GUI

CS196 - Lecture 8
Instructor: Dr. V

Popular Python GUI libraries

Tkinter	Win, MacOS, Linux	Free	easy, built-in	limited features
PySimpleGui	Win, MacOS, Linux	Free	easiest	limited features
Kivy	iOS, Android, Win, MacOS, Linux	Free	mostly easy	modern UI, high performance
PyQT	Win, MacOS, Linux (maybe mobile)	Release code or pay	tougher	enterprise grade
PySide		Free		
wxPython	Win, MacOS, Linux	Free	mostly easy	advanced customizable widgets

Use pip (or pip3) to install libraries

- download and install libraries
  - pip (python default package manager); e.g.,

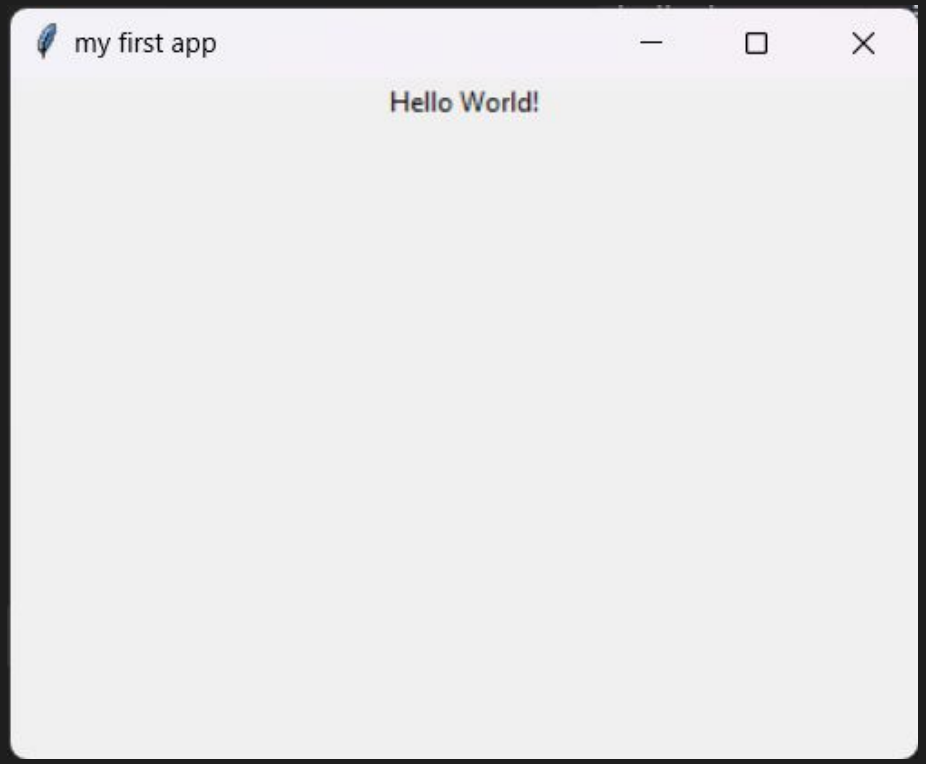
```
> pip install kivy
> pip install mysimplegui
> pip install ttkbootstrap
```

tkinter hello world

```python
import tkinter as tk

# create window
window = tk.Tk()

# set window title and size
window.title("my first app")
window.geometry('400x300')

# add label to window
greeting = tk.Label(text="Hello World!")
greeting.pack()

# keep window open, listen for GUI events
window.mainloop()
```

tkinter hello world

```python
import tkinter as tk

# create window
window = tk.Tk()

# set window title and size
window.title("my first app")
window.geometry('400x300')

# add label to window
greeting = tk.Label(text="Hello World!")
greeting.pack()

# keep window open, listen for GUI events
window.mainloop()
```

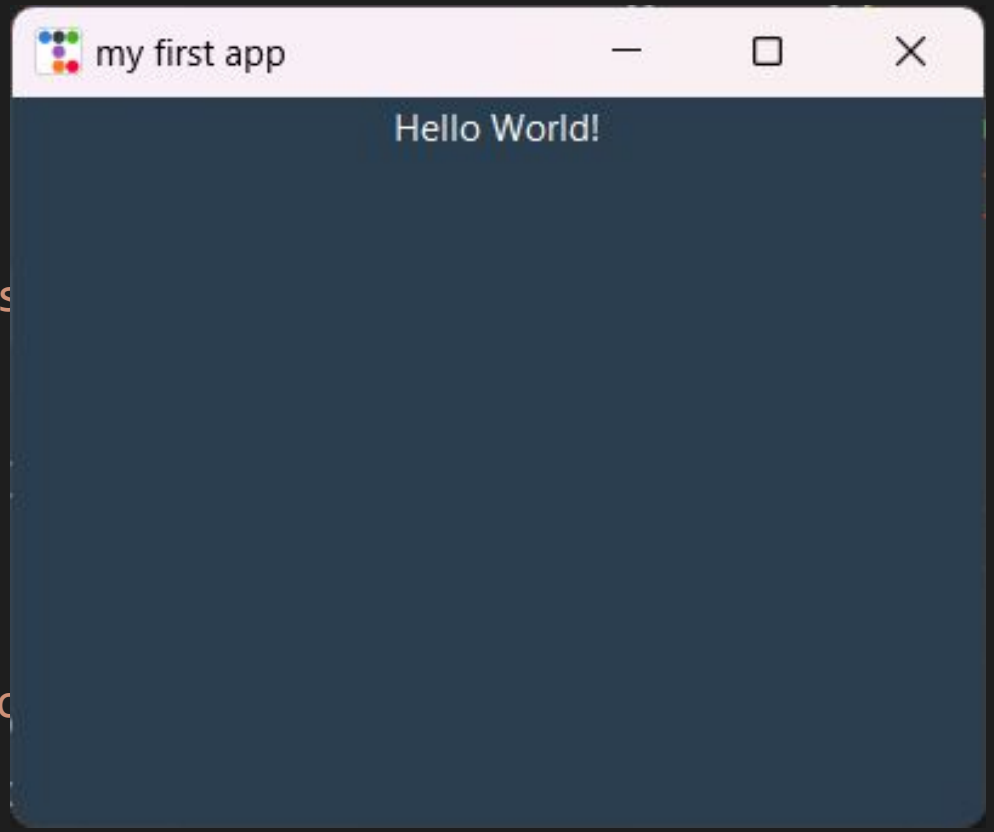


ttkbootstrap hello world

```python
import ttkbootstrap as tk

# create window
window = tk.Window(themename='superhero')

# set window title and size
window.title("my first app")
window.geometry('400x300')

# add label to window
greeting = tk.Label(text="Hello World!")
greeting.pack()

# keep window open, listen for GUI events
window.mainloop()
```

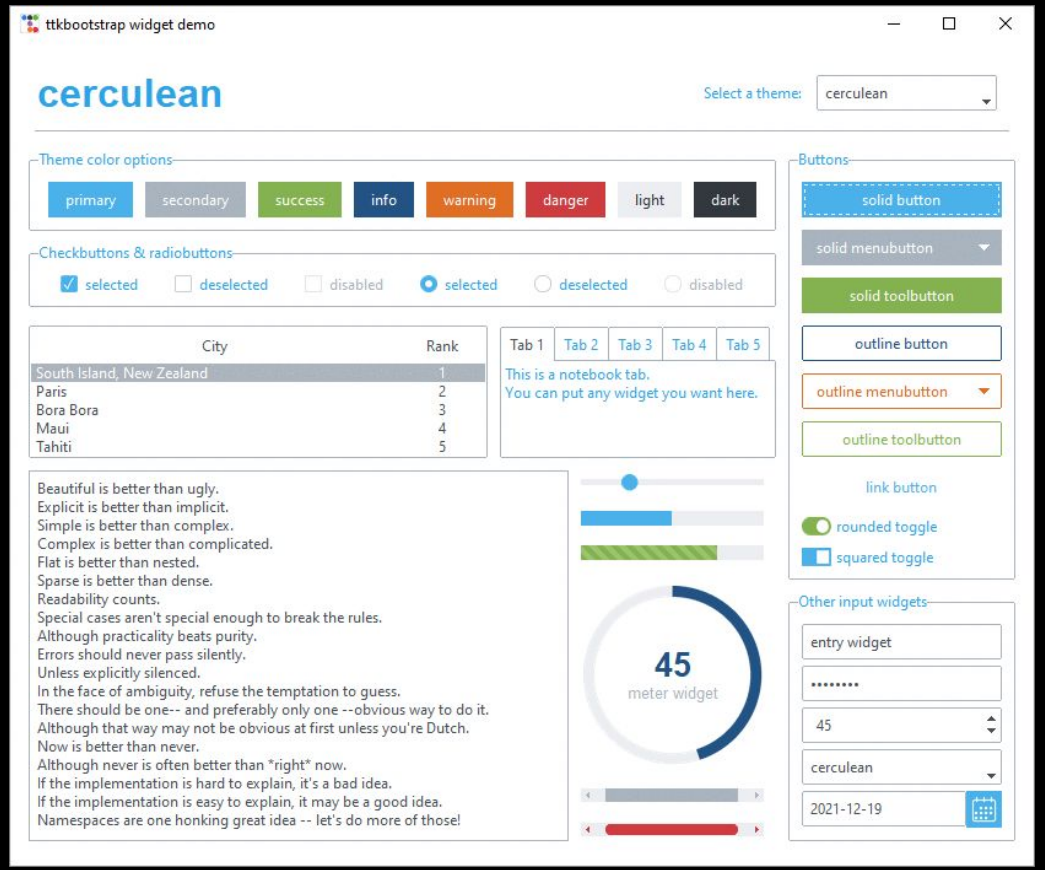
ttkbootstrap hello world

```
1  import ttkbootstrap as tk
2
3  # create window
4  window = tk.Window(themename='s
5
6  # set window title and size
7  window.title("my first app")
8  window.geometry('400x300')
9
10 # add label to window
11 greeting = tk.Label(text="Hello
12 greeting.pack()
13
14 # keep window open, listen for GUI events
15 window.mainloop()
```



ttkbootstrap

- everything that tkinter has, but also
  - predefined themes
  - custom themes
  - extra widgets
  - predefined widget styles (e.g., primary, secondary, success; like bootstrap)

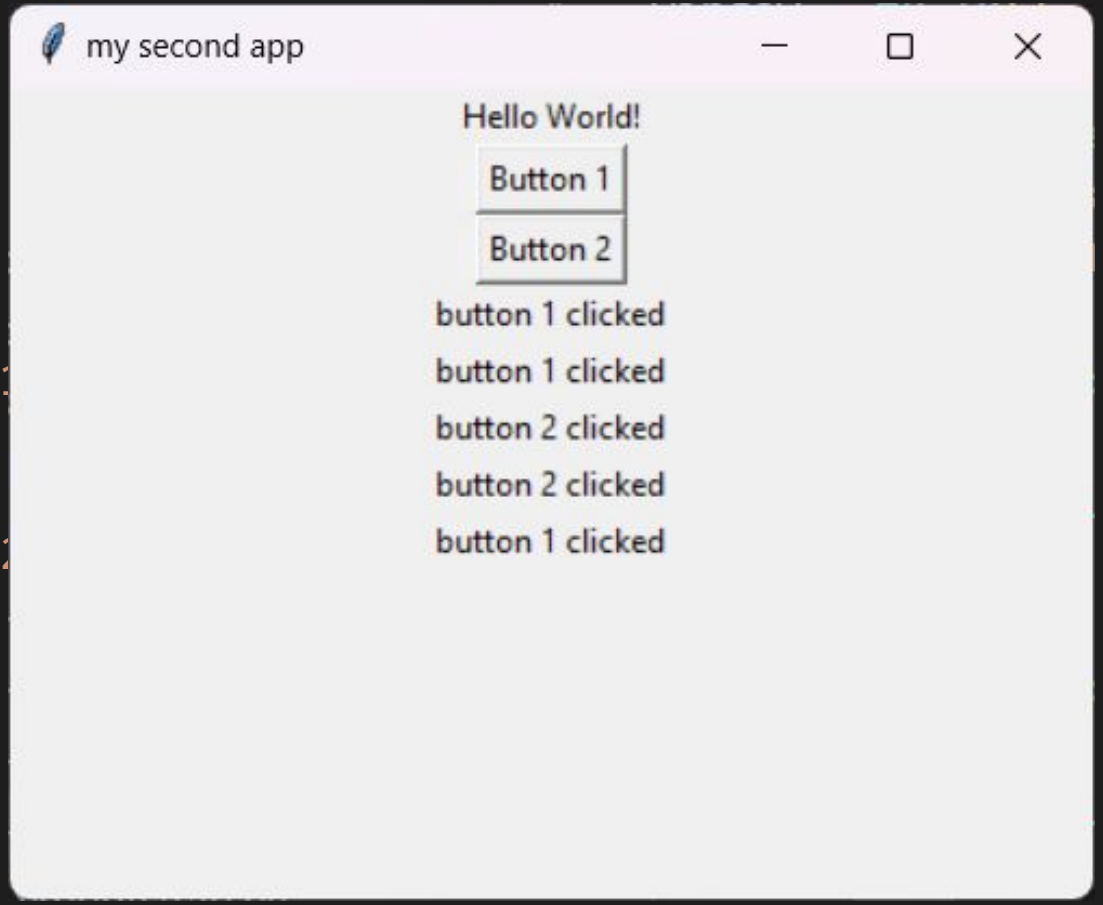


tkinter buttons

```python
import tkinter as tk
window = tk.Tk()
window.geometry('400x300')

def btn1clicked():
    tk.Label(text="button 1 clicked").pack()

def btn2clicked():
    tk.Label(text="button 2 clicked").pack()

tk.Label(text="Hello World!").pack()
tk.Button(text="Button 1", command=btn1clicked).pack()
tk.Button(text="Button 2", command=btn2clicked).pack()

window.mainloop()
```

tkinter buttons

```
1  import tkinter as tk
2  window = tk.Tk()
3  window.geometry('400x300')
4
5  def btn1clicked():
6      tk.Label(text="button 1
7
8  def btn2clicked():
9      tk.Label(text="button 2
10
11 tk.Label(text="Hello World
12 tk.Button(text="Button 1",
13 tk.Button(text="Button 2",
14
15 window.mainloop()
```

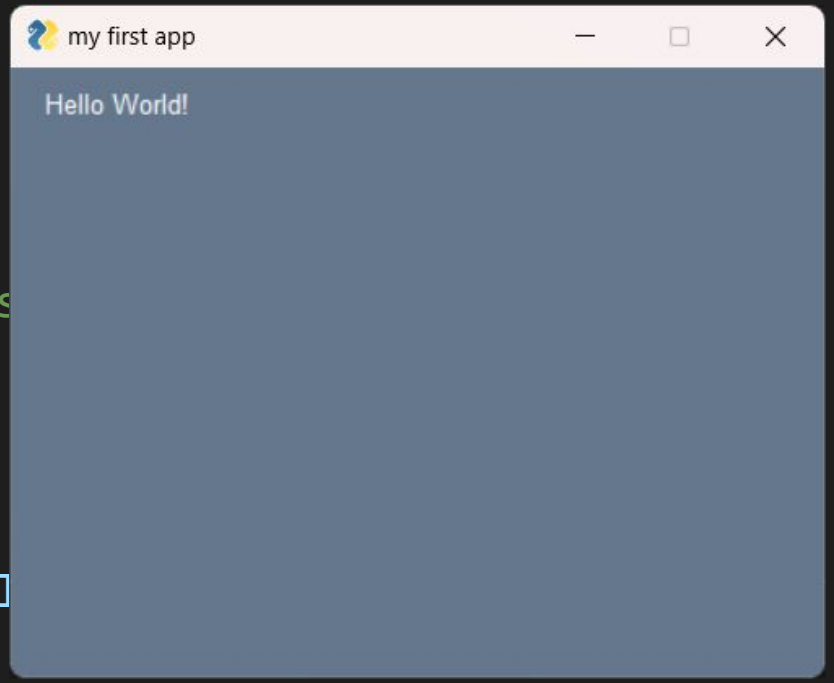


PySimpleGUI hello world

```
1  import PySimpleGUI as gui
2
3  layout = [                          # set up layout
4      [gui.Text("Hello World!")]
5  ]
6
7  # create window with our layout, and set its title and size
8  window = gui.Window("my first app", layout, size=(400,300))
9
10 while True:                         # event loop
11     event, values = window.read() #  read event from window
12     if event == gui.WIN_CLOSED:   #  if user is closing the window
13         break                     #   break out of the loop
14
15 window.close()
```

PySimpleGUI hello world

```
1  import PySimpleGUI as gui
2
3  layout = [                                  # s
4      [gui.Text("Hello World!")]
5  ]
6
7  # create window with our layout, and
8  window = gui.Window("my first app", l
9
10 while True:                                 # event loop
11     event, values = window.read() #  read event from window
12     if event == gui.WIN_CLOSED:   #  if user is closing the window
13         break                     #   break out of the loop
14
15 window.close()
```

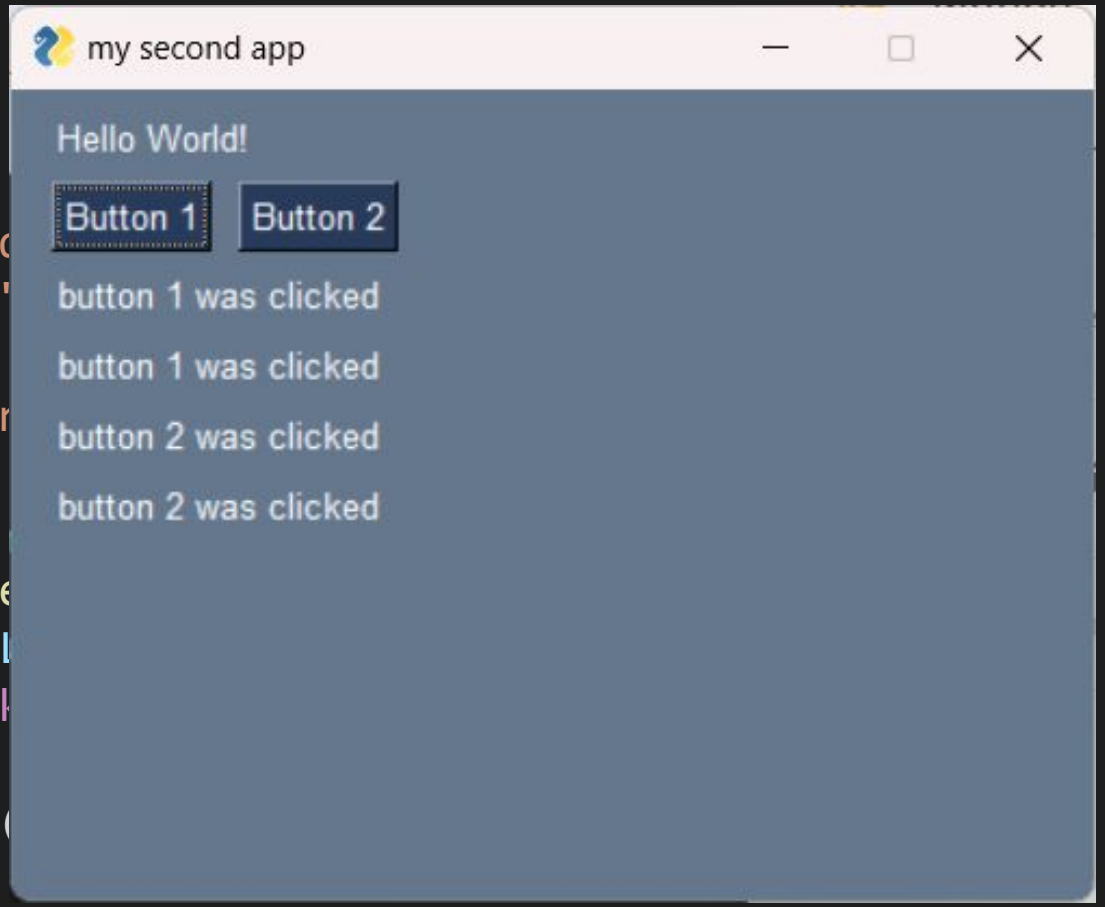


PySimpleGUI buttons

```
1  import PySimpleGUI as gui
2  layout = [
3      [gui.Text("Hello World!")],
4      [gui.Button("Button 1"), gui.Button("Button 2")] ]
5
6  win = gui.Window("my second app", layout, size=(400,300))
7
8  while True:
9      event, values = win.read()
10     if event == gui.WIN_CLOSED:
11         win.close(); break
12     elif event == "Button 1":
13         win.extend_layout(win, [[gui.Text("button1 clicked")]])
14     elif event == "Button 2":
15         win.extend_layout(win, [[gui.Text("button2 clicked")]])
```

PySimpleGUI buttons

```
1  import PySimpleGUI as gui
2  layout = [
3      [gui.Text("Hello Worl
4      [gui.Button("Button 1"
5
6  win = gui.Window("my secon
7
8  while True:
9      event, values = win.re
10     if event == gui.WIN_CL
11         win.close(); break
12     elif event == "Button
13         win.extend_layout(
14     elif event == "Button
15         win.extend_layout(win, [[gui.Text("button2 clicked")]])
```

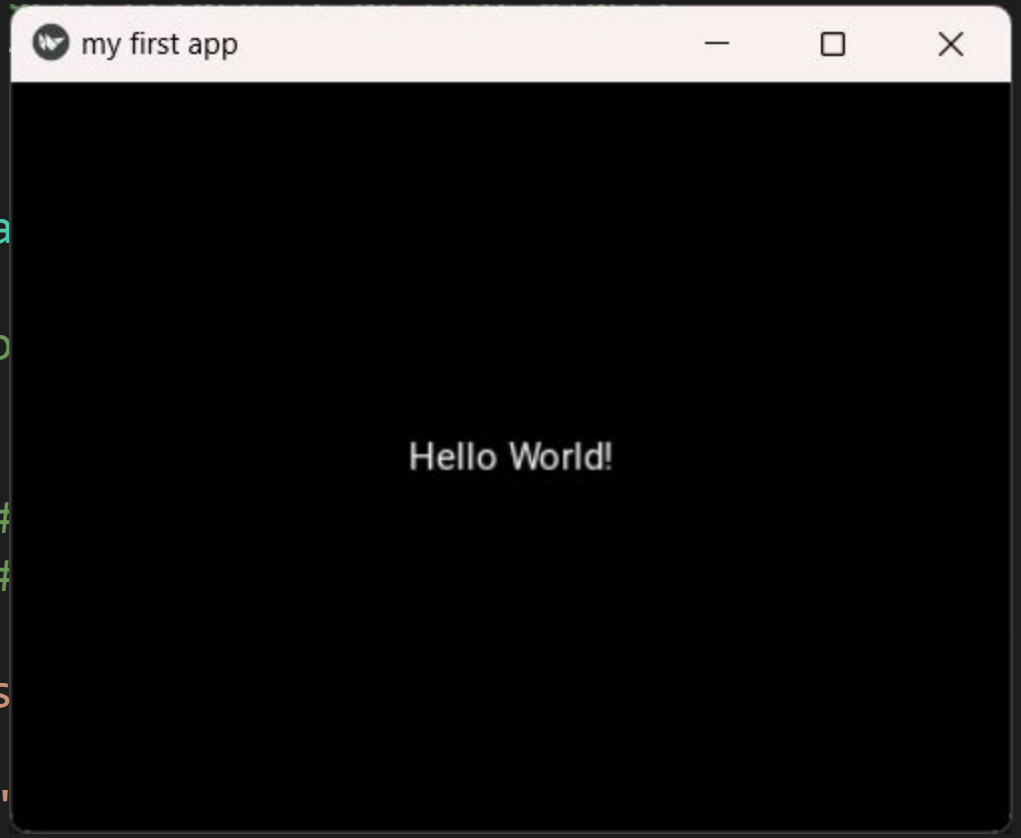


Kivy hello world

```python
from kivy.app import App
from kivy.uix.label import Label
from kivy.core.window import Window
# set window size (kivy can only have 1 window)
Window.size = (400,300)

class MyApp(App):            # create class that extends App class
    def build(self):         # override build() method
        # set window title
        self.title = 'my first app'
        # create text label, return it as root widget
        return Label(text = "Hello World!")

# create instance of MyApp, start app by calling its run() method
MyApp().run()
```

Kivy hello world

```
1  from kivy.app import App
2  from kivy.uix.label import La
3  from kivy.core.window import
4  # set window size (kivy can o
5  Window.size = (400,300)
6
7  class MyApp(App):              #
8      def build(self):           #
9          # set window title
10         self.title = 'my firs
11         # create text label,
12         return Label(text = "
13
14 # create instance of MyApp, start app by calling its run() method
15 MyApp().run()
```



KV design language

- You can use the **kv** design language to create your Kivy applications
  - https://kivy.org/doc/stable/guide/lang.html
- Easiest way to use it is to create a kv file called **somename.kv** file, and in your .py file create a class called **SomenameApp**, where somename and Somename are the same (except for capitalization)

Kivy hello world using a .kv file

```python
from kivy.app import App

class HelloApp(App): pass

HelloApp().run()



# hello.py
```

HelloApp refers to hello.kv

```
#:kivy 2.2
Label:
    text: 'Hello World'
    font_size: '48pt'



# hello.kv
```

Kivy hello world using a .kv file

```python
from kivy.app import App

class HelloWorldApp(App): pass

HelloWorldApp().run()



# hello.py
```

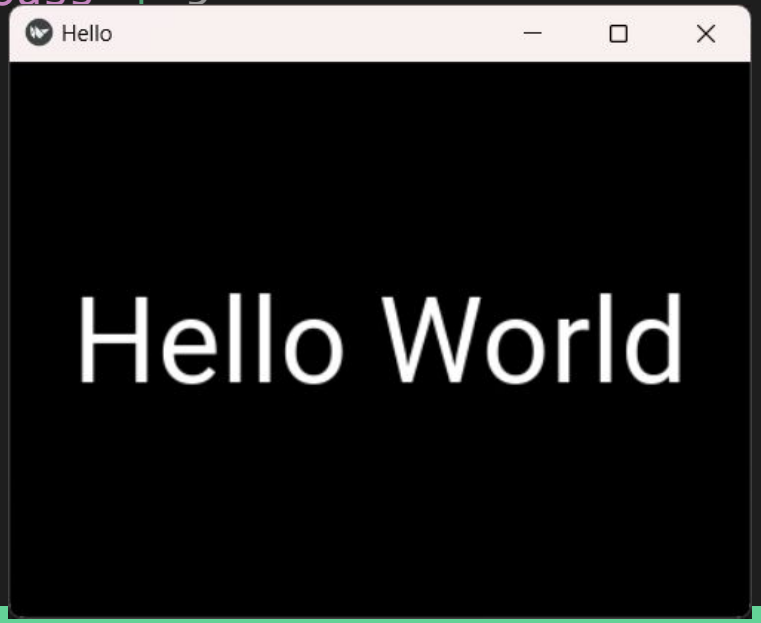
```
#:kivy 2.2
Label:
    text: 'Hello World'
    font_size: '48pt'



# helloworld.kv
```

HelloWorldApp refers to helloworld.kv

Kivy hello world using a .kv file

```python
from kivy.app import App

Window.size = (400,300)

class HelloWorldApp(App): pass

HelloWorldApp().run()




# hello.py
```

```kivy
#:kivy 2.2
Label:
    text: 'Hello World'
    font_size: '48pt'

```



Kivy app with 2 buttons, using a .kv file (hello2.kv)

```
GridLayout: # root widget will be of class GridLayout
    cols: 1 #    grid will have one column
    Label:
        text: 'Hello World'
        size_hint: 0.8, 0.1
    Button:
        text: 'Button 1'
        size_hint: 0.8, 0.1
        on_press: app.button1clicked()
    Button:
        text: 'Button 2'
        size_hint: 0.8, 0.1
        on_press: app.button2clicked()
    Label:
        id: content
```

Kivy app with 2 buttons, using a .kv file (hello2.py)

```python
from kivy.app import App
from kivy.core.window import Window

Window.size = (400,600)

class Hello2App(App):
    def button1clicked(self):
        self.root.ids.content.text += 'button 1 clicked\n'
    def button2clicked(self):
        self.root.ids.content.text += 'button 2 clicked\n'

Hello2App().run()
```

Kivy app with 2 buttons, using a .kv

```
1  from kivy.app import App
2  from kivy.core.window import Window
3
4  Window.size = (400,600)
5
6  class Hello2App(App):
7      def button1clicked(self):
8          self.root.ids.content.text += 'butt
9      def button2clicked(self):
10         self.root.ids.content.text += 'butt
11
12 Hello2App().run()
13
14
15
```

Kivy app with 2 buttons, using a hello2.kv file

```python
from kivy.app import App
from kivy.core.window \
    import Window

Window.size = 400, 600

class Hello2App(App):
    def btn1(self):
        self.root.ids.content.\
           text+='clicked 1\n'
    def btn2(self):
        self.root.ids.content.\
           text+='clicked 2\n'

Hello2App().run()
```

```kv
GridLayout:
    cols: 1
    Label:
        text: 'Hello World'
        size_hint: 0.8, 0.1
    Button:
        text: 'Button 1'
        size_hint: 0.8, 0.1
        on_press: app.btn1()
    Button:
        text: 'Button 2'
        size_hint: 0.8, 0.1
        on_press: app.btn2()
    Label:
        id: content
```

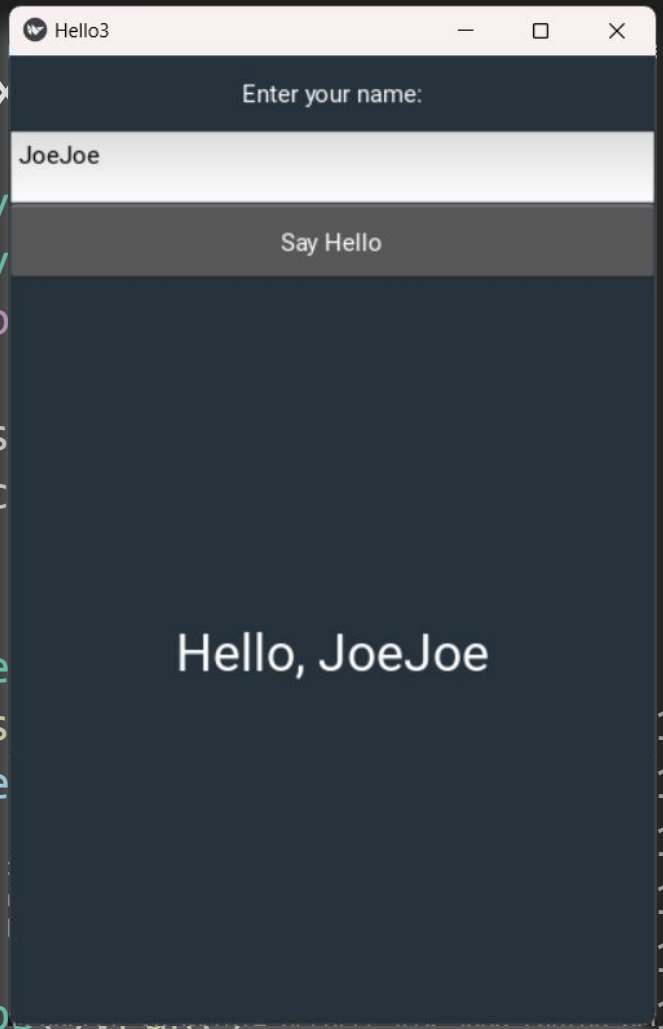
Kivy text input

```python
from kivy.app import App
from kivy.core.window \
    import Window

Window.size = 400, 600
Window.clearcolor = \
             .15, .20, .24, 1

class Hello3App(App):
   def sayHello(self):
      self.root.ids.content.\
        text = 'Hello, ' +  \
        self.root.ids.name.text

Hello3App().run()
```

```yaml
GridLayout:
    cols: 1
    Label:
      text: 'Enter your name:'
      size_hint: 1, 0.1
    TextInput:
      id: name
      size_hint: 1, 0.1
    Button:
      text: 'Say Hello'
      size_hint: 1, 0.1
      on_press: app.sayHello()
    Label:
      id: content
      font_size: '24pt'
```

Kivy tex

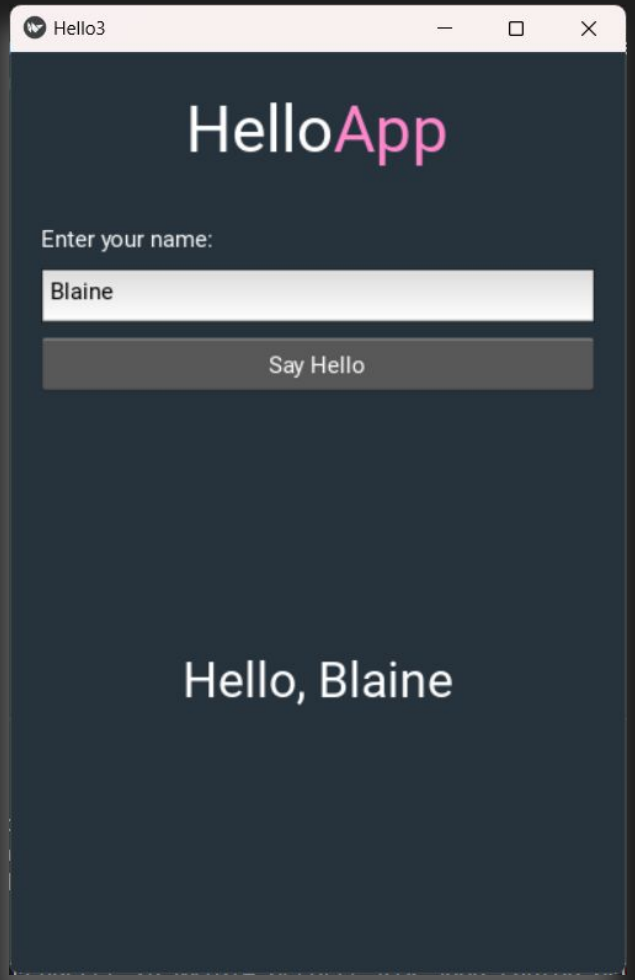
```
1  from kiv
2  from kiv
3      impo
4
5  Window.s
6  Window.c
7
8
9  class He
10     def s
11         se
12
13
14
15 Hello3Ap
```




```
1  GridLayout:
2      cols: 1
3      Label:
4        text: 'Enter your name:'
5        size_hint: 1, 0.1
6      TextInput:
7        id: name
8        size_hint: 1, 0.1
9      Button:
10       text: 'Say Hello'
11       size_hint: 1, 0.1
12       on_press: app.sayHello()
13     Label:
14       id: content
15       font_size: '24pt'
```

Kivy sizing, positioning, markup

```
1  GridLayout:
2    cols: 1
3    size_hint: 0.9, 0.95
4    pos_hint: {"center_x":.5, "center_y":.5}
5    spacing: 10
6    Label:
7      size_hint: 1, 0.2
8      text: 'Hello[color=ff88cc]App[/color]'
9      font_size: '32pt'
10     markup: True
11   Label:
12     size_hint: 1, 0.1
13     text_size: self.size
14     halign: 'left'
15     text: 'Enter your name:'...
```

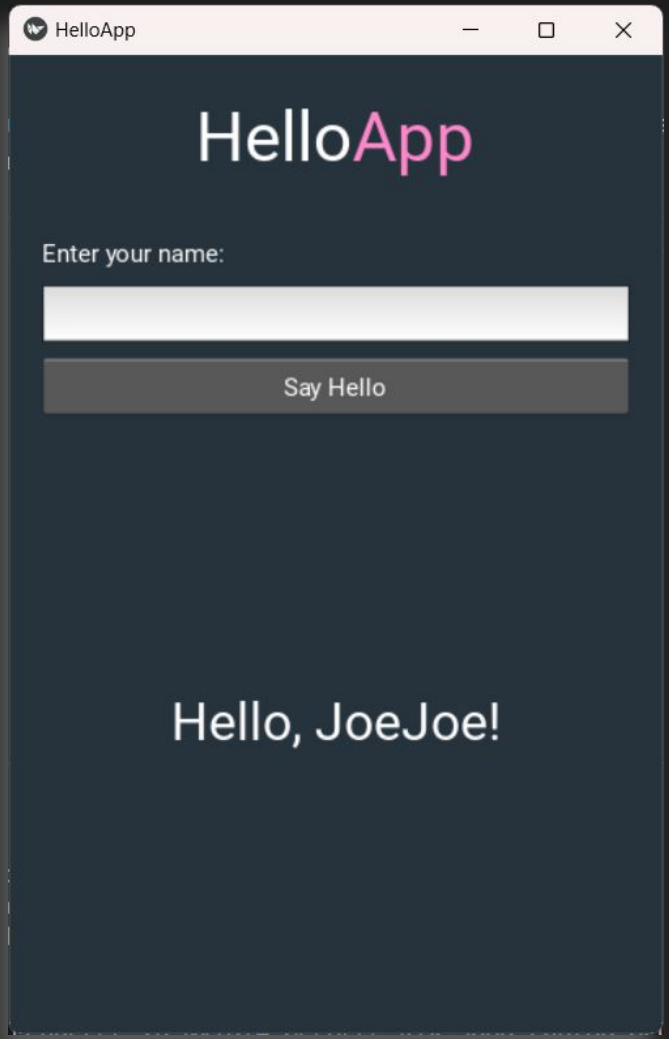


Kivy sizing, positioning, markup

```
1  from kivy.app import App
2  from kivy.core.window import Window
3
4  Window.size = 400, 600
5  Window.clearcolor = .15, .20, .24, 1
6
7  class Hello3App(App):
8    def build(self):
9      self.title='HelloApp'
10   def sayHello(self):
11     self.root.ids.content.text = \
12       f'Hello, {self.root.ids.name.text}!'
13     self.root.ids.name.text=''
14
15 Hello3App().run()
```

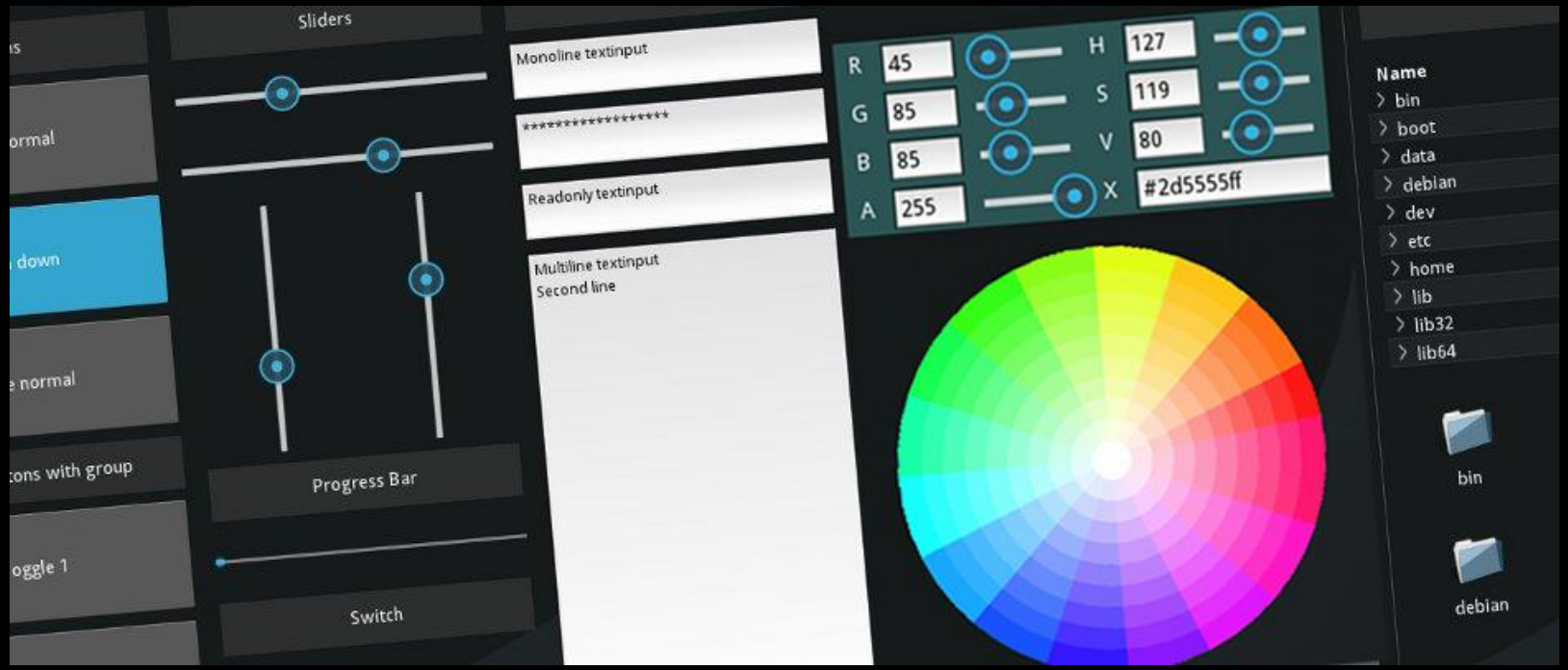
Kivy sizing, positioning, markup

```python
from kivy.app import App
from kivy.core.window import Window

Window.size = 400, 600
Window.clearcolor = .15, .20, .24, 1

class Hello3App(App):
  def build(self):
    self.title='HelloApp'
  def sayHello(self):
    self.root.ids.content.text = \
      f'Hello, {self.root.ids.name.text}!'
    self.root.ids.name.text=''

Hello3App().run()
```



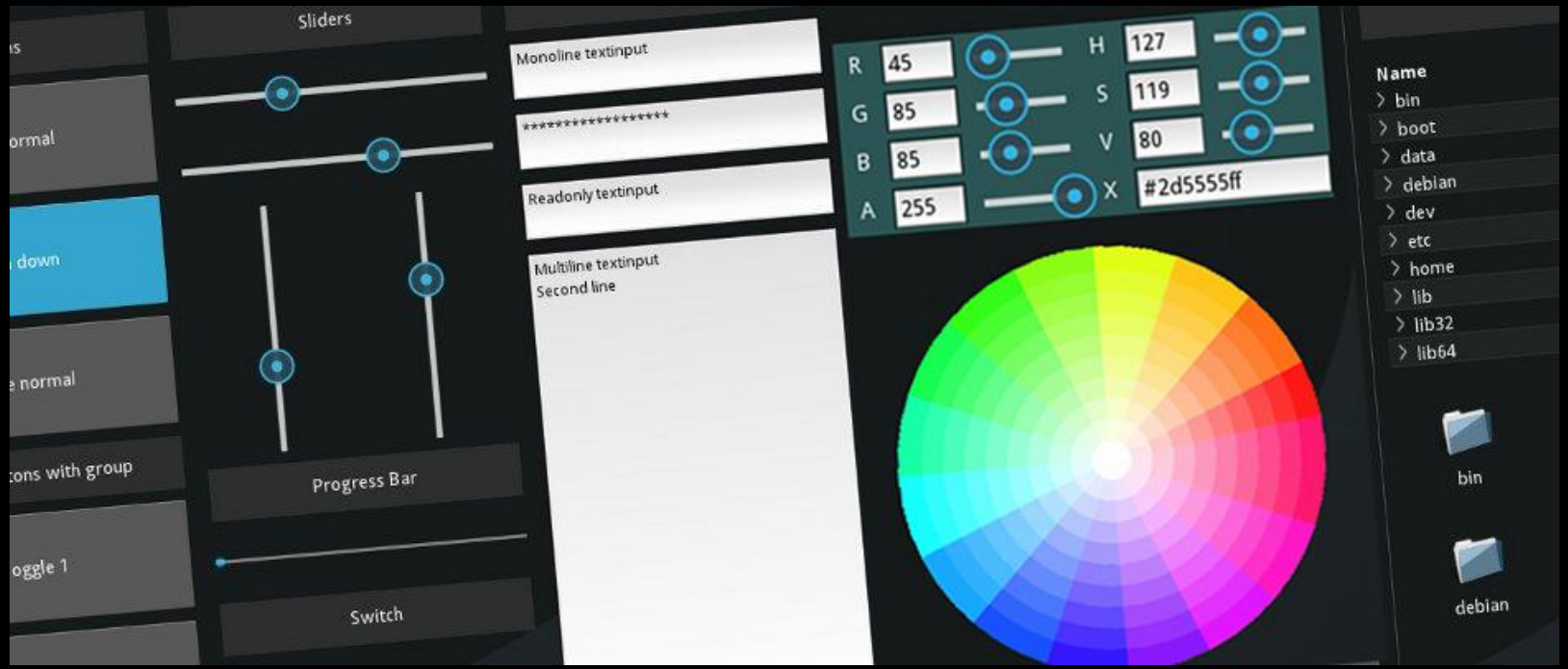
Kivy widgets and layouts

- https://kivy.org/doc/stable/api-kivy.uix.html



Kivy UX widgets (e.g., kivy.uix.label.Label)

- Label, Button, CheckBox, Image, Slider, ProgressBar, TextInput, ToggleButton, Switch, Video, ColorWheel...



Kivy layouts (e.g., kivy.uix.gridlayout.GridLayout)

- AnchorLayout
- BoxLayout
- FloatLayout
- GridLayout
- PageLayout
- RelativeLayout
- ScatterLayout
- StackLayout

pyinstaller and buildozer

- bundle applications from your python code
  - for desktop (Windows, Linux, OS X)
    - pyinstaller
      - https://pyinstaller.org/en/stable/
  - for mobile (iOS, Android)
    - buildozer
      - https://buildozer.readthedocs.io/en/latest/installation.html
      - https://buildozer.readthedocs.io/en/latest/quickstart.html
      - will only work on Linux or MacOS
        - to use on Windows, you have to use WSL
          - https://learn.microsoft.com/en-us/windows/wsl/install

Building a kivy app with PyInstaller

- https://kivy.org/doc/stable/guide/packaging.html

Assignment 7 (due before next lecture)

- Build an interactive Kivy app:
  - Make sure it has at least the following elements
    - at least 2 labels, 3 buttons, 2 inputs
      - each button must do a different thing
      - at least 1 of the buttons must do something based on inputs
  - Up to you whether you want to use a kv design file
  - Submit your source code (and if you have an accompanying .kv file, submit that too) on blackboard
- Extra credit (50pts per OS; up to 200pts; anytime before May 10):
  - Use pyinstaller or buildozer to turn your python code into a standalone executable
  - Record a 15s video of your standalone app being launched, text being entered, buttons being clicked, app closing
  - Submit video on blackboard along with your code