Using Multiprocessing module for updating Tkinter GUI

Asked 8 years, 6 months ago Active 7 years, 11 months ago Viewed 11k times

4

4

I have been trying to use Multiprocessing module for updating Tkinter GUI but when I run this code, it is giving Pickling error.

```
# Test Code for Tkinter with threads
import Tkinter
from multiprocessing import Queue
import multiprocessing
import time

# Data Generator which will generate Data
def GenerateData():
    global q
    for i in range(10):
        print "Generating Some Data, Iteration %s" %(i)
        time.sleep(2)
        q.put("Some Data from iteration %s \n" %(i))

def QueueHandler():
    global q, text_wid
    while True:
        if not q.empty():
            str = q.get()
            text_wid.insert("end", str)

# Main Tkinter Application
def GUI():
    global text_wid
    tk = Tkinter.Tk()
    text_wid = Tkinter.Text(tk)
    text_wid.pack()
    tk.mainloop()

if __name__ == '__main__':
# Queue which will be used for storing Data
    tk = Tkinter.Tk()
    text_wid = Tkinter.Text(tk)
    q = multiprocessing .Queue()
    t1 = multiprocessing.Process(target=GenerateData,args=(q,))
    t2 = multiprocessing.Process(target=QueueHandler,args=(q,text_wid))
    t1.start()
    t2.start()
    text_wid.pack()
    tk.mainloop()
```

Error:

```
PicklingError: Can't pickle <type 'thread.lock'>: it's not found as thread.lock
```

UPDATE :

I modified code so as to call the function to update the GUI whenever there is value in queue, thus preventing Tkinter widgets from being passed to separate process. Now, I am not getting any error but the widget is not updated with the data. However if i use mix of `Threading` and `Multiprocessing` module i.e. create a separate thread for handling data from the queue, then it works fine. My question why didn't it worked when i run the handler code in separate process. Am I not passing the data correctly. Below is the modified code:

```
# Test Code for Tkinter with threads
import Tkinter
import multiprocessing
from multiprocessing import Queue
import time
import threading

# Data Generator which will generate Data
def GenerateData(q):
    for i in range(10):
        print "Generating Some Data, Iteration %s" %(i)
        time.sleep(2)
        q.put("Some Data from iteration %s \n" %(i))

def QueueHandler(q):
    while True:
        if not q.empty():
            str = q.get()
            update_gui(str)
            #text_wid.insert("end", str)

# Main Tkinter Application
def GUI():
    global text_wid
    tk = Tkinter.Tk()
    text_wid = Tkinter.Text(tk)
    text_wid.pack()
    tk.mainloop()

def update_gui(str):
    global text_wid
    text_wid.insert("end", str)

if __name__ == '__main__':
# Queue which will be used for storing Data
    tk = Tkinter.Tk()
    text_wid = Tkinter.Text(tk)
    q = multiprocessing.Queue()
    t1 = multiprocessing.Process(target=GenerateData,args=(q,))
    t2 = multiprocessing.Process(target=QueueHandler,args=(q,))
    t1.start()
    t2.start()
    text_wid.pack()
    tk.mainloop()
```

python tkinter multiprocessing

sarbjit
3,264 9 30 52

2 Answers

Active | Oldest | Votes

3

You missed out an important part, you should protect your calls with a `__main__` trap:

```
if __name__ == '__main__':
    q = Queue.Queue()
    # Create a thread and run GUI & QueueHadnler in it
    t1 = multiprocessing.Process(target=GenerateData,args=(q,))
    t2 = multiprocessing.Process(target=QueueHandler,args=(q,))

    ....
```

Note that the Queue is passed as a parameter rather than using a global.

Edit: just spotted another issue, you should be using `Queue` from the `multiprocessing` module, not from `Queue`:

```
from multiprocessing import Queue
```

Share Follow

edited Nov 5 '12 at 9:35

answered Nov 5 '12 at 9:06

cdarke
37.4k 5 67 77

I modified the code as suggested and now I am getting the Pickling error (Error that I am getting in my original application code) PicklingError: Can't pickle <type 'thread.lock'>: it's not found as thread.lock – sarbjit Nov 5 '12 at 9:26

Have you moved the Queue object out of global space? Probably a good idea to pass `text_wid` as a parameter as well. Generally, if you ever wondered why you should not use globals then doing any form of multitasking will show you good reasons! – cdarke Nov 5 '12 at 9:32

Yes, I moved queue object out of global space and passing through arguments. I tried passing "text_wid" as argument but didn't helped. I will be posting modified code. – sarbjit Nov 5 '12 at 9:38

See my new edit about using Queue from the multiprocessing module, I think this is where your pickle issue is. – cdarke Nov 5 '12 at 9:40

Now it is giving error for TkApp module. `PicklingError: Can't pickle 'tkapp' object: <tkapp object at 0x02906AD8>` . Any substitute for it as well? – sarbjit Nov 5 '12 at 9:48

```
# Test Code for Tkinter with threads
import Tkinter as Tk
```

```
class GuiApp(object):
   def __init__(self,q):
      self.root = Tk.Tk()
      self.root.geometry('300x100')
      self.text_wid = Tk.Text(self.root,height=100,width=100)
      self.text_wid.pack(expand=1,fill=Tk.BOTH)
      self.root.after(100,self.CheckQueuePoll,q)

   def CheckQueuePoll(self,c_queue):
      try:
         str = c_queue.get(0)
         self.text_wid.insert('end',str)
      except Empty:
         pass
      finally:
         self.root.after(100, self.CheckQueuePoll, c_queue)

# Data Generator which will generate Data
def GenerateData(q):
   for i in range(10):
      print "Generating Some Data, Iteration %s" %(i)
      time.sleep(2)
      q.put("Some Data from iteration %s \n" %(i))


if __name__ == '__main__':
# Queue which will be used for storing Data

   q = multiprocessing.Queue()
   q.cancel_join_thread() # or else thread that puts data will not term
   gui = GuiApp(q)
   t1 = multiprocessing.Process(target=GenerateData,args=(q,))
   t1.start()
   gui.root.mainloop()

   t1.join()
   t2.join()
```

Share Follow

answered Jun 7 '13 at 17:18

user2464430

19 2

8 This answer would be considerably more useful if you gave at least a little description of your solution. What lines did you change, add, or remove, and why? – Bryan Oakley Jun 7 '13 at 17:42

1 Agreed, some comments would probable be beneficial to the community. The solution works very well, though. I have @user2464430 's example here for a similar problem and it's working like a charm. I will note that for a proper start of the multiprocessing thread on windows I had to add `multiprocessing.freeze_support()` immediately after the **main** instantiation or I got an error and no start of this thread. – Matthew Aug 15 '13 at 16:41