JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

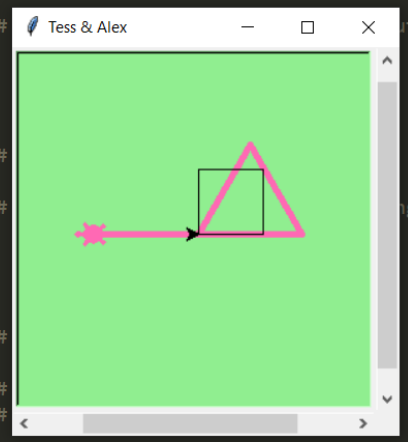
EN.540.635
Software Carpentry

Lecture 9
Building Graphical User Interface with TKinter

Python's Turtle

- An in-built functionality present in python that helps when introducing programming to beginners
- Originally part of and is based off of an old programming language called LOGO

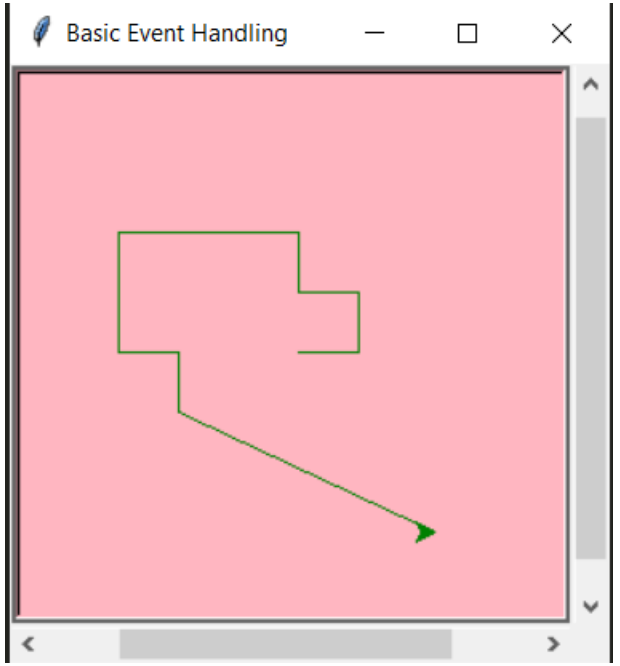
```
turtle.setup(300, 300)
wn = turtle.Screen()          # Set up the window and its attributes
wn.bgcolor("lightgreen")
wn.title("Tess & Alex")

tess = turtle.Turtle()        #
tess.shape('turtle')
tess.color("hotpink")
tess.pensize(5)

alex = turtle.Turtle()        #

tess.forward(80)              #
tess.left(120)
tess.forward(80)
tess.left(120)
tess.forward(80)
tess.left(120)                #

tess.right(180)               #
tess.forward(80)              #

alex.forward(50)              # Make alex draw a square
alex.left(90)
alex.forward(50)
alex.left(90)
alex.forward(50)
alex.left(90)
alex.forward(50)
alex.left(90)

wn.mainloop()
```



Event Handling

```
import turtle

# ignore my use of global variables here
# I was sleepy


def forward():
    tess.forward(30)


def turn_left():
    tess.left(90)


def turn_right():
    tess.right(90)


def translate(x, y):
    tess.goto(x, y)


def sleep():
    wn.bye()
```

```
if __name__ == '__main__':

    turtle.setup(300, 300)
    wn = turtle.Screen()
    wn.title('Basic Event Handling')
    wn.bgcolor('lightpink')
    tess = turtle.Turtle()
    tess.color('green')

    wn.onkey(forward, "Up")
    wn.onkey(turn_left, "Left")
    wn.onkey(turn_right, "Right")
    wn.onkey(sleep, "q")
    wn.onclick(translate)

    wn.listen()
    wn.mainloop()
```



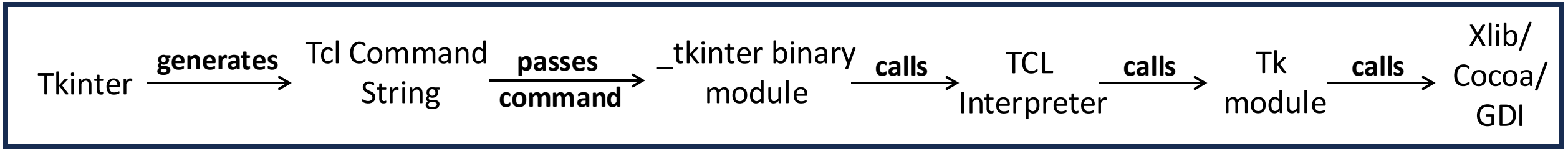
Example adapted from 'How to think like a computer scientist'

What is TKinter

- It is the standard Python interface to the Tk GUI toolkit. Tkinter is just a wrapper of code around the Tk module with added functionality to make it more Pythonic
- Tk is standard GUI toolkit for many dynamics languages like Tcl (Tool Command Line), Python, Javascript, Lua and Ruby.
- Turtle uses the built python library Tkinter to interact with the user
- There are many other GUI packages for python : Kivy, PyQT, WxPython
- Internally Tk uses the facilities of underlying OS : Xlib on Unix/Linux, Cocoa on Mac, GDI on Windows

What is TKinter

Under the hood



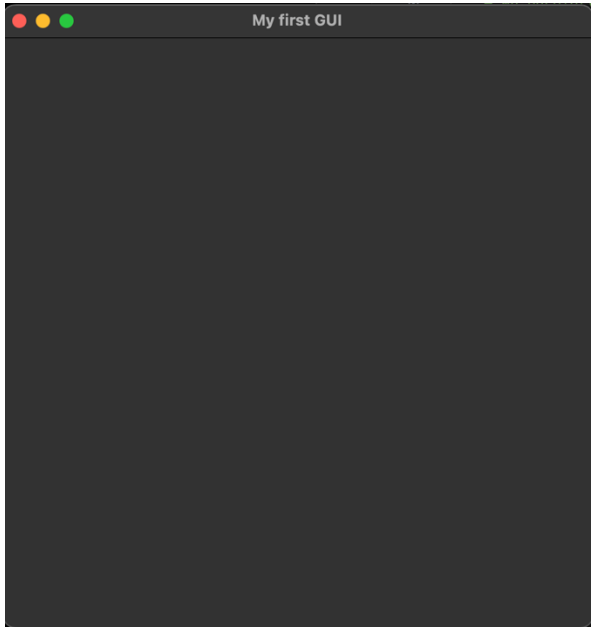
Terminal
username@machinename ~ python3 –m tkinter

Your First Tkinter Application

my_first_tkinter_app.py

```
import tkinter as tk

if __name__ == '__main__':
        root = tk.Tk()
        root.title('My first GUI')
        root.resizable(False, False)
        root.geometry('500x500')

        root.mainloop()
```

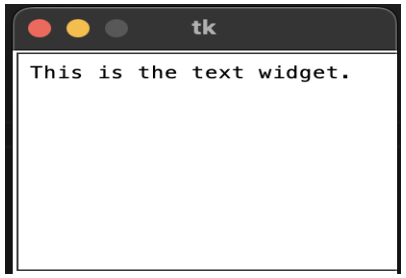
- `import tkinter as tk` — Creating alias
- `root = tk.Tk()` — Launches the TCL interpreter, creates window for app
- `root.title('My first GUI')` — Add title to GUI
- `root.resizable(False, False)` — Does not allow window resize
- `root.geometry('500x500')` — Set the size of the window
- `root.mainloop()` — Displays the window and responds to user input until window terminated



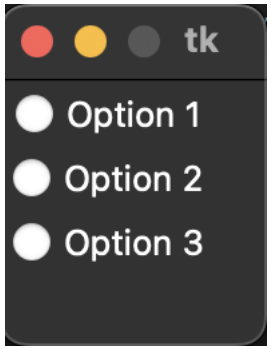
Tkinter widgets

- Widgets add fuctionality to your App.
- Tkinter offers several types of widgets:
  - Frame : A container for other widgets
  - LabelFrame : Displays a label on the border of the frame.
  - Label : Displays text and images
  - Text : A complete text editor inside a window
  - Radiobutton : Single choice selection

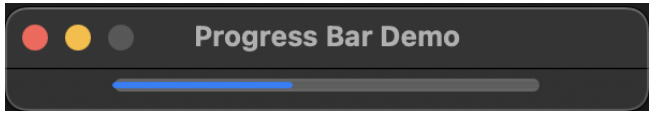
LabelFrame



Text



Radiobutton



Progressbar

- An event occurs when a user interacts with anything inside the GUI.
- Events trigger event handlers that respond to the event.
- For each widget there is a function called bind that links the event handler to the event.

my_first_tkinter_app.py

```
#When you hit enter, the submit event hand is called.
root.bind('<Return>', submit)
```

Adding widgets to the GUI

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

my_first_tkinter_app.py

```
import tkinter as tk

if __name__ == '__main__':
        root = tk.Tk()
        root.title('My first GUI')
        root.resizable(False, False)
        root.geometry('450x200')

        frame = tk.Frame(root, bg='white', bd=4, relief='ridge', height=50,
                         width=450)
        frame.grid()
        frame.grid_propagate(0)

        fn_label = tk.Label(frame, fg='black', bg='white', text='First Name:')
        fn_lablel.grid(row=0, column=0, padx=10, pady=10)

        fn = tk.StringVar()
        fn_entry = tk.Entry(frame, textvariable=fn, width=10)
        fn_entry.grid(row=0, column=1)

        ln_label = tk.Label(frame, fg='black', bg='white', text='Last Name:')
        ln_label.grid(row=0, column=2, padx=10, pady=10)

        ln = tk.StringVar()
        ln_entry = tk.Entry(frame, textvariable=ln, width=10)
        ln_entry.grid(row=0, column=3)

        root.mainloop()
```

Container in which frame is placed in

Border width & design

Geometry manager

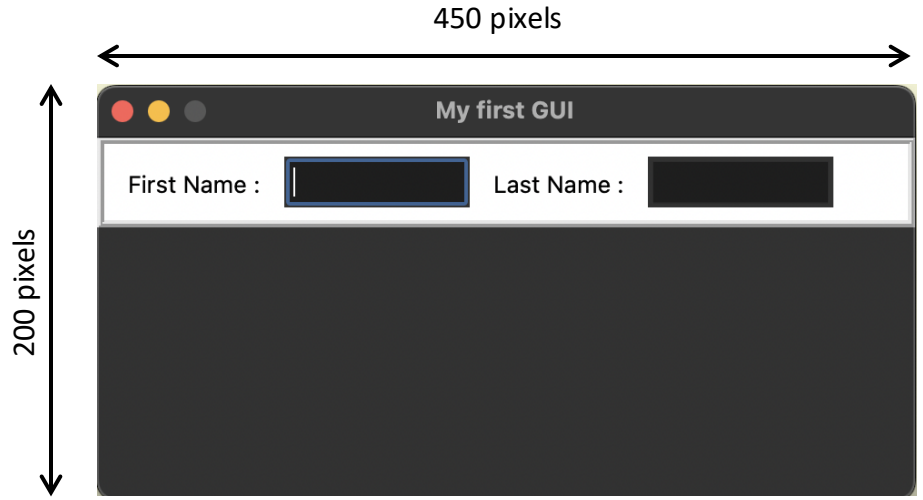
Disables auto adjustment

Place label inside frame

Adds Left and right padding outside widget

Variable to hold first name

Store user entered val. in this variable



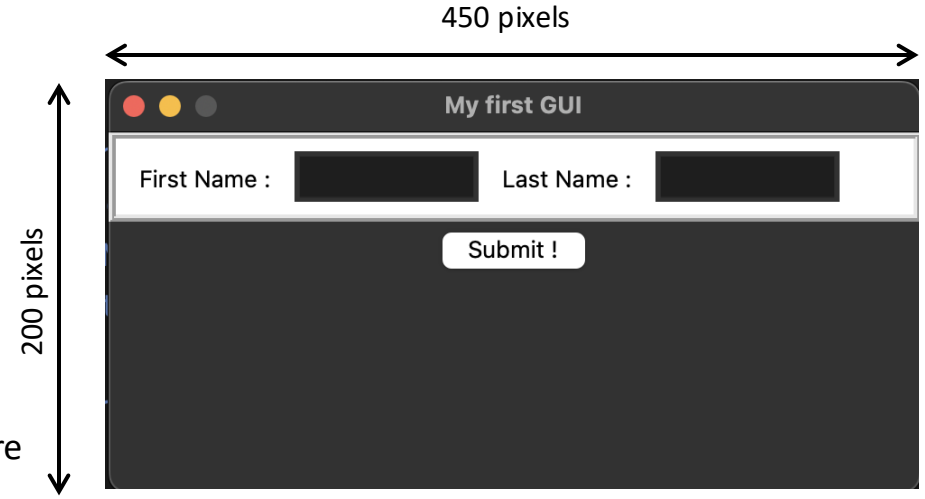
Adding Buttons …

my_first_tkinter_app.py

```
import tkinter as tk

if __name__ == '__main__':

                    .
                    .
                    .

        def submit(event):
                    print('Submitted !')

        submit_button = tk.Button(root, text='Submit !',
                    command=lambda : print('Submitted ! (from button)')
        submit_button.grid(row=1, column=0)

        root.bind('<Return>', submit)

        root.mainloop()
```

Event handler, must always an argument. Here we name it as event.

Whenever enter key is pressed in the root frame, calls the event handler



Writing Tkinter Application Using Classes

- Written in procedural format.
- Makes use of global variables.
- Cannot be imported into other python scripts

```python
if __name__ == '__main__':

    root = tk.Tk()
    root.title('My first GUI')
    root.resizable(False, False)
    root.geometry('450x200')

    frame = tk.Frame(root, bg='white', bd=4, relief='ridge', height=50, width=450)
    frame.grid(row=0, column=0)
    frame.grid_propagate(0)

    first_name_label = tk.Label(frame, fg='black', bg='white', text='First Name :')
    first_name_label.grid(row=0, column=0, padx=10, pady=10)

    first_name = tk.StringVar()
    first_name_entry = tk.Entry(frame, textvariable=first_name, width=10)
    first_name_entry.grid(row=0, column=1)

    last_name_label = tk.Label(frame, fg='black', bg='white', text='Last Name :')
    last_name_label.grid(row=0, column=2, padx=10, pady=10)

    last_name = tk.StringVar()
    last_name_entry = tk.Entry(frame, textvariable=last_name, width=10)
    last_name_entry.grid(row=0, column=3)

    def submit(event):
        print('Submitted !')

    submit_button = tk.Button(root, text='Submit !', command=lambda : print('Submitted!'))
    submit_button.grid(row=1)

    root.bind('<Return>', submit)
```

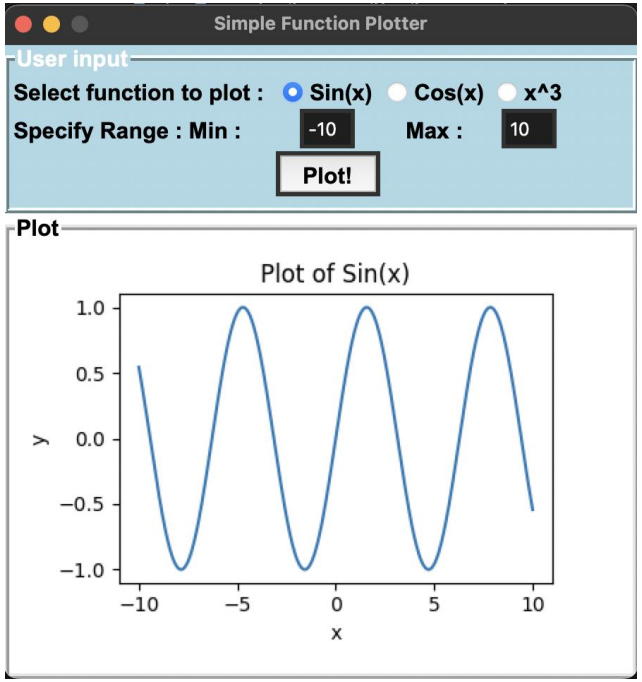
- Allows importing into other python scripts.
- Can easily extend and add more UI elements through inheritance.

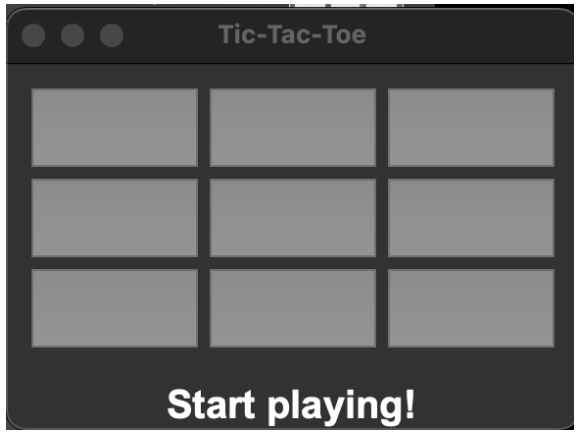
```python
class MyFirstGUI(tk.Frame):
    def __init__(self, main=None, set_height=None, set_width=None):
        super().__init__(main)
        self.main = main
        self.app_height = set_height
        self.app_width = set_width
        main.title('My First GUI')
        main.resizable(False, False)
        main.geometry(f'{self.app_width}x{self.app_height}')
        self.create_user_layout()
        self.main.bind('<Return>', self.submit)

    def create_user_layout(self):
        self.user_frame = tk.Frame(self.main, bg='white', bd=4, relief='ridge')
        self.user_frame['height'] = 50
        self.user_frame['width'] = self.app_width
        self.user_frame.grid()
        self.user_frame.grid_propagate(0)

        self.first_name_label = tk.Label(self.user_frame, fg='black', bg='white', text='First Name :')
        self.first_name_label.grid(row=0, column=0, padx=10, pady=10)

        self.first_name = tk.StringVar()
        self.first_name_entry = tk.Entry(self.user_frame, textvariable=self.first_name, width=10)
        self.first_name_entry.grid(row=0, column=1)

        self.last_name_label = tk.Label(self.user_frame, fg='black', bg='white', text='Last Name :')
        self.last_name_label.grid(row=0, column=2, padx=10, pady=10)

        self.last_name = tk.StringVar()
        self.last_name_entry = tk.Entry(self.user_frame, textvariable=self.last_name, width=10)
        self.last_name_entry.grid(row=0, column=3)

        self.button_frame = tk.Frame(self.main)
        self.button_frame.grid(row=1)

        self.submit_button = tk.Button(self.button_frame, text='Submit !',
                                       command=lambda : tk.messagebox.showinfo('Information', f'Hello {self.first_name.get()} {self.last_name.get()}'))
        self.submit_button.grid(ipadx=1, ipady=1)

        self.quit_button = tk.Button(self.button_frame, text='QUIT', fg='red', command=self.main.destroy)
        self.quit_button.grid(row=0, column=1, ipadx=1, ipady=1)

    def submit(self, event):
        tk.messagebox.showinfo('Information', f'Hello {self.first_name.get()} {self.last_name.get()}')
```

Getting User Input through .get() function

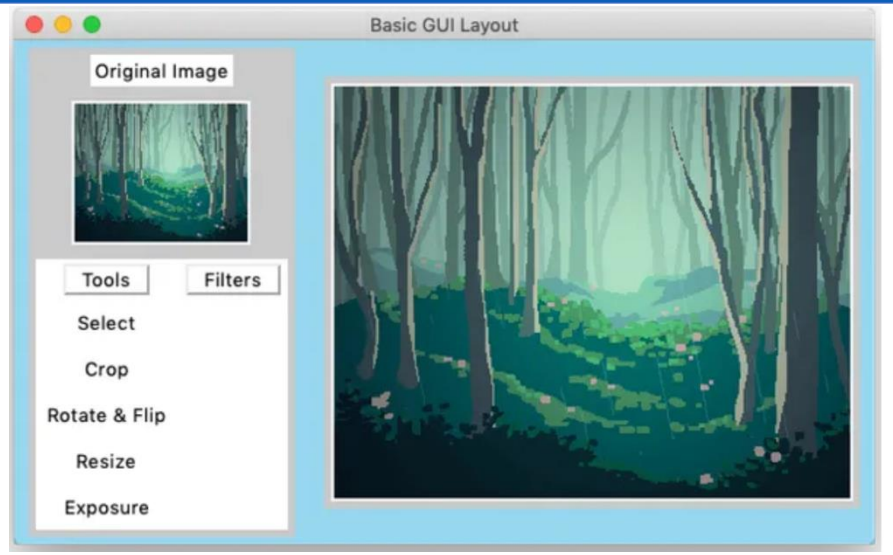
Event handler uses messagebox

Some Examples of Applications Using Tkinter

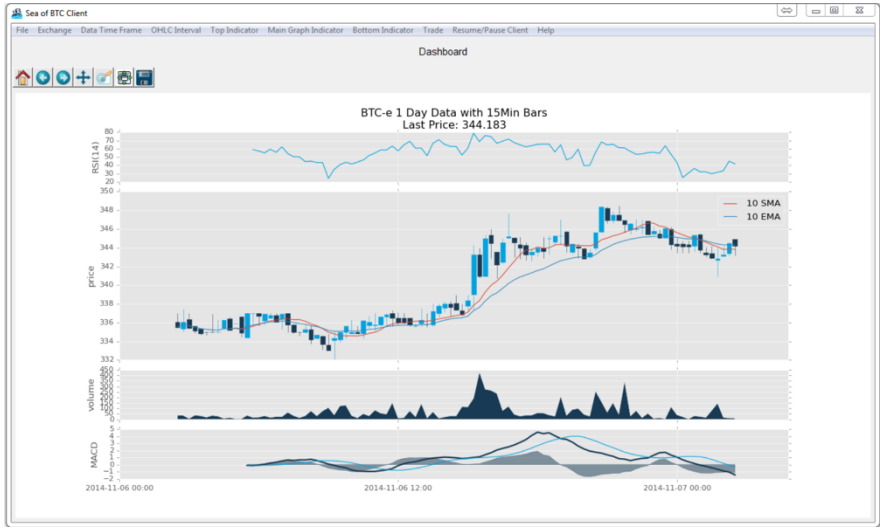




Ref : https://github.com/SarcasticMinion/Tic_Tac_Toe

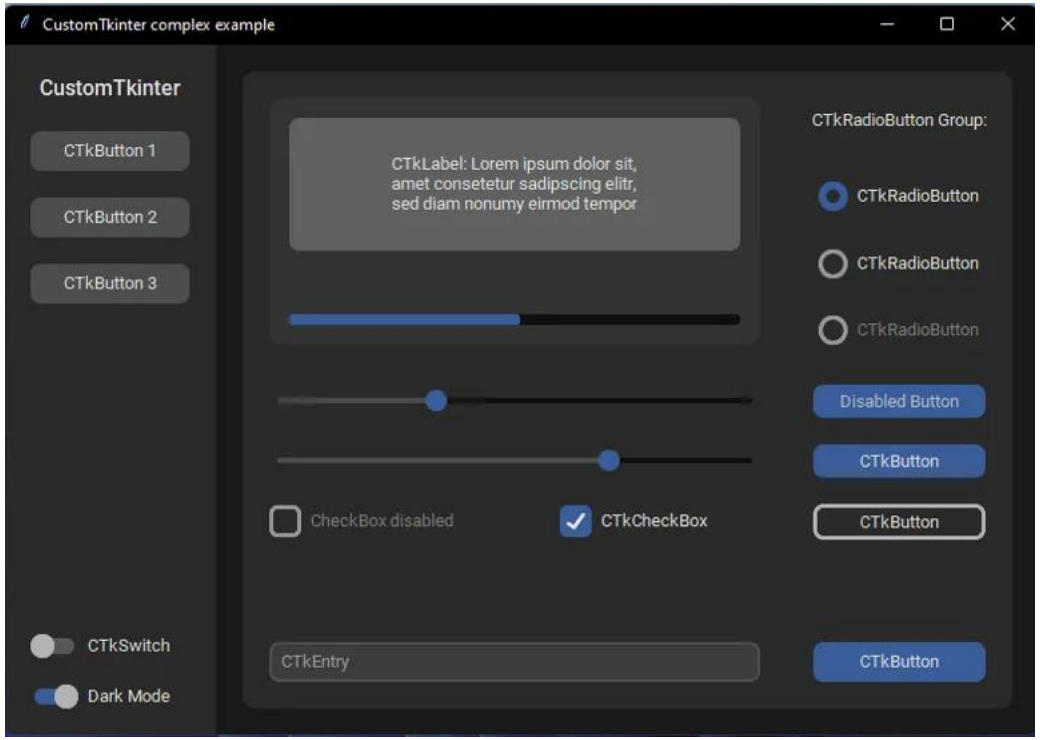


Ref : https://www.pythonguis.com/faq/pack-place-and-grid-in-tkinter/



Ref : https://pythonprogramming.net/tkinter-depth-tutorial-making-actual-program/

Some Examples of Applications Using Tkinter (Cont.)



There are two things to remember:

1. If your idea is new that never existed before, a beautiful user interface (GUI) is not required. People will use your product without hesitation.
2. If your idea is old, then a beautiful user interface (GUI) is what makes them stick to your product.

Ref : https://medium.com/@fareedkhandev/modern-gui-using-tkinter-12da0b983e22

References:

[1] https://tkdocs.com/shipman/tkinter.pdf
[2] https://docs.python.org/3/library/tk.html

Oct 29 : Demo – Building a simple calculator using Tkinter !