Questions for Django Trainee at Accuknox

Topic: **Django 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.
 Yes by default django signals are synchronous. Signals are used to do actions when an event occurs. In the django project,I have a model called Blog and a signal handler function called pre_save_blog. So before the instance is saved in the database , the pre_save signal helps to create slug and pub_date for the blog .I added a sleep(5) in the pre_save_blog to show it is synchronous.

```
class Blog(models.Model):
   title = models.CharField(max_length=200)
   pub_date = models.DateTimeField('date published',null=True,blank=True)
   slug = models.SlugField(max_length=200, unique=True,null=True,blank=True)
   body = models.TextField()

@receiver(pre_save, sender=Blog)
def pre_save_blog(sender, instance, **kwargs):
   print("Handler function Thread Id-", threading.get_ident())
   print(f"pre_save_blog started at:
{datetime.datetime.now().strftime('%H:%M:%S')}")
   if not instance.slug:
       instance.slug = slugify(instance.title, allow_unicode=True)
       instance.pub_date = timezone.now()
       print("pre_save_blog")
       print("slug,",instance.slug)
       print("pub_date",instance.pub_date)
   sleep(5)
   print(f"pre_save_blog ended at:
{datetime.datetime.now().strftime('%H:%M:%S')}")
```

```
def index(request):
   print("Main Thread Id-", threading.get_ident())
```

```python
   if request.method == 'POST':
       title = request.POST.get('title')
       body = request.POST.get('body')
       print(f"1st print statement in View  before saving the instance started at: {datetime.datetime.now().strftime('%H:%M:%S')}")
       try:
           with transaction.atomic():
               blog_instance = Blog.objects.create(
               title=title,
               body=body
               )

       except Exception as e:
           blog =Blog.objects.filter(title = title)
           print("Checking if the blog is saved in database ?",blog.exists())
           print(e)

       print(f"2nd print statement in View  after saving the instance is saved started at: ended at: {datetime.datetime.now().strftime('%H:%M:%S')}")
       return redirect('blog_list')
   return render(request,'app/home.html')
```

The start time before the instance is saved and instance in pre_save_blog and after the instance is saved is captured .Below is the output from the terminal when you submit to create a new blog post from the frontend form.

```
Main Thread Id- 49000
1st print statement in View  before saving the instance started at: 22:36:16
Handler function Thread Id- 49000
pre_save_blog started at: 22:36:16
pre_save_blog
slug, sample-blog-5
pub_date 2024-10-31 05:36:16.442215+00:00
pre_save_blog ended at: 22:36:21
Creating an exception after saving the instance
Checking if the blog is saved in database ? False
Exception raised after saving the instance
2nd print statement in View  after saving the instance is saved started at: ended at:
22:36:21
```

Since the second print statement in the view is printed after all the actions in the pre_save_blog handler is executed, It shows that the signals are executed synchronously. However we can make signals asynchronous by using celery or other libraries.

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.
Yes, by default signals run in the same thread. In the above output from the terminal, we can see that both the Main thread and the signal thread is 49000.However there are ways we can do multithreading using the multithreading module.

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.
Yes, signal runs in the same database transaction if both the signal and the caller are in the same transaction.Then any exception in the handler should rollback all the transaction activities. We can see this by making the Blog instance creation as atomic by using transaction.atomic().Create an exception in post_save. Before printing the exception, try to check the instance is saved.We got False,as all the database transactions was rolled back.

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>}`**

```
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}

instance = Rectangle(10,20)
for i  in instance:
   print(i)
```