Python TUTORIAL

How to Display a Progress Bar while a Thread is Running in Tkinter

If this Python Tutorial saves you hours of work, please whitelist it in your ad blocker 😭 and

Donate Now (https://www.pythontutorial.net/donation/)

to help us ❤️ pay for the web hosting fee and CDN to keep the website running.

Summary: in this tutorial, you'll learn to display a progressbar while a thread is running in a Tkinter application.

This tutorial assumes that you know how to use the `after()` (https://www.pythontutorial.net/tkinter/tkinter-after/) method and understand how threadings (https://www.pythontutorial.net/tkinter/tkinter-thread/) work in Python. Also, you should know how to switch between frames using the tkraise() method (https://www.pythontutorial.net/tkinter/tkraise/) .

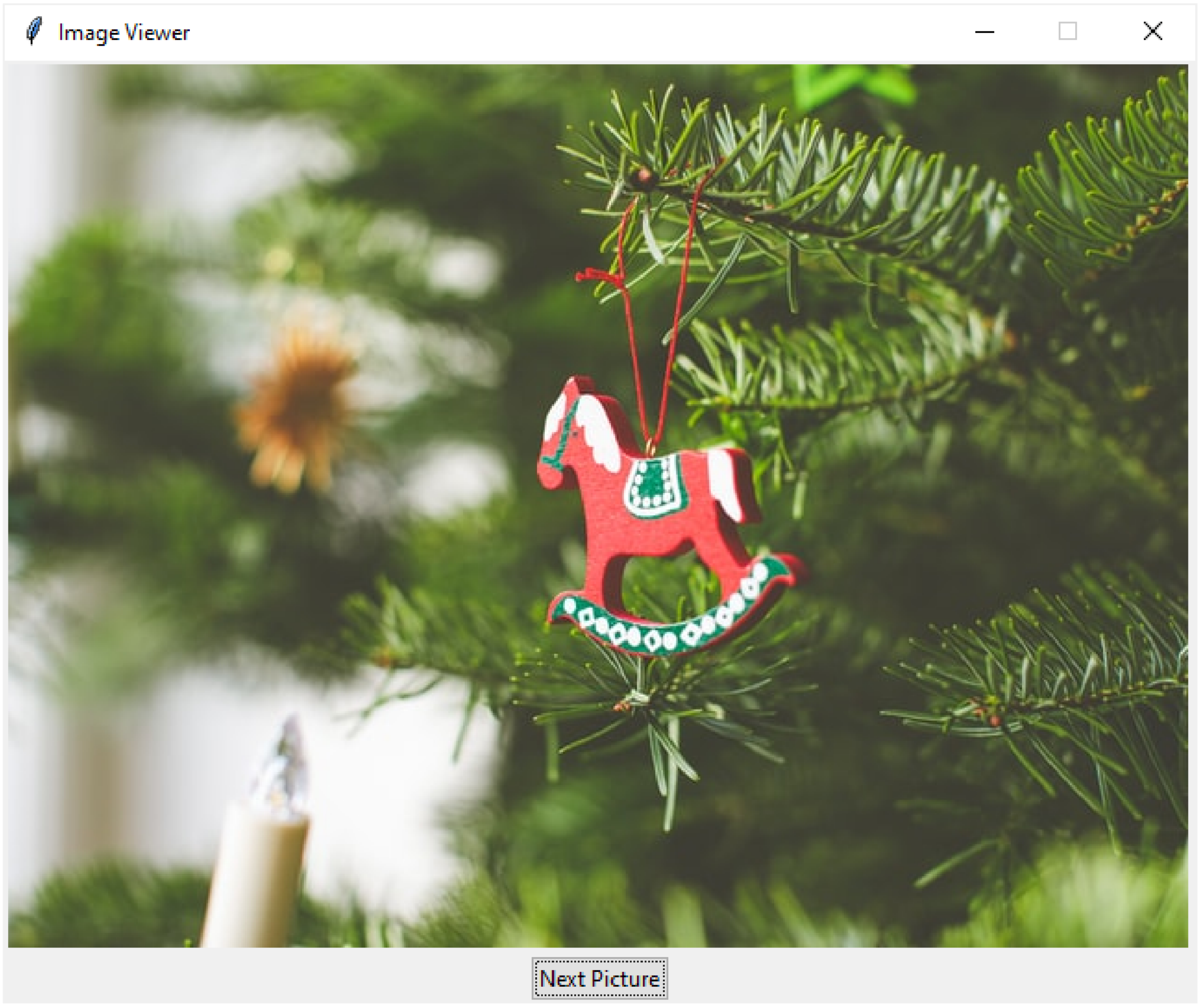
In this tutorial, you'll build a picture viewer that shows a random picture from unsplash.com using its API.

If you make an HTTP request to the following API endpoint:

```
https://source.unsplash.com/random/640x480
```

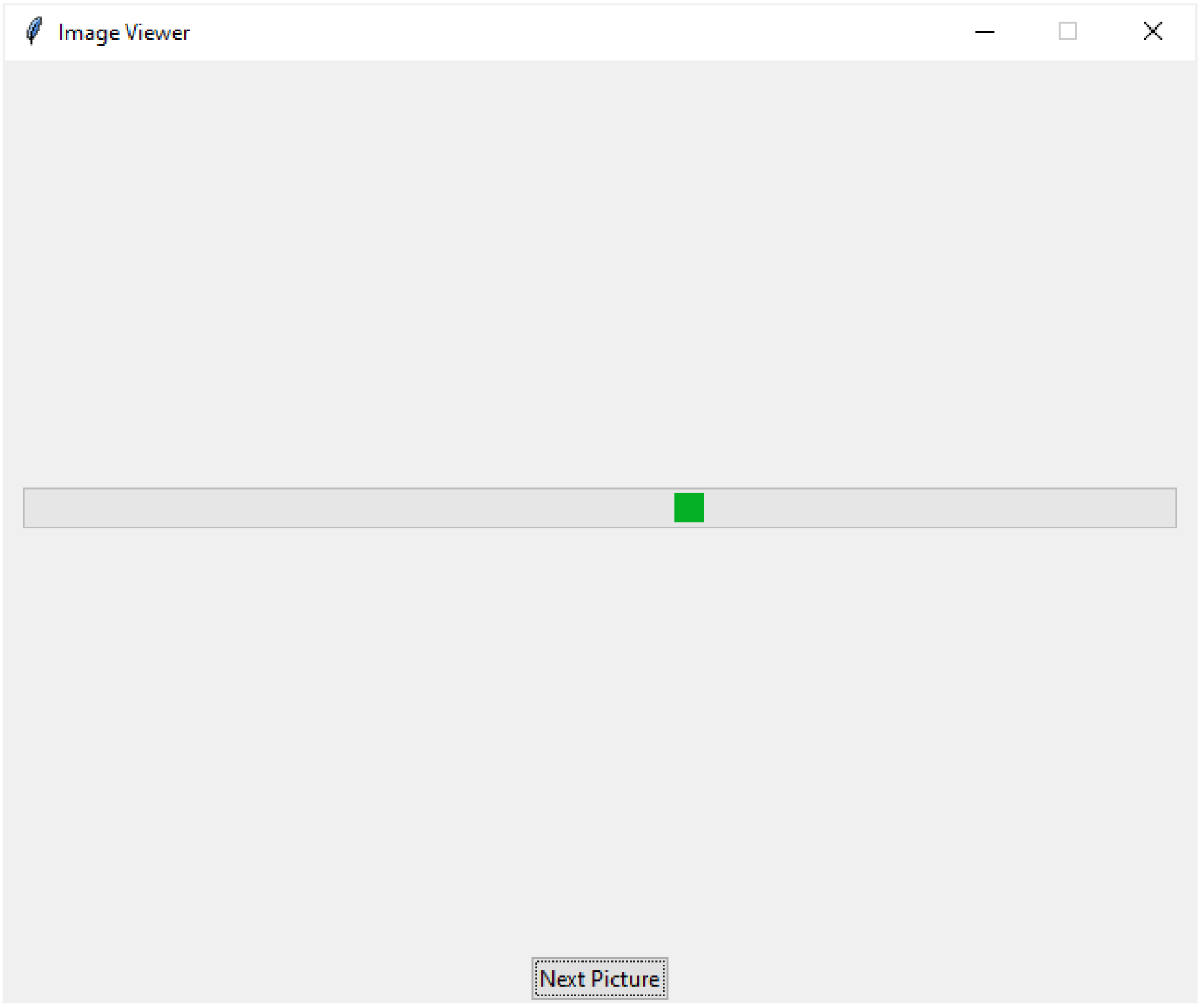
…you'll get a random picture with the size of 640×480.

The following picture shows the final Image Viewer application:



When you click the **Next Picture** button, the program calls the API from unsplash.com to download a random picture and displays it on the window.

It'll also show a progress bar (https://www.pythontutorial.net/tkinter/tkinter-progressbar/) while the picture is downloading, indicating that the download is in progress:



To call the API, you use the `requests` module (https://pypi.org/project/requests/) .

First, install the `requests` module if it’s not available on your computer:

```
pip install requests
```

Second, define a new class (https://www.pythontutorial.net/python-oop/python-class/) that inherits from the `Thread` (https://www.pythontutorial.net/advanced-python/python-threading/) class:

```
class PictureDownload(Thread):
    def __init__(self, url):
        super().__init__()

        self.picture_file = None
        self.url = url
    def run(self):
        """ download a picture and save it to a file """
        # download the picture
        response = requests.get(self.url, proxies=proxyDict)
        picture_name = self.url.split('/')[-1]
        picture_file = f'./assets/{picture_name}.jpg'

        # save the picture to a file
        with open(picture_file, 'wb') as f:
            f.write(response.content)

        self.picture_file = picture_file
```

In this `PictureDownload` class, the `run()` method calls the API using the `requests` module.

The `run()` method downloads a picture and saves it to the `/assets/` folder. Also, it assigns the path of the downloaded picture to the `picture_file` instance attribute.

Third, define an `App` class that inherits the `Tk` class. The `App` class represents the root window.

The `root` window has two frames, one for displaying the Progressbar (https://www.pythontutorial.net/tkinter/tkinter-progressbar/) and the other for showing the Canvas which holds the downloaded picture:

```
def __init__(self, canvas_width, canvas_height):
    super().__init__()
    self.resizable(0, 0)
    self.title('Image Viewer')

    # Progress frame
    self.progress_frame = ttk.Frame(self)

    # configrue the grid to place the progress bar is at the center
    self.progress_frame.columnconfigure(0, weight=1)
    self.progress_frame.rowconfigure(0, weight=1)

    # progressbar
```

```
        self.pb = ttk.Progressbar(
            self.progress_frame, orient=tk.HORIZONTAL, mode='indeterminate')
        self.pb.grid(row=0, column=0, sticky=tk.EW, padx=10, pady=10)

        # place the progress frame
        self.progress_frame.grid(row=0, column=0, sticky=tk.NSEW)

        # Picture frame
        self.picture_frame = ttk.Frame(self)

        # canvas width &amp; height
        self.canvas_width = canvas_width
        self.canvas_height = canvas_height

        # canvas
        self.canvas = tk.Canvas(
            self.picture_frame,
            width=self.canvas_width,
            height=self.canvas_height)
        self.canvas.grid(row=0, column=0)

        self.picture_frame.grid(row=0, column=0)
```

When you click the `Next Picture` button, the `handle_download()` method executes:

```
    def handle_download(self):
        """ Download a random photo from unsplash """
        self.start_downloading()

        url = 'https://source.unsplash.com/random/640x480'
        download_thread = PictureDownload(url)
        download_thread.start()

        self.monitor(download_thread)
```

The `handle_download()` method shows the progress frame by calling the `start_downloading()` method and starts the progress bar:

```
def start_downloading(self):
    self.progress_frame.tkraise()
    self.pb.start(20)
```

It also creates a new thread that downloads the random picture and calls the `monitor()` method to monitor the status of the thread.

The following shows the `monitor()` method:

```
def monitor(self, download_thread):
    """ Monitor the download thread """
    if download_thread.is_alive():
        self.after(100, lambda: self.monitor(download_thread))
    else:
        self.stop_downloading()
        self.set_picture(download_thread.picture_file)
```

The `monitor()` method checks the status of the thread. If the thread is running, it schedules another check after 100ms.

Otherwise, the `monitor()` method calls the `stop_downloading()` method to stop the progressbar, display the picture frame, and show the image.

The following shows the `stop_downloading()` method:

```
def stop_downloading(self):
    self.picture_frame.tkraise()
    self.pb.stop()
```

The following shows the complete Image Viewer program:

```
import requests
import tkinter as tk
```

```
from threading import Thread
from PIL import Image, ImageTk
from tkinter import ttk
from proxies import proxyDict


class PictureDownload(Thread):
    def __init__(self, url):
        super().__init__()

        self.picture_file = None
        self.url = url

    def run(self):
        """ download a picture and save it to a file """
        # download the picture
        response = requests.get(self.url, proxies=proxyDict)
        picture_name = self.url.split('/')[-1]
        picture_file = f'./assets/{picture_name}.jpg'

        # save the picture to a file
        with open(picture_file, 'wb') as f:
            f.write(response.content)

        self.picture_file = picture_file


class App(tk.Tk):
    def __init__(self, canvas_width, canvas_height):
        super().__init__()
        self.resizable(0, 0)
        self.title('Image Viewer')

        # Progress frame
        self.progress_frame = ttk.Frame(self)
        # configrue the grid to place the progress bar is at the center
        self.progress_frame.columnconfigure(0, weight=1)
        self.progress_frame.rowconfigure(0, weight=1)

        # progressbar
        self.pb = ttk.Progressbar(
            self.progress_frame, orient=tk.HORIZONTAL, mode='indeterminate')
        self.pb.grid(row=0, column=0, sticky=tk.EW, padx=10, pady=10)

        # place the progress frame
        self.progress_frame.grid(row=0, column=0, sticky=tk.NSEW)

        # Picture frame
        self.picture_frame = ttk.Frame(self)

        # canvas width &amp; height
        self.canvas_width = canvas_width
        self.canvas_height = canvas_height

        # canvas
        self.canvas = tk.Canvas(
            self.picture_frame,
            width=self.canvas_width,
            height=self.canvas_height)
        self.canvas.grid(row=0, column=0)

        self.picture_frame.grid(row=0, column=0)

        # Button
        btn = ttk.Button(self, text='Next Picture')
        btn['command'] = self.handle_download
        btn.grid(row=1, column=0)

    def start_downloading(self):
        self.progress_frame.tkraise()
        self.pb.start(20)
    def stop_downloading(self):
        self.picture_frame.tkraise()
        self.pb.stop()

    def set_picture(self, file_path):
        """ Set the picture to the canvas """
        pil_img = Image.open(file_path)

        # resize the picture
        resized_img = pil_img.resize(
            (self.canvas_width, self.canvas_height),
            Image.ANTIALIAS)

        self.img = ImageTk.PhotoImage(resized_img)

        # set background image
        self.bg = self.canvas.create_image(
            0,
            0,
            anchor=tk.NW,
            image=self.img)

    def handle_download(self):
        """ Download a random photo from unsplash """
        self.start_downloading()

        url = 'https://source.unsplash.com/random/640x480'
        download_thread = PictureDownload(url)
        download_thread.start()

        self.monitor(download_thread)

    def monitor(self, download_thread):
        """ Monitor the download thread """
        if download_thread.is_alive():
            self.after(100, lambda: self.monitor(download_thread))
        else:
            self.stop_downloading()
            self.set_picture(download_thread.picture_file)


if __name__ == '__main__':
    app = App(640, 480)
    app.mainloop()
```

In this tutorial, you've learned how to display a progressbar that connects to a running thread to indicate that an operation is still in progress.