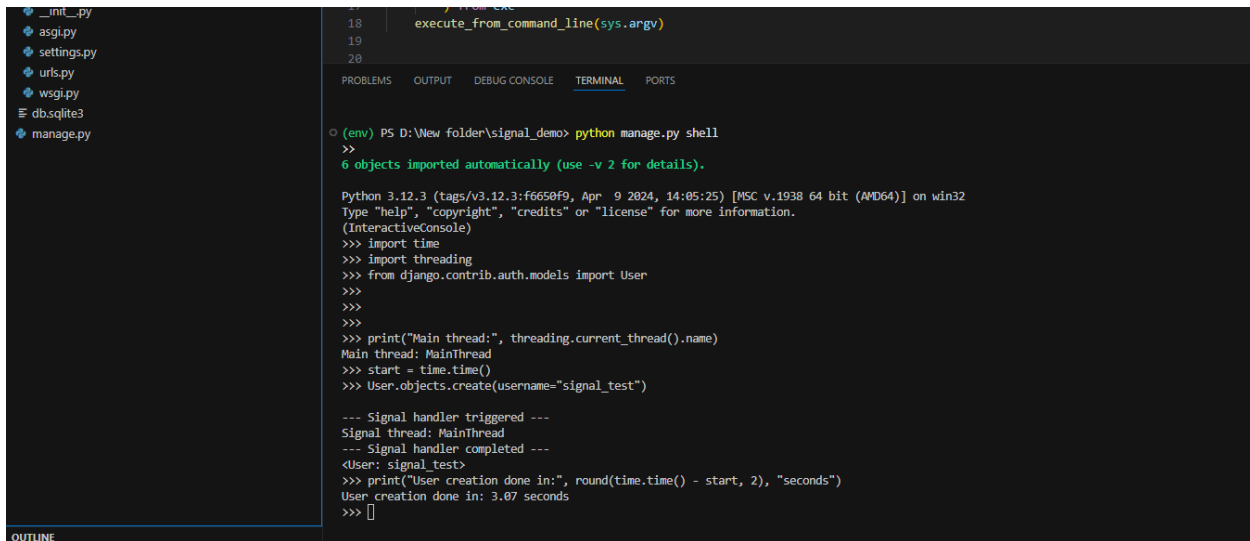


Questions for Django Trainee at Accuknox

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.

Answer: Django signals are executed synchronously by default.



```
18 execute_from_command_line(sys.argv)
19
20
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

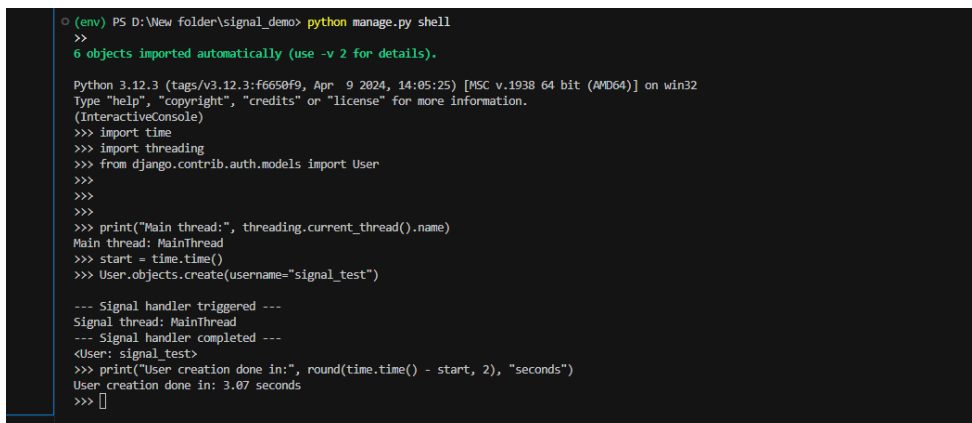
(env) PS D:\Vew folder\signal_demo> python manage.py shell
>>
6 objects imported automatically (use -v 2 for details).

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> import time
>>> import threading
>>> from django.contrib.auth.models import User
>>>
>>>
>>> print("Main thread:", threading.current_thread().name)
Main thread: MainThread
>>> start = time.time()
>>> User.objects.create(username="signal_test")

--- Signal handler triggered ---
Signal thread: MainThread
--- Signal handler completed ---
<User: signal test>
>>> print("User creation done in:", round(time.time() - start, 2), "seconds")
User creation done in: 3.07 seconds
>>> []
```

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.

Answer: Yes, The django signals run in the same thread as the caller by default.



```
(env) PS D:\Vew folder\signal_demo> python manage.py shell
>>
6 objects imported automatically (use -v 2 for details).

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> import time
>>> import threading
>>> from django.contrib.auth.models import User
>>>
>>>
>>> print("Main thread:", threading.current_thread().name)
Main thread: MainThread
>>> start = time.time()
>>> User.objects.create(username="signal_test")

--- Signal handler triggered ---
Signal thread: MainThread
--- Signal handler completed ---
<User: signal test>
>>> print("User creation done in:", round(time.time() - start, 2), "seconds")
User creation done in: 3.07 seconds
>>> []
```

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.

Answer: By default Django signals run in the same database transaction as the caller.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

>>> from django.contrib.auth.models import User
>>>
>>> try:
...     User.objects.create(username="rollback_test_2")
... except Exception as e:
...     print("Caught exception:", e)
...

--- Signal handler triggered ---
Signal thread: MainThread
--- Signal handler completed ---
<User: rollback_test_2>
>>> print("User saved?", User.objects.filter(username="rollback_test_2").exists())
User saved? True
>>>
```

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>}`

```
[ ]: #Description: You are tasked with creating a Rectangle class with the following requirements:  
  
#An instance of the Rectangle class requires length:int and width:int to be initialized.  
#We can iterate over an instance of the Rectangle class  
#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}
```

```
[3]: 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}
```

```
[8]: r = Rectangle(5, 10)  
      print("length:",r.length,"width:",r.width)  
  
      length: 5 width: 10
```