AccuKnox Assignment – Django Trainee

By – Sahitya Singh

Solution 1: - By default, Django signals are executed synchronously. This means that the signal handler is called immediately within the same execution flow as the sender. It can be seen in the following example:-

```
def user_saved(sender, instance, **kwargs):
        print("Signal received. Processing...")
        time.sleep(3)
        print("Signal processed.")

user = User.objects.create(username="accuknox")
print("User saved!")
```

In this the signal is triggered as soon as the user is saved, and the signal handler runs immediately. The delay caused by time.sleep(3) proves that signals are processed synchronously. If it were asynchronous, the "Signal processed." message would not cause a delay in the main thread.

Solution 2: - Yes, Django signals are run in the same thread as the caller. The following code proves this:-

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

user = User.objects.create(username="accuknox")
print(f"Caller is running in thread: {threading.current_thread().name}")
```

When the user is saved, both the caller and the signal handler will print the current thread name. Since the thread names match, this indicates that both the signal and the caller are executing in the same thread.

Solution 3: - Yes, Django signals run in the same database transaction as the caller, unless you manually alter the behavior. For example:-

```
@receiver(post_save, sender=User)
def user_saved(sender, instance, **kwargs):
        if transaction.get_autocommit():
                print("Signal is outside of a transaction.")
        else:
                print("Signal is within a transaction.")

with transaction.atomic():
        user = User.objects.create(username="accuknox")
```

In this code, we create a user within a transaction block (transaction.atomic()), and the signal checks if it's running within a transaction using transaction.get_autocommit(). Since Django signals share the same transaction context by default, the signal will print "Signal is within a transaction."

- Topic: Custom Classes in Python – Implementation: -

  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}

rect = Rectangle(length=5, width=3)
for attribute in rect:
        print(attribute)
```

When you iterate over an instance of Rectangle, it first yields the length and then the width as dictionaries.