Running a Tkinter form in a separate thread

Asked 9 years ago Active 4 years, 10 months ago Viewed 21k times

8

I have written a short module that can be passed an image and simply creates a Tkinter window and displays it. The problem that I am having is that even when I instantiate and call the method that displays the image in a separate thread, the main program will not continue until the Tkinter window is closed.

Here is my module:

```python
import Image, ImageTk
import Tkinter


class Viewer(Tkinter.Tk):
    def __init__(self,parent):
        Tkinter.Tk.__init__(self,parent)
        self.parent = parent
        self.initialize()

    def initialize(self):
        self.grid()

    def show(self,img):
        self.to_display = ImageTk.PhotoImage(img)
        self.label_image = Tkinter.Label(self,image=self.to_display)
        self.label_image.grid(column = 0, row = 0, sticky = "NSEW")
        self.mainloop()
```

It seems to work fine, except when I call it from my test program like the one below, it will not seem to allow my test program to continue, even when started in a different thread.

```python
import Image
from viewer import Viewer
import threading

def showimage(im):
    view = Viewer(None)
    view.show(im)

if __name__ == "__main__":
    im = Image.open("gaben.jpg")
    t = threading.Thread(showimage(im))
    t.start()
    print "Program keeps going..."
```

I think that perhaps my problem is that I should be creating a new thread within the module itself, but I was wanting to just try and keep it simple, as I am new to Python.

edit: To clarity, I am just trying to make a module that will display an image in a Tkinter window, so that I can use this module any time I want to display an image. The problem that I am having is that any time a program uses this module, it cannot resume until the Tkinter window is closed.

python multithreading tkinter

Share Improve this question Follow

edited May 11 '12 at 23:35

asked May 11 '12 at 18:10
derricw
5,430 2 24 31

2 Just a heads up for next visitor or maybe OP 6years later: The main problem besides tkInter and threads is that "threading.Thread(showimage(im))" does not execute "showimage" in a thread, but calls "showimage(im)" and provides the return (None) as argument to the Thread constructor. "Thread(target=partial(showimage, im))" would be correct thread creation wise. – SleepProgger Oct 21 '18 at 7:57

4 Answers

Active Oldest Votes

5

From your comments it sound's like you do not need a GUI at all. Just write the image to disk and call an external viewer.

On most systems it should be possible to launch the default viewer using something like this:

```
import subprocess

subprocess.Popen("yourimage.png")
```

Share Improve this answer Follow

edited Jul 12 '16 at 10:21
Delgan
14.6k 6 75 118

answered Jan 23 '13 at 14:50
Gonzo
1,676 1 19 29

1 On which system does it work to execute an image file as a process? This really tries to execute the file directly, no shell involved, and no code that looks at the file type or something like that. – BlackJack Jun 7 '20 at 10:25

I agree completely with @BlackJack...it is REALLY dangerous to do this...malware could inpersonate the file...you should NOT run it directly – TheTechRobo36414519 Jun 18 '20 at 23:33

18

Tkinter isn't thread safe, and the general consensus is that Tkinter doesn't work in a non-main thread. If you rewrite your code so that Tkinter runs in the main thread, you can have your workers run in other threads.

If all you're doing is showing images, you probably don't need threading at all. Threading is only useful when you have a long running process that would otherwise block the GUI. Tkinter can easily handle hundreds of images and windows without breaking a sweat.

Share Improve this answer Follow edited May 11 '12 at 18:33 answered May 11 '12 at 18:28 Bryan Oakley **308k** 36 436 581

Thanks a lot for the advice. I've noticed that if I run self.mainloop() in the viewer module, like the above example, it will never continue with my actual program until the window is closed. It does the same thing if I place this line in the program (for example view.mainloop()). I had thought that placing this line in a separate thread could solve the problem. You are saying that I should be able to do this without threading, do you mean that there is a way to display the window without mainloop()? – derricw May 11 '12 at 22:50

@ballsdotballs: you need to call `mainloop` if you intend to create an interactive GUI. By its very nature, because you are running an event loop you likely don't need threads. It's true that after calling `mainloop` "it will never continue with [your] actual program", but that is how it is designed to work. You establish the loop, then your program just responds to events. `mainloop` is almost always the very last line of code in your application startup logic. – Bryan Oakley May 11 '12 at 22:59

I think that the problem is, I simply want to write a module that displays an image. I need a way for my program to continue running after it uses this module, instead of waiting on the user to close the image. Do you mean that there is no way to use Tkinter to do this? – derricw May 11 '12 at 23:04

@ballsdotballs: generally speaking, no GUI toolkit works like that. GUI toolkits have event loops and respond to events. That being said, you can still run other code after starting the event loop, but while that code is running your GUI will freeze. Without knowing what you are trying to really accomplish it's hard to be more specific. I don't know if it will help, but look for examples of tkinter's `after` method. With it you can ask the event loop to run something in the future, so you can create a function with your extra code, then run it a few milliseconds after you call `mainloop` – Bryan Oakley May 11 '12 at 23:13

Thank you so much for the help with this. I know I seem dense, but I think that you are missing my fundamental problem. I don't want the GUI to do anything else at all. It does exactly what I want it to do already, which is just display the image. I want the program (the second block above) that creates the viewer to be able to continue while the viewer is running (as in I want to somehow get to the print statement line without closing the GUI). – derricw May 11 '12 at 23:23

3

From what I can tell, Tkinter doesn't like playing in other threads. See this post...I Need a little help with Python, Tkinter and threading

The work around is to create a (possibly hidden) toplevel in your main thread, spawn a separate thread to open images, etc - and use a shared queue to send messages back to the Tk thread.

Are you required to use Tkinter for your project? I like Tkinter. It's "quick and dirty." - but there are (many) cases where other GUI kits are the way to go.

Share Improve this answer Follow edited May 23 '17 at 12:17 Community ♦ **1** 1 answered May 11 '12 at 18:22 parselmouth **1,358** 7 8

2

I have tried to run tkinter from a separate thread, not a good idea, it freezes. There is one solution that worked. Run the gui in the main thread, and send events to the main gui. This is similar example, it just shows a label.

```
import Tkinter as t
global root;
root = t.Tk()
root.title("Control center")
root.mainloop()

def new_window(*args):
    global root
    print "new window"
    window = t.Toplevel(root)
    label = t.Label(window, text="my new window")
    label.pack(side="top", fill="both", padx=10, pady=10)
    window.mainloop()

root.bind("<<newwin>>",new_window)

#this can be run in another thread
root.event_generate("<<newwin>>",when="tail")
```

Share Improve this answer Follow

answered Mar 22 '14 at 19:12
Aftershock
4,568 3 47 59