Django는 주로 요청-응답 모델에 기반한 웹 프레임워크이지만, 실시간으로 실행되는 함수를 구현하는 방법도 있습니다. 일반적으로 Django는 실시간 웹 애플리케이션을 지원하기 위해 Channels 라이브러리를 사용합니다.

Channels는 Django 애플리케이션을 확장하여 WebSocket 연결과 같은 실시간 기능을 추가할 수 있게 해주는 라이브러리입니다. 아래는 Django Channels를 사용하여 실시간으로 실행되는 함수를 구현하는 간단한 예제입니다.

1. Channels 설치하기:

```
pip install channels
```

2. Django 프로젝트의 `settings.py` 파일에 Channels를 등록합니다:

```
INSTALLED_APPS = [
    # 기타 앱들...
    'channels',
]
```

3. 실시간으로 실행되는 함수를 정의합니다. 예를 들어, 실시간으로 현재 시간을 전송하는 함수를 만들어 보겠습니다. `consumers.py` 파일을 생성하고 다음 코드를 추가합니다:

```
from channels.generic.websocket import AsyncWebsocketConsumer
import asyncio
import datetime

class TimeConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        # WebSocket 연결이 수립될 때 호출됩니다.
        await self.accept()

        while True:
            now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            await self.send(now)
            await asyncio.sleep(1)
```

4. `routing.py` 파일을 생성하고 다음 코드를 추가합니다:

```
from django.urls import re_path

from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/time/$', consumers.TimeConsumer.as_asgi()),
]
```

5. `settings.py` 파일에서 Channels를 사용하도록 설정합니다:

```
ASGI_APPLICATION = '<your_project_name>.routing.application'
```

6. 서버를 실행합니다:

```
python manage.py runserver
```

이제 `/ws/time/` 경로로 WebSocket 연결을 생성하면 실시간으로 현재 시간이 전송됩니다. 이 예제는 매 초마다 현재 시간을 클라이언트로 전송합니다.

Channels를 사용하여 Django에서 더 복잡한 실시간 기능을 구현할 수 있습니다. Channels는 WebSocket 이외에도 HTTP 장고 뷰, Django ORM, 외부 API 호출 등과 통합할 수 있는 다양한 기능을 제공합니다. Channels 문서(https://channels.readthedocs.io/)를 참조하면 더 많은 정보를 얻을 수 있습니다.