NED University of Engineering and Technology
Department of Computer and Information Systems Engineering

Fall Semester 2020

FE Batch 2020

CS-115 Computer Programming

Online Lecture 12a (Week 10)

Graphical User Interface (GUI)

Dr. Maria Waqas

Previously . . .

- Files

Today's Session . . .

- Graphical User Interface (GUI)

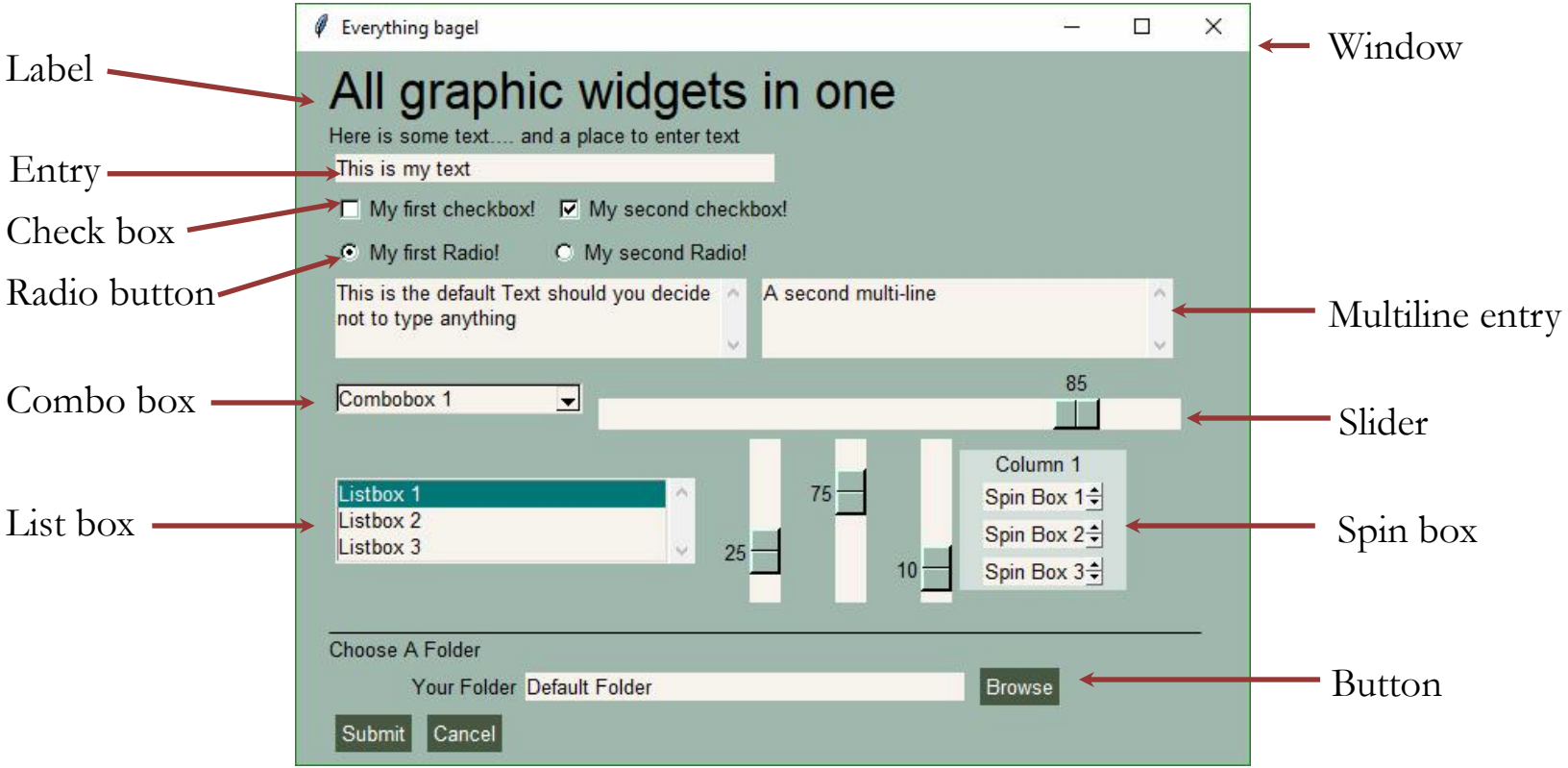
User Interfaces

- **Command Line Interface (CLI)**
  - Text-based user interface.
  - Requires commands to be typed on a computer keyboard.

- **Graphical User Interface (GUI)**
  - A graphical user interface (GUI) is a type of user interface through which users interact with electronic devices via visual indicator representations.
  - GUI objects include icons, cursors, buttons, etc.
  - These graphical elements are sometimes enhanced with sounds, or visual effects like colors, transparency and drop shadows.

- A GUI is considered to be more user-friendly than a text-based command-line interface.

Graphical User Interfaces (GUI)



Graphical User Interfaces (GUIs)

- Graphical User Interface (GUI) consists of basic visual building blocks called widgets
  Examples of widgets: buttons, labels, text entry forms, menus, checkboxes, scrollbars, etc.

- Advantages:
  - Gives a better overview of what an application does.
  - Makes it easier to use the application.

- Event-driven programming is an approach for developing applications in which tasks are executed in response to events (such as button clicks).

- GUI toolkit:
  - `tkinter` – part of Python's Standard Library
  - Other GUI development libraries: `Kivy`, `PyQT`, `WxPython`

Graphical User Interfaces (GUIs)

- **Window:** A rectangular area on display screen.

- **Widget:** Building blocks that makeup the display.
  - There is one main widget, the window; other widgets are placed within it.

- **Frame:** Basic unit of organization for complex layouts.
  - A frame is a rectangular area that contains other widgets.

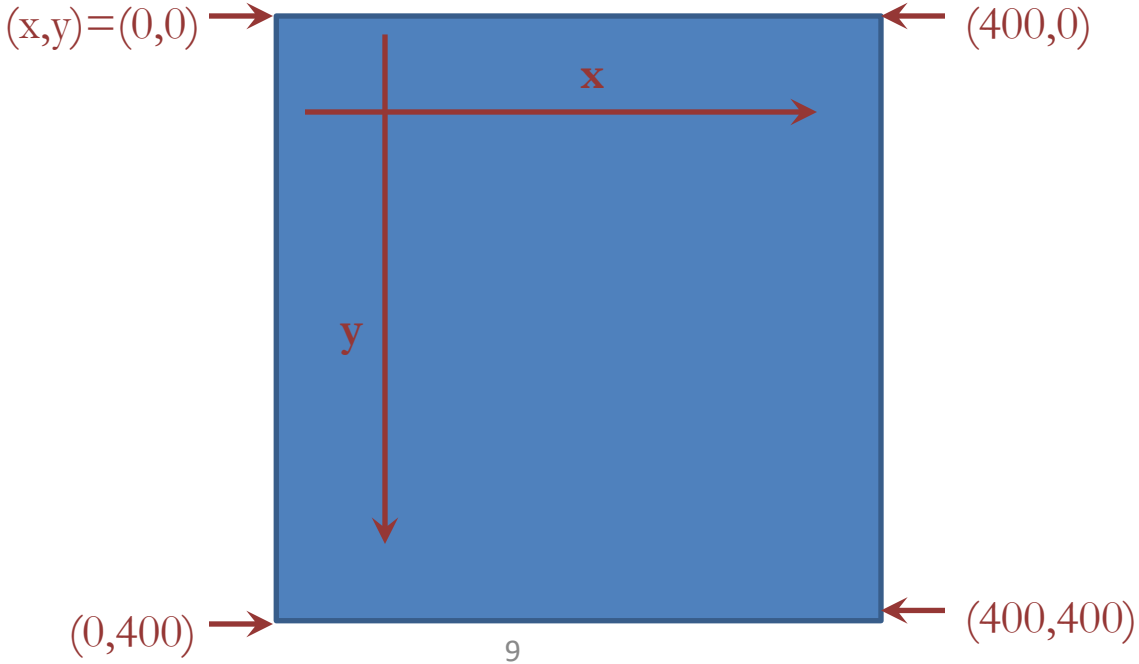
- **Child/parent widgets:** When a widget is created inside a widget, the former is called child and the later is called parent.

Creating a `tkinter` Application

- Import the module `tkinter`.
- Create the main window/container/widget - the parent.
- Add any number of child widgets to the main window.
- Apply event trigger in the widgets.

Window Coordinates x and y

- The ordered pair (x,y) represents a pixel on screen/window
- Consider a window of dimensions 400x400



Creating the Main Container

- Class `Tk` creates the main GUI window of an application.

Example

`root = Tk()` → an object `root` of type `Tk` is instantiated

- It initializes the window manager; a blank window with close, maximize and minimize buttons at the top.

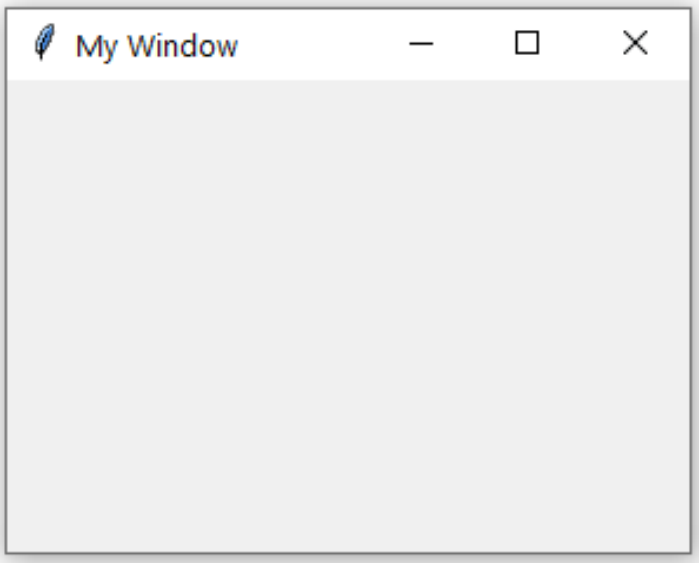
Common methods on `Tk`:

- `title(string=None)`: sets the title for the window.
- `geometry(newGeometry=None)`: sets window geometry in the form 'widthxheight+x+y' (string); x and y are the coordinates of the screen where the window appears.
- `maxsize(width=None,height=None)`: sets dimensions for minimized window.
- `minsize(width=None,height=None)`: sets dimensions for maximized window.
- `destroy()`: destroys the current and all descendants widgets; equivalent to closing a window through close button.
- `mainloop()`: Event listener which runs in infinite loop, until window is closed.

Creating the Main Container

Example 1

```
import tkinter
r=tkinter.Tk()
r.title('My Window')
r.geometry('100x100+200+0')
r.mainloop()
```



- Try using methods minsize and maxsize.

Adding widgets to the GUI Window

- Any number of widgets can be added to the main GUI window or sub-windows

Example

```
myWidget= <widget_name>(MASTER,options)
```

- an object `myWidget` of class `<widget_name>` is made.
- constructs a widget with parent `MASTER`.

- Common widgets / widget classes:

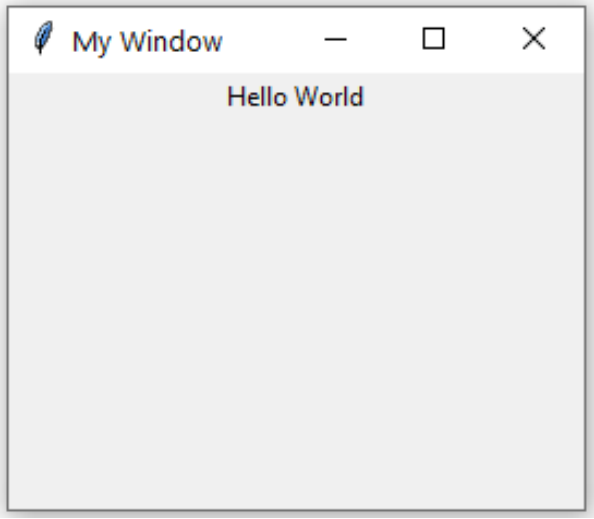
- `Label`: to display text and images
- `Frame`: to organize a group of widgets
- `Entry`: to input single line text from user
- `Button`: to place a clickable button
- `Radiobutton`: used in groups, only one can be selected in a group

} Event-based widgets

The Label Widget

- Label is a display box where some text or image can be displayed.

```
l= Label(MASTER,options)
```



Example 2

```
from tkinter import Tk,Label
r=Tk()
r.title('My Window')
r.geometry('200x200+400+400')
lab1=Label(r,text='Hello World')
lab1.pack()
r.mainloop()
```

places lab1 in r. Default placement is top, middle.

parent widget

Customizing Look of a Widget

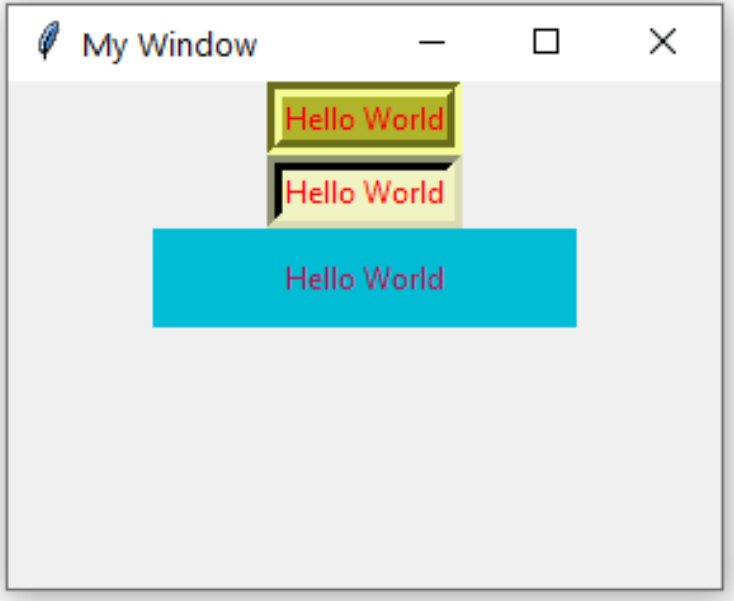
Common options to specify the look of a widget:

- `text`: text to display.
- `image`: image to display.
- `width`: width of widget in pixels (for images) or characters (for text) ; if omitted, size is calculated based on content.
- `height`: height of widget in pixels (for images) or characters (for text); if omitted, size is calculated based on content.
- `relief`: border style; possibilities are 'flat' (default, string), 'groove', 'raised', 'ridge', 'sunken'.
- `borderwidth`: width of border, default is 0 (no border).
- `background`: background color name (as a string).
- `foreground`: foreground color name (as a string).
- `font`: font descriptor (as a tuple with font family name, font size, and optionally, a font style).
- `padx`, `pady`: padding added to the widget along the x- or y-axis.
- `command`: name of function to be called; for event-based widgets only.

Customizing the Label Widget

Example 3

```
from tkinter import Tk,Label
r=Tk()
r.title('My Window')
r.geometry('200x200+400+400')
lab1=Label(r,text='Hello World',
           fg='red',bg='#AFB42B',
           borderwidth=6,relief='groove')
lab1.pack()
lab2=Label(r,text='Hello World',
           fg='red',bg='#F0F4C3',
           borderwidth=6,relief='sunken')
lab2.pack()
lab3=Label(r,text='Hello World',
           fg='#C2185B',bg='#00BCD4',padx=50,pady=10)
lab3.pack()
r.mainloop()
```



Displaying Images using the `Label` Widget

Example 4

```
from tkinter import Tk,Label,PhotoImage
r=Tk()
r.title('My Window')
lab1=Label(r,text='Let\'s Learn Python')
lab1.pack()
photo1=PhotoImage(file='F:/realPython.png')
lab2=Label(r,image=photo1)
lab2.pack()
photo2=PhotoImage(file='python.png')
lab3=Label(r,image=photo2)
lab3.pack()
r.mainloop()
```

Packing Widgets

- The `tkinter` geometry manager is responsible for the placement of widgets within their master.

- If multiple widgets must be laid out, the placement will be computed by the geometry manager using sophisticated layout algorithms (that attempt to ensure that the layout looks good) and using directives given by the programmer.

- `tkinter` has three built-in layout managers:
  - the pack method - organizes widgets in horizontal and vertical boxes.
  - the grid method - places widgets in a two dimensional grid.
  - the place method - positions widgets using absolute positioning.

Packing Widgets – The `pack` Method

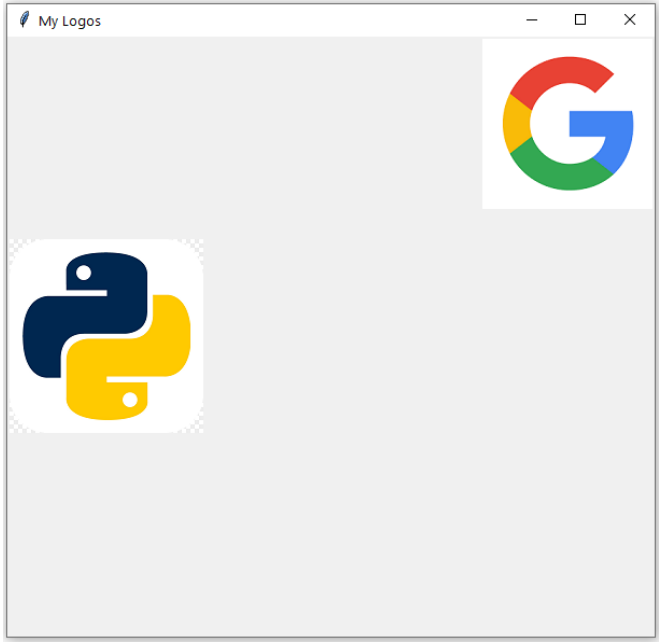
Some useful parameters:

`side`: sets position; possible values are 'top' (default, string), 'bottom', 'left' and 'right'.

`anchor`: sets direction; possible values are 'n', 'nw', 'ne', 'w', 'center', 'e', 'sw', 's', 'se'.

Example 5

```
photo1=PhotoImage(file='F:/python.png')
lab1=Label(r,image=photo1)
lab1.pack(side='left')
photo2=PhotoImage(file='F:/google.png')
lab2=Label(r,image=photo2)
lab2.pack(side='right',anchor='ne')
```



Packing Widgets – The `grid` Method

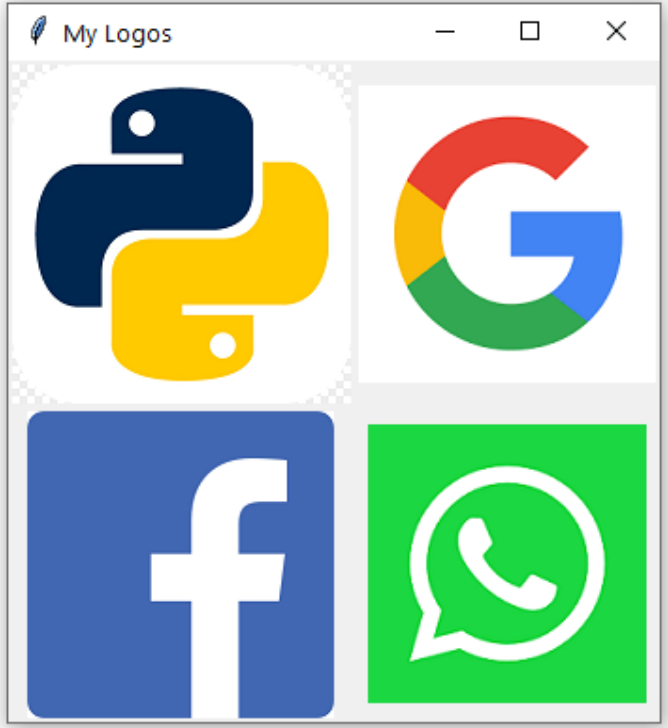
Some useful parameters:

`column`: specifies the column for the widget; default is column 0.

`row`: specifies the row for the widget; default is row 0.

Example 6

```
photo1=PhotoImage(file='F:/python.png')
lab1=Label(r,image=photo1)
lab1.grid(row=0,column=0)
photo2=PhotoImage(file='F:/google.png')
lab2=Label(r,image=photo2)
lab2.grid(row=0,column=1)
photo3=PhotoImage(file='F:/facebook.png')
lab3=Label(r,image=photo3)
lab3.grid(row=1,column=0)
photo4=PhotoImage(file='F:/whatsapp.png')
lab4=Label(r,image=photo4)
lab4.grid(row=1,column=1)
```



Packing Widgets – The `place` Method

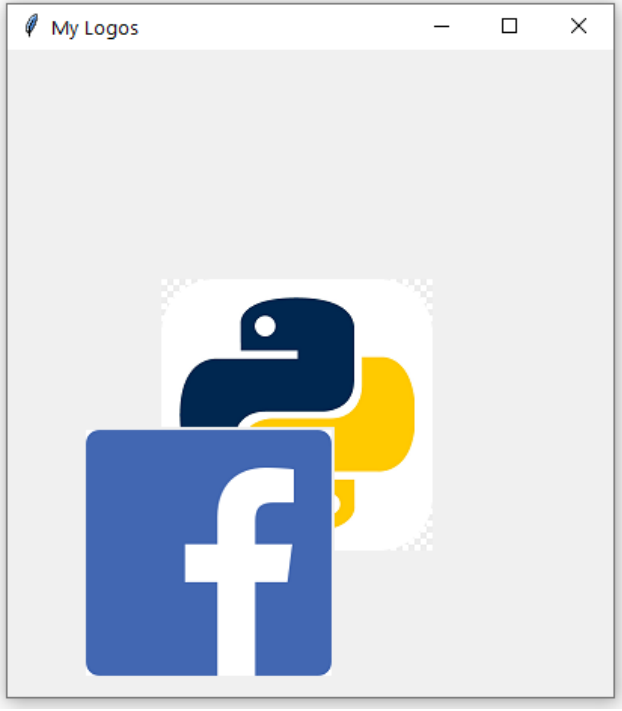
Some useful parameters:

`x`: specifies the location of the widget at position x of `MASTER`

`y`: specifies the location of the widget at position y of `MASTER`

Example 7

```
photo1=PhotoImage(file='F:/python.png')
lab1=Label(r,image=photo1)
lab1.place(x=100,y=150)
photo2=PhotoImage(file='F:/facebook.png')
lab2=Label(r,image=photo2)
lab2.place(x=50,y=250)
```

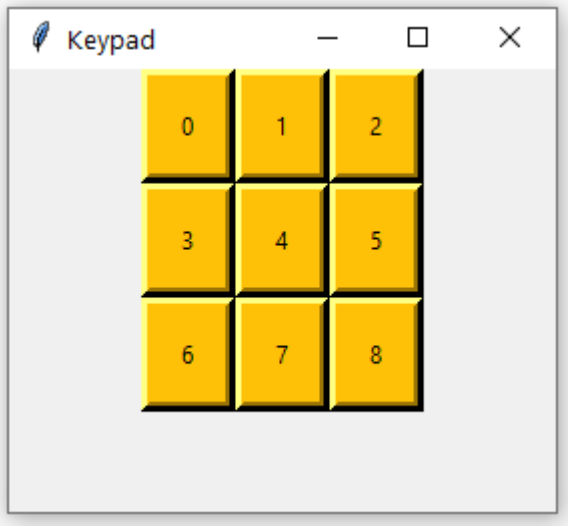


The Frame Widget

- A frame organizes a group of widgets.
  `f= Frame(MASTER,options)`

Example 8: Create the following display using the label widget.

```
from tkinter import Tk,Label,Frame
r=Tk()
r.title('Keypad')
value=0
f=Frame(r)
for i in range(3):
    for j in range(3):
        lab=Label(f,text=value,border=5,...)
        lab.grid(row=i,column=j)
        value+=1
f.pack()
r.mainloop()
```



Thank you!