Python - Conjuntos de Threads

O que é um pool de threads?

Um pool de threads é um mecanismo que gerencia automaticamente um pool de threads de trabalho. Cada thread no pool é chamado de trabalhador ou thread de trabalho. Threads de trabalho podem ser reutilizados quando a tarefa for concluída. Um único thread é capaz de executar uma única tarefa uma vez.

Um pool de threads controla quando os threads são criados e o que os threads devem fazer quando não estão sendo usados.

O pool é significativamente eficiente para usar um pool de threads em vez de iniciar, gerenciar e fechar threads manualmente, especialmente com um grande número de tarefas.

Vários Threads em Python executam simultaneamente uma determinada função. A execução assíncrona de uma função por vários threads pode ser alcançada pela classe ThreadPoolExecutor definida no módulo concurrent.futures.

O módulo concurrent.futures inclui a classe Future e duas classes Executor - ThreadPoolExecutor e ProcessPoolExecutor.

A aula do futuro

A classe concurrent.futures.Future é responsável por lidar com a execução assíncrona de qualquer chamada, como uma função. Para obter um objeto da classe Future, você deve chamar o método submit() em qualquer objeto Executor. Não deve ser criado diretamente pelo seu construtor.

Métodos importantes na classe Future são -

resultado(tempo limite=Nenhum)

Este método retorna o valor retornado pela chamada. Se a chamada ainda não tiver sido concluída, esse método aguardará o tempo limite em segundos. Se a chamada não for concluída em segundos de tempo limite, um TimeoutError será gerado. Se o tempo limite não for especificado, não haverá limite para o tempo de espera.

cancelar()

Este método tenta cancelar a chamada. Se a chamada estiver sendo executada ou concluída e não puder ser cancelada, o método retornará False, caso contrário, a chamada será cancelada e o método retornará True.

cancelado()

Este método retorna True se a chamada foi cancelada com sucesso.

correndo()

Este método retorna True se a chamada estiver sendo executada e não puder ser cancelada.

feito()

Este método retorna True se a chamada foi cancelada ou concluída com sucesso.

A classe ThreadPoolExecutor

Esta classe representa um conjunto de threads de trabalho com um número máximo especificado para executar chamadas de forma assíncrona.

```
concurrent.futures.ThreadPoolExecutor(max_threads)
```

Exemplo

```
from concurrent.futures import ThreadPoolExecutor
from time import sleep
def square(numbers):
   for val in numbers:
      ret = val*val
      sleep(1)
      print("Number:{} Square:{}".format(val, ret))
def cube(numbers):
   for val in numbers:
      ret = val*val*val
      sleep(1)
      print("Number:{} Cube:{}".format(val, ret))
if __name__ == '__main__':
   numbers = [1,2,3,4,5]
   executor = ThreadPoolExecutor(4)
   thread1 = executor.submit(square, (numbers))
   thread2 = executor.submit(cube, (numbers))
   print("Thread 1 executed ? :",thread1.done())
   print("Thread 2 executed ? :",thread2.done())
   sleep(2)
   print("Thread 1 executed ? :",thread1.done())
   print("Thread 2 executed ? :",thread2.done())
```

Ele produzirá a seguinte **saída** -

```
Thread 1 executed ? : False
Thread 2 executed ? : False
Number:1 Square:1
Number:1 Cube:1
Number:2 Square:4
Number:2 Cube:8
Thread 1 executed ? : False
Thread 2 executed ? : False
Number:3 Square:9
Number:3 Cube:27
Number:4 Square:16
Number:4 Cube:64
Number:5 Square:25
Number:5 Cube:125
Thread 1 executed ? : True
Thread 2 executed ? : True
```