Python - Prioridade de Thread

O módulo **queue** na biblioteca padrão do Python é útil na programação threaded quando as informações devem ser trocadas com segurança entre vários threads. A classe Priority Queue neste módulo implementa toda a semântica de bloqueio necessária.

Com uma fila de prioridade, as entradas são mantidas classificadas (usando o módulo **heapq**) e a entrada de menor valor é recuperada primeiro.

Os objetos Queue possuem os seguintes métodos para controlar a Queue -

- **get()** - O get() remove e retorna um item da fila.
- **put()** - O put adiciona um item a uma fila.
- **qsize()** - O qsize() retorna o número de itens que estão atualmente na fila.
- **vazio()** - O vazio() retorna True se a fila estiver vazia; caso contrário, Falso.
- **full()** - full() retorna True se a fila estiver cheia; caso contrário, Falso.

```
queue.PriorityQueue(maxsize=0)
```

Este é o Construtor para uma fila de prioridade. maxsize é um número inteiro que define o limite superior do número de itens que podem ser colocados na fila. Se maxsize for menor ou igual a zero, o tamanho da fila será infinito.

As entradas de menor valor são recuperadas primeiro (a entrada de menor valor é aquela que seria retornada por min(entries)). Um padrão típico para entradas é uma tupla na forma -

```
(priority_number, data)
```

Exemplo

```
from time import sleep
from random import random, randint
from threading import Thread
from queue import PriorityQueue

queue = PriorityQueue()

def producer(queue):
   print('Producer: Running')
   for i in range(5):
```

```
      # create item with priority
      value = random()
      priority = randint(0, 5)
      item = (priority, value)
      queue.put(item)
   # wait for all items to be processed
   queue.join()

   queue.put(None)
   print('Producer: Done')

def consumer(queue):
   print('Consumer: Running')

   while True:

      # get a unit of work
      item = queue.get()
      if item is None:
         break

      sleep(item[1])
      print(item)
      queue.task_done()
   print('Consumer: Done')

producer = Thread(target=producer, args=(queue,))
producer.start()

consumer = Thread(target=consumer, args=(queue,))
consumer.start()

producer.join()
consumer.join()
```

Ele produzirá a seguinte **saída** -

```
Producer: Running
Consumer: Running
(0, 0.15332707626852804)
(2, 0.4730737391435892)
(2, 0.8679231358257962)
(3, 0.051924220435665025)
```

```
(4, 0.23945882716108446)
Producer: Done
Consumer: Done
```