user = User.objects.create(username="testuser")
except ValueError:
    print("Transaction rolled back!")

# Check if the user was saved
print(User.objects.filter(username="testuser").exists())  # Should be False
```
#### Expected Output:
```
Signal received, raising an exception!
Transaction rolled back!
False
```
Since the transaction was rolled back, Django signals execute in the same database
transaction by default.

---

# Topic: Custom Classes in Python

## Rectangle Class Implementation with Iteration Support

To make an instance of `Rectangle` iterable, we need to define the `__iter__` method.

### Code Implementation:

```python
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}

# Example Usage:
rect = Rectangle(10, 5)

for item in rect:
~~~~

```
    print(item)
```

#### Expected Output:

```
{'length': 10}
{'width': 5}
```

This confirms that:

- The class initializes with `length` and `width`.
- The class supports iteration, yielding the length first and then the width.