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

Django signals are executed synchronously by default.

```
from django.core.signals import request_finished
from django.dispatch import receiver

@receiver(request_finished)
def  my_receiver(sender, **kwargs):
    print("Request finished signal received.")
    import time
    time.sleep(5) #
request_finished.send(sender=None)
print("After signal emission”)
```

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

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

```
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 post_save_receiver(sender, instance, **kwargs):
      print(f"Signal is running in thread: {threading.current_thread().name}")
def create_user():
      print(f"Main function is running in thread: {threading.current_thread().name}")
      user = User.objects.create(username='test_user')

if __name__ == '__main__':
      create_user()
```

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

Django signals run in the same database transaction as the caller when they are triggered by database operations by default.

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, 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 attribute in rectangle:
     print(attribute)
```