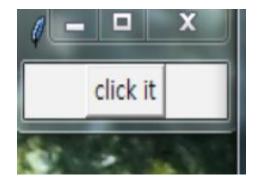
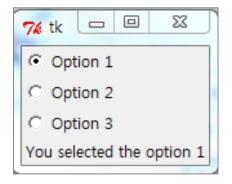
#### TKINTER:

# 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......

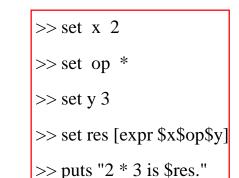






# **Tkinter History**

- 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 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]

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
- There are currently 15 types of widgets in Tkinter.
- We present these widgets as well as a brief description in the following table –

Operator	Description	
<b>Button</b>	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 your 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.	
<b>Frame</b>	The Frame widget is used as a container widget to organize other widgets.	
<u>Label</u>	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.	
<b>Listbox</b>	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.	
<u>Menu</u>	The Menu widget is used to provide various commands to a user. These commands are contain ed inside Menubutton.	



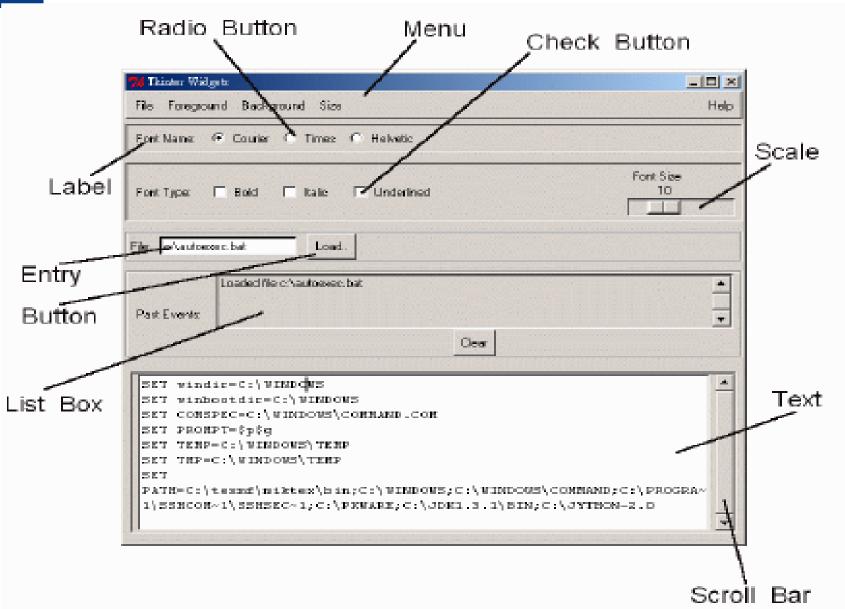
# Tkinter Widgets [2/2]

모든 Widget은 Class 구조!

Operator	Description	
<u>Message</u>	The Message widget is used to display multiline text fields for accepting values from a us er.	
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.	
<u>Scale</u>	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 bo xes.	
<u>Text</u>	The Text widget is used to display text in multiple lines.	
<b>Toplevel</b>	The Toplevel widget is used to provide a separate window container.	
<b>Spinbox</b>	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.	
<b>PanedWindow</b>	A PanedWindow is a container widget that may contain any number of panes, arranged ho rizontally or vertically.	
<b>LabelFrame</b>	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or cont ainer 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()
```

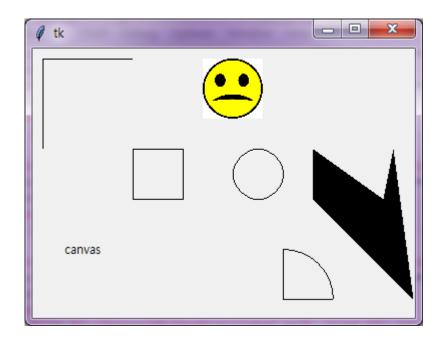


Microsoft Windows 2000



#### Canvas widget for Computer Graphics: Drawing

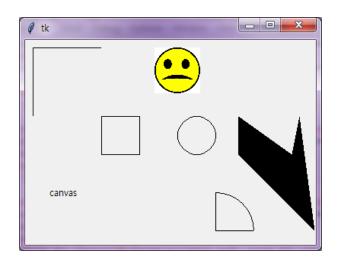
- 그림이나 복잡한 layout을 표현하기 위한 직사각형 모양의 판
- Canvas상에는 도형이나 문자, widget 등을 넣을 수 있다





# **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**

■ 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



# **Canvas Code Example**

root.mainloop()

```
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
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)
```



- - X

### **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)
```

Result:



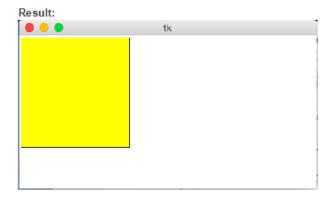
tk

### **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")
```

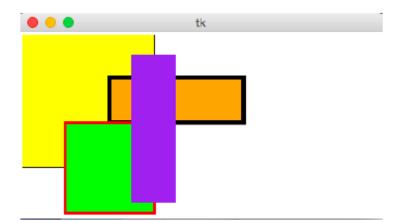




### **Practicing Canvas Drawing** [3/9]

```
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)
```

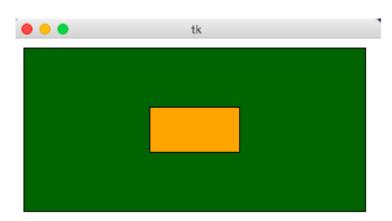
#### 3. Draw Multiple Rectangles





### **Practicing Canvas Drawing** [4/9]

#### 4. Draw Centered Rectangles





### **Practicing Canvas Drawing** [5/9]

#### 5. Draw Custom Colors

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

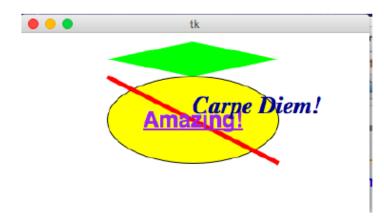
def draw(canvas, width, height):
    pistachio = rgbString(147, 197, 114)
    maroon = rgbString(176, 48, 96)
    canvas.create_rectangle(0, 0, width/2, height/2, fill=pistachio)
    canvas.create_rectangle(width/2, height/2, width, height, fill=maroon)
```





### **Practicing Canvas Drawing** [6/9]

#### Draw Other Shapes and Text

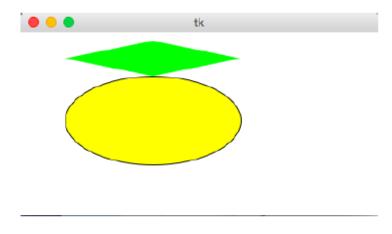




### **Practicing Canvas Drawing** [7/9]

#### 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")
```





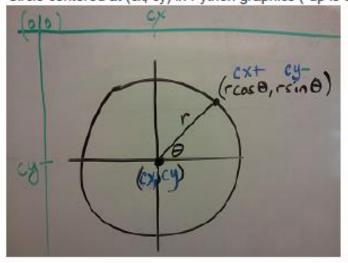
### **Practicing Canvas Drawing** [8/9]

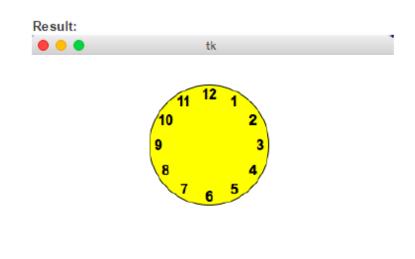
#### 8. Graphics Helper Functions

```
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
                                                                 Result:
                                                                  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)
```

### **Practicing Canvas Drawing** [9/9]

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





```
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)
- <u>image</u> (a <u>BitmapImage</u> or <u>PhotoImage</u> instance)
- <u>line</u>
- oval (a circle or an ellipse)
- polygon
- rectangle
- <u>text</u>
- 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:

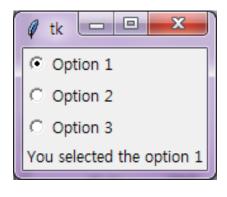
ALL (or the string "all") matches all items on the canvas.

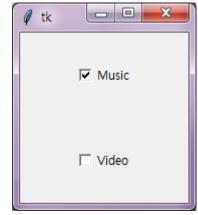
**CURRENT** (or "current") matches the item under the mouse pointer, if any. This can be used inside mouse event bindings to refer to the item that triggered the callback.



### Python TkInter Graphical User Interface (GUI)







button

radio button

check button





menu

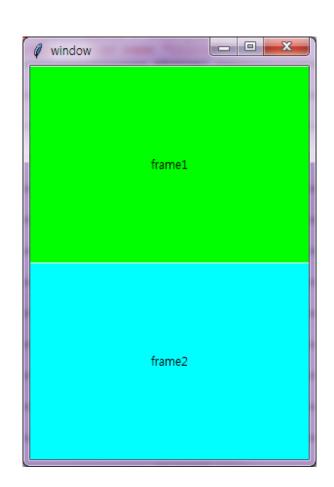


#### **Basic of Tkinter: Window & Frame**

- 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

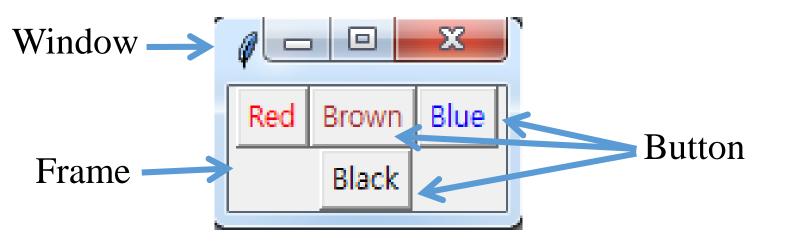
```
frame1 = Frame(root)
frame2 = Frame(root)
```





### 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 Tkinter program

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



Microsoft Windows 2000



#### **First Tkinter Window**

```
# first Tkinter window
```

```
from tkinter import *
```

```
root = Tk() # Create the root (base) window where all widgets go
w = Label(root, text = "Hello, world!") # Create a label with words
w.pack() # Put the label into the window
root.mainloop() # Start the event loop
```





# Explain the code

# first Tkinter window

from tkinter import \*

root = Tk()

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!")

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()

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

root.mainloop()

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

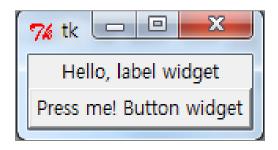
```
# label and button

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
                                                                            Blue
                                                           Red
                                                                  Brown
frame = Frame(root)
frame.pack()
                                         bottom
                                                                   Black
bottomframe = Frame(root)
                                         frame
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")
                                                              fg 

foreground
blackbutton.pack( side = BOTTOM)
                                                              bottom frame에는 button이
```

34/65

root.mainloop()

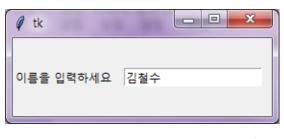
한개만 있어서 side=LEFT

로 해도 중앙에 위치한다

### Tkinter widgets for 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)

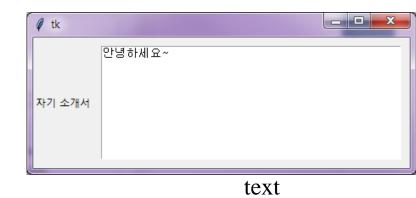
createTextBox(root)

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

# Tkinter widgets for input [2/3]

## Text widget

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



from tkinter import \*

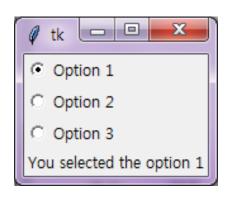
root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, "Just a text Widget \n in two lines \n ")
mainloop()

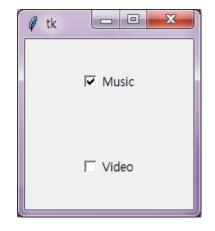


#### Tkinter widgets for input [3/3]

- 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.







radio button check button



#### 2 Buttons Example

```
from tkinter import *

root=Tk()

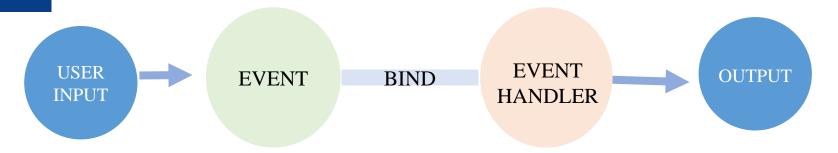
b1 = Button(root, text = "OK", height = 5, width = 20)
b1.pack()
b2 = Button(root, text = "Cancel", height=5, width = 20)
b2.pack()
```



button

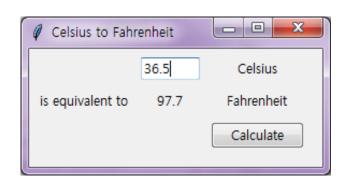


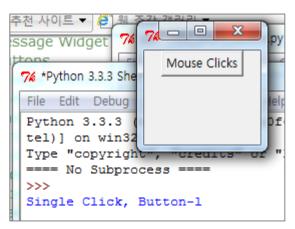
#### **User Interaction: Events and binds**



- 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







#### Levels of binding

- Instance binding
  - 한 event를 특정한 widget에 묶을 수 있다.
    - self.canv.bind('<Button-2>', self.\_\_drawOrangeBlob)
- Class binding
  - Class의 모든 widget에 하나의 event를 묶을 수 있다.
    - self.bind\_class('Canvas', '<Button-2>', self.\_\_drawOrangeBlob)
- Application binding
  - Widget이 focus되어있거나 마우스 아래에 있는 것과 상관없이 확실하 게 event가 handler를 호출할 수 있도록 binding하는 것.
    - self.bind\_all('<Key-Print>', self.\_\_printScreen)



# List of Event patterns

Event patterns	Description
<button-1></button-1>	첫 번째 mouse button을 누른다.
<keypress-h></keypress-h>	H key를 누른다.
<control-shift-keypress-h></control-shift-keypress-h>	control-shift-H를 누른다.
<b1-motion></b1-motion>	왼쪽버튼을 누른채로 마우스를 드래그한다.
<double-button-1></double-button-1>	더블클릭
<enter></enter>	마우스 포인터가 widget안에 있다.
<leave></leave>	마우스 포인터를 widget 밖으로 둔다.
<return></return>	Enter key를 누른다.
<configure></configure>	크기를 바꾼다.



# Event object의 attribute 리스트

■ 모든 event handler들은 단일 매개변수를 전달

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 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)

SSage Widget 76

SSage Widget 76

Whouse Clicks

System File Edit Debug

Python 3.3.3 (tel)] on win32

Type "copyright", "crears" or ":

==== No Subprocess ====

>>>

Single Click, Button-1

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

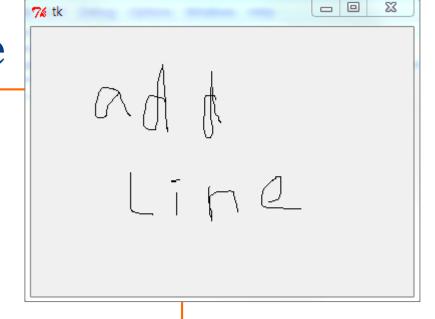
root.mainloop()

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



## 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:
      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)
```

## **Button click Example**

■ 누르면 시간을 나타내는 Button의 생성

```
from tkinter import Tk, Button
from time import strftime, localtime
def clicked():
   time = strftime("Day: %d %b %Y ₩nTime: %H : %M : %S %p₩n", localtime())
   print(time)
root = Tk()
#create button
but = Button( root, text="click it", command=clicked)
but.pack()
root.mainloop()
```





#### 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].	
%н	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].	
8m	Month as a decimal number [01,12].	
8M	Minute as a decimal number [00,59].	
<del>l</del> p	Locale's equivalent of either AM or PM.	(1)
%S	Second as a decimal number [00,61].	(2)
дУ	Year without century as a decimal number [00,99].	
%Y	Year with century as a decimal number.	

## 3 Radiobuttons Example

```
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)
                                                                        \Sigma S
                                                                    7% tk
label = Label(root)
                                                              Option 1
label.pack()
                                                              Option 2
root.mainloop()
                                                              Option 3
                                                              You selected the option 1
```



#### 2 Checkbuttons Examples

from tkinter import \*

top = Tk()

```
• 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

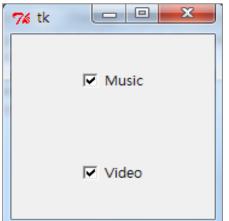
```
CheckVar1 = IntVar()
CheckVar2 = IntVar()
```

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()

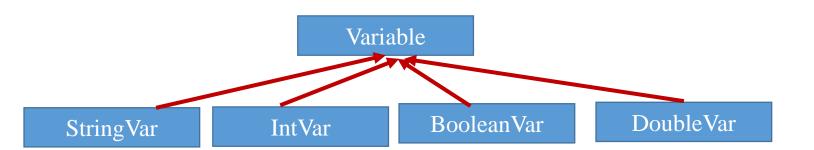
top.mainloop()





#### Tkinter variables

- Variables can be used as widget options to hold values associated with them (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

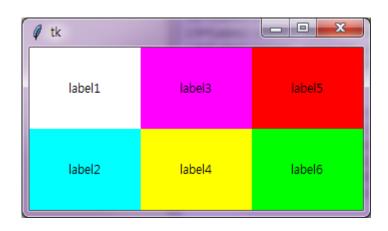




## Tkinter widgets for output [1/2]

#### Label widget

- Text or image을 보여주는 가장 기본적인 방법.
- StringVar를 이용하여 text값을 저장
- PhotoImage/BitmapImage를 이용하여 image를 저장
  - Image(gif, pgm, ppm)와 bitmap(x11)을 각각 보여준다.
  - 실제 widget보다 class 객체(objects)에 가깝다.





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



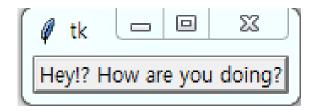
#### Label Widget Example for Text

```
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)

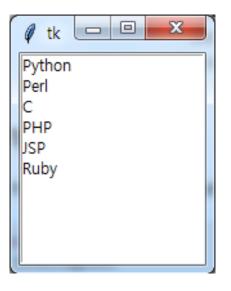
peace.pack()
root.mainloop()
```





#### Listbox Widget Example

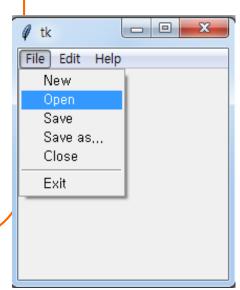
```
from tkinter import *
top = Tk()
Lb1 = Listbox(top)
Lb1.insert(1, "Python")
Lb1.insert(2, "Perl")
Lb1.insert(3, "C")
Lb1.insert(4, "PHP")
Lb1.insert(5, "JSP")
Lb1.insert(6, "Ruby")
Lb1.pack()
top.mainloop()
```





#### 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)
```



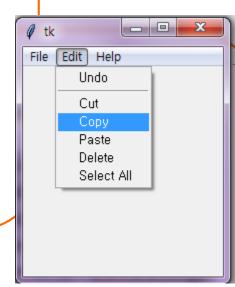


#### Menu Widget Example [2/2]

```
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)

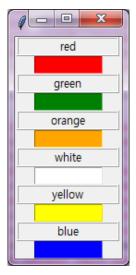
root.config(menu=menubar)
root.mainloop()

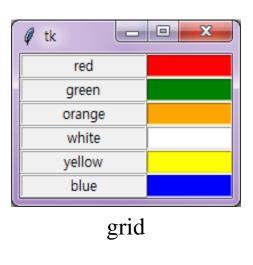


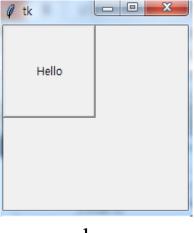


#### Geometry Management in TKinter

- 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.







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/ <b>CENTER</b>

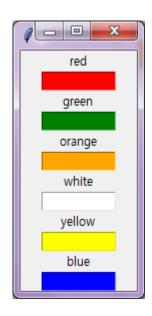


#### Layout example using pack()

```
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 *)

<sup>\*</sup> CENTER is not supported with the sticky option



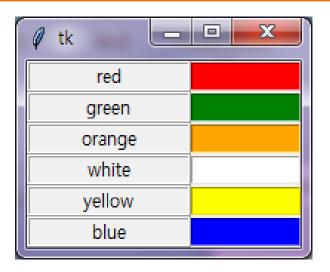
#### Layout example using grid()

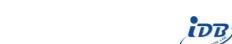
```
from tkinter import *

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/ <b>NW</b> /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



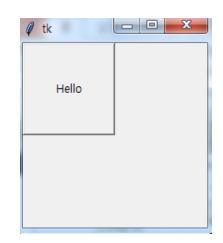
#### Layout example using place()

```
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()  $\rightarrow$  Lower one



# **Options for Tkinter widgets**

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	배경 색상
foreground	전경 색상
font	폰트이름 지정



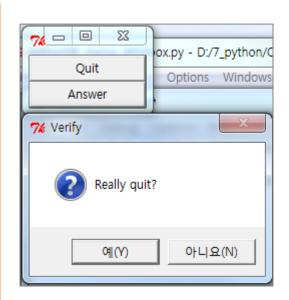
#### **Dialog Box widgets**

- 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()



76 - 0 33

File Open

바탕 화면

라이브러리

컴퓨터

네트워크

찾는 위치(I): 🚹 Class\_material

se cat dog

python\_schedule\_NMT

Python\_syllabus\_dabeaz

python\_syllabus\_princeton

python\_syllabus\_NYU

python syllabus RICE

Python01\_InstallDemo

python01\_scriptMode

All Files (\*,\*)

Python02\_Basics

python02\_input

파일 이름(N):

파일 형식(T):

Python\_강의진행안

ibo fibo

Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (In

2013-12-30 오전 10...

2014-01-09 오후 5:...

2013-12-31 오전 10...

2014-01-07 오후 5:...

2014-01-17 오후 2:...

2013-12-31 오전 10... Microsoft V

2013-12-31 오전 10... Microsoft V

2013-12-31 오전 10... Adobe Acrc

2013-12-31 오전 10... Microsoft V

2014-01-02 오후 11... Python File

2014-01-04 오전 11... Microsoft P

2014-01-04 오전 10... Python File

열기(0)

취소

#### **Color Chooser Dialog Box**

```
from tkinter import *
                                                                                         빨강(R): 106
from tkinter.colorchooser import *
                                                                                   채도(S): 50
                                                                                         녹색(G): 150
                                                                             색I단색(0) 명도(L): 117 파랑(U): 98
                                                                 취소
                                                                                 사용자 지정 색에 추가(A)
                                                           확인
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()
```

\_ O

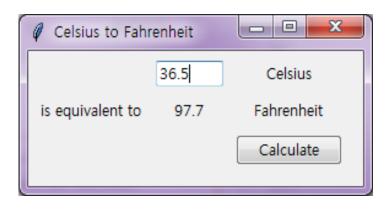
Choose Color Colour Chooser 기본 색(B):



#### Real World Practices of Tkinter



#### Practice 1: C-F converter [1/2]



```
from tkinter import *
from tkinter import ttk

def calculate(*args):
    try:
       value = float(celsius.get())
       fahrenheit.set(value * 9 / 5 + 32)
    except ValueError:
       pass

root = Tk()
root.title("Celsius to Fahrenheit")
```



#### Practice 1: C-F converter [2/2]

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

```
Celsius to Fahrenheit
#ttk.Frame()
mainframe = ttk.Frame(root, padding="3 3 12 12")
                                                                          36.5
                                                                                      Celsius
mainframe.grid(column=0, row=0)
                                                              is equivalent to
                                                                            97.7
                                                                                    Fahrenheit
celsius = StringVar()
                                                                                     Calculate
fahrenheit = StringVar()
#ttk.Entry
cel_entry = ttk.Entry(mainframe, width=7, textvariable=celsius)
cel entry.grid(column=2, row=1)
ttk.Label(mainframe, text="Celsius").grid(column=3, row=1)
ttk.Label(mainframe, text="is equivalent to").grid(column=1, row=2)
ttk.Label(mainframe, textvariable=fahrenheit).grid(column=2, row=2)
ttk.Label(mainframe, text="Fahrenheit").grid(column=3, row=2)
ttk.Button(mainframe, text="Calculate", command=calculate).grid(column=3, row=3)
for child in mainframe.winfo children():
   child.grid configure(padx=5, pady=5)
#entry and run calculate
cel entry.focus()
```

72/65

#### Practice 2 [1/2]

#button

button.pack()

```
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
#Frame
frame = ttk.Frame()
frame['padding'] = (5,10)
frame['borderwidth'] = 2
frame['relief']='sunken'
frame.pack()
#Label
label = ttk.Label(text='Full name;')
resultsContents = StringVar()
label['textvariable'] = resultsContents
resultsContents.set('New value to display')
label.pack()
#image
image = PhotoImage(file='gif.gif')
label['image'] = image
label.pack()
def hello():
   showinfo('자기소개', '안녕하세요~ {0}입니다'.format(name.get()))
```

button = ttk.Button(text='자기소개', command=hello)

73/65





## **Practice 2 [2/2]**

```
#Entry
username = StringVar()
name = ttk.Entry(textvariable=username)
name.insert(0, "이름")
name.pack(pady = 5)
#radiobutton
location = StringVar()
home = ttk.Radiobutton(text='집', variable=location, value='집')
school = ttk.Radiobutton(text='학교', variable=location, value='학교')
library = ttk.Radiobutton(text='도서관', variable=location, value='도서관')
home.pack()
school.pack()
library.pack()
#combobox
countryvar = StringVar()
country = ttk.Combobox(textvariable=countryvar)
country.bind('<<ComboboxSelected>>')
country.set("거주 지역")
country['values'] = ('서울', '경기', '경상', '전라', '충청', '강원', '제주')
country.pack()
```



## **Practice 3 [1/6]**

- 회원 등록을 위해 아래의 기능을 포함하는 화면을 구성한다
  - 이름을 입력한다
  - 성별을 선택한다
  - 생년월일을 선택한다
  - 전화번호를 입력한다
  - 이메일을 입력한다
  - 관심 언어를 최대 3개까지 선택한다





#### **Practice 3 [2/6]**

```
from tkinter import * from tkinter import ttk
```

```
root = Tk()
root.title("회원 등록")
```

```
main = Frame(root, width=400, height=400, bd=1, relief=SUNKEN)
main.pack(fill=X, padx=5, pady=5)
#이름
name label = Label(main, text="이름")
name_label.grid(row=1, column=1, padx = 5, pady = 5, sticky=W)
name_entry = Entry(main, bd = 3, width = 10)
name entry.grid(row=1, column=2, padx = 5, pady = 5, sticky=W)
#성별
gender_label = Label(main, text="성별")
gender label.grid(row=2, column=1, padx = 5, pady = 5, sticky=W)
#성별 입력
gender frame = Frame(main)
gender_frame.grid(row=2, column=2, padx = 5, pady = 5, sticky=W)
varG = IntVar()
gender_rb1 = Radiobutton(gender_frame, text="남성", variable=varG, value=1)
gender_rb1.grid(row=1, column=1, padx = 5)
```



gender rb2 = Radiobutton(gender frame, text="여성", variable=varG, value=2)

gender\_rb2.grid(row=1, column=2, padx = 5)

```
∅ 회원 등록
 Practice 3 [3/6]
                                                                              이름
#생년월일
birth_label = Label(main, text="생년월일")
birth label.grid(row=3, column=1, padx = 5, pady = 5, sticky=W)
                                                                              전화번호 010 ▼ - 2222 - 1111
#생년월일 입력
                                                                              이메일
birth frame = Frame(main)
                                                                              관심언어
birth frame.grid(row=3, column=2, padx = 5, pady = 5, sticky=W)
                                                                                    최대 3개만 선택 가능합니다.
varY = IntVar()
birth combo1 = ttk.Combobox(birth frame, textvariable=varY, state='readonly', width=4)
birth combo1['values'] = list(range(2015, 1899, -1))
birth combo1.set("년")
birth_combo1.grid(row=1, column=1)
year_label = Label(birth_frame, text="년")
vear label.grid(row=1, column=2)
varM = IntVar()
birth combo2 = ttk.Combobox(birth frame, textvariable=varM, state='readonly', width=2)
birth combo2['values'] = list(range(1, 13))
birth_combo2.set("월")
birth combo2.grid(row=1, column=3)
month label = Label(birth frame, text="월")
month label.grid(row=1, column=4)
varD = IntVar()
birth combo3 = ttk.Combobox(birth frame, textvariable=varD, state='readonly', width=2)
birth combo3['values'] = list(range(1, 32))
birth combo3.set("일")
birth_combo3.grid(row=1, column=5)
day_label = Label(birth_frame, text="일")
                                            77/65
day label.grid(row=1, column=6)
```

\_ D X

□ Python

▼ C#

▼ PHP

## **Practice 3 [4/6]**

phone\_entry1.grid(row=1, column=3)

phone\_label3.grid(row=1, column=4)

phone\_entry2.grid(row=1, column=5)

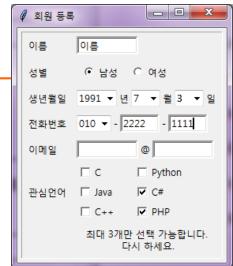
phone label3 = Label(phone frame, text="-")

phone entry2 = Entry(phone frame, bd = 3, width=6)

```
#전화번호
phone_label1 = Label(main, text="전화번호")
phone_label1.grid(row=4, column=1, padx = 5, pady = 5, sticky=W)
#전화번호 입력
phone_frame = Frame(main)
phone frame.grid(row=4, column=2, padx = 5, pady = 5, sticky=W)
varP = IntVar()
phone_combo = ttk.Combobox(phone_frame, textvariable=varP, state='readonly', width=3)
phone combo['values'] = ['010', '02', '031', '032', '033', '041', '042', '043', '044', '051', '052',
'053', '054', '055', '061', '062', '063', '064']
phone combo.current(0)
phone combo.grid(row=1, column=1)
                                                                                   회원 등록
phone_label2 = Label(phone_frame, text="-")
phone label2.grid(row=1, column=2)
phone_entry1 = Entry(phone_frame, bd = 3, width=6)
```

#### **Practice 3 [5/6]**

```
#이메일
mail label1 = Label(main, text="이메일")
mail_label1.grid(row=5, column=1, padx = 5, pady = 5, sticky=W)
                                                                            이메일
#이메일 입력
mail frame = Frame(main)
                                                                            관심언어
mail frame.grid(row=5, column=2, padx = 5, pady = 5, sticky=W)
mail entry1 = Entry(mail frame, bd = 3, width=10)
mail_entry1.grid(row=1, column=1)
mail_label2 = Label(mail_frame, text="@")
mail label2.grid(row=1, column=2)
mail_entry2 = Entry(mail_frame, bd = 3, width=10)
mail_entry2.grid(row=1, column=3)
#관심언어
interest label1 = Label(main, text="관심언어")
interest_label1.grid(row=6, column=1, padx = 5, pady = 5, sticky=W)
#관심언어 입력
interest frame = Frame(main)
interest_frame.grid(row=6, column=2, padx = 5, sticky=W)
def check():
  if varCB1.get()+varCB2.get()+varCB3.get()+varCB4.get()+varCB5.get()+varCB6.get()>3:
     varCB1.set(0)
     varCB2.set(0)
     varCB3.set(0)
     varCB4.set(0)
     varCB5.set(0)
                                            79/65
```



IDB (MERINET DATABLE UNIT

## **Practice 3 [6/6]**

```
이름
이름
        ⓒ 남성 ○ 여성
생년월일 1991 ▼ 년 7 ▼ 월 3 ▼ 일
전화번호 010 ▼ - 2222 - 1111
이메일
               □ Python
관심언어 🗀 Java

✓ C#
       □ C++

▼ PHP

        최대 3개만 선택 가능합니다.
```

```
varCB1 = IntVar()
varCB2 = IntVar()
varCB3 = IntVar()
varCB4 = IntVar()
varCB5 = IntVar()
varCB6 = IntVar()
CB1 = Checkbutton(interest_frame, text="C", variable=varCB1, onvalue=1, offvalue=0, command=check)
CB1.grid(row=1, column=1, sticky=W)
CB2 = Checkbutton(interest_frame, text="Python", variable=varCB2, onvalue=1, offvalue=0, command=check)
```

CB3 = Checkbutton(interest\_frame, text="Java", variable=varCB3, onvalue=1, offvalue=0, command=check)

CB3.grid(row=2, column=1, sticky=W)

CB4 = Checkbutton(interest frame, text="C#", variable=varCB4, onvalue=1, offvalue=0, command=check)

CB4.grid(row=2, column=2, padx= 20, sticky=W)

CB2.grid(row=1, column=2, padx= 20, sticky=W)

CB5 = Checkbutton(interest frame, text="C++", variable=varCB5, onvalue=1, offvalue=0, command=check)

CB5.grid(row=3, column=1, sticky=W)

CB6 = Checkbutton(interest frame, text="PHP", variable=varCB6, onvalue=1, offvalue=0, command=check)

CB6.grid(row=3, column=2, padx= 20, sticky=W)

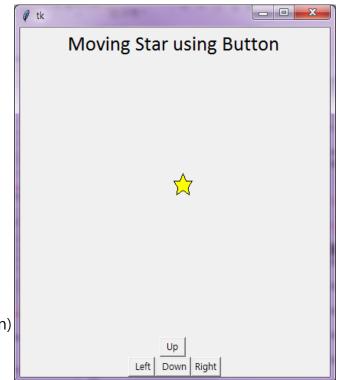
```
interest label2 = Label(main)
interest_label2.qrid(row=7, column=2, padx = 5, pady = 5)
```

root.mainloop()



### **Practice 4: Moving picture on clicking arrow**

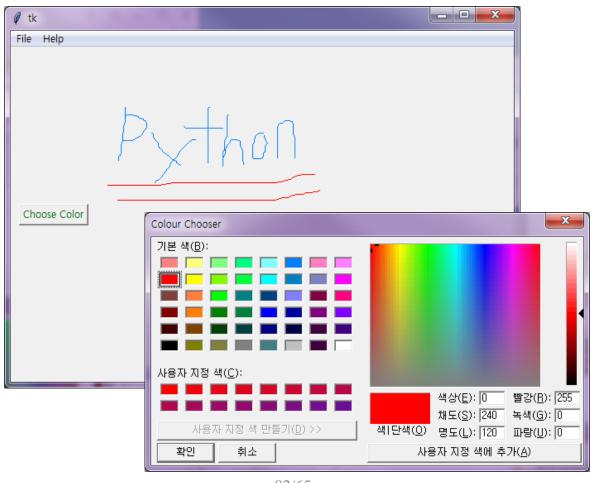
```
from tkinter import *
tk = Tk()
canvas = Canvas(tk, width=400, height=400)
canvas.pack()
canvas.create text(200,20, font=('Calibri', 20), text='Moving Star using Button')
fill='yellow', outline='black')
def keyup():
  canvas.move(star,0,-5)
  canvas.itemconfig(star, fill='yellow')
def keydown():
  canvas.move(star,0,5)
  canvas.itemconfig(star, fill='pink')
def keyleft():
  canvas.move(star,-5, 0)
  canvas.itemconfig(star, fill='lightblue')
def keyright():
  canvas.move(star,5,0)
  canvas.itemconfig(star, fill='white')
btn frame = Frame(tk)
btn frame.pack()
btn_up = Button(btn_frame, text=" Up ", command=keyup)
btn down = Button(btn frame, text=" Down", command=keydown)
btn left = Button(btn frame, text=" Left", command=keyleft)
btn right = Button(btn frame, text="Right", command=keyright)
btn up.grid(row=1, column=2)
btn_left.grid(row=2, column=1)
btn down.grid(row=2, column=2)
btn right.grid(row=2, column=3)
```





# Practice 5: drawing board [1/3]

- 색상 선택 가능
- 파일 열기, 저장





### **Practice 5: Drawing Board [2/3]**

```
from tkinter import *
from tkinter.colorchooser import *
def NewFile() :
  print("New File!")
def OpenFile() :
   print("Open File!")
def About() :
  print("This is a simple example of a menu")
#menu
root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label="File", menu=filemenu)
filemenu.add_command(label="New", command=NewFile)
filemenu.add_command(label="Open...", command=OpenFile)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label="Help", menu=helpmenu)
helpmenu.add_command(label="About...", command=About)
#canvas
canvas = Canvas(root, width=1000, height=500)
color = "red"
result = '#476042'
```



# Practice 5: Drawing Board [3/3]

```
def callback() :
   global result
   result = askcolor(title = "Colour Chooser")
   result = result[1]
button = Button(root, text='Choose Color', fg="darkgreen", command=callback)
button.pack(side=LEFT, padx=10)
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), fill=result)
   lastx, lasty = event.x, event.y
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)
canvas.pack()
canvas.bind("<Button-1>", xy)
canvas.bind("<B1-Motion>", addLine)
root.mainloop()
```

