Python Class 17

Saving data and GUI

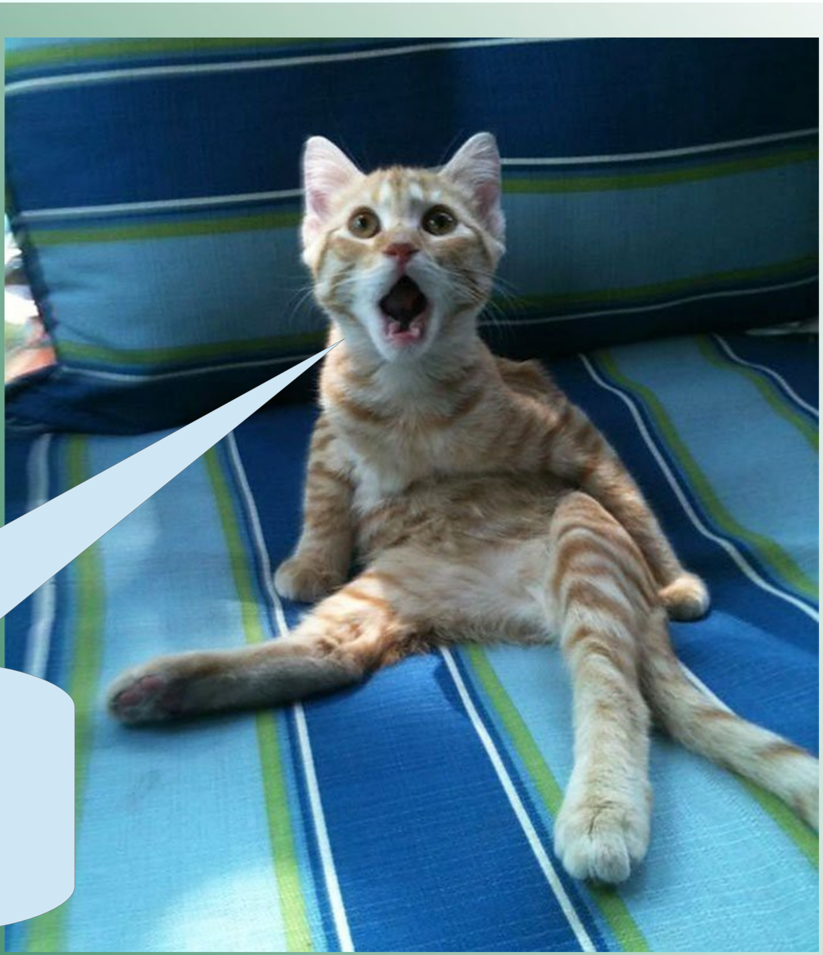
Saving data

Order (順番)

- Import pickle
- Create the data
- Save the data to a file
- Load the data from the file

Import

import pickle



Remember to import!

What is import and what is a library?

- Try this: type

```
help('modules')
```

in Python Idle shell

Delicious!!

Imagine two cases

- Buy chicken, ketchup, kochijian, honey, lemon juice, chili powder, flour, sesame seeds, sesame oil
- Cut the chicken, bread the chicken.
- Make the sauce.
- Fry the chicken, then add the sauce. Top with sesame seeds.

A

- Buy frozen fried chicken, sesame seeds, and yang nyeom sauce.
- Cook the fried chicken, then fry with the sauce.
- Top with sesame seeds.

B

Which is easier?

Save time and effort

- We can save time by using a library.
- Many programmers have worked hard to create these libraries.

Check for data and Create data

```
import os
import pickle
data = MyData()
if os.path.exists(_yourfile_):
    file = open(_yourfile_, “rb”)
    data = pickle.load(file)
    file.close()
else:
    file = open(_yourfile_, “wb”)
   …............
    file.close()
```

This checks if the file exists (存在しているかどうか)

Make the data here

Save the data

```
file = open(_yourfile_, “wb”)

pickle.dump(data, _yourfile_)

file.close()
```

Be careful about 3 things here.

Things to check

- Did you import pickle?
- Did you create the save file?
- Are you using dump and load correctly?
- Did you close the file?
- Are you using the correct mode (“rb” or “wb”)?

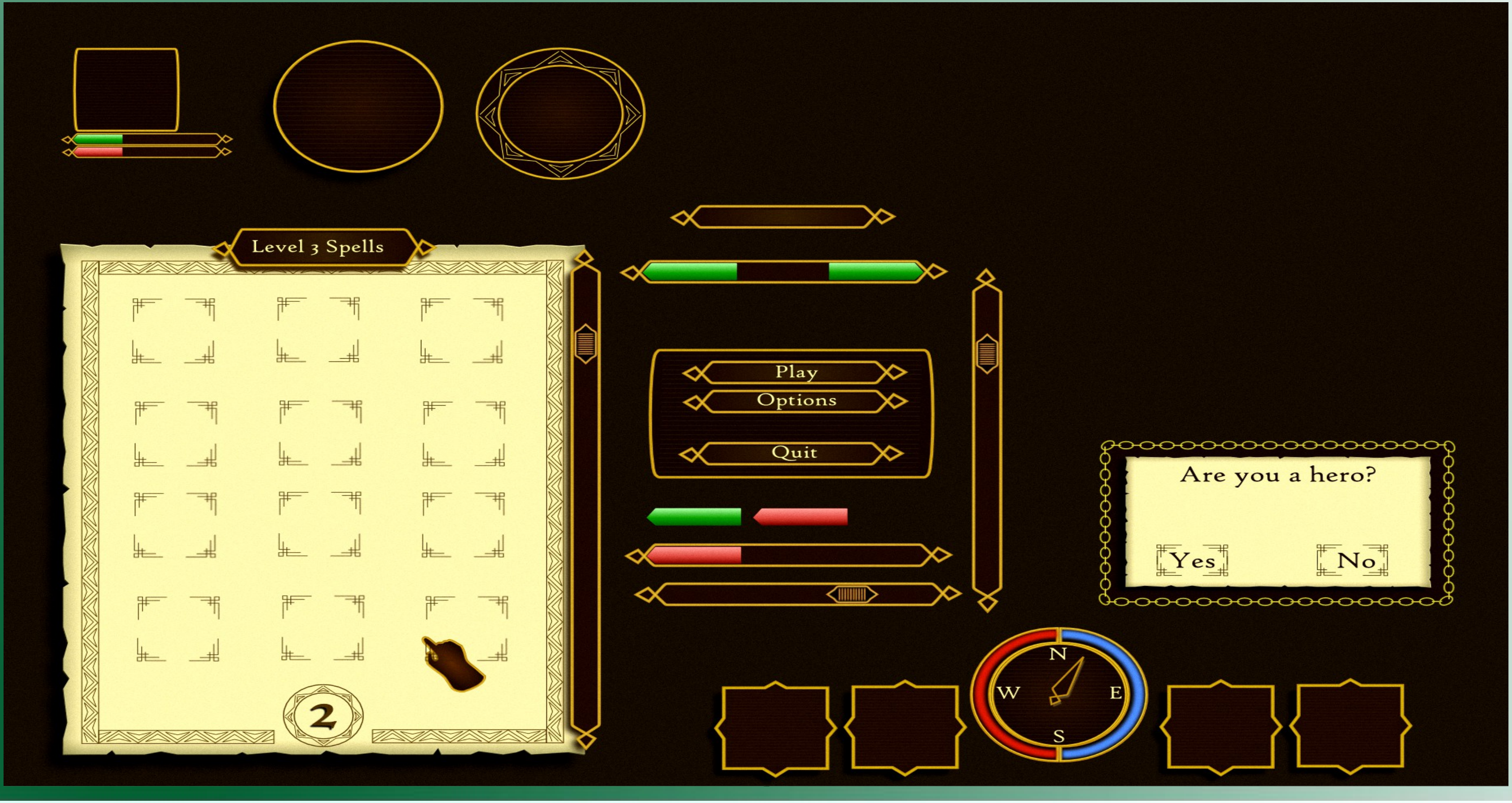
Practice!!!

- Pick one of the projects we made in previous classes: recipe, manga-ka, vending machine, to-do list.
- Use pickle to save the data to a file.
- Use pickle to load the data.

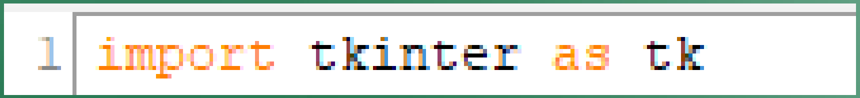
Challenge – make a new class, save the data, and load it

GUI

- GUI (graphical user interface)

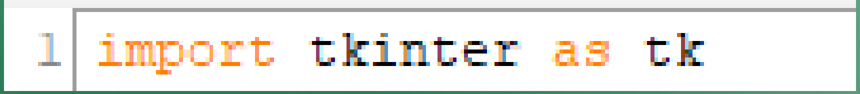


tkinter

- To make a GUI, we will use a library called tkinter
- 

```python
import tkinter as tk
```

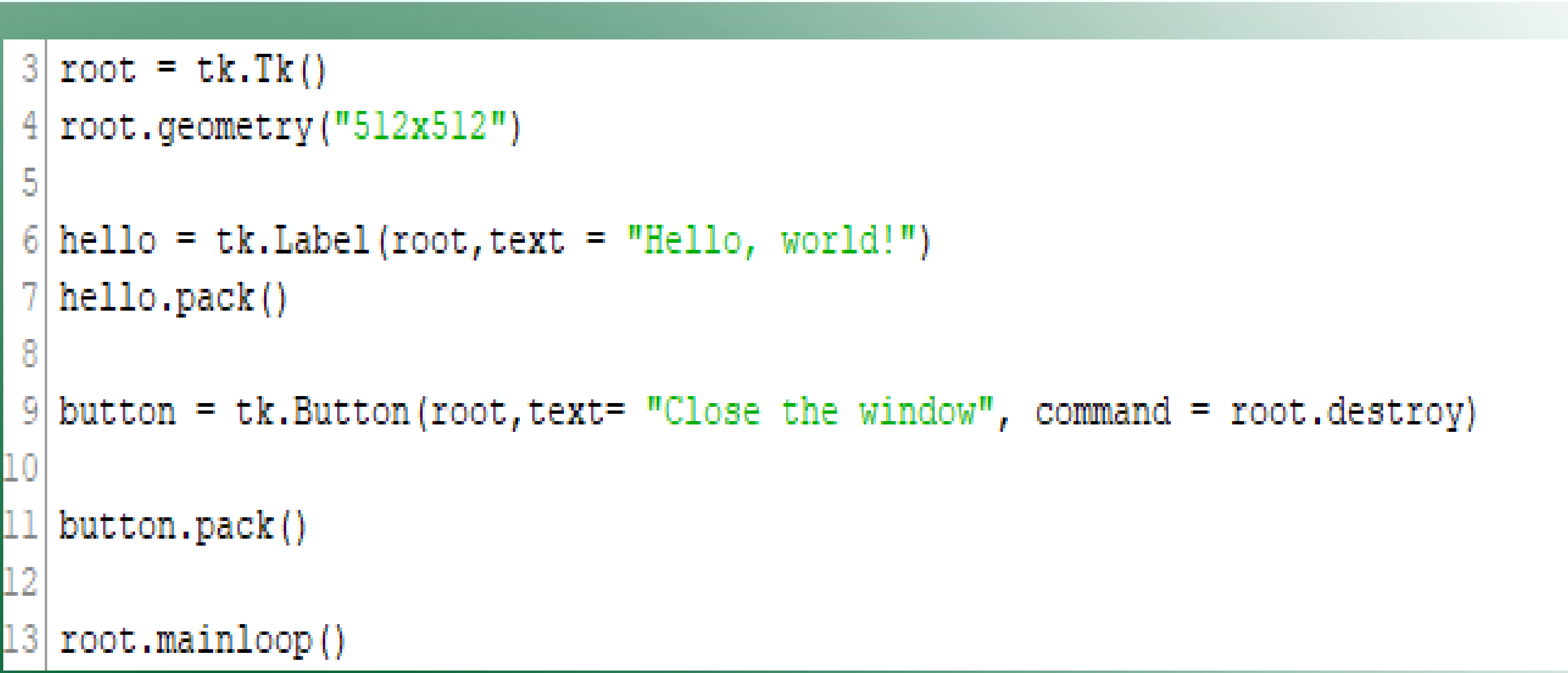
tkinter

- To make a GUI, we will use a library called tkinter
- 

```python
import tkinter as tk
```

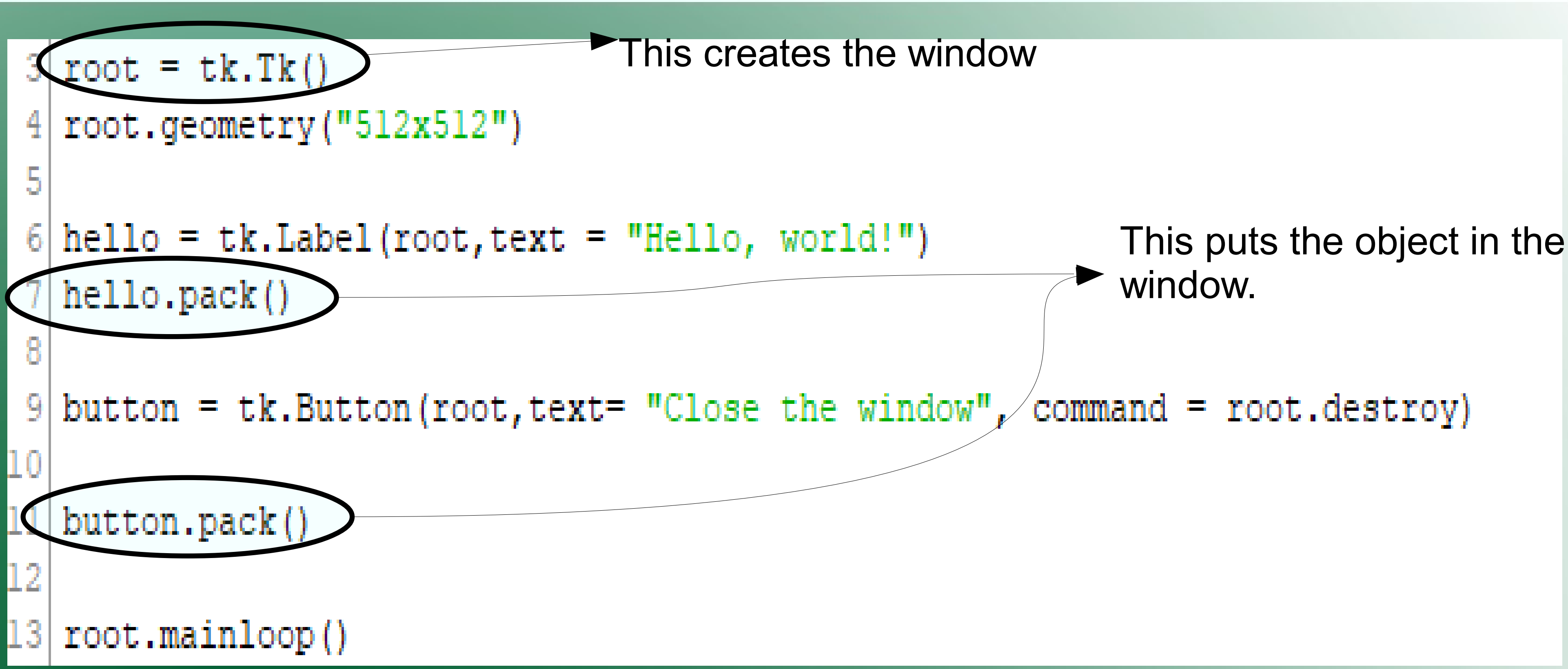
Remember! When you need to use a library (like datetime, random, pickle, or tkinter) you need to use import

Making a basic window



```
root = tk.Tk()
root.geometry("512x512")

hello = tk.Label(root,text = "Hello, world!")
hello.pack()

button = tk.Button(root,text= "Close the window", command = root.destroy)

button.pack()

root.mainloop()
```

Making a basic window

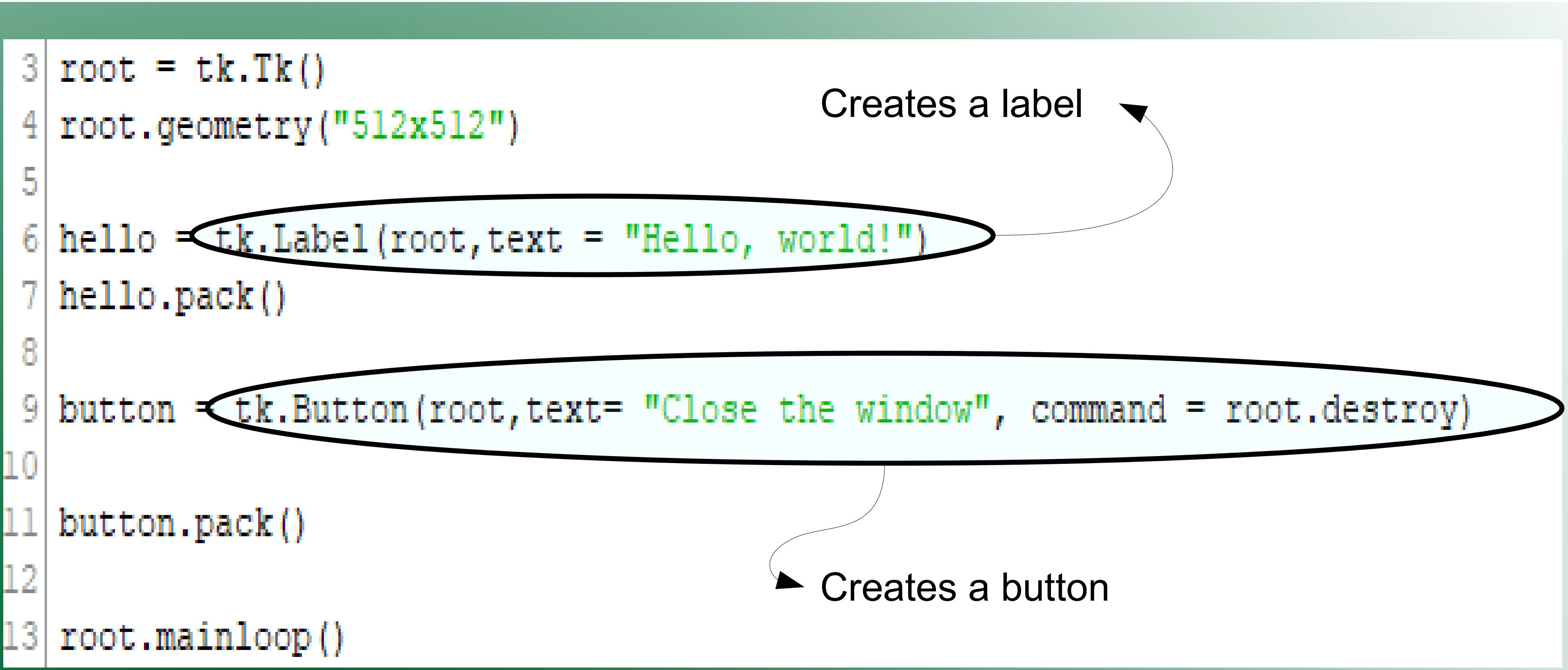


```
root = tk.Tk()
root.geometry("512x512")

hello = tk.Label(root,text = "Hello, world!")
hello.pack()

button = tk.Button(root,text= "Close the window", command = root.destroy)

button.pack()

root.mainloop()
```

This creates the window

This puts the object in the window.

Making a basic window

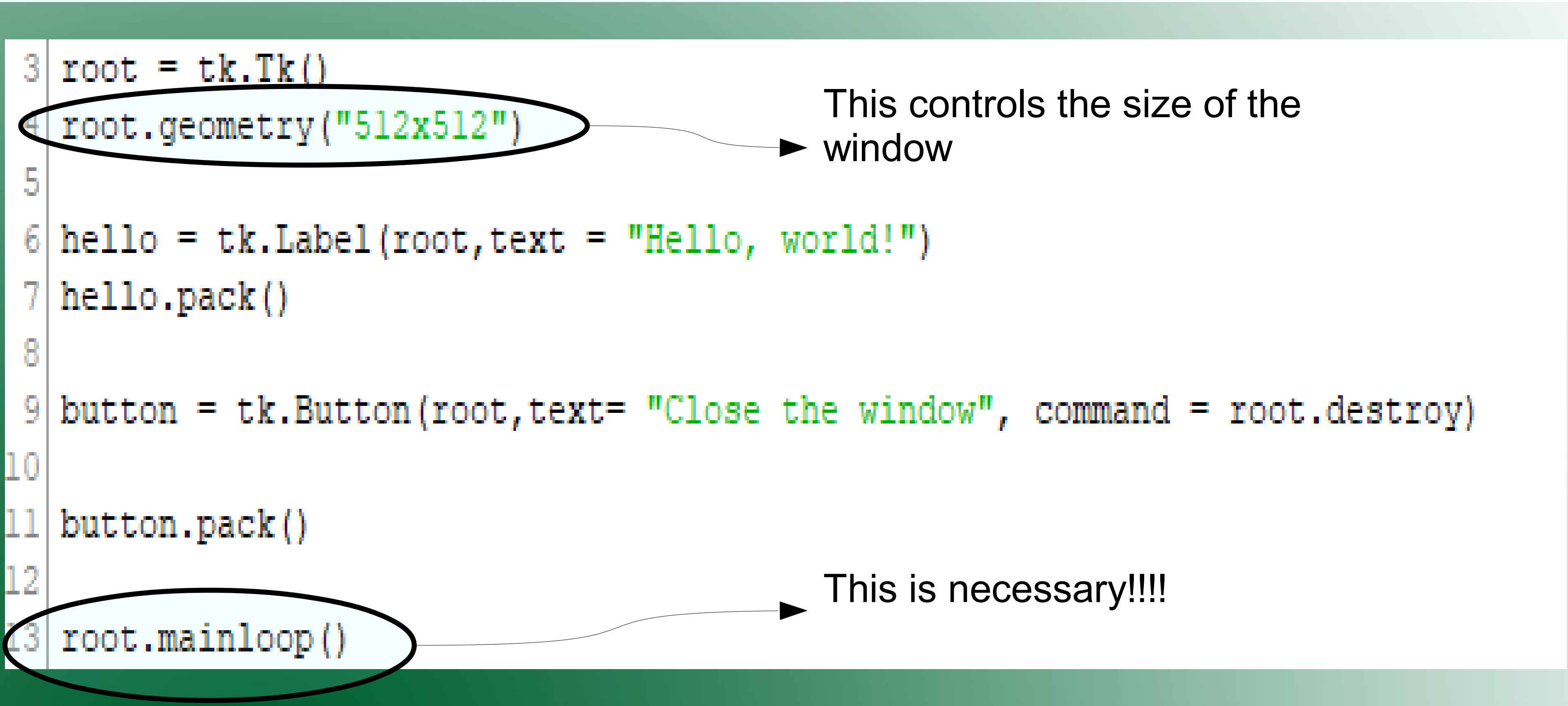


```
root = tk.Tk()
root.geometry("512x512")

hello = tk.Label(root,text = "Hello, world!")
hello.pack()

button = tk.Button(root,text= "Close the window", command = root.destroy)

button.pack()

root.mainloop()
```

Creates a label

Creates a button

Making a basic window



```python
root = tk.Tk()
root.geometry("512x512")

hello = tk.Label(root,text = "Hello, world!")
hello.pack()

button = tk.Button(root,text= "Close the window", command = root.destroy)

button.pack()

root.mainloop()
```

This controls the size of the window

This is necessary!!!!

Explaining the options

- la = tk.Label(*parent*, *option* = , …)
- *parent* (母体；親) is the main object – if parent moves, then la moves too. For now, it will always be the root window.
- There are many *options*

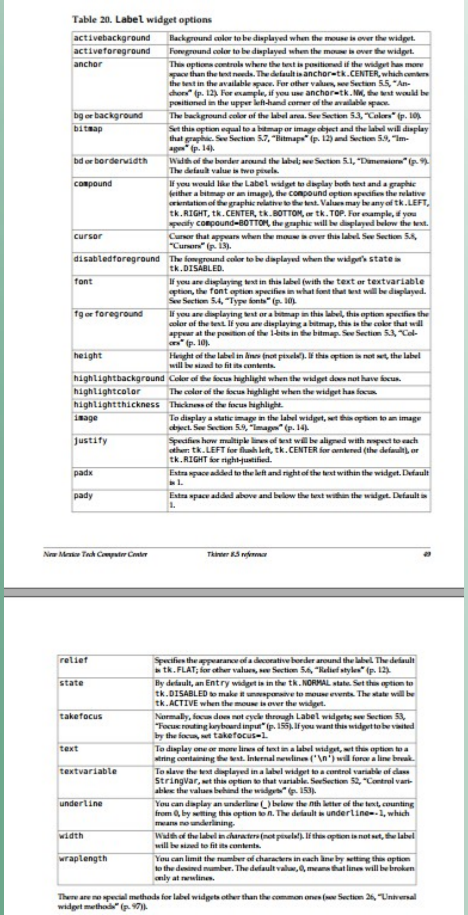


Table 20. `Label` widget options

`activebackground`	Background color to be displayed when the mouse is over the widget.
`activeforeground`	Foreground color to be displayed when the mouse is over the widget.
`anchor`	This options controls where the text is positioned if the widget has more space than the text needs. The default is `anchor=tk.CENTER`, which centers the text in the available space. For other values, see Section 5.5, "Anchors" (p. 12). For example, if you use `anchor=tk.NW`, the text would be positioned in the upper left-hand corner of the available space.
`bg` or `background`	The background color of the label area. See Section 5.3, "Colors" (p. 10).
`bitmap`	Set this option equal to a bitmap or image object and the label will display that graphic. See Section 5.7, "Bitmaps" (p. 12) and Section 5.9, "Images" (p. 14).
`bd` or `borderwidth`	Width of the border around the label; see Section 5.1, "Dimensions" (p. 9). The default value is two pixels.
`compound`	If you would like the `Label` widget to display both text and a graphic (either a bitmap or an image), the `compound` option specifies the relative orientation of the graphic relative to the text. Values may be any of `tk.LEFT`, `tk.RIGHT`, `tk.CENTER`, `tk.BOTTOM`, or `tk.TOP`. For example, if you specify `compound=BOTTOM`, the graphic will be displayed below the text.
`cursor`	Cursor that appears when the mouse is over this label. See Section 5.8, "Cursors" (p. 13).
`disabledforeground`	The foreground color to be displayed when the widget's `state` is `tk.DISABLED`.
`font`	If you are displaying text in this label (with the `text` or `textvariable` option, the `font` option specifies in what font that text will be displayed. See Section 5.4, "Type fonts" (p. 10).
`fg` or `foreground`	If you are displaying text or a bitmap in this label, this option specifies the color of the text. If you are displaying a bitmap, this is the color that will appear at the position of the 1-bits in the bitmap. See Section 5.3, "Colors" (p. 10).
`height`	Height of the label in *lines* (not pixels!). If this option is not set, the label will be sized to fit its contents.
`highlightbackground`	Color of the focus highlight when the widget does not have focus.
`highlightcolor`	The color of the focus highlight when the widget has focus.
`highlightthickness`	Thickness of the focus highlight.
`image`	To display a static image in the label widget, set this option to an image object. See Section 5.9, "Images" (p. 14).
`justify`	Specifies how multiple lines of text will be aligned with respect to each other: `tk.LEFT` for flush left, `tk.CENTER` for centered (the default), or `tk.RIGHT` for right-justified.
`padx`	Extra space added to the left and right of the text within the widget. Default is 1.
`pady`	Extra space added above and below the text within the widget. Default is 1.
`relief`	Specifies the appearance of a decorative border around the label. The default is `tk.FLAT`; for other values, see Section 5.6, "Relief styles" (p. 12).
`state`	By default, an `Entry` widget is in the `tk.NORMAL` state. Set this option to `tk.DISABLED` to make it unresponsive to mouse events. The state will be `tk.ACTIVE` when the mouse is over the widget.
`takefocus`	Normally, focus does not cycle through `Label` widgets; see Section 53, "Focus: routing keyboard input" (p. 155). If you want this widget to be visited by the focus, set `takefocus=1`.
`text`	To display one or more lines of text in a label widget, set this option to a string containing the text. Internal newlines (`'\n'`) will force a line break.
`textvariable`	To slave the text displayed in a label widget to a control variable of class `StringVar`, set this option to that variable. SeeSection 52, "Control variables: the values behind the widgets" (p. 153).
`underline`	You can display an underline (_) below the *n*th letter of the text, counting from 0, by setting this option to *n*. The default is `underline=-1`, which means no underlining.
`width`	Width of the label in *characters* (not pixels!). If this option is not set, the label will be sized to fit its contents.
`wraplength`	You can limit the number of characters in each line by setting this option to the desired number. The default value, 0, means that lines will be broken only at newlines.

New Mexico Tech Computer Center *Tkinter 8.5 reference* 49

There are no special methods for label widgets other than the common ones (see Section 26, "Universal widget methods" (p. 97)).

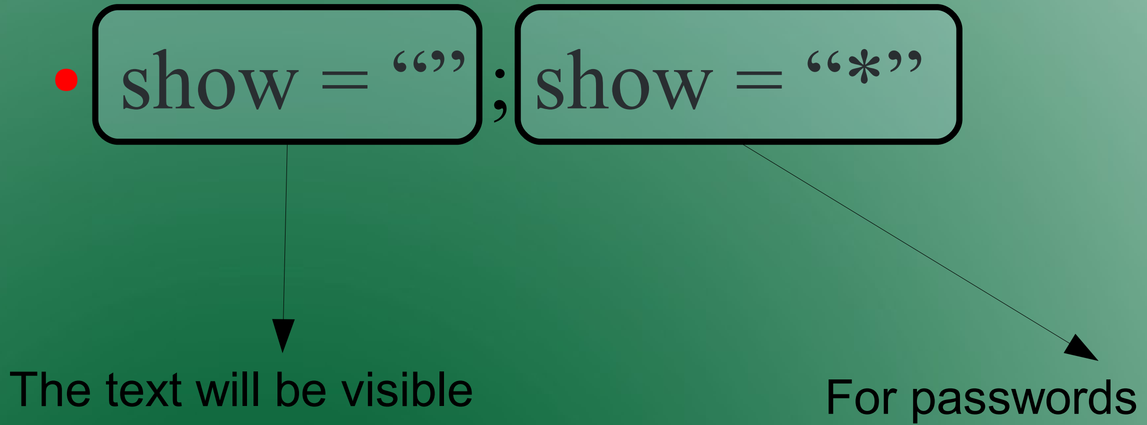
Explaining the options

- text = “”
- cursor = “”

- This is the text/string that will be displayed
- This will change the cursor (fleur, dotbox, X_cursor, heart...)

Entering Data

- Use the Entry class
- e = tk.Entry(*parent*, *option* = ,...)
- show = “”; show = “*”

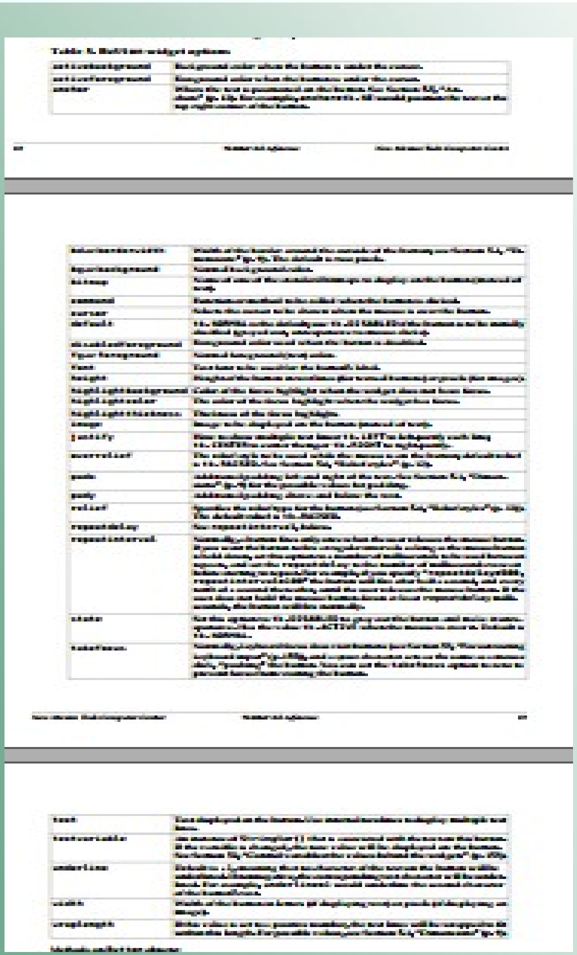


The text will be visible

For passwords

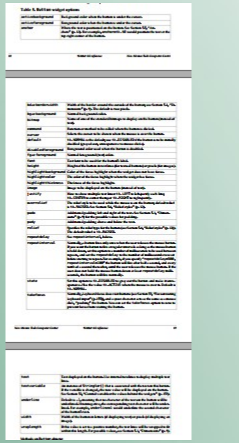
Button

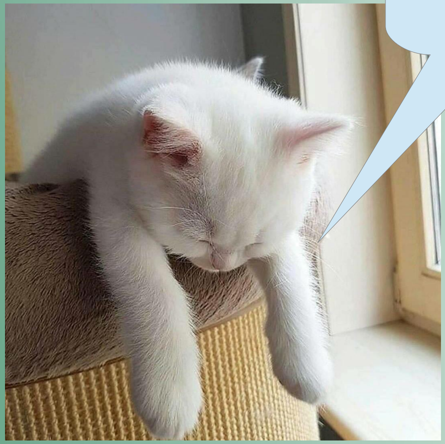
- button = tk.Button*(parent, option =)*
- There are many many options!!!



Button

- button = tk.Button*(parent, option =)*
- There are many many options!!!





Too many!

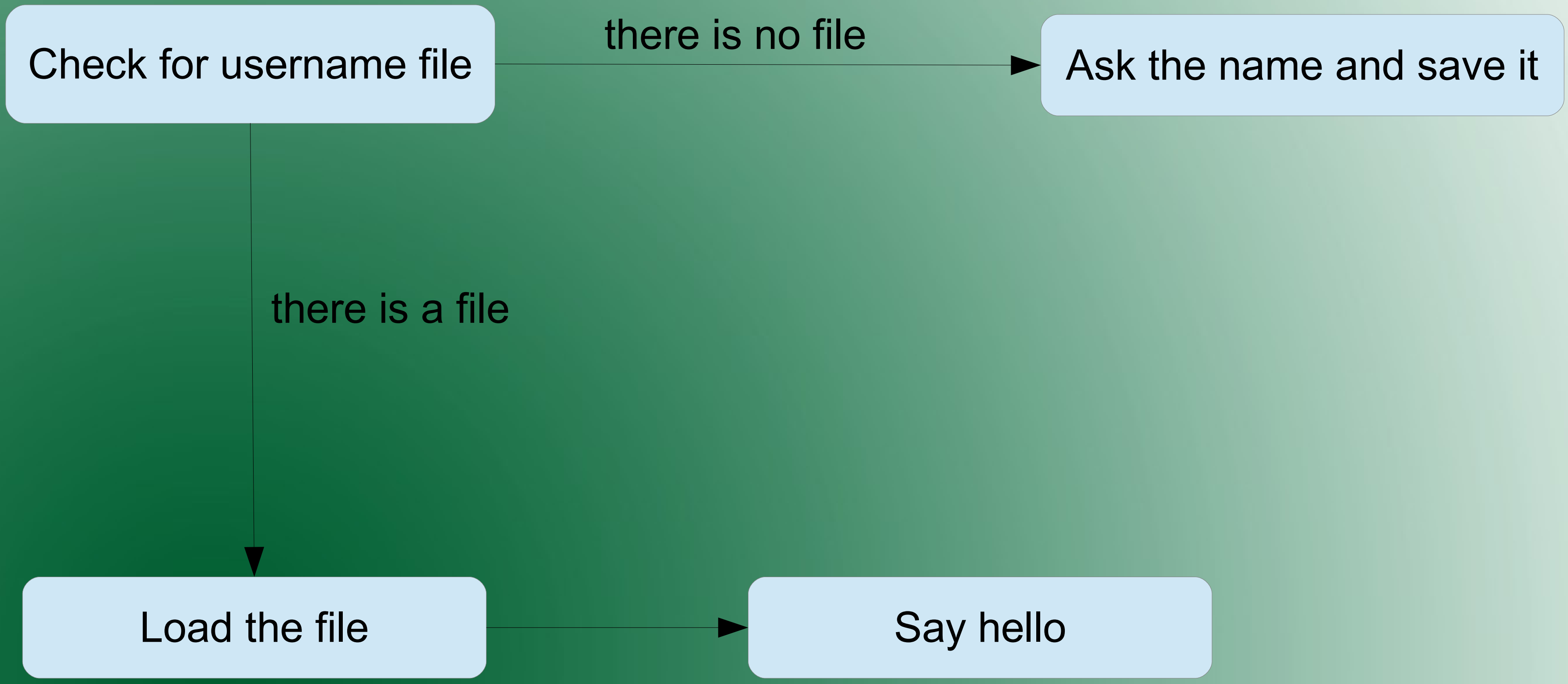
Button options

- text = “”
- command = ….

- Same as Label
- This is amazing! We use a function for command. If you press the button, the function will be used.

Goal – Program that says Hello

- Use pickle, os, and tkinter


Check for username file
there is no file
Ask the name and save it
there is a file
Load the file
Say hello