"""

Charlie Ang

CSC 4800 Python Applications Programming

Lab # 7 Producer/Consumer with Multiple Consumers

Dr. Tindall

February 27, 2017

This program is a producer/consumer with any number of consumer threads which can process

or consume more than one item from the Queue at any given moment.

"""

#!/usr/bin/env python

# Chun Example 4-12, changes by mht

from random import randrange

from time import sleep

from queue import Queue

from myThread3 import MyThread

NITEMS = 10 # number of items to be produced

NREADERS = 3 # number of readers (consumers)

WRITERDELAY = float(1) # delay time between items

WRITERFINISHED = False # global variable producer modifies and consumer checks

# True when writer (producer) has finished its operations

# writing item into the queue (producer)

def writeQ(queue, item):

print('Writer producing object %d for Q...' % item, end='')

queue.put(item, True)

print("size now", queue.qsize())

#reading item from queue (consumer)

def readQ(queue, threadName):

if(WRITERFINISHED == True and queue.qsize() == 0): # end of prod-cons processing

return -1 # exit thread

try: # wrtierfinished == FALSE (producer not done)

val = queue.get(False) # non-blocking operation; returns next val or throws exception

print(' ', threadName, 'consumed object %d from Q... size now' % val, queue.qsize())

sleep(1)

return val

except: # queue is empty

print(' ', threadName, 'polling empty queue')

if(WRITERFINISHED == True):

return -1

def writer(queue, loops):

for i in range(loops):

writeQ(queue, i)

#sleep(randrange(1, 3)) #1-2 sec

sleep(float(WRITERDELAY)) # delay specified # sec between items

global WRITERFINISHED

WRITERFINISHED = True # producer finishes its normal operations

def reader(queue, loops, threadName):

for i in range(loops):

item = readQ(queue, threadName)

if (item == -1):

return # exit thread

sleep(2)

funcs = [writer]

for i in range(0, int(NREADERS)): #0-2 for 3 readers

funcs.append(reader)

nfuncs = range(len(funcs)) # length of funcs

def main():

nloops = NITEMS

q = Queue(32)

threads = []

for i in nfuncs: # putting writers and readers into thread

if (i == 0): # writer

t = MyThread(funcs[i], (q, nloops), funcs[i].\_\_name\_\_)

threads.append(t)

else: # reader

t = MyThread(funcs[i], (q, nloops, 'Reader-{}'.format(i - 1)), funcs[i].\_\_name\_\_)

threads.append(t)

for i in nfuncs:

threads[i].start()

for i in nfuncs:

threads[i].join()

print('all DONE')

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Example Output: NTIEMS = 10, NREADERS = 3, WRITERDELAY = 1 second between items**

Writer producing object 0 for Q...size now 1

Reader-0 consumed object 0 from Q... size now 0

Reader-1 polling empty queue

Reader-2 polling empty queue

Writer producing object 1 for Q...size now 1

Writer producing object 2 for Q...size now 2

Reader-2 consumed object 1 from Q... size now 1

Reader-1 consumed object 2 from Q... size now 0

Reader-0 polling empty queue

Writer producing object 3 for Q...size now 1

Writer producing object 4 for Q...size now 2

Reader-0 consumed object 3 from Q... size now 1

Writer producing object 5 for Q...size now 2

Reader-1 consumed object 4 from Q... size now 1

Reader-2 consumed object 5 from Q... size now 0

Writer producing object 6 for Q...size now 1

Writer producing object 7 for Q...size now 2

Reader-0 consumed object 6 from Q... size now 1

Reader-1 consumed object 7 from Q... size now 0

Reader-2 polling empty queue

Writer producing object 8 for Q...size now 1

Writer producing object 9 for Q...size now 2

Reader-2 consumed object 8 from Q... size now 1

Reader-0 consumed object 9 from Q... size now 0

all DONE

**Example Output: NTIEMS = 10, NREADERS = 3, WRITERDELAY = 0.5 second between items**

Writer producing object 0 for Q...size now 1

Reader-0 consumed object 0 from Q... size now 0

Reader-1 polling empty queue

Reader-2 polling empty queue

Writer producing object 1 for Q...size now 1

Writer producing object 2 for Q...size now 2

Writer producing object 3 for Q...size now 3

Reader-1 consumed object 1 from Q... size now 2

Reader-2 consumed object 2 from Q... size now 1

Writer producing object 4 for Q...size now 2

Writer producing object 5 for Q...size now 3

Reader-0 consumed object 3 from Q... size now 2

Writer producing object 6 for Q...size now 3

Writer producing object 7 for Q...size now 4

Writer producing object 8 for Q...size now 5

Writer producing object 9 for Q...size now 6

Reader-1 consumed object 4 from Q... size now 5

Reader-2 consumed object 5 from Q... size now 4

Reader-0 consumed object 6 from Q... size now 3

Reader-1 consumed object 7 from Q... size now 2

Reader-2 consumed object 8 from Q... size now 1

Reader-0 consumed object 9 from Q... size now 0

all DONE

**Example Output: NTIEMS = 10, NREADERS = 5, WRITERDELAY = 0.1 second between items**

Writer producing object 0 for Q...size now 1

Reader-0 consumed object 0 from Q... size now 0

Reader-1 polling empty queue

Reader-2 polling empty queue

Reader-3 polling empty queue

Reader-4 polling empty queue

Writer producing object 1 for Q...size now 1

Writer producing object 2 for Q...size now 2

Writer producing object 3 for Q...size now 3

Writer producing object 4 for Q...size now 4

Writer producing object 5 for Q...size now 5

Writer producing object 6 for Q...size now 6

Writer producing object 7 for Q...size now 7

Writer producing object 8 for Q...size now 8

Writer producing object 9 for Q...size now 9

Reader-1 consumed object 1 from Q... size now 8

Reader-2 consumed object 2 from Q... size now 7

Reader-3 consumed object 3 from Q... size now 6

Reader-4 consumed object 4 from Q... size now 5

Reader-0 consumed object 5 from Q... size now 4

Reader-1 consumed object 6 from Q... size now 3

Reader-2 consumed object 7 from Q... size now 2

Reader-3 consumed object 8 from Q... size now 1

Reader-4 consumed object 9 from Q... size now 0

all DONE