# Send messages to telegram users using Python

#### **About Telegram message sender project:**

Telegram is a communication application used by millions of people across the world. Telegram also provides API support which enables us to create bots and automate sending messages as well. Thus in this project, we will send messages to a telegram user using python.

#### **Python Telegram message sender project:**

Here, we create a simple telegram message sender using tkinter and telethon

#### **Project Prerequisites:**

Telethon is a library that serves to use the API of Telegram. It is easy to follow and hence does not require any prerequisite knowledge. Tkinter contains widgets, thus we use it to create the GUI of the application.

It is available in python by default. Import to see if it is available

import tkinter

If the package is available for use, it will not show any error. If not, install it using the command

sudo apt-get install python3-tkinter. Linux users and windows users can reinstall python or follow the link [Tkinter installation on windows](https://tkdocs.com/tutorial/install.html#install-win-python) and proceed with those steps.

#### **Download Telegram message sender project code:**

You can download the source code for the telegram message sender project in the given link: <add link>

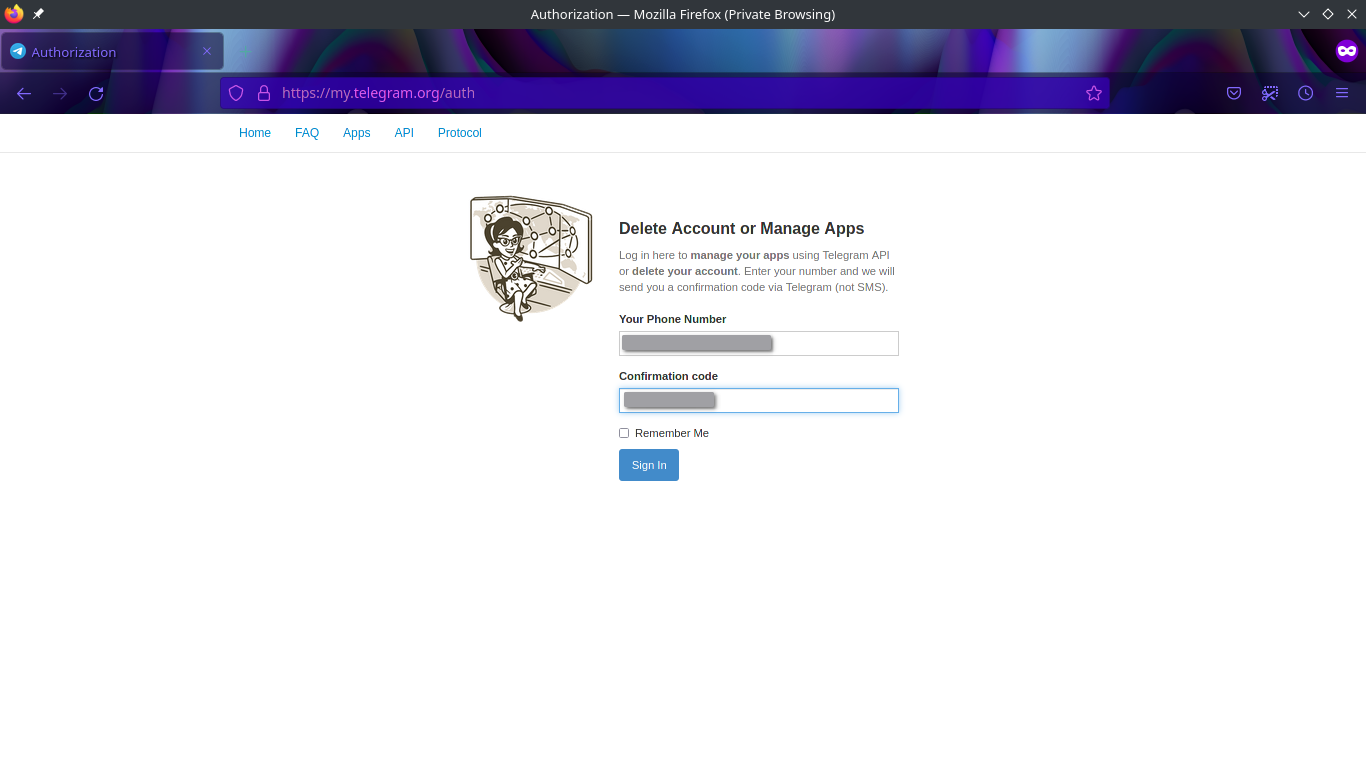
#### **Project File Structure:**

Below is the flow of the project.

1. Obtaining the API ID and Hash code
2. Importing libraries
3. Creating the sender function
4. Creation of the application interface

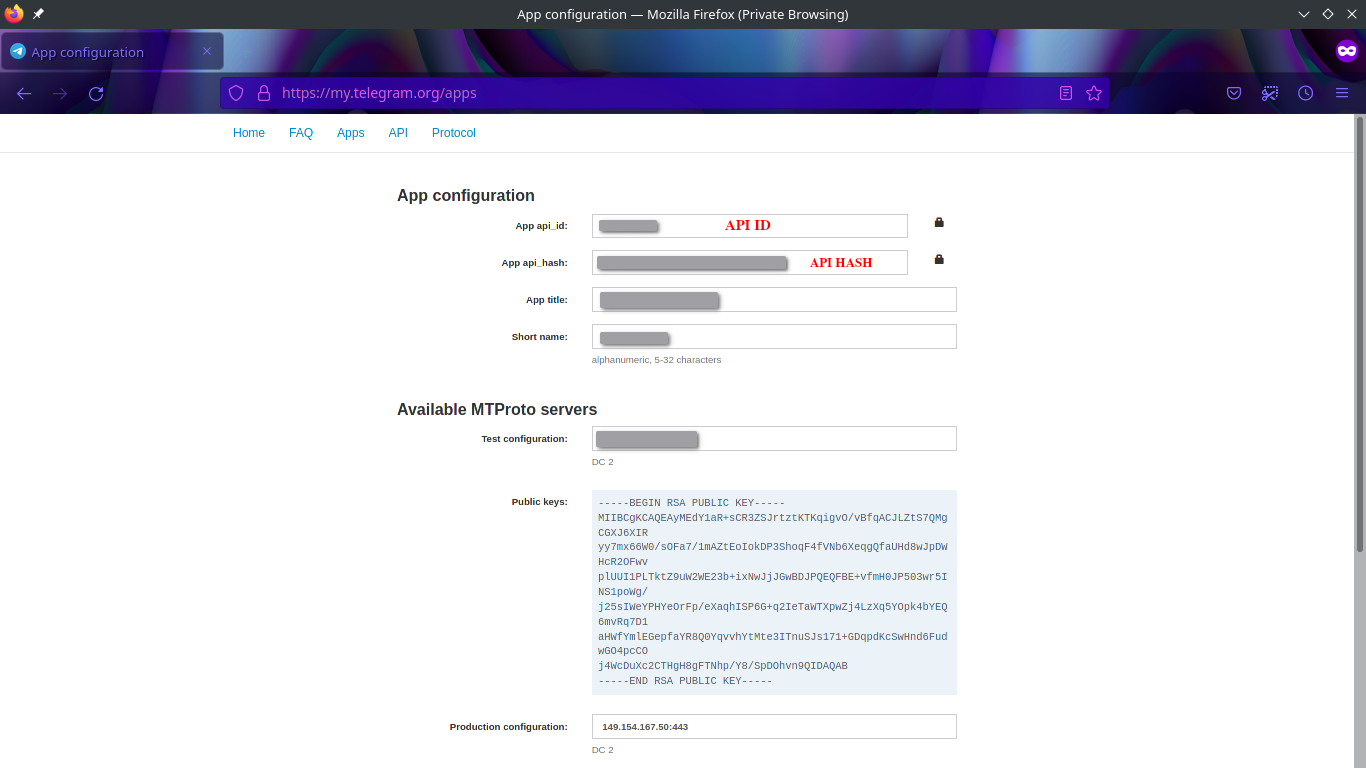
##### **Obtaining the API ID and Hash code:**

To obtain the hash code and the API ID visit [Telegram API link](https://my.telegram.org/). In the resulting page, key in your phone number with the country code and provide the confirmation code or the passcode obtained through telegram after selecting the Next button.



In the next page, type in the requested details. These details do not impact your API hash and ID values.

Finally you will obtain your API ID and Hash codes.



##### **Importing libraries:**

#DataFlair Tutorial for sending messages to telegram users using python

#Import necessary modules

from telethon import TelegramClient, events, sync

from tkinter import \*

from tkinter import messagebox

**Code Explanation:**

* **from telethon import TelegramClient, events, sync:** Telethon allows us to send a message to a telegram user either by phone number or by the username. For further understanding of the python library, refer to [Telethon Documentation](https://docs.telethon.dev/en/latest/)
* **from tkinter import \*:** To make use of tkinter widgets for creating the user interface, we import tkinter
* **from tkinter import messagebox:** Messagebox contains dialog boxes to prompt the user with some information, error or warning.

##### **Creating the sender function:**

#Sender function

def send\_message():

#API details

user\_details = user\_entry.get()

message\_content = message\_entry.get("1.0","end-1c")

#Raise a warning if no input is given

if (len(user\_details) <=0) & (len(message\_content)<=1):

messagebox.showerror(message = "ENTER USER DETAILS" )

else:

#These API codes wont work, hence create your own

api\_id = 1234567

api\_hash = '1234567890abcdefghijklmopqrt123'

#Initialise telegram client with API codes

client = TelegramClient('session\_name', api\_id, api\_hash)

#Start the process

client.start()

#Send the message

client.send\_message(user\_details, message\_content)

messagebox.showinfo(message = "MESSAGE SENT (:" )

**Code explanation:**

* **def send\_message():** Declaration of the sender function, send\_message
* **user\_details = user\_entry.get():** Since user\_entry is an Entry widget, we use get() to obtain the user input.
* **message\_content = message\_entry.get("1.0","end-1c"):** Message\_entry is a Text widget. Thus we extract its contents by specifying the start index (“1.0”) and the finishing index (“end-1c”)
* **if (len(user\_details) <=0) & (len(message\_content)<=1)**: The length of an empty entry widget is 0 and that of a text widget is 1. Here we check if the user provided all the necessary inputs.
* **messagebox.showerror(message = "ENTER USER DETAILS" ):** If the condition hold True ie. the user fails to provide either of the two or both the inputs, then we raise an error prompt. Showerror displays an error prompt and the parameter message contains the message the prompt must show.
* **api\_id and api\_hash:** API ID and API Hash are the results from step 1.
* **client = TelegramClient('session\_name', api\_id, api\_hash):** TelegramClient initialises a telegram client to communicate with the telegram API. Session\_name is a name for the current interaction and api\_is and api\_hash are the initialised variables.
* **client.start():** The client-API interaction commences here.
* **client.send\_message(user\_details, message\_content):** To send the message to the desired user, you use either the username or the phone number with country code (user\_details) and the message to be sent (message\_content)
* **messagebox.showinfo(message = "MESSAGE SENT (:" ):** To notify the user of a successful message transmission.

##### **Creation of the application interface:**

#Define the user interface

telegram\_mess = Tk()

telegram\_mess.geometry("400x300")

telegram\_mess.title("DataFlair's Telegram Message Sender")

bg\_img = PhotoImage(file = "/home/deepika/Downloads/internship/telegram\_deepika/pictures/bg\_image.png")

# Show image using label

label1 = Label(telegram\_mess, image = bg\_img, bd=0)

label1.pack()

#Application Title in the window

title\_label = Label(telegram\_mess, text="DataFlair's Telegram Message Sender", bg="#1d8bd6")

title\_label.place(x=80,y=10)

#Input for user entry

user\_label = Label(telegram\_mess, text="ENTER USER DETAILS:", bg="#2591D9")

info\_label = Label(telegram\_mess, text="Note: Phone number with code or username with @", bg="#2C99DC")

user\_label.place(x=0,y=40)

info\_label.place(x=0,y=70)

user\_entry = Entry(telegram\_mess, width=20)

user\_entry.place(x=160,y=40)

#Message input

message\_label = Label(telegram\_mess, text="ENTER MESSAGE:", bg ="#35A1DF")

message\_label.place(x=0,y=100)

message\_entry = Text(telegram\_mess, width=30, height=3)

message\_entry.place(x=130,y=99)

#send button

send\_button = Button(telegram\_mess, text="Send Button", command=send\_message, relief= RAISED)

send\_button.place(relx=0.5,rely=0.59, anchor=CENTER)

telegram\_mess.mainloop()

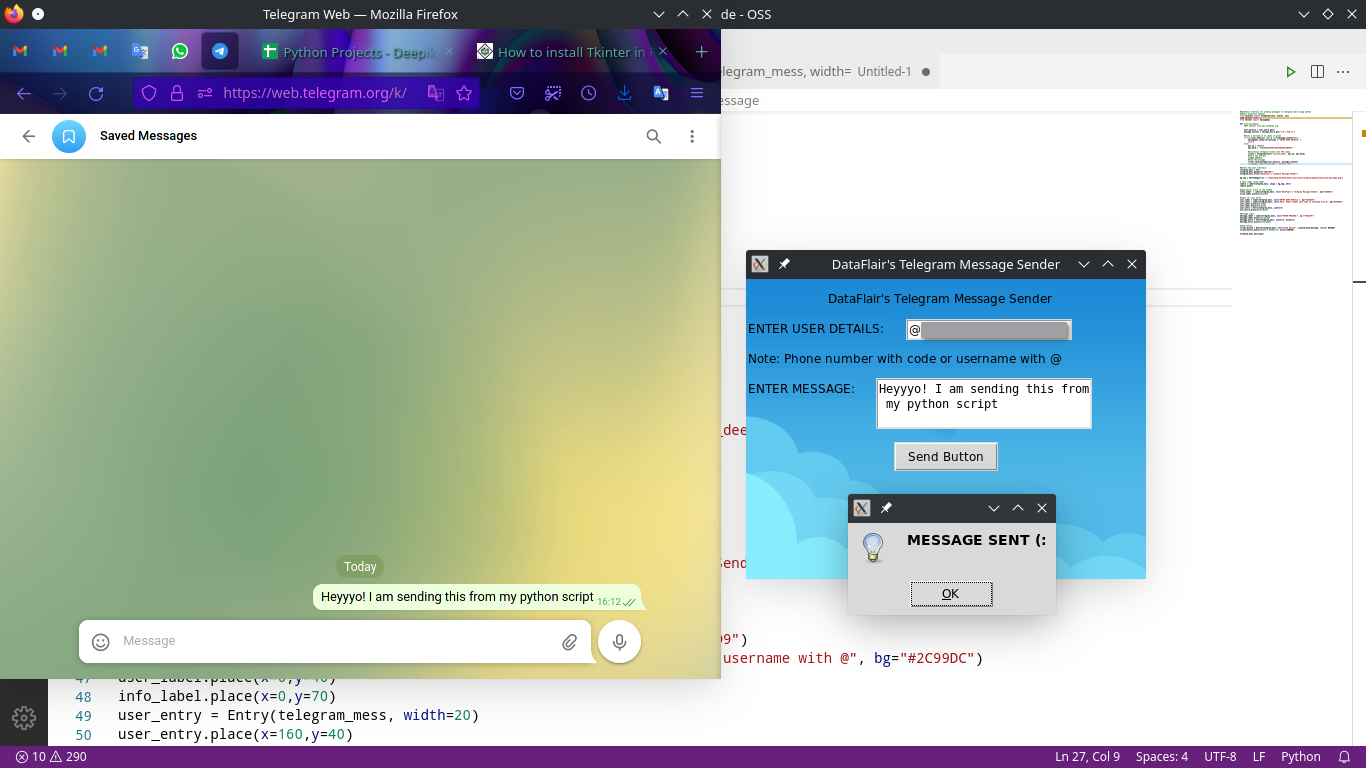
**Code explanation:**

* **telegram\_mess:** To use the tkinter widgets, we call the constructor and assign it to telegram\_mess. This is also the window of the application. Set the window dimensions using geometry(widthxheight) and set a title using title().
* **bg\_img = PhotoImage():** To add a background image, we use the PhotoImage function in tkinter. This function reads PNG images.
* **label1:** Display the image using a label. A label contains text and images that cannot be copied or edited. The parameter image contains the image and bd=0 implies no border. Pack() positions the element in the first row below the top margin.
* **title\_label, user\_label, info\_label, message\_label:** These labels are created using the Label() widget. It contains a text parameter where the text is added and the window of the application where it must reside, and bg: the background colour of the label. It defaults to gray85. Use a colour picker to get the codes for background colours
* **title\_label.place(), info\_lable.place(), user\_label.place(), message\_label.place():** A widget can be shown only if it has a position. Thus we use place, another positioning function like pack(), to fix the position of the widget. The parameters are x and y, where x is the distance from the left margin and y is the distance from the top margin.
* **user\_entry:** An Entry widget allows the user to provide the input. The window of the application and the width of the entry widget is given as parameters
* **message\_entry:** We use text widgets to allow for multiple line user input. The parameters are the same as an entry widget with an additional height parameter
* **send\_button:** Button() creates a button with the parameters, window of the application, text: name of the button, command: the function to call, relief: styling of the button. RAISED creates a shadow effect for the button. Besides positioning the widget with x and y parameters which are nothing but pixel values, we can use relx and rely. Relx and rely are distances from the left and y margins respectively, but they are in percentages. Thus relx=0.50 implies to place the widget at 50% of the distance from the left margin. This option is very useful because it positions the widget based on a set percentage of the window row and column size irrespective of the window size. Anchor tag is an alignment tag.
* **telegram\_mess.mainloop():** When the app is run, widgets and functions above mainloop() will be visible. The rest are simply ignored. Thus when a user selects exit, the control flows out of mainloop and the app terminates.

**NOTE:** Telegram will ask to provide a phone number and a code sent to this number to authorise access while running it for the first time. This will occur at the command line. You can also verify the added device containing the short name given for the application in the first step.

#### **Project output:**

Provide a username (with ‘@’) or phone number (with “+countrycode”) and view the message sent from your end to the user.



#### **Summary**

Thus we created a simple telegram message sender in python. The project introduces the use of Telegram APIs and one simple application of it. We also learnt tkinter to create GUI in python.