

Modelo de Actor

Conceptos fundamentales del modelo de actor

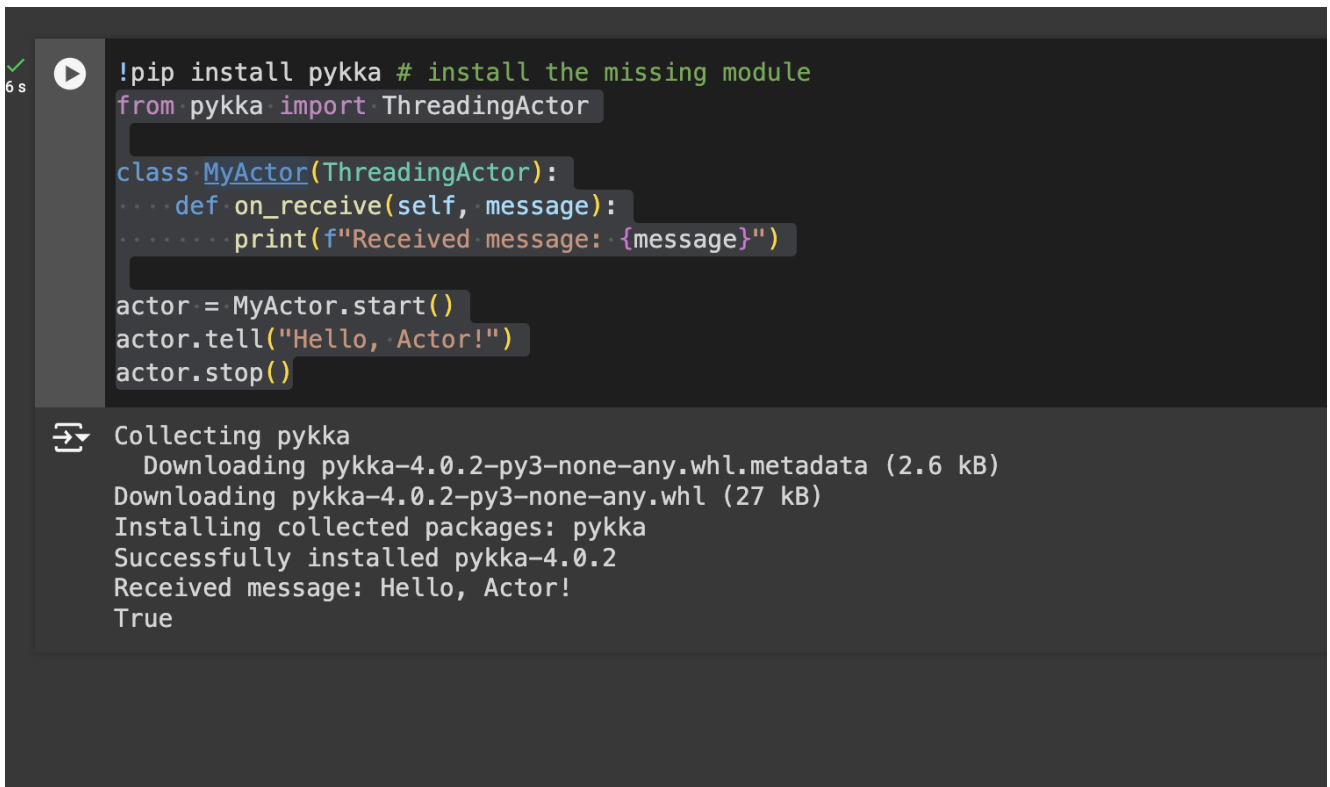
1. **Actor:**
2. **Comunicación mediante mensajes:**
3. **Comportamiento del actor:**
4. **Creación de actores:**
5. **Escalabilidad y paralelismo:**

Beneficios del modelo de actor

- **Tolerancia a fallos:** Dado que los actores son entidades aisladas, los fallos en un actor pueden ser manejados sin afectar a otros. Además, los actores supervisores pueden reiniciar actores fallidos de manera controlada.
- **Concurrencia segura:** Como los actores no comparten memoria y la interacción se realiza mediante mensajes, no es necesario preocuparse por problemas comunes de la programación concurrente, como las condiciones de carrera o los bloqueos.
- **Escalabilidad:** El modelo de actor es adecuado para sistemas distribuidos y escalables porque se adapta bien a diferentes arquitecturas. Los actores pueden distribuirse fácilmente en múltiples máquinas o núcleos de CPU.
- **Simplicidad en la sincronización:** No es necesario usar primitivas de sincronización como semáforos o cerrojos. El paso de mensajes secuencial asegura que el actor procese un mensaje a la vez, simplificando la gestión del estado.

Ejemplo de uso de actores en Python

[crayon-6a9d4d52ebe26778123595/]



```
!pip install pykka # install the missing module
from pykka import ThreadingActor

class MyActor(ThreadingActor):
    ... def on_receive(self, message):
    ...     print(f"Received message: {message}")

actor = MyActor.start()
actor.tell("Hello, Actor!")
actor.stop()
```

Collecting pykka
 Downloading pykka-4.0.2-py3-none-any.whl.metadata (2.6 kB)
 Downloading pykka-4.0.2-py3-none-any.whl (27 kB)
 Installing collected packages: pykka
 Successfully installed pykka-4.0.2
 Received message: Hello, Actor!
 True