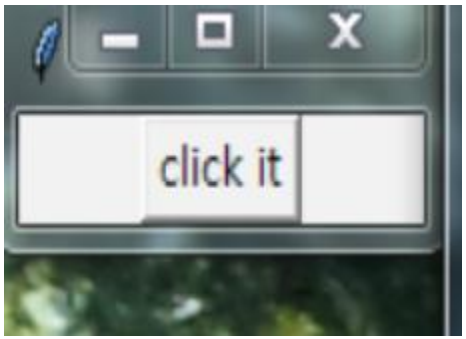
TKINTER:

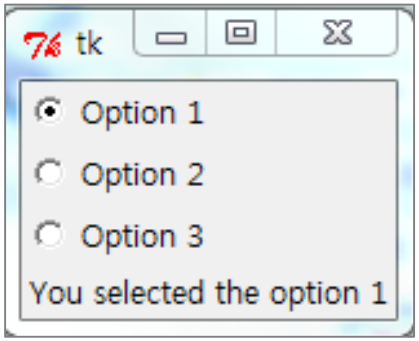
Python Module for Graphics & GUI Programming

What is Python Tkinter?

- Drawing과 GUI (graphical user interface)를 위한 Widget set (= Library)
- By Fredrik Lundh (2005)

- Widget?
  - 컴퓨터와 사용자가 상호 작용하는 GUI요소
  - GUI에서 어플리케이션을 만드는 블록을 구성하는 포괄적인 용어
  - 창, 캔버스, 텍스트 상자, 버튼……





Tkinter History

```
>> set  x  2
>> set  op  *
>> set y 3
>> set res [expr $x$op$y]
>> puts "2 * 3 is $res."
```

- **Tcl** (originally from **Tool Command Language**)
  - a scripting language intended to be embedded into applications
  - created by John Ousterhout in 1988
  - All operations are commands, including language structures
  - All data types can be manipulated as strings, including source code.

  - Tk toolkit (Tcl extension) is a GUI library (1991, by John Ousterhout)
    - Tk provides a number of (GUI) widgets commonly needed to develop desktop applications such as button, menu, canvas, text, frame, label, etc
    - Tk has been ported to run on most flavors of Linux, Mac OS, Unix, and Microsoft Windows

- **Tkinter** is a Python interface to the Tcl/Tk GUI toolkit
  - Python's *de facto* standard GUI
  - The name *Tkinter* comes from *Tk interface*
  - Tkinter was written by Fredrik Lundh (2005)

Tkinter Widgets [1/2]

모든 Widget은 Class 구조!

- Tkinter provides various control mechanisms (called widgets), such as buttons, labels and text boxes used in a GUI application
- There are currently 15 types of widgets in Tkinter

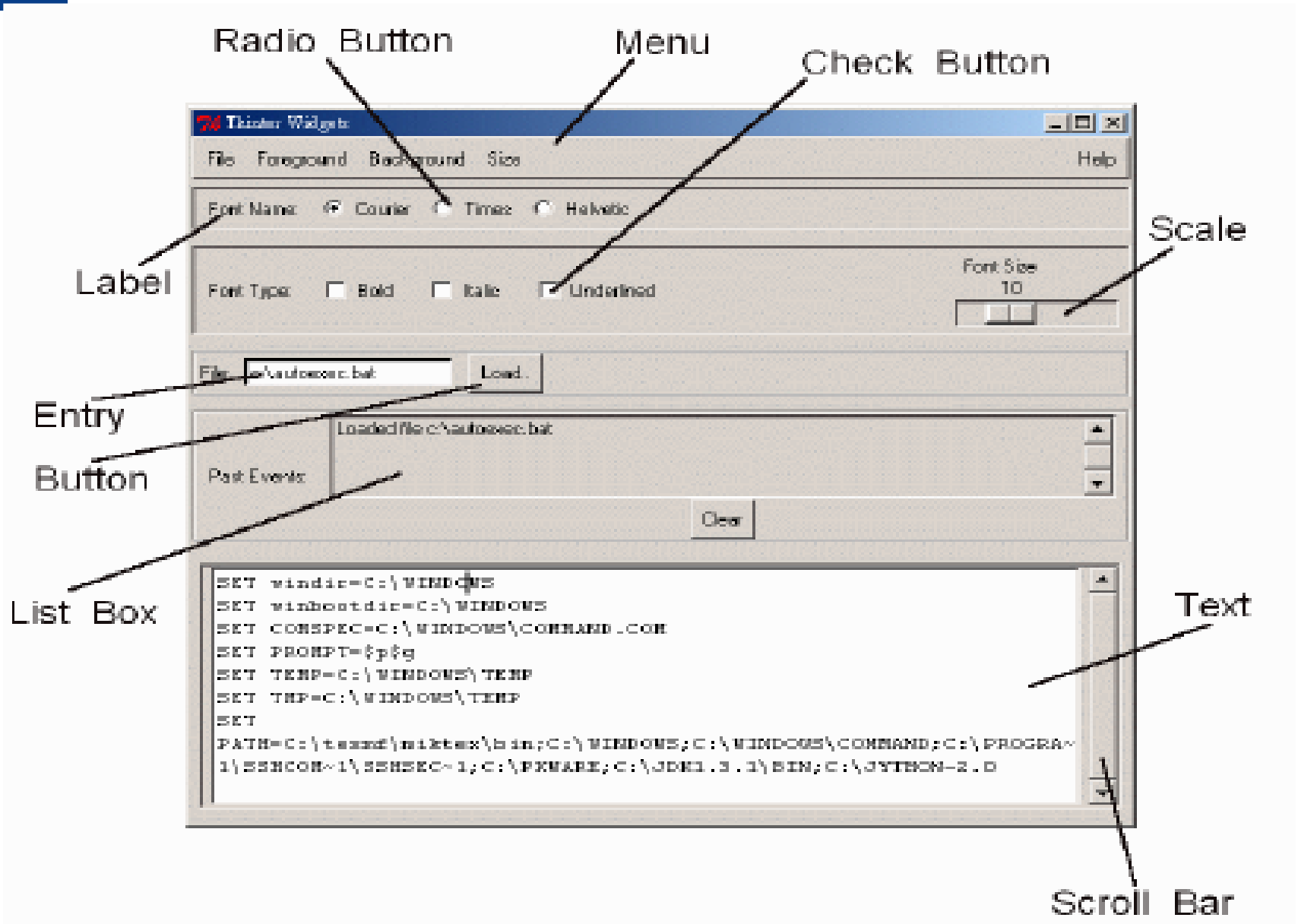
Operator	Description
Button	The Button widget is used to display buttons in your application.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in yo ur application.
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can se lect multiple options at a time.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.
Frame	The Frame widget is used as a container widget to organize other widgets.
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Listbox	The Listbox widget is used to provide a list of options to a user.
Menubutton	The Menubutton widget is used to display menus in your application.
Menu	The Menu widget is used to provide various commands to a user. These commands are contain ed inside Menubutton.

Tkinter Widgets [2/2]

모든 Widget은 Class 구조!

Operator	Description
Message	The Message widget is used to display multiline text fields for accepting values from a user.
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
Scale	The Scale widget is used to provide a slider widget.
Scrollbar	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
Text	The Text widget is used to display text in multiple lines.
Toplevel	The Toplevel widget is used to provide a separate window container.
Spinbox	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
PanedWindow	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox	This module is used to display message boxes in your applications.

Various Widgets for Graphical User Interface



Pros and Cons of TKinter

- Pros
  - Layered Implementation
  - High Portability
  - High Availability (at almost all OSs)
- Cons
  - Rather slow execution speed

==========

Computer Graphics
Human Computer Interface (HCI)
Graphical User Interface (GUI)

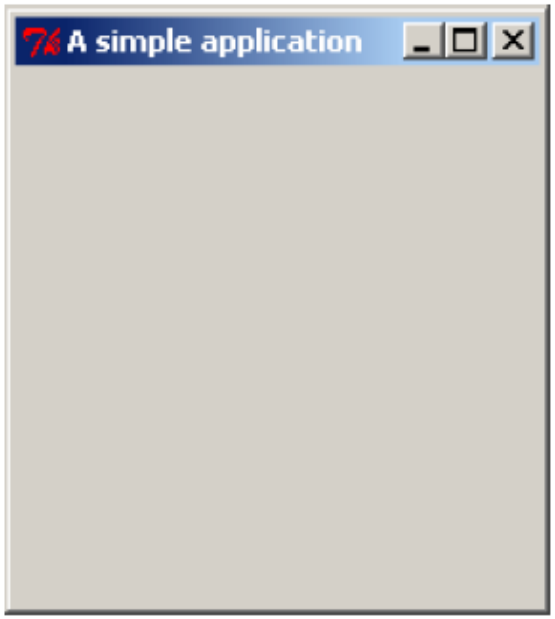
Window Creation Tkinter program

```
from tkinter import *
root = Tk()
root.title("A simple application")
root.mainloop()
```

Tk Class에서 instance생성

Tk Class의 instance function

Tk Class의 instance function

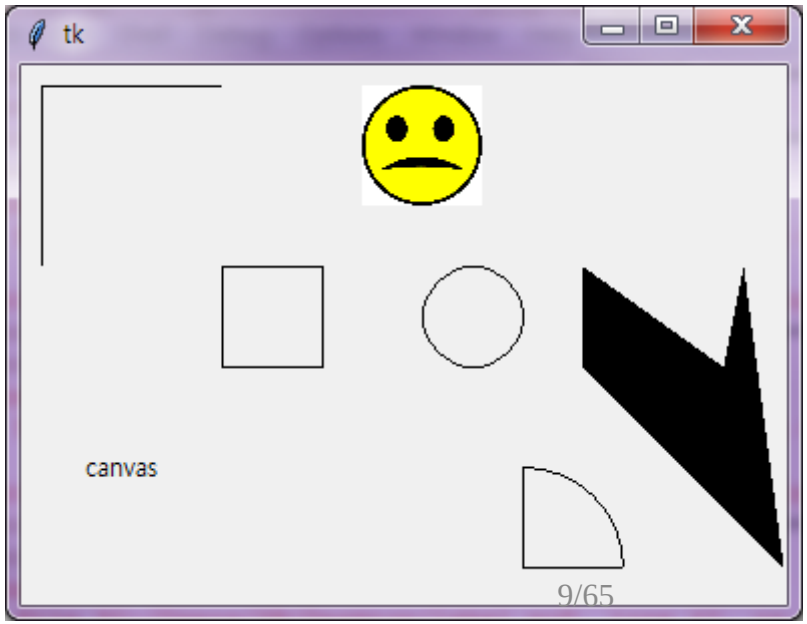


Canvas widget for Computer Graphics (Drawing)

- 그림이나 복잡한 layout을 표현하기 위한 직사각형 모양의 판
- Canvas상에는 도형이나 문자, widget 등을 넣을 수 있다

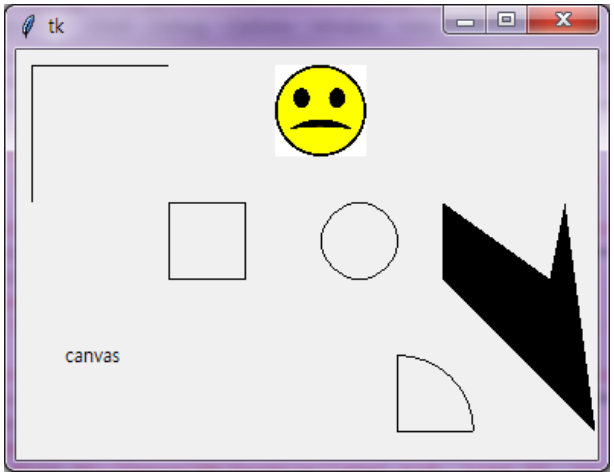
```
root = Tk()
canvas = Canvas(root)
```

→ Canvas Class에서 instance생성



Python Computer Graphics

- Canvas Widget: 그림이나 복잡한 layout을 표현하기 위한 직사각형 모양의 판
- Canvas상에는 도형이나 문자, widget 등을 넣을 수 있다



- Drawings on Canvas Widgets

```
root = Tk()
canvas = Canvas(root, width=width, height=height)
canvas.pack()
draw(canvas, width, height)
root.mainloop()
```

Canvas Widget Functions for Graphics

- Canvas widget에 다양한 개체들을 생성할 수 있는 함수들 제공

function	Description
create_line()	canvas상에 선 생성
create_text()	canvas상에 문자 생성
create_oval()	canvas상에 원 생성
create_rectangle()	canvas상에 직사각형 생성
create_polygon()	canvas상에 다각형 생성
create_arc()	canvas상에 호 생성
create_image()	canvas상에 이미지 생성
create_bitmap()	canvas상에 비트맵 생성
create_window()	canvas상에 window 생성

- http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html

Option Parameters for Tkinter Widgets and Their Functions

option	description
text	글자를 화면에 출력
image	그림을 화면에 출력
width	이미지나 글자의 넓이 지정, 지정하지 않으면 원래 데이터의 크기로 보인다
height	이미지나 글자의 높이 지정, 지정하지 않으면 원래 데이터의 크기로 보인다
relief	보더스타일(flat(default), Groove, Raised, Ridge, Sunken..)
borderwidth	보더의 넓이, 지정하지 않으면 '0'로 선이 그어지지 않는다
background (bg)	배경 색상
foreground (fg)	전경 색상
font	폰트이름 지정

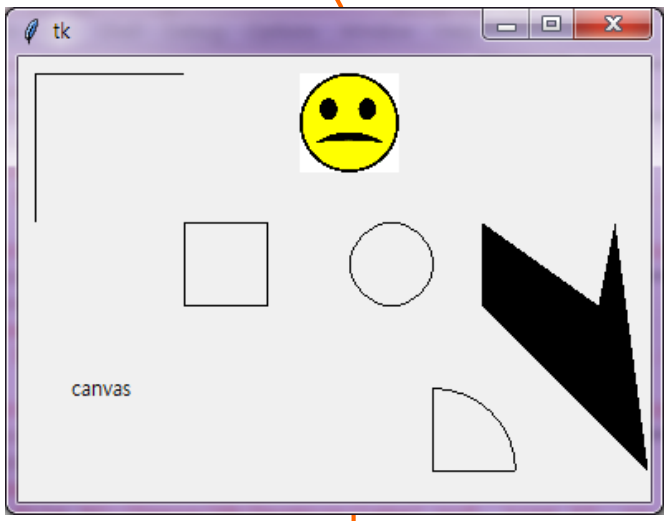
```
root = Tk()
canvas = Canvas(root, width=width, height=height)
```

Widget class & function 들에 대한 parameter들은 help()를 이용하거나 예제에서 참고!

위 Table외에도 많이 있다! Ex. fill, textvariable, relief, master, …

Canvas Drawing Code Example

```
from tkinter import *

root = Tk()
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)

canvas = Canvas(root)
canvas.grid(column=0, row=0, sticky=(N, W, E, S))
canvas.create_line((10,10,10,100))
canvas.create_line((10,10,100,10))
canvas.create_rectangle(100,100,150,150)
canvas.create_oval(200,100,250,150)
canvas.create_arc(200,200,300,300)
canvas.create_polygon(280,150,280,100,350,150,360,100,380,250)
canvas.create_text(50,200, text="canvas")
image = PhotoImage(file="sad.gif")
canvas.create_image(200,40,image=image)

root.mainloop()
```

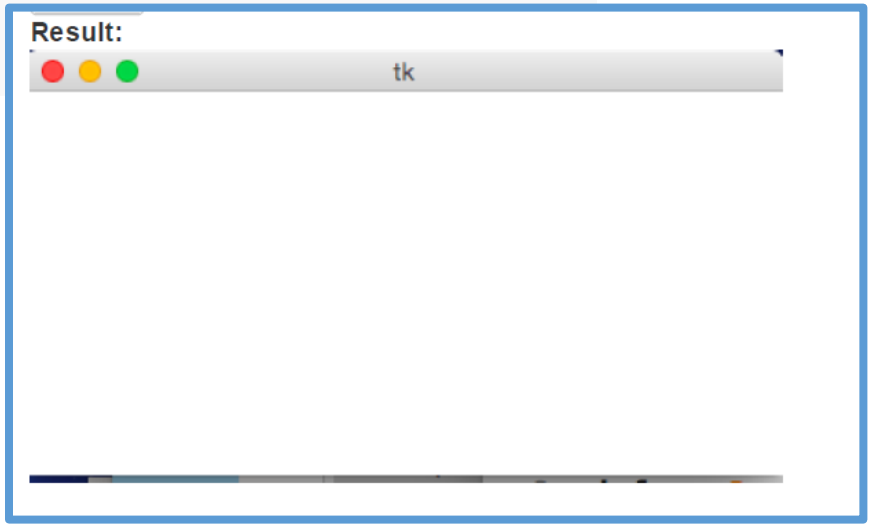


Canvas class의 function들에 대한 parameter들은 help()를 이용하거나 예제에서 참고!

Practicing Canvas Drawing [1/9]

1. **Create an Empty Canvas**

```
from tkinter import *

def draw(canvas, width, height):
    pass # replace with your drawing code!

def runDrawing(width=300, height=300):
    root = Tk()
    canvas = Canvas(root, width=width, height=height)
    canvas.pack()
    draw(canvas, width, height)
    root.mainloop()
    print("bye!")

runDrawing(400, 200)
```



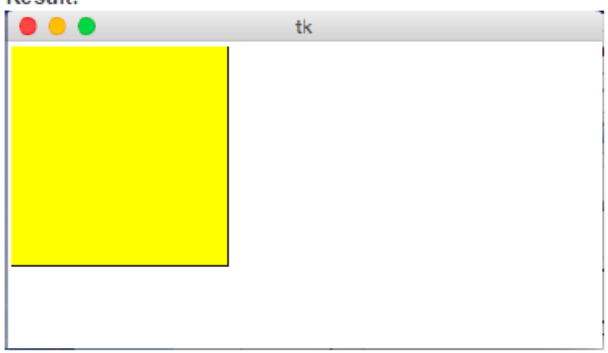
Practicing Canvas Drawing [2/9]

```
def runDrawing(width=300, height=300):
    root = Tk()
    canvas = Canvas(root, width=width, height=height)
    canvas.pack()
    draw(canvas, width, height)
    root.mainloop()
    print("bye!")

runDrawing(400, 200)
```

2. Draw a Rectangle

```
def draw(canvas, width, height):
    canvas.create_rectangle(0,0,150,150, fill="yellow")
```

Result:

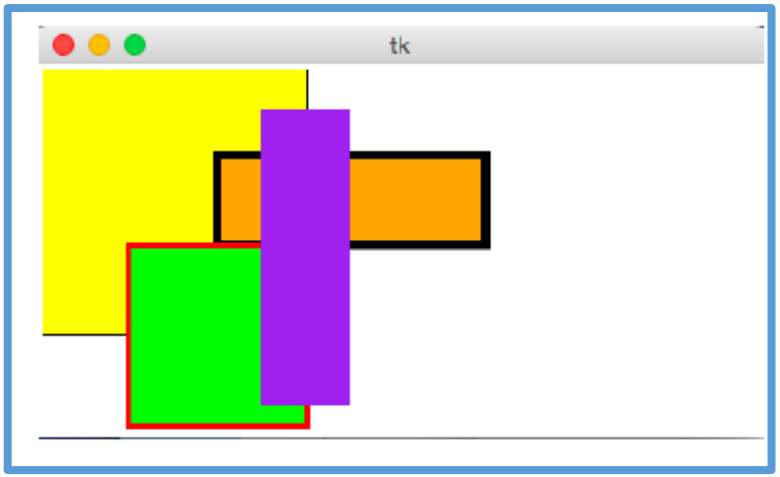


Practicing Canvas Drawing [3/9]

```
def runDrawing(width=300, height=300):
    root = Tk()
    canvas = Canvas(root, width=width, height=height)
    canvas.pack()
    draw(canvas, width, height)
    root.mainloop()
    print("bye!")

runDrawing(400, 200)
```

3. Draw Multiple Rectangles

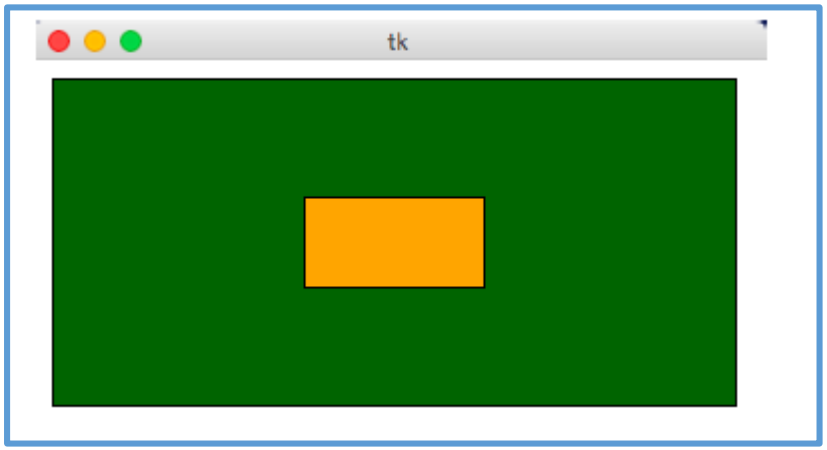
```
def draw(canvas, width, height):
    canvas.create_rectangle(  0,   0, 150, 150, fill="yellow")
    canvas.create_rectangle(100,  50, 250, 100, fill="orange", width=5)
    canvas.create_rectangle( 50, 100, 150, 200, fill="green",
                                                outline="red", width=3)
    canvas.create_rectangle(125,  25, 175, 190, fill="purple", width=0)
```



Practicing Canvas Drawing [4/9]

4. Draw Centered Rectangles

```
def draw(canvas, width, height):
    margin = 10
    # Approach #1: Add margin to top/left, subtract margin from bottom/right:
    canvas.create_rectangle(margin, margin, width-margin, height-margin,
                            fill="darkGreen")
    # Approach #2: add/subtract width/height from center (cx, cy)
    (cx, cy) = (width/2, height/2)
    (rectWidth, rectHeight) = (width/4, height/4)
    canvas.create_rectangle(cx - rectWidth/2, cy - rectHeight/2,
                            cx + rectWidth/2, cy + rectHeight/2,
                            fill="orange")
```

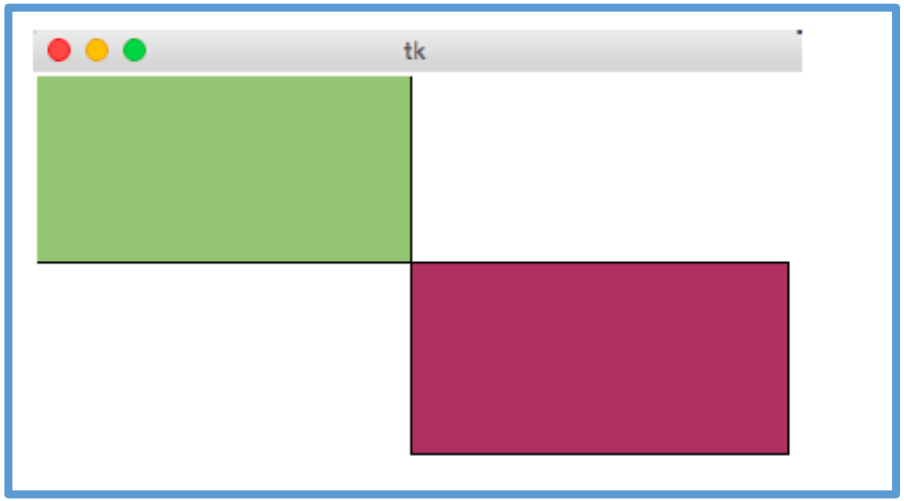


Practicing Canvas Drawing [5/9]

5. Draw Custom Colors

```
def rgbString(red, green, blue):
    return "#%02x%02x%02x" % (red, green, blue)

def draw(canvas, width, height):
    pistachio = rgbString(147, 197, 114)
    maroon = rgbString(176, 48, 96)
    canvas.create_rectangle(0, 0, width/2, height/2, fill=pistachio)
    canvas.create_rectangle(width/2, height/2, width, height, fill=maroon)
```

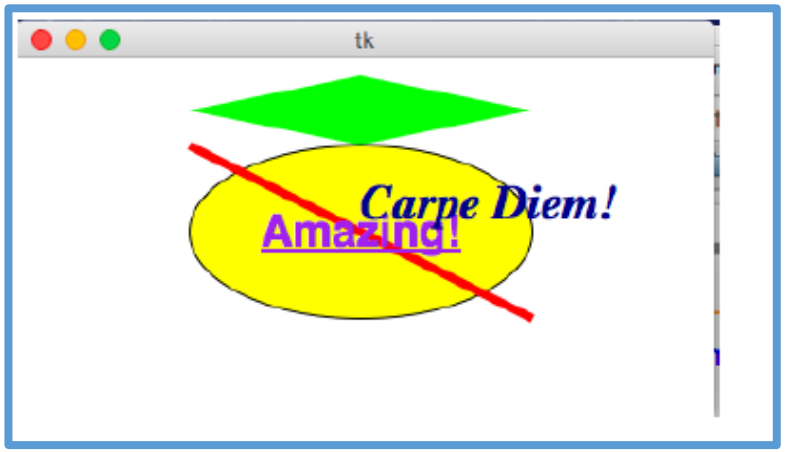
Color를 생성하는 function



Practicing Canvas Drawing [6/9]

6. Draw Other Shapes and Text

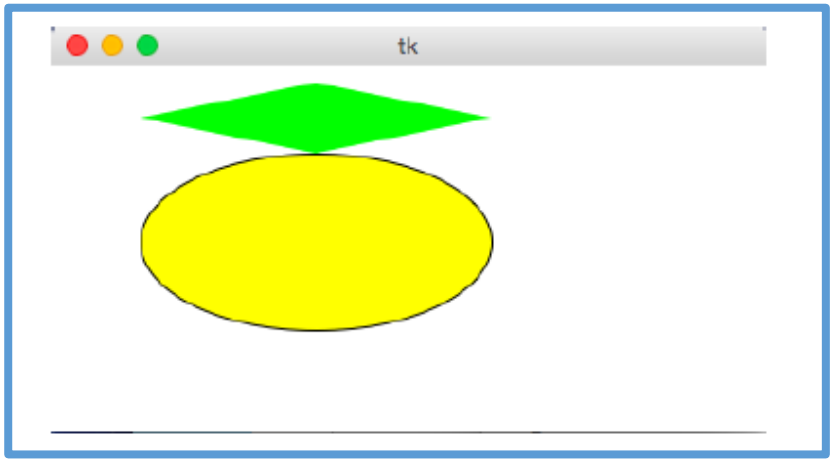
```
def draw(canvas, width, height):
    canvas.create_oval(100, 50, 300, 150, fill="yellow")
    canvas.create_polygon(100,30,200,50,300,30,200,10, fill="green")
    canvas.create_line(100, 50, 300, 150, fill="red", width=5)
    canvas.create_text(200, 100, text="Amazing!",
                       fill="purple", font="Helvetica 26 bold underline")
    canvas.create_text(200, 100, text="Carpe Diem!", anchor=SW,
                       fill="darkBlue", font="Times 28 bold italic")
```



Practicing Canvas Drawing [7/9]

7. Tuples and List-of-Tuples as Parameters

```
def draw(canvas, width, height):
    canvas.create_oval((50, 50), (250, 150), fill="yellow")
    canvas.create_polygon([(50,30),(150,50),(250,30),(150,10)], fill="green")
```

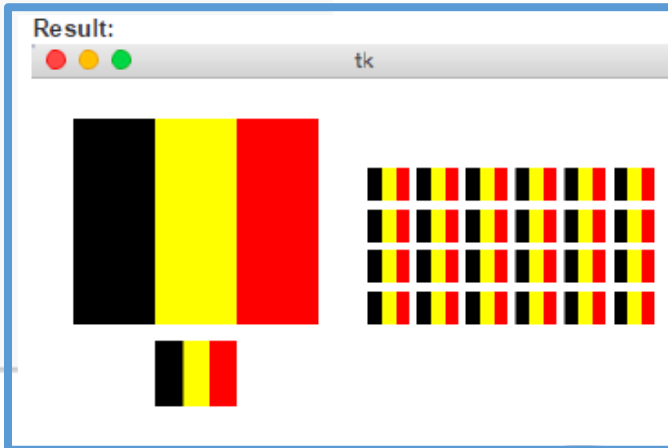


Practicing Canvas Drawing [8/9]

8. Graphics Helper Functions

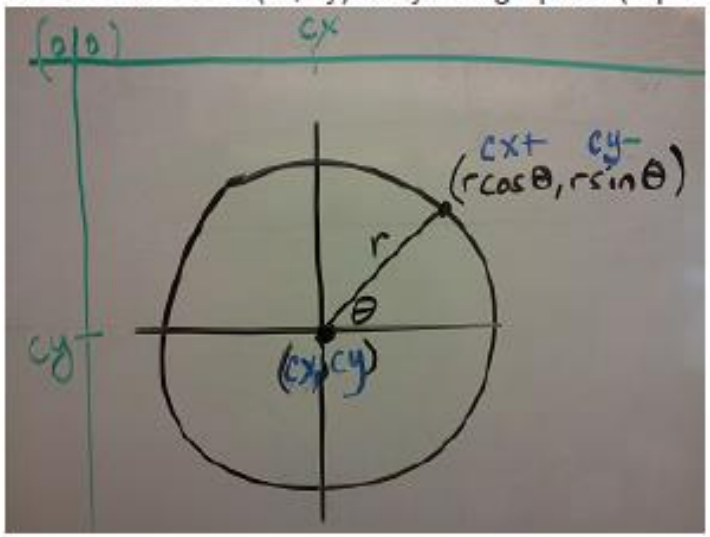
```
def drawBelgianFlag(canvas, x0, y0, x1, y1):
    # draw a Belgian flag in the area bounded by (x0,y0) in
    # the top-left and (x1,y1) in the bottom-right
    width = (x1 - x0)
    canvas.create_rectangle(x0, y0, x0+width/3, y1, fill="black", width=0)
    canvas.create_rectangle(x0+width/3, y0, x0+width*2/3, y1,
                            fill="yellow", width=0)
    canvas.create_rectangle(x0+width*2/3, y0, x1, y1, fill="red", width=0)

def draw(canvas, width, height):
    # Draw a large Belgian flag
    drawBelgianFlag(canvas, 25, 25, 175, 150)

    # And draw a smaller one below it
    drawBelgianFlag(canvas, 75, 160, 125, 200)

    # Now let's have some fun and draw a whole grid of Belgian flags!
    flagWidth = 30
    flagHeight = 25
    margin = 5
    for row in range(4):
        for col in range(6):
            left = 200 + col * flagWidth + margin
            top = 50 + row * flagHeight + margin
            right = left + flagWidth - margin
            bottom = top + flagHeight - margin
            drawBelgianFlag(canvas, left, top, right, bottom)
```

Result:

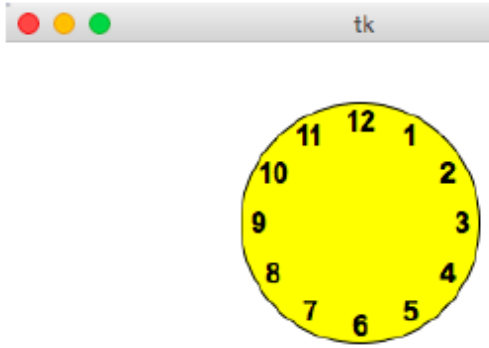


Practicing Canvas Drawing [9/9]

- Circle centered at (cx, cy) in Python graphics ("up is down!")



Result:




```
import math

def draw(canvas, width, height):
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
    canvas.create_oval(cx-r, cy-r, cx+r, cy+r, fill="yellow")
    r *= 0.85 # make smaller so time labels lie inside clock face
    for hour in range(12):
        hourAngle = math.pi/2 - (2*math.pi)*(hour/12)
        hourX = cx + r * math.cos(hourAngle)
        hourY = cy - r * math.sin(hourAngle)
        label = str(hour if (hour > 0) else 12)
        canvas.create_text(hourX, hourY, text=label, font="Arial 16 bold")
```

The Tkinter Canvas Widget

The **Canvas** widget provides structured graphics facilities for Tkinter. This is a highly versatile widget which can be used to draw graphs and plots, create graphics editors, and implement various kinds of custom widgets.

When to use the Canvas Widget

The canvas is a general purpose widget, which is typically used to display and edit graphs and other drawings.

Another common use for this widget is to implement various kinds of custom widgets. For example, you can use a canvas as a completion bar, by drawing and updating a rectangle on the canvas.

Patterns

To draw things in the canvas, use the **create** methods to add new items.

```
from Tkinter import *

master = Tk()

w = Canvas(master, width=200, height=100)
w.pack()

w.create_line(0, 0, 200, 100)
w.create_line(0, 100, 200, 0, fill="red", dash=(4, 4))

w.create_rectangle(50, 25, 150, 75, fill="blue")
```

Note that items added to the canvas are kept until you remove them. If you want to change the drawing, you can either use methods like **coords**, **itemconfig**, and **move** to modify the items, or use **delete** to remove them.

```
i = w.create_line(xy, fill="red")

w.coords(i, new_xy) # change coordinates
w.itemconfig(i, fill="blue") # change color

w.delete(i) # remove

w.delete(ALL) # remove all items
```

Concepts

To display things on the canvas, you create one or more *canvas items*, which are placed in a stack. By default, new items are drawn on top of items already on the canvas.

Tkinter provides lots of methods allowing you to manipulate the items in various ways. Among other things, you can attach (*bind*) event callbacks to individual canvas items.

The **Canvas** widget supports the following standard items:

- **arc** (arc, chord, or pieslice)
- **bitmap** (built-in or read from XBM file)
- **image** (a BitmapImage or PhotoImage instance)
- **line**
- **oval** (a circle or an ellipse)
- **polygon**
- **rectangle**
- **text**
- **window**

Chords, pieslices, ovals, polygons, and rectangles consist of both an outline and an interior area, either of which can be made transparent (and if you insist, you can make both transparent).

Window items are used to place other Tkinter widgets on top of the canvas; for these items, the Canvas widget simply acts like a geometry manager.

You can also write your own item types in C or C++ and plug them into Tkinter via Python extension modules.

Coordinate Systems

The **Canvas** widget uses two coordinate systems; the window coordinate system (with (0, 0) in the upper left corner), and a canvas coordinate system which specify where the items are drawn. By scrolling the canvas, you can specify which part of the canvas coordinate system to show in the window.

The **scrollregion** option is used to limit scrolling operations for the canvas. To set this, you can usually use something like:

The **Canvas** widget allows you to identify items in several ways. Everywhere a method expects an item specifier, you can use one of the following:

- item handles (integers)
- tags
- **ALL**
- **CURRENT**

Item handles are integer values used to identify a specific item on the canvas. Tkinter automatically assigns a new handle to each new item created on the canvas. Item handles can be passed to the various canvas methods either as integers or as strings.

Tags are symbolic names attached to items. Tags are ordinary strings, and they can contain anything except whitespace (as long as they don't look like item handles).

An item can have zero or more tags associated with it, and the same tag can be used for more than one item. However, unlike the **Text** widget, the **Canvas** widget doesn't allow you to create bindings or otherwise configure tags for which there are no existing items. Tags are owned by the items, not the widget itself. All such operations are ignored.

You can either specify the tags via an option when you create the item, set them via the **itemconfig** method, or add them using the **addtag_withtag** method. The **tags** option takes either a single tag string, or a tuple of strings.

```
item = canvas.create_line(0, 0, 100, 100, tags="uno")
canvas.itemconfig(item, tags=("one", "two"))
canvas.addtag_withtag("three", "one")
```

To get all tags associated with a specific item, use **gettags**. To get the handles for all items having a given tag, use **find_withtag**.

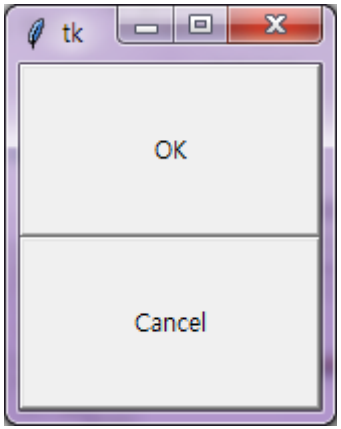
```
>>> print canvas.gettags(item)
('one', 'two', 'three')
>>> print canvas.find_withtag("one")
(1,)
```

The **Canvas** widget also provides two predefined tags:

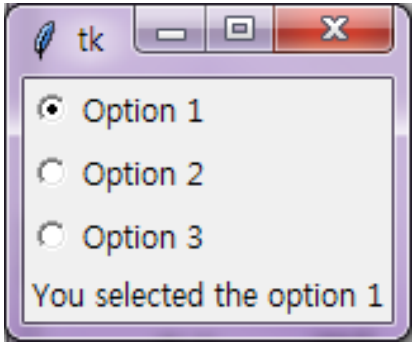
ALL (or the string "all") matches all items on the canvas.

CURRENT (or "current") matches the item under the mouse pointer, if any. This can be used inside mouse event bindings to refer to the item that triggered the callback.

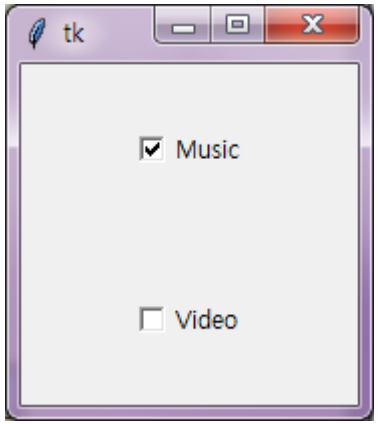
Python TkInter Graphical User Interface (GUI)



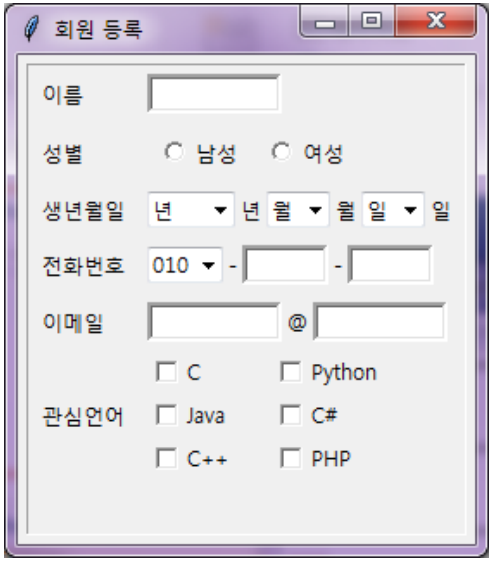
button

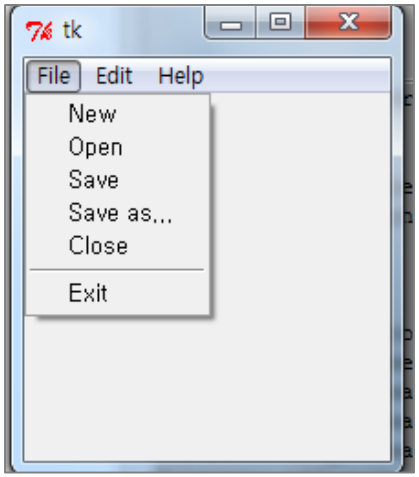


radio button



check button



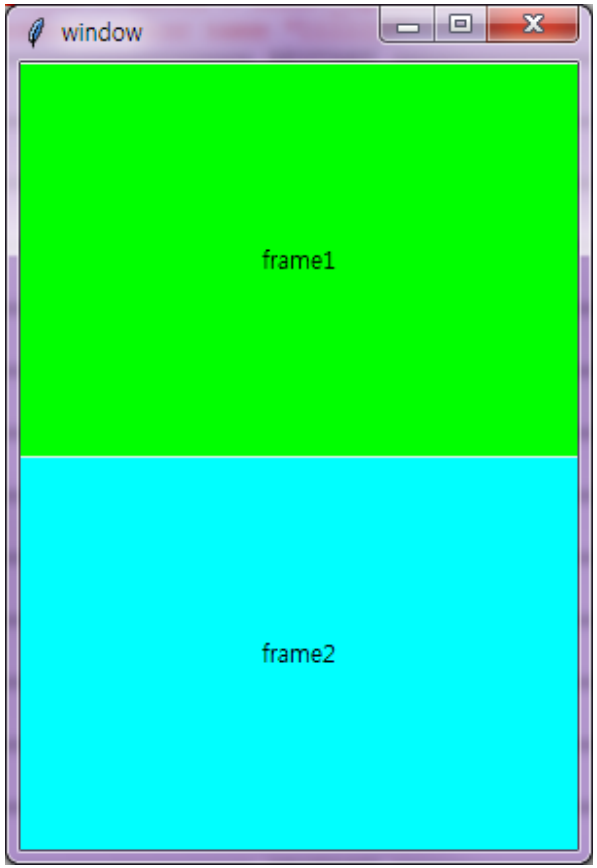


menu

Basic of Tkinter: Window & Frame

- Windows란?
  - 일반적으로 결과를 보여주는 직사각형 스크린

```
root = tk()     // root window is created
```

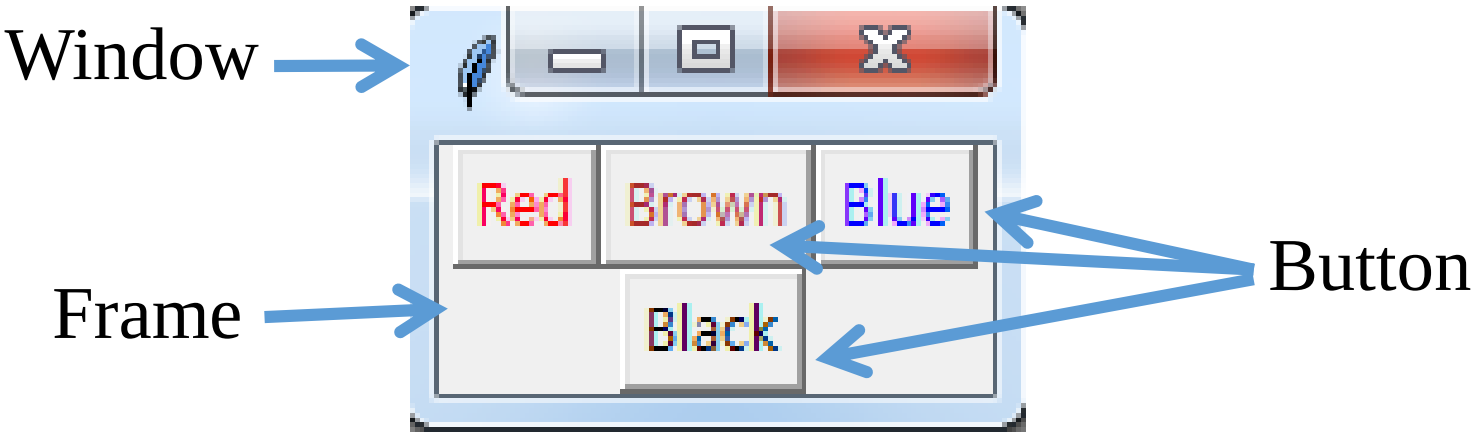


- Frame widget이란?
  - 복잡한 레이아웃을 조직하는 기본 단위
  - 다른 Widget을 포함하는 직사각형 영역
  - Window는 Frame 포함이 가능하나, Frame은 Window를 포함할 수 없다

```
frame1 = Frame(root)
frame2 = Frame(root)
```

Window, Frame, and Buttons…..

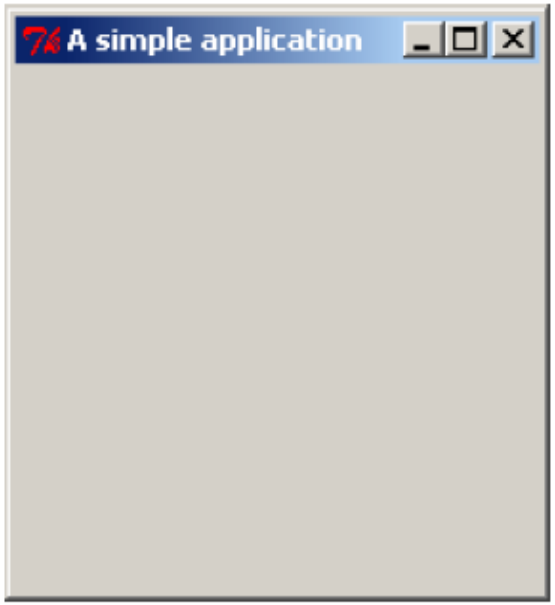
- Parent-Child 관계
  - 어떤 Widget이 만들어질 때, child-parent 관계가 발생
  - Window를 여러 개의 frame들로 나눌 수 있음
  - Button이 frame 안에 있을 경우: frame이 parent, button이 child



- Parent-Child관계: Window vs Red button, Window vs Brown button
  Window vs Blue button, Window vs Black button

Window Creation with Title

```
from tkinter import *
root = Tk()
root.title("A simple application")
root.mainloop()
```



Window Creation with Label

```
# first Tkinter window

from tkinter import *

root = Tk() # Create the root (base) window where all widgets go
```

Create the parent window. All applications have a "root" window. This is the parent of all other widgets, you should create only one!

```
w = Label(root, text = "Hello, world!") # Create a label with words
```

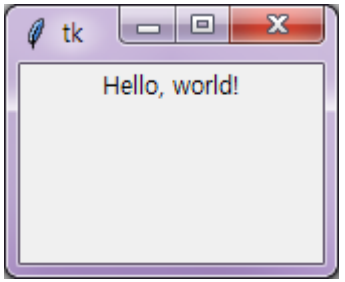
A label is a widget that holds text. This one has a parent of "root". That is the mandatory first argument to the Label's constructor.

```
w.pack() # Put the label into the window
```

Tell the label to place itself into the root window and display. Without calling pack the Label will NOT be displayed!!!

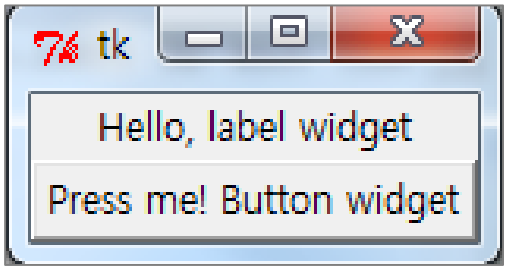
```
root.mainloop() # Start the event loop
```

Windows go into an "**event loop**" where they wait for things to happen (buttons pushed, text entered, mouse clicks, etc...) or Windowing operations to be needed (redraw, etc..). You must tell the root window to enter its event loop or the window won't be displayed!



1 Window, 1 Label, and 1 Button example

```
# label and button

from tkinter import *
root = Tk()

mylabel = Label(root, text="Hello, label widget")
mylabel.pack()

mybutton = Button(root, text="Press me! Button widget")
mybutton.pack()

root.mainloop()
```



The *pack()* Method - organizes widgets in blocks before placing them in the parent widget.

Tk also provides 3 geometry managers:

- place - which positions widgets at absolute locations
- grid - which arranges widgets in a grid
- pack - which packs widgets into a cavity

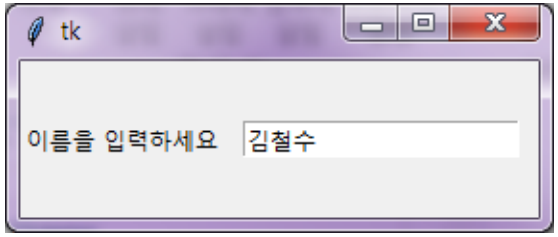
1 Window, 2 Frames, and 4 Buttons Example

```
from tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()

bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text="Red", fg="red")
redbutton.pack( side = LEFT)

greenbutton = Button(frame, text="Brown", fg="brown")
greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text="Blue", fg="blue")
bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text="Black", fg="black")
blackbutton.pack( side = BOTTOM)

root.mainloop()
```

- fg ➔ foreground
- bottom frame에는 button이 한개만 있어서 side=LEFT 로 해도 중앙에 위치한다

Tkinter Widgets for User Input [1/3]

- Entry widget
  - 가장 기본적인 text box.
  - 보통 User가 한 줄의 text를 입력.
  - 다양한 서식 설정을 허용하지 않음.



entry

```
from tkinter import *

root=Tk()

w = Label(root, text = "이름을 입력하세요",
height = 7)
w.grid(row=1, column=1)

tBox = Entry(root)
tBox.grid(row=1, column=2, padx = 10)
```

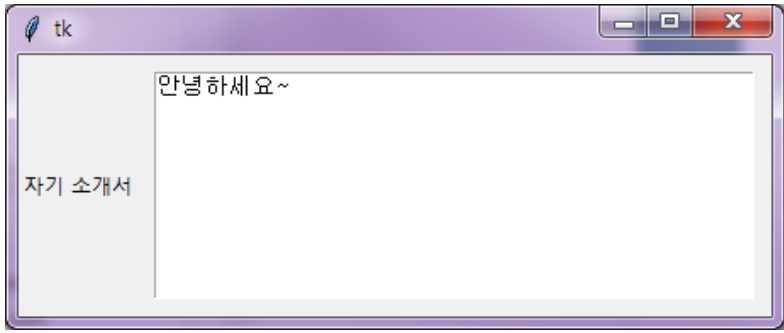
```
from tkinter import *

root=Tk()

def createTextBox(parent):
  tBox = Entry(parent)
  tBox.grid(row=1, column=2, padx = 10)

createTextBox(root)

w = Label(root, text = "이름을 입력하세요",
height = 7)
w.grid(row=1, column=1)
```

Row를 0 로 하던 2로 하던 화면모습은 변화가 없다

Tkinter Widgets for User Input [2/3]

Text widget

User에게 여러 줄의 text를 입력.
입력된 그 text를 저장.
다양한 서식 설정 옵션 제공. (style, attributes)

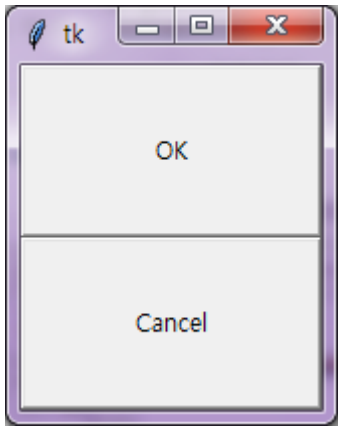


text

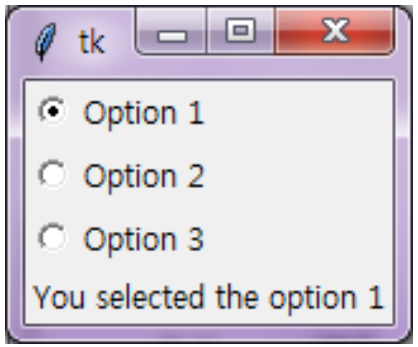
```
from tkinter import *

root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, "Just a text Widget \n in two lines \n ")
mainloop()
```

Tkinter Widgets for User Input [3/3]

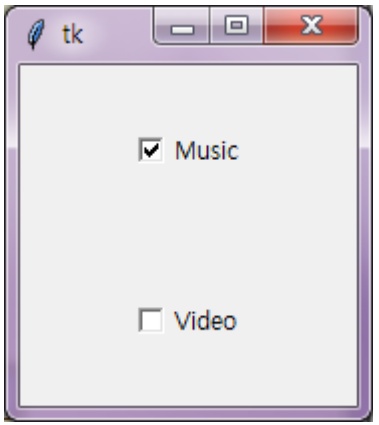
- Button widget
  - User가 GUI에 명령을 수행하도록 하는 기본방법.
  - 눌러서 뭔가 다른 기능을 수행하게 하는 역할
- Radio Button widget
  - User가 선택 가능한 리스트로부터 하나의 옵션을 선택 가능.
- Check Button widget
  - User가 선택 가능한 리스트로부터 여러 개의 옵션을 선택 가능.



button



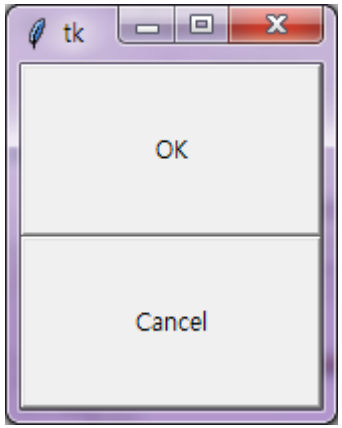
radio button



check button

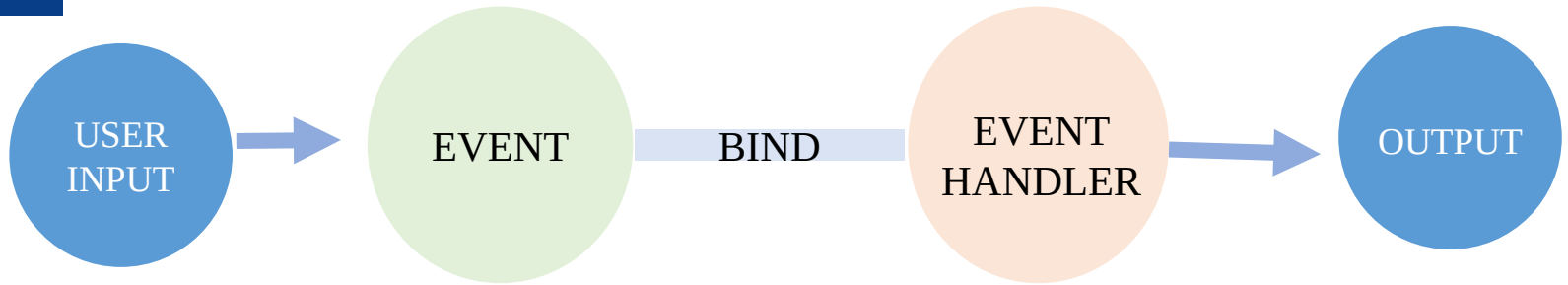
Window with 2 Buttons Example

```
from tkinter import *

root=Tk()

b1 = Button(root, text = "OK", height = 5, width = 20)
b1.pack()
b2 = Button(root, text = "Cancel", height=5, width = 20)
b2.pack()
```

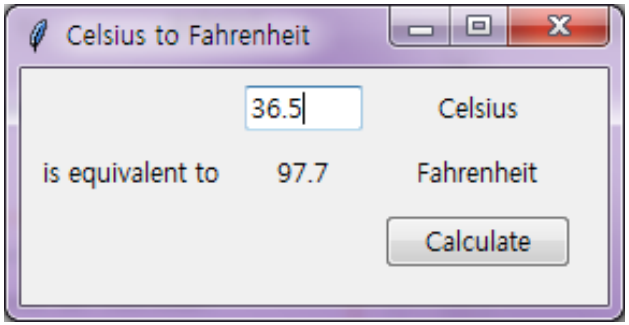


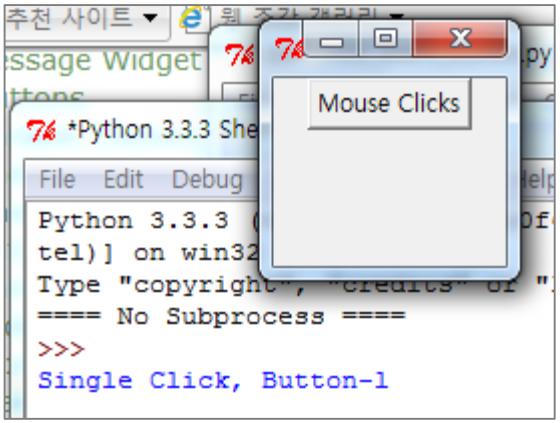
button

User Interaction: Event and Binding



- Events: 어플리케이션에 발생하는 user interaction
  - Mouse click, Button click, Mouse dragging, Key-board typing
- Event handler: Event가 발생시 호출되는 응용 프로그램
- Binding
  - Widget에 event가 발생할 때, 어플리케이션이 event handler을 호출하여 준비하도록 연결해주는 것

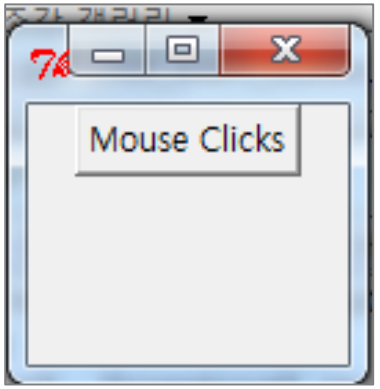


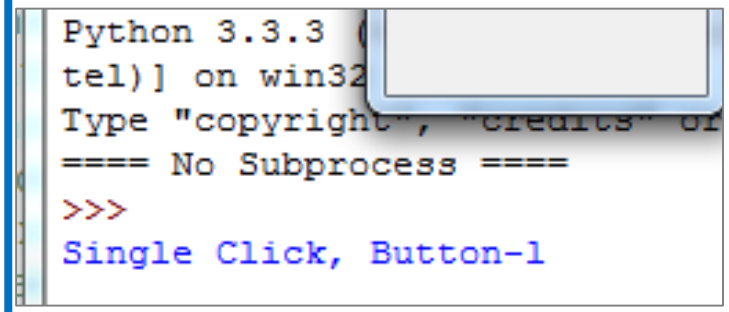


Event Binding Example

```
from tkinter import *
root = Tk()

def hello(event) :
    print("Single Click, Button-l")

def quit(event) :
    print("Double Click, so let's stop")
    import sys;
    sys.exit()

wbutton = Button(root, text='Mouse Clicks')
wbutton.pack()

wbutton.bind('<Button-1>', hello)
wbutton.bind('<Double-Button-1>', quit)

root.mainloop()
```

Event Handler





hello() event handler에게 '<Button-1>' event를 넘김.

quit() event handler에게 '<Double-Button-1>' event를 넘김

List of Event Patterns

Event patterns	Description
<Button-1>	첫 번째 mouse button을 누른다.
<KeyPress-H>	H key를 누른다.
<Control-Shift-KeyPress-H>	control-shift-H를 누른다.
<B1-Motion>	왼쪽버튼을 누른채로 마우스를 드래그한다.
<Double-Button-1>	더블클릭
<Enter>	마우스 포인터가 widget안에 있다.
<Leave>	마우스 포인터를 widget 밖으로 둔다.
<Return>	Enter key를 누른다.
<Configure>	크기를 바꾼다.

Note: '<Button-1>'

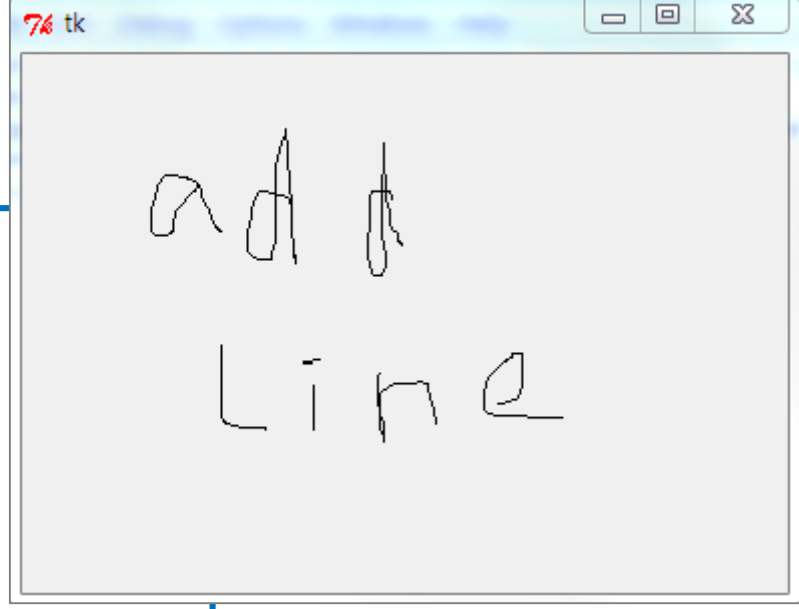
Event Object의 Attribute 리스트

Attributes	Description
widget	Event가 발생하는 widget(이름이 아닌 reference로)
x, y	현재 마우스 위치(픽셀)
x-root, y-root	스크린의 좌측상단 코너를 기준으로 한 마우스 포인터 위치(픽셀)
char	The character code (keyboard events only), as a string
keysym	The key symbol (keyboard events only)
keycode	The key code (keyboard events only)
num	The button number (mouse button events only)
width, height	Widget의 새 크기, in pixels (Configure events only)
type	The event type

- event.x, event.y, event.keysym, event.keycode, event.type…

Mouse Event Example

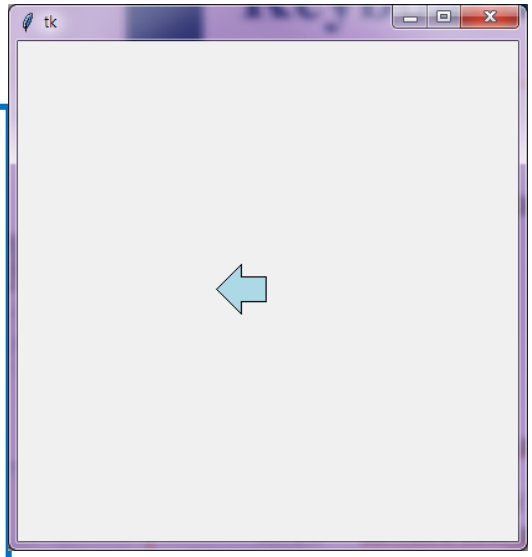
```
from tkinter import *

lastx, lasty = 0, 0

def xy(event) :
    global lastx, lasty
    lastx, lasty = event.x, event.y

def addLine(event) :
    global lastx, lasty
    canvas.create_line((lastx, lasty, event.x, event.y))
    lastx, lasty = event.x, event.y

root = Tk()
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)

canvas = Canvas(root)
canvas.grid(column=0, row=0, sticky=(N, W, E, S))
canvas.bind("<Button-1>", xy)
canvas.bind("<B1-Motion>", addLine)

root.mainloop()
```



Mouse를 처음 click하는 위치가 event.x, event.y 로 잡히고, 그것을 lastx, lasty에 저장을 하고, Mouse dragging하면서 mouse의 pixel위치가 바뀔때마다 (lastx, lasty)에서 (event.x, event.y)에 작은 line을 그리고, (lastx, lasty)값을 update한다

Keyboard Event example

```
from tkinter import *
tk = Tk()

canvas = Canvas(tk, width=400, height=400)
canvas.pack()
Arrow = 0

def printarrow(event) :
    global Arrow
    canvas.delete(Arrow)
    if   event.keysym=='Up' :
        Arrow = canvas.create_polygon( 210, 200, 190, 200, 190, 180, 180,
180, 200, 160, 220, 180, 210, 180, fill='yellow', outline='black')
    elif event.keysym=='Down' :
        Arrow = canvas.create_polygon( 210, 200, 190, 200, 190, 220, 180,
220, 200, 240, 220, 220, 210, 220, fill='pink', outline='black')
    elif event.keysym=='Left' :
        Arrow = canvas.create_polygon( 200, 190, 200, 210, 180, 210, 180,
220, 160, 200, 180, 180, 180, 190, fill='lightblue', outline='black')
    else :  # event.keysym == 'Right'
        Arrow = canvas.create_polygon( 200, 190, 200, 210, 220, 210, 220,
220, 240, 200, 220, 180, 220, 190, fill='white', outline='black')

canvas.bind_all('<KeyPress-Up>',      printarrow)
canvas.bind_all('<KeyPress-Down>',  printarrow)
canvas.bind_all('<KeyPress-Left>',    printarrow)
canvas.bind_all('<KeyPress-Right>',  printarrow)
```



bind_all: mouse위치와 상관없이 무조건 event가 발생하면 실행

“command” paramer inside Button Widget

- 누르면 시간을 나타내는 Button의 생성

```
from tkinter import Tk, Button
from time import strftime, localtime

def clicked():      # not a event handler
    time = strftime("Day: %d %b %Y ₩n Time: %H : %M : %S %p ₩n", localtime())
    print(time)

root =Tk()

#create button
but = Button( root, text="click it", command=clicked)
but.pack()
root.mainloop()
```

참고사항

time.localtime()

➔ (tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec, tm_wday, tm_yday, tm_isdst)

`time.`**`strftime`**(*format*[, *t*])

Convert a tuple or `struct_time` representing a time as returned by `gmtime()` or `localtime()` to a string as specified by the *format* argument. If *t* is not provided, the current time as returned by `localtime()` is used. *format* must be a string. `ValueError` is raised if any field in *t* is outside of the allowed range.

0 is a legal argument for any position in the time tuple; if it is normally illegal the value is forced to a correct one.

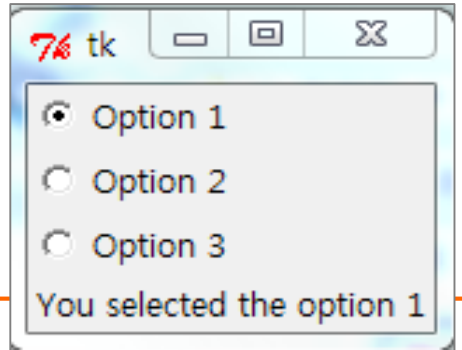
The following directives can be embedded in the *format* string. They are shown without the optional field width and precision specification, and are replaced by the indicated characters in the `strftime()` result:

Directive	Meaning	Notes
`%a`	Locale's abbreviated weekday name.	
`%A`	Locale's full weekday name.	
`%b`	Locale's abbreviated month name.	
`%B`	Locale's full month name.	
`%c`	Locale's appropriate date and time representation.	
`%d`	Day of the month as a decimal number [01,31].	
`%H`	Hour (24-hour clock) as a decimal number [00,23].	
`%I`	Hour (12-hour clock) as a decimal number [01,12].	
`%j`	Day of the year as a decimal number [001,366].	
`%m`	Month as a decimal number [01,12].	
`%M`	Minute as a decimal number [00,59].	
`%p`	Locale's equivalent of either AM or PM.	(1)
`%S`	Second as a decimal number [00,61].	(2)
`%y`	Year without century as a decimal number [00,99].	
`%Y`	Year with century as a decimal number.	

Locale: 로우켈: 장소, 현장

“IntVar” class and “command” in Radiobutton Widget

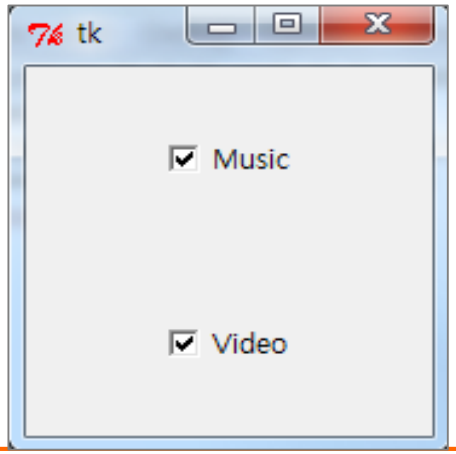
```
from tkinter import *

def sel():
   selection = "You selected the option " + str(var.get())
   label.config(text = selection)

root = Tk()
var = IntVar()

R1 = Radiobutton(root, text="Option 1” , variable=var, value=1, command=sel)
R1.pack( anchor = W )

R2 = Radiobutton(root, text="Option 2” , variable=var, value=2, command=sel)
R2.pack( anchor = W )

R3 = Radiobutton(root, text="Option 3” , variable=var, value=3, command=sel)
R3.pack( anchor = W)

label = Label(root)
label.pack()

root.mainloop()
```



“IntVar()” in Checkbutton Widget

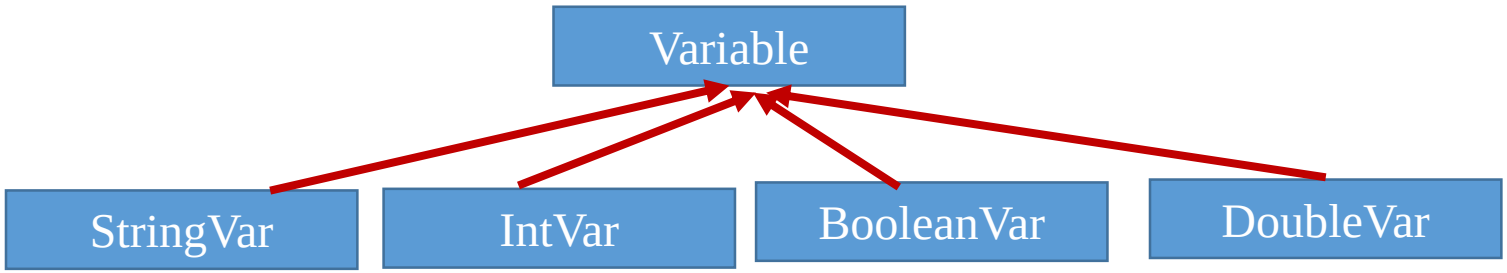
- **CheckVar1(2)**: Assign Checkbutton’s value to these variable
- If Checkbutton C1(C2) is checked, **onvalue 1** is assigned to CheckVar1(2)
- Else, **offvalue 0** is assigned

```
from tkinter import *

top = Tk()

CheckVar1 = IntVar()
CheckVar2 = IntVar()

C1 = Checkbutton(top, text = "Music", variable = CheckVar1, onvalue = 1,
offvalue = 0, height=5, width = 20)

C2 = Checkbutton(top, text = "Video", variable = CheckVar2, onvalue = 1,
offvalue = 0, height=5, width = 20)

C1.pack()
C2.pack()

top.mainloop()
```



Tkinter Variables

- Variables can be used as widget options to hold values associated with widgets (eg. Value option for RadioButtons)
- Tkinter provides a way for the widget to adjust to a change in the value of such a variable
- Tkinter provides the Variable class
- Variable class has 4 subclasses
  - StringVar class, IntVar class, BooleanVar class, DoubleVar class



Tkinter Widgets for Output [1/2]

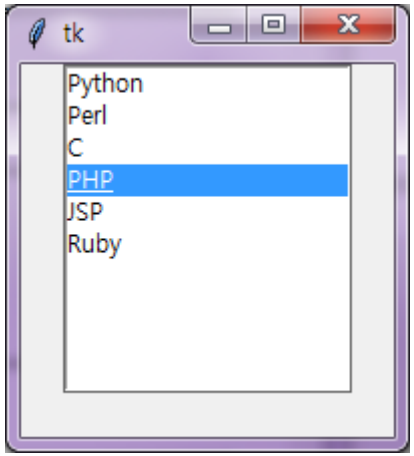
- Label widget
  - Text or image을 보여주는 가장 기본적인 방법
  - StringVar class를 이용하여 text값을 저장
  - PhotoImage class / BitmapImage class를 이용하여 image를 저장
    - Image(gif, pgm, ppm)와 bitmap(x11)을 각각 보여준다.

label

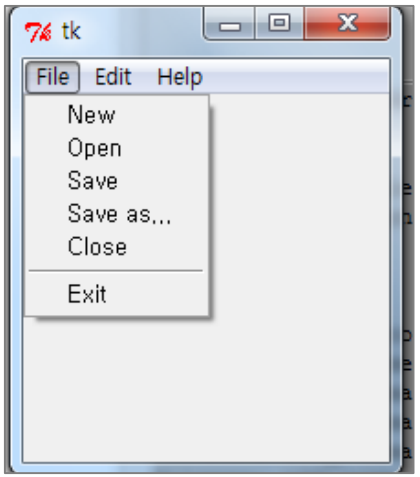
image

Tkinter Widgets for Output [2/2]

- Listbox widget: User가 강조할 수 있는 Text items의 목록을 보여줌
- Menu widget : User가 menu를 만들 수 있는 기능을 제공



listbox

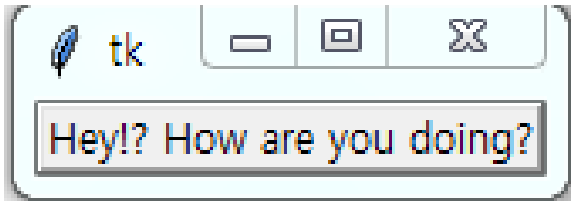


menu

- More Widgets in Tkinter and their descriptions
  - http://www.tutorialspoint.com/python/python_gui_programming.htm

Label Widget Example for Text

```
from tkinter import *

root = Tk()

var = StringVar()
label = Label( root, textvariable=var, relief=RAISED )

var.set("Hey!? How are you doing?")
label.pack()
root.mainloop()
```

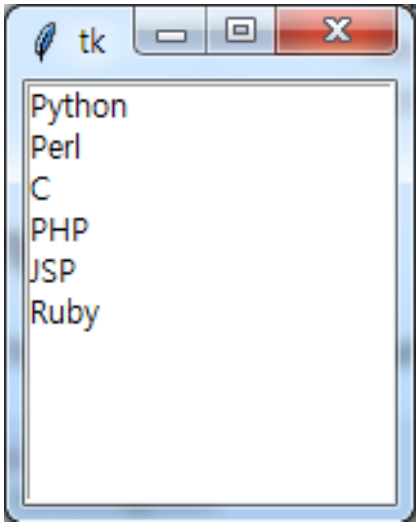


Label Widget for Displaying Image Example

```
from tkinter import *

root =Tk()

#image
photo = PhotoImage(file='c:₩suzi.gif')

peace = Label( master=root, image=photo, width=350, height=700)

peace.pack()
root.mainloop()
```

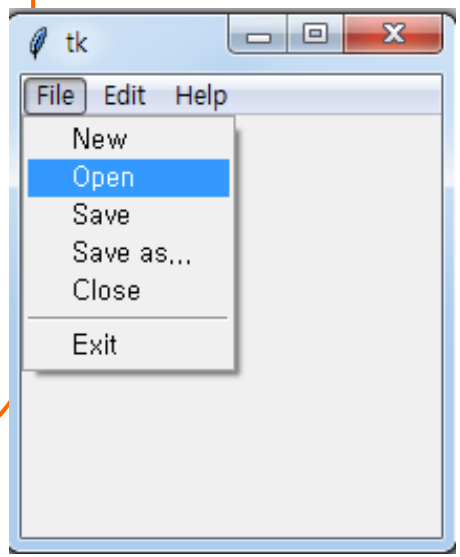
Listbox Widget Example

```
from tkinter import *

top = Tk()

Lb1 = Listbox(top)
Lb1.insert(1, "Python")
Lb1.insert(2, "Perl")
Lb1.insert(3, "C")
Lb1.insert(4, "PHP")
Lb1.insert(5, "JSP")
Lb1.insert(6, "Ruby")

Lb1.pack()
top.mainloop()
```



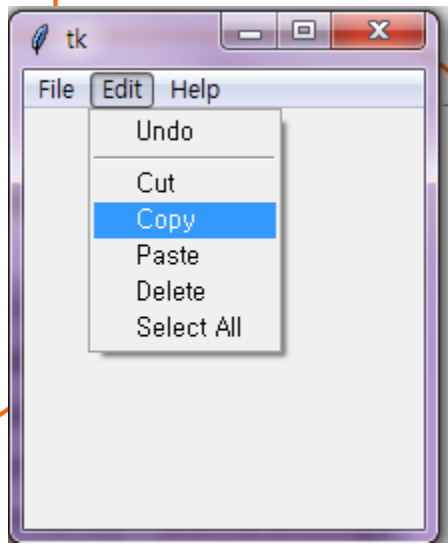
Menu Widget Example [1/2]

```
from tkinter import *
def donothing():
   filewin = Toplevel(root)
   button = Button(filewin, text="Do nothing button")
   button.pack()

root = Tk()
menubar = Menu(root)

filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
```



Menu Widget Example [2/2]

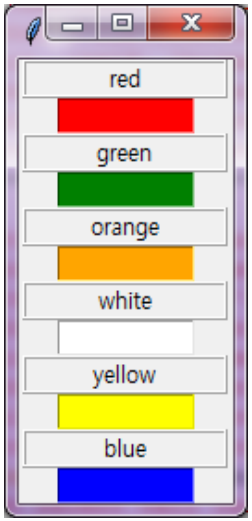
```
editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)
editmenu.add_separator()
editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Delete", command=donothing)
editmenu.add_command(label="Select All", command=donothing)
menubar.add_cascade(label="Edit", menu=editmenu)

helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)

root.config(menu=menubar)
root.mainloop()
```



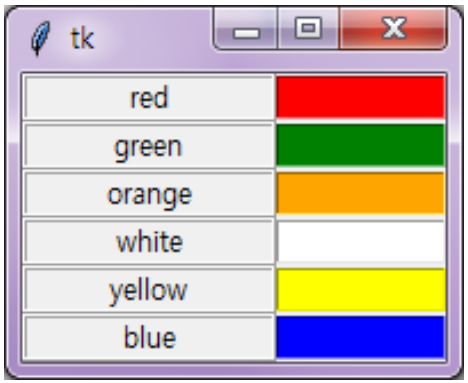
Geometry Management in TKinter

- Geometry management consists of widget placement and sizing of screen
  - organizing widgets throughout the parent widget area

- Tkinter exposes the following 3 geometry manager classes
  - The *pack()* Method - organizes widgets in blocks before placing them in the parent widget.
  - The *grid()* Method - organizes widgets in a table-like structure in the parent widget.
  - The *place()* Method -organizes widgets by placing them in a specific position in the parent widget.



pack



grid

place

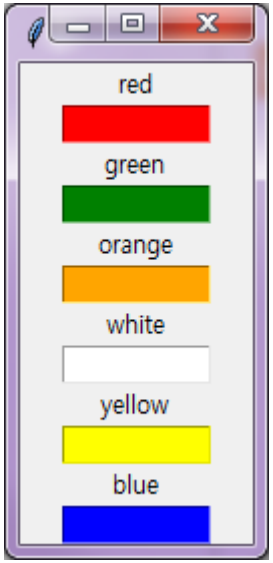
The Pack Geometry Manager

- Quickest and most common way to design interfaces
- Positioning is done relative to the container widgets (top, bottom, left, right)
- Widgets are packed from edge to center of the container, using space left available by previous pack operations
- Options to the pack() method

Option	Possible Values
expand	YES / **NO**
fill	**NONE** / X / Y / BOTH
side	**TOP** / BOTTOM / RIGHT / LEFT
in_('in')	Widget
padx, pady	Integer values
ipadx, ipady	Integer values
anchor	N / S / E / W / NW / SW / NE / SE / NS / EW / NSEW / **CENTER**

Layout Example using pack()

```
from tkinter import *

colours = ['red','green','orange','white','yellow','blue']

r = 0
for c in colours:
    Label(text=c, relief=RIDGE, width=15).pack()
    Entry(bg=c, relief=SUNKEN, width=10).pack()
    r = r + 1
mainloop()
```



pack

The Grid Geometry Manager

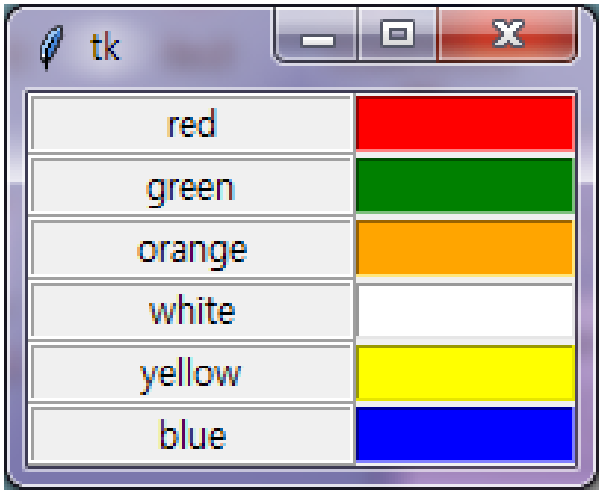
- Used for more complex layouts
- Allows for container to be divided in rows and columns
- Options to the grid() method

Option	Possible Values
row, column	Positive integer values
rowspan, columnspan	Positive integer values
in_('in')	Widget
padx, pady	Integer values
ipadx, ipady	Integer values
sticky	N / S / E / W / NW / SW / NE / SE / NS / EW / NSEW (Note: Default is to center widgets *)

* CENTER is not supported with the sticky option

Layout Example using grid()

```
from tkinter import *

colours = ['red','green','orange','white','yellow','blue']

r = 0
for c in colours:
    Label(text=c, relief=RIDGE, width=15).grid(row=r , column=0)
    Entry(bg=c, relief=SUNKEN, width=10).grid(row=r , column=1)
    r = r + 1

mainloop()
```



grid

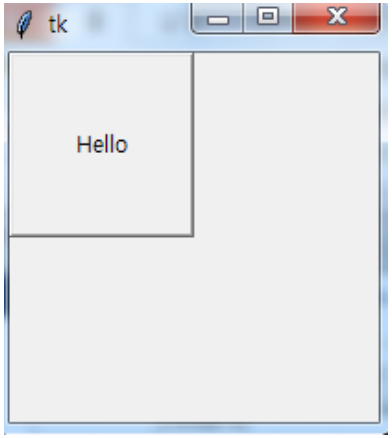
The Place Geometry Manager

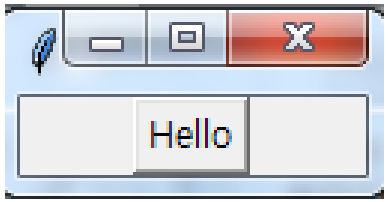
- Most powerful manager
- Allows exact placement of widgets in a container
  - Exact coordinates, size percentage
- Options to the place() method

Option	Possible Values
anchor	N / NE / E / SE / SW / W / **NW** / CENTER
bordermode	INSIDE / OUTSIDE
in_('in')	Widget
relwidth, relheight	Float [0.0, 1.0]
relx, rely	Float [0.0, 1.0]
width, height	Integer values
x, y	Integer values

Layout Example using place()

```
import tkinter

top = tkinter.Tk()

B = tkinter.Button(top, text ="Hello" )

B.pack()
B.place(height=100, width=100)
top.mainloop()
```





- B.place(height=100, width=100) ➔ Upper one
- B.place() ➔ Lower one

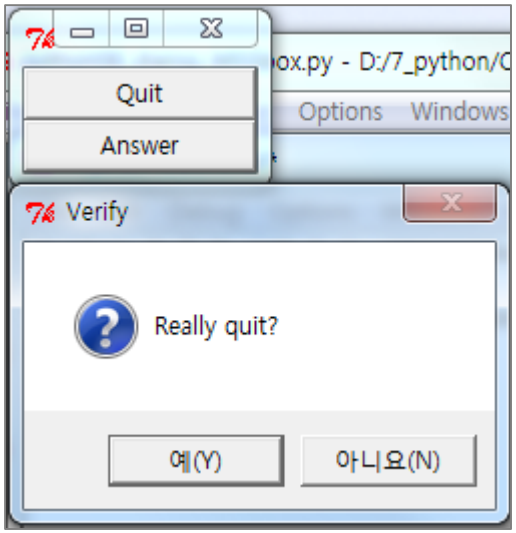
Dialog Box widgets

- A small modal window that lets you
  - Ask a question or Show a message (Message Dialog Box)
  - Do file-related dialogs (File Dialog Box)
  - Ask for a number or string (SimpleDialog Dialog Box)
  - Choose colors (Color Chooser Dialog Box)
  - Do other things in a separate window from the main window

- Each dialog box widgets has a corresponding useful functions
  - tkinter.messagebox: showerror(), askyesno(), showwarning()
  - tkinter.filedialog: askopenfilename(),
  - tkinter.simpledialog: askstring(), askinteger(), askfloat(),…
  - tkinter.colorchooser: askcolor(),….

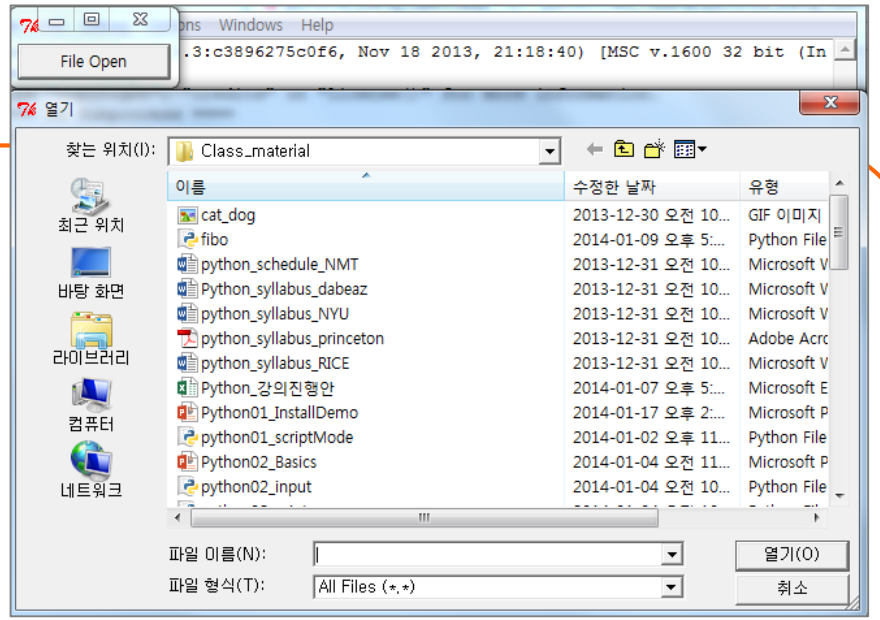
Message Dialog Box

```
from tkinter import *
from tkinter.messagebox import *

def answer() :
    showerror("Answer", "Sorry, no answer available")

def callback() :
    if askyesno('Verify', 'Really quit?') :
        showwarning('Yes', 'Not yet implemented')
    else :
        showinfo('No', 'Quit has been cancelled')

Button(text='Quit',      command=callback).pack(fill=X)
Button(text='Answer', command=answer).pack(fill=X)
mainloop()
```



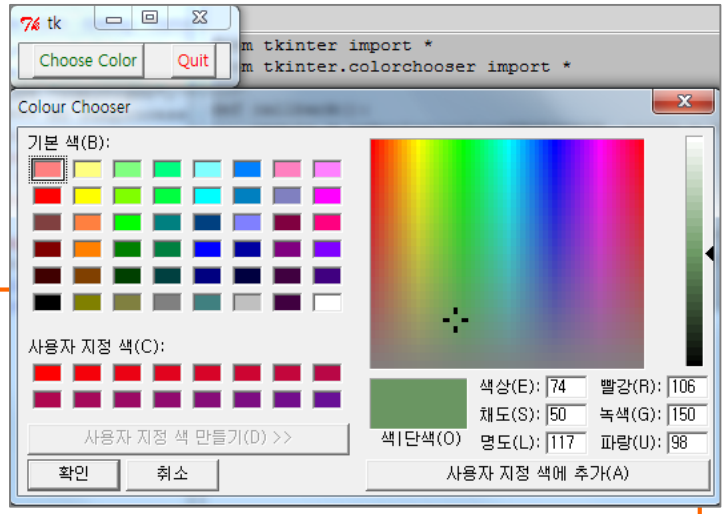

```
askokcancel(title=None, message=None, **options)
askquestion(title=None, message=None, **options)
askretrycancel(title=None, message=None, **options)
askyesno(title=None, message=None, **options)
askyesnocancel(title=None, message=None, **options)
showerror(title=None, message=None, **options)
showinfo(title=None, message=None, **options)
showwarning(title=None, message=None, **options)
```

File Dialog Boxes




```
from tkinter import *
from tkinter.filedialog import *

def callback() :
    name= askopenfilename()
    print(name)

errmsg = 'Error!'
Button(text='File Open', command=callback).pack(fill=X)

mainloop()
```

Color Chooser Dialog Box




```
from tkinter import *
from tkinter.colorchooser import *

def callback() :
    result = askcolor(color="#6A9662", title = "Colour Chooser")
    print(result)

root = Tk()

Button(root, text='Choose Color', fg="darkgreen", command=callback).pack(side=LEFT,
padx=10)
Button(text='Quit', command=root.quit, fg="red").pack(side=LEFT, padx=10)

root.mainloop()
```