Why ttk Progressbar appears after process in Tkinter

Asked 8 years, 1 month ago Active 1 year, 10 months ago Viewed 10k times

5

3

I want to create a large text upon `Tkinter` menu command and provide visual support by a progress bar. Although the progress bar is meant to start before the subsequent time-consuming loop, the progress bar shows up only after the large text was created and displayed.

```
def menu_bar(self):
    self.create_menu.add_command(label="Create large file",
    command=self.create_large_file)

def create_large_file(self):
    self.progressbar = ttk.Progressbar(self.master, mode='indeterminate')
    self.progressbar.pack()
    self.progressbar.start()
    self.text.delete(1.0, 'end')
    self.file_content = []

i = 0
while i < 2000000:
    line = lfd.input_string
    self.file_content.append(line + "\n")
    i += 1

self.file_content = ''.join(self.file_content)
self.text.insert(1.0, self.file_content)
```

python progress-bar ttk

Share Improve this question

Follow

edited Jan 28 '15 at 2:58

nbro

12.3k 19 85 163

asked May 6 '13 at 14:08

solarisman

236 3 16

The default update interval for a `mode='indeterminate'` progressbar is 50ms, so maybe your loop isn't all that time-consuming. Try specifying a smaller time value argument when you call its `start()` method. – martineau May 6 '13 at 14:46

That doesn't seem to help. I specified a smaller time value as update interval and I increased the process. I wait approx. 3 seconds while the loop is being processes, only after that the process bar appears. – solarisman May 6 '13 at 14:54

2 Answers

Active Oldest Votes

7

I think the problem is that the time-consuming loop is preventing the `tkinter` event loop, `mainloop()` , from running. In other words, when your work intensive function runs in the same thread as the GUI, it interferes with it by hogging the interpreter.

To prevent this you can use a secondary Thread to run your function and run the GUI and its progressbar in the main thread. To give you an idea of how to do this, here's a simple example I derived from code in another (unrelated) progressbar question to show how easily something like that can be done. *Note:* It's generally recommended that secondary threads not be given direct access to the main thread's `tkinter` objects.

```
from Tkinter import *
import ttk

import time
import threading

def foo():
    time.sleep(5) # simulate some work

def start_foo_thread(event):
    global foo_thread
    foo_thread = threading.Thread(target=foo)
    foo_thread.daemon = True
    progressbar.start()
    foo_thread.start()
    root.after(20, check_foo_thread)

def check_foo_thread():
    if foo_thread.is_alive():
        root.after(20, check_foo_thread)
    else:
        progressbar.stop()

root = Tk()
mainframe = ttk.Frame(root, padding="3 3 12 12")
mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
mainframe.columnconfigure(0, weight=1)
mainframe.rowconfigure(0, weight=1)
progressbar = ttk.Progressbar(mainframe, mode='indeterminate')
progressbar.grid(column=1, row=100, sticky=W)

ttk.Button(mainframe, text="Check",
           command=lambda:start_foo_thread(None)).grid(column=1, row=200,
                                                       sticky=E)

for child in mainframe.winfo_children():
    child.grid_configure(padx=5, pady=5)
root.bind('<Return>', start_foo_thread)

root.mainloop()
```

Share Improve this answer

Follow

edited Aug 2 '19 at 9:34

answered May 7 '13 at 5:28

martineau

99.5k 22 140 249

martineau, upon your good answer, I posted another question, because I have difficulties implementing your solution into my programm. Maybe you also have an idea for this: stackoverflow.com/questions/16420041/... – solarisman May 7 '13 at 14:05

@solarisman: I'll take a look at the other question. I was wondering why you didn't also up-vote my answer here after accepting it. – martineau May 7 '13 at 14:42

sorry, I am still pretty new here. I forgot to upvote, now I did. – solarisman May 7 '13 at 14:45

3

Here's another considerably simpler solution that doesn't require mixing Tkinter and multi-threading. To use it requires the ability to call the progressbar widget's `update_idletasks()` method multiple times during the time-consuming function.

```
from Tkinter import *
import ttk

import time

def foo(progressbar):
    progressbar.start()
    for _ in range(50):
        time.sleep(.1) # simulate some work
        progressbar.step(10)
        progressbar.update_idletasks()
    progressbar.stop()

root = Tk()

mainframe = ttk.Frame(root, padding="3 3 12 12")
mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
mainframe.columnconfigure(0, weight=1)
mainframe.rowconfigure(0, weight=1)
progressbar = ttk.Progressbar(mainframe, mode='indeterminate')
progressbar.grid(column=1, row=100, sticky=W)

ttk.Button(mainframe, text="Check",
           command=lambda:foo(progressbar)).grid(column=1, row=200, sticky=E)

for child in mainframe.winfo_children():
    child.grid_configure(padx=5, pady=5)
root.bind('<Return>', lambda event:foo(progressbar))

root.mainloop()
```

Share Improve this answer Follow

edited Oct 5 '15 at 15:49

answered May 11 '13 at 14:20

martineau
99.5k 22 140 249

Just came across this ... am I correct that the ...step(10) works a bit like a velocity param for the progress bar - which is an interesting 'enhancement' of the normal indeterminate mode. – RFlack Oct 5 '15 at 15:15

@RFlack: Yes, effectively it controls how quickly the bouncing rectangle moves back and forth within the `'indeterminate'` mode `Progressbar`. – martineau Oct 5 '15 at 15:40

I have been having a number of problems getting this all to work right. Im trying to have my ProgressBar class in a separate module. Im finding that I have to use update() which I gather is dangerous; update_idletasks() results in blanking the progress bar window. I will do some more digging but are you an expert on this? I have a question posted already that hasnt had much comment – RFlack Oct 5 '15 at 20:34

@RFlack: Sorry, no, I'm not a `tkinter` expert. I did take a quick peek at your other question and one thing I noticed is that the `bar.Progressbar.start()`, `bar.Progressbar.set()`, and `bar.Progressbar.stop()` methods of `ttk.ProgressBar` instance are never called in the implementation of your own `ProgressBar` class. All of those will need to be called as shown in the code in my answer if you want yours to work. After doing so, `update_idletasks()` is all you should need to use. – martineau Oct 5 '15 at 23:04

@RFlack: BTW, you shouldn't capitalize the first letter of the names of your class's methods — see the PEP 8 - Style Guide for Python Code. It makes your code confusing and more difficult to understand if you don't follow conventions. – martineau Oct 5 '15 at 23:07