Advanced Python

Lecture 14

Creating GUI with TkInter

Svitlana M. Kovalenko

GUI

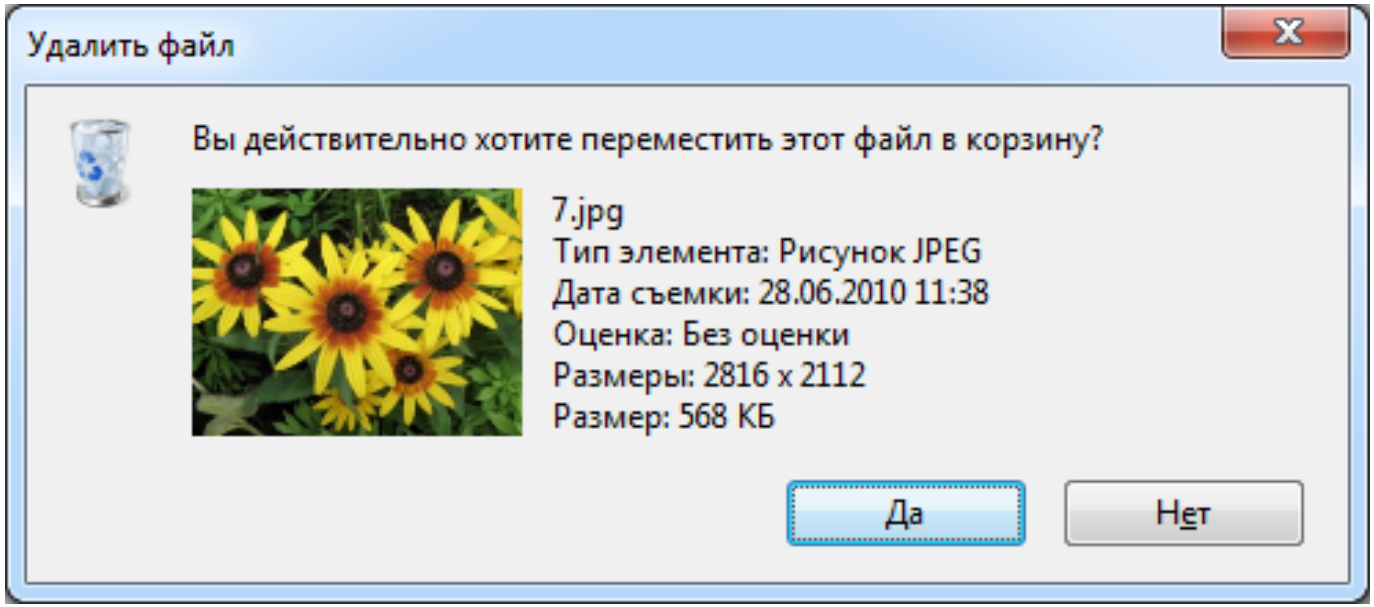
GUI stands for **Graphical User Interface**. In this three-word acronym, the User seems to be the most obvious part.

Interface – it's a tool used by the user to command a device and to receive its responses.

Visual programming

- The term stresses the fact that an application's look is as important as its functionality, but it's not just a matter of what you see on the screen, but also what you can do to change its state, and how you force the application to submit to your will.
- A working GUI application externalizes its existence by creating a window (or windows) visible on the screen.

Widgets


Удалить файл
Вы действительно хотите переместить этот файл в корзину?
7.jpg
Тип элемента: Рисунок JPEG
Дата съемки: 28.06.2010 11:38
Оценка: Без оценки
Размеры: 2816 x 2112
Размер: 568 КБ
Да
Нет

Widgets

- The user interacts with the GUI by using gestures: a mouse's movements, clicks targeting selected GUI elements, or by dragging other elements. Touch screens may offer something more: tapping (single or double or even more complex), swiping, and pinching.
- The GUI elements designed to receive such gestures are called **controls** or **widgets**.

Classical event-handler pseudo-code

```
while True:
    wait_for_user_action()
    if user_pressed_button_yes():
    :
    elif user_pressed_button_no():
    :
    elif user_move_mouse_coursor_over_button_yes():
    :
    elif user_move_mouse_coursor_over_button_no():
    :
    elif user_pressed_Tab_key():
        if isfocused(button_yes):
        :
        elif isfocused(button_no):
       :
    :
    :
```

Classical vs. event-driven paradigm

- In the classical paradigm we would have to:
  - discover the click and check if it happened over our button;
  - redraw the button to reflect the click (e.g., to show that it is actually pressed)
  - invoke the function.
- In the event-driven paradigm our duties look completely different:
  - the event controller detects the clicks on its own;
  - it identifies the target of the click on its own;
  - it invokes the desired function on its own;
  - all these actions take place behind the scenes.

Events

- pressing the mouse button;
- releasing the mouse button (actually, an ordinary mouse click consists of these two subsequent events)
- moving the mouse cursor;
- dragging something under the mouse cursor;
- pressing and releasing a key;
- tapping a screen;
- tracking the passage of time;
- monitoring a widget's state change;
- and many, many more…

TkInter

- **Tk:** widget toolkit, a GUI toolkit, or a UX library

Here are some of its features:

- it's free and open (we don't need to pay for anything)
- it has been developed since 1991 (which means it's stable and mature)
- it defines and serves more than thirty different universal widgets (which is enough even for quite complex applications)
- its implementation is available for many programming languages (of course, for Python too)

- The module that brings Tk to the Python world is named TkInter, which is short for Tk Interface. It's free and open

Importing TkInter

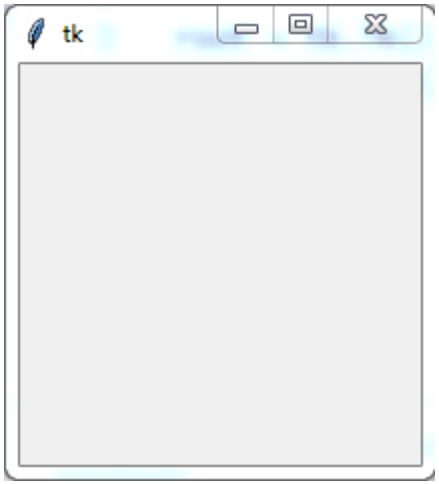
- The GUI application itself consists of four essential elements:
- **importing** the needed tkinter components;
- **creating** an application's main window;
- adding a set of necessary **widgets** to the window;
- **launching** the event controller.

First app

app.py

app.py > ...

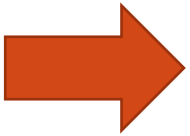
```
1   import tkinter as tk
2
3   root = tk.Tk()
4   root.mainloop()
```

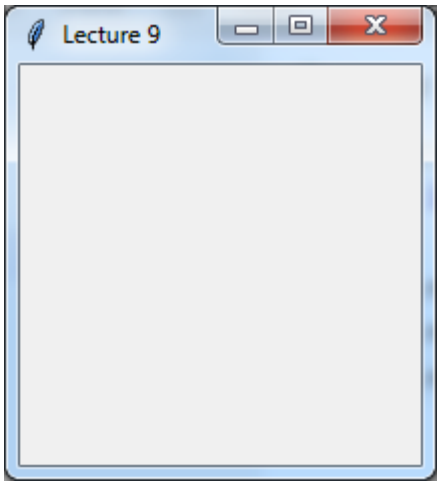


app.py

app.py > ...

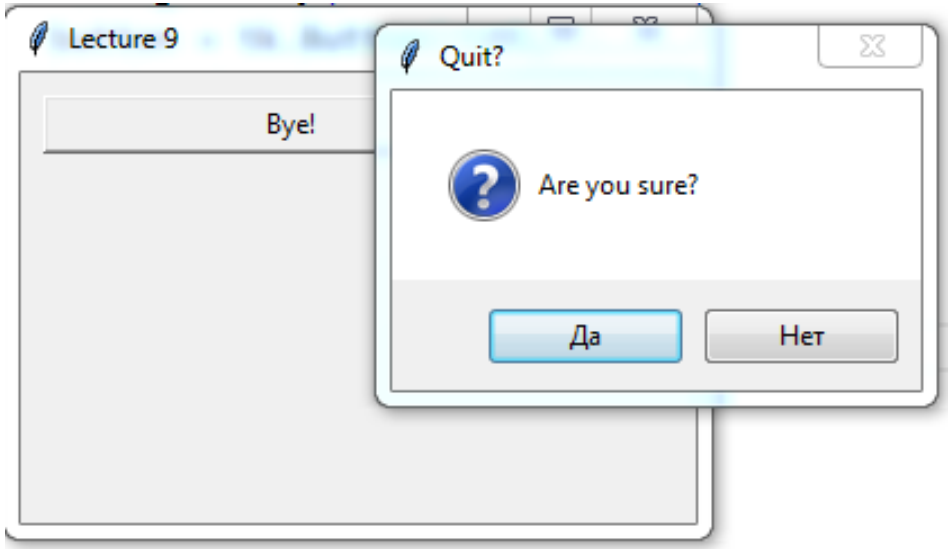
```
1   import tkinter as tk
2
3   root = tk.Tk()
4   root.title("Lecture 9")
5   root.mainloop()
```





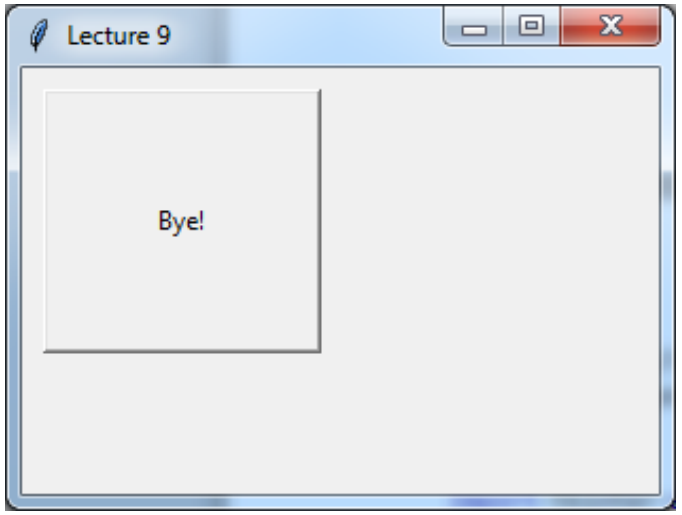
First app

```
import tkinter as tk
from tkinter import messagebox

def Click():
    reply = messagebox.askquestion("Quit?", "Are you sure?")
    if reply == 'yes':
        root.destroy();

root = tk.Tk()
root.title("Lecture 9")
root.geometry("300x200+400+300")
button = tk.Button(root,
                   text="Bye!",
                   command=Click,
                   width=30)

button.place(x=10, y=10)

root.mainloop()
```

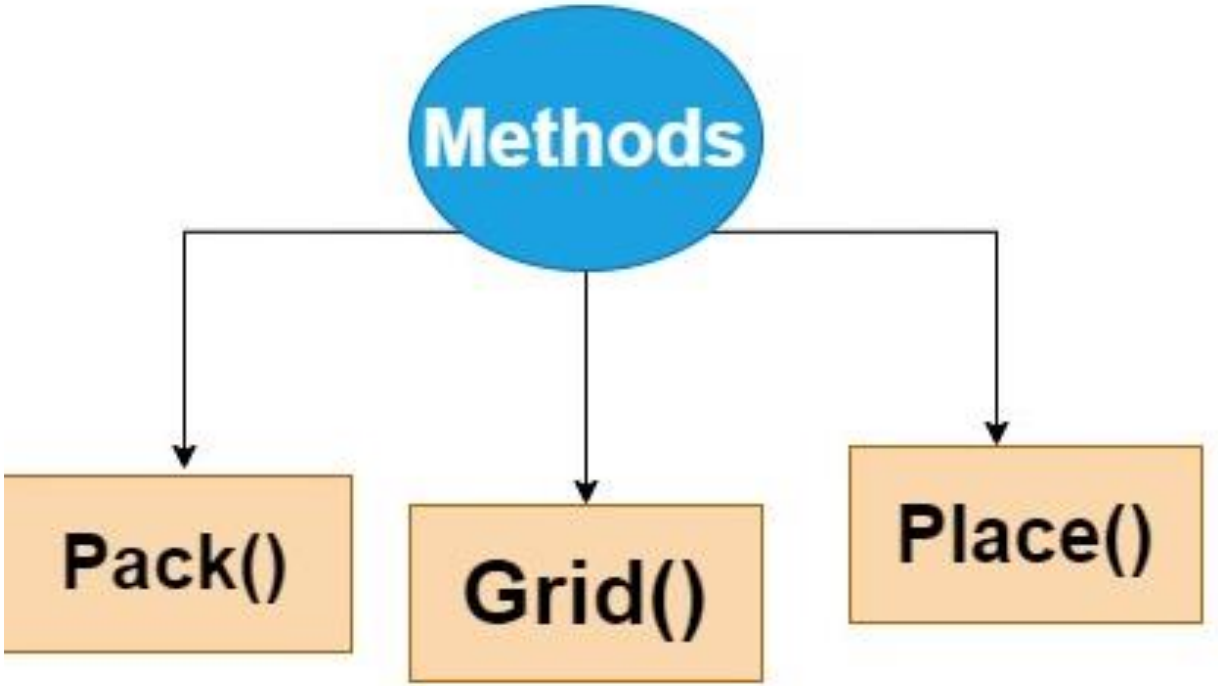


First app

```
import tkinter as tk

root = tk.Tk()
root.title("Lecture 9")
root.geometry("300x200")

button = tk.Button(root, text="Bye!", padx=50, pady=50)
button.place(x=10, y=10)

root.mainloop()
```

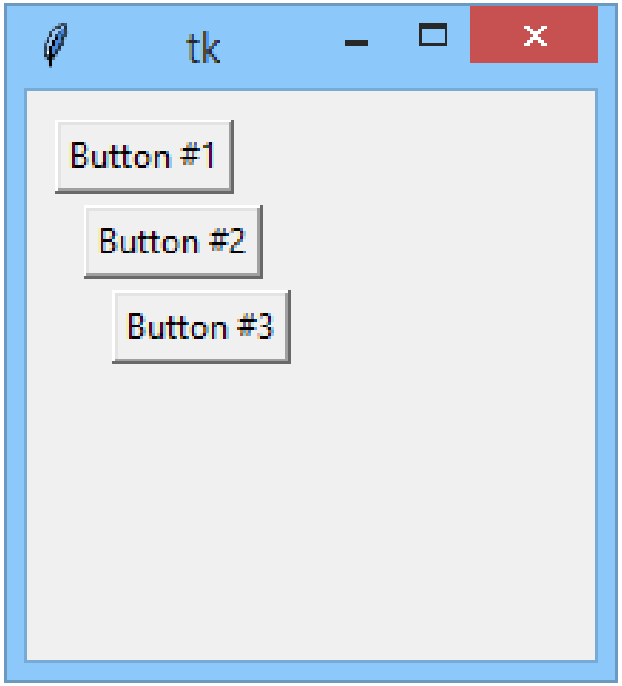


Geometry managers



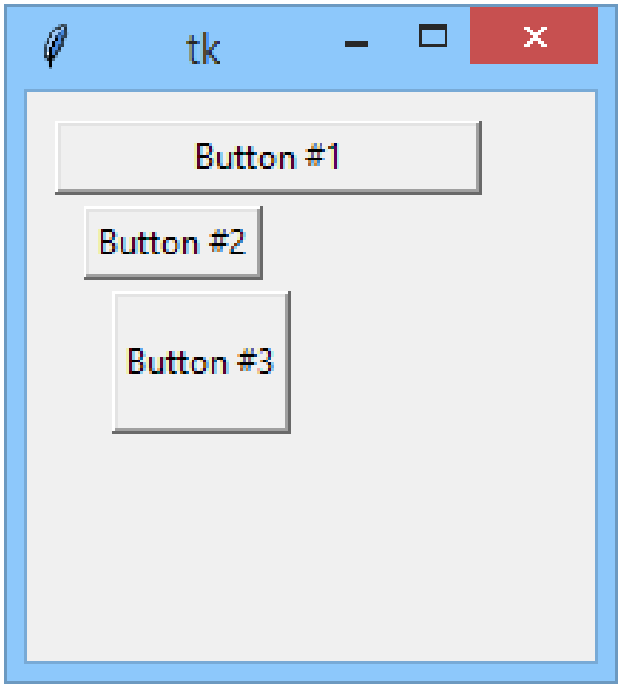
Place()

```
import tkinter as tk

window = tk.Tk()

button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")

button_1.place(x=10, y=10)
button_2.place(x=20, y=40)
button_3.place(x=30, y=70)

window.mainloop()
```



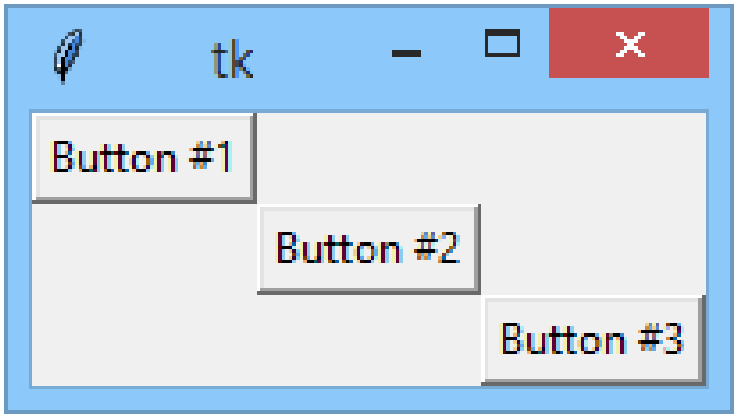
Place()

```
import tkinter as tk

window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
button_1.place(x=10, y=10, width=150)
button_2.place(x=20, y=40)
button_3.place(x=30, y=70, height=50)

window.mainloop()
```



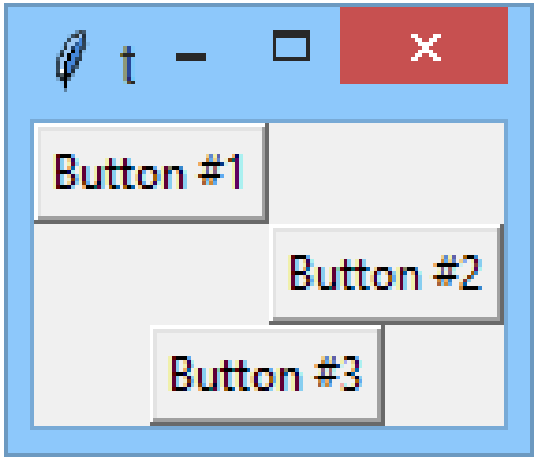
grid()

```
import tkinter as tk

window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
button_1.grid(row=0, column=0)
button_2.grid(row=1, column=1)
button_3.grid(row=2, column=2)

window.mainloop()
```



grid()

```
import tkinter as tk

window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
button_1.grid(row=0, column=0)
button_2.grid(row=1, column=1)
button_3.grid(row=2, column=0, columnspan=2)
window.mainloop()
```

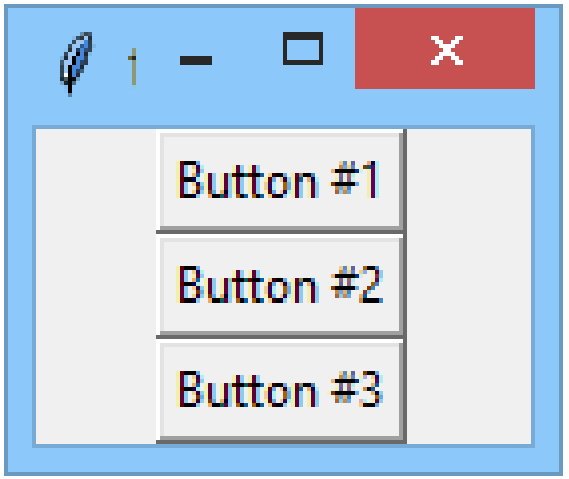


pack()

- side=s – forces the manager to pack the widgets in a specified **direction**, where s can be specified as:
  - TOP – the widget is packed toward the window's **top** (it's manager's default behavior)
  - BOTTOM – the widget is packed toward the window's **bottom**;
  - LEFT – toward the window's **left** boundary;
  - RIGHT – toward the window's **right** boundary;
- fill=f – suggests to the manager how to expand the widget if you want it to occupy more space than the default, while f should be specified as:
  - NONE – do not expand the widget (default behavior)
  - X – expand it in the **horizontal** direction;
  - Y – expand it in the **vertical** direction;
  - BOTH – expand it in **both** directions;

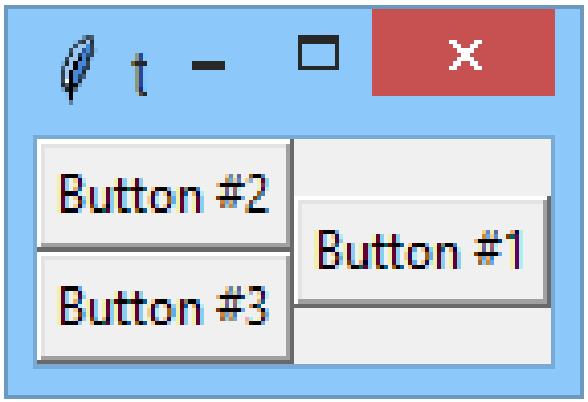
pack()

```
import tkinter as tk


window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
button_1.pack()
button_2.pack()
button_3.pack()
window.mainloop()
```



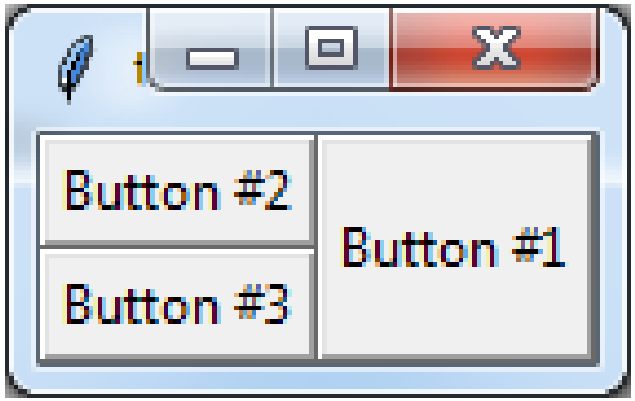
pack()

```
import tkinter as tk

window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
button_1.pack(side=tk.RIGHT)
button_2.pack()
button_3.pack()
window.mainloop()
```



pack()

```
import tkinter as tk

window = tk.Tk()
button_1 = tk.Button(window, text="Button #1")
button_2 = tk.Button(window, text="Button #2")
button_3 = tk.Button(window, text="Button #3")
)
button_1.pack(side=tk.RIGHT, fill=tk.Y)
button_2.pack()
button_3.pack()

window.mainloop()
```



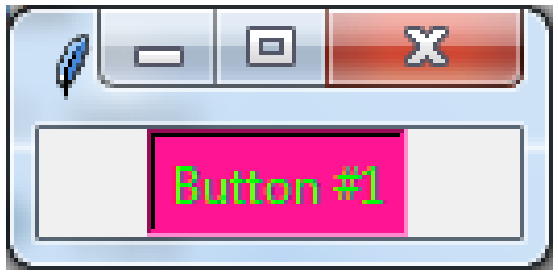
Widgets

Widget Class	Description
Label	A widget used to display text on the screen
Button	A button that can contain text and can perform an action when clicked
Entry	A text entry widget that allows only a single line of text
Text	A text entry widget that allows multiline text entry
Frame	A rectangular region used to group related widgets or provide padding between widgets

Set colors

HTML Colors - https://htmlcolorcodes.com/color-names/

```
button = tk.Button(window, text="Button #1",
                   background="yellow",
                #  bg ='yellow' ,
                   foreground="#0000ff",
                #  fg = "#0000ff"
                   activeforeground="#00ff00",
                   activebackground="DeepPink")
```



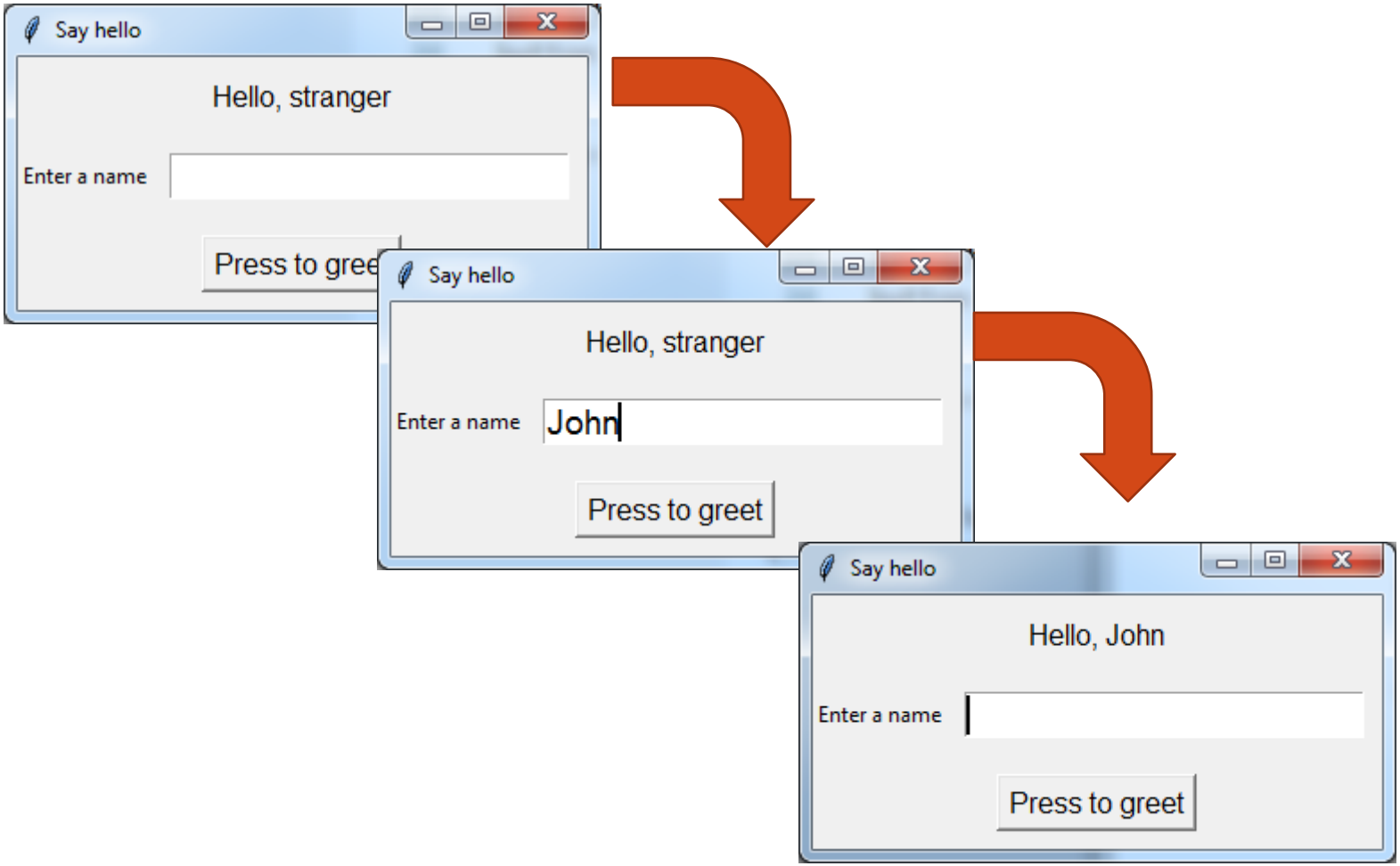
Example. Greeting. Part1

```
import tkinter as tk

def Click():
    name = entry.get()
    if not name:
        greetings = 'Hello, stranger'
    else:
        entry.delete(0,tk.END)
        greetings =f'Hello, {name}'
    label_hello.config(text=greetings)


window = tk.Tk()
window.geometry('+300+400')
window.title('Say hello')
```

Example. Greeting. Part2

```
label_name = tk.Label(text='Enter a name')
entry = tk.Entry(font=('Arial',14))
label_hello = tk.Label(text="Hello, stranger", font = 14)
button = tk.Button(window,
                   text="Press to greet",
                   font = 14,
                   command=Click
                    )

label_hello.grid(row = 0, column=0, columnspan=2, padx=10,pady=10)
label_name.grid(row=1,column=0)
entry.grid(row=1,column=1,padx=10,pady=10)

button.grid(row = 2, column=0, columnspan=2, padx=10,pady=10)

window.mainloop()
```

Example. Result



Tkinter widgets

- https://coderslegacy.com/python/list-of-tkinter-widgets/
- https://www.studytonight.com/tkinter/python-tkinter-widgets