

# Chapter 10

## Basic GUI Programming Using Tkinter



# Motivations

Tkinter is not only a useful tool for developing GUI projects, but also a valuable pedagogical tool for learning object-oriented programming.



# Getting Started with Tkinter

Getting started with Tkinter with a simple example.



SimpleGUI



```
# file: SimpleGUI.py
import tkinter as tk

window = tk.Tk() # Create a root window
window.title("Simple GUI TTK") # Set title corresponding to file name
window.geometry("300x80") # Set window size to make title visible
label = tk.Label(window, text = "Welcome to Python") # Create a label
button = tk.Button(window, text = "Click Me") # Create a button
label.pack() # Display the label in the window
button.pack() # Display the button in the window

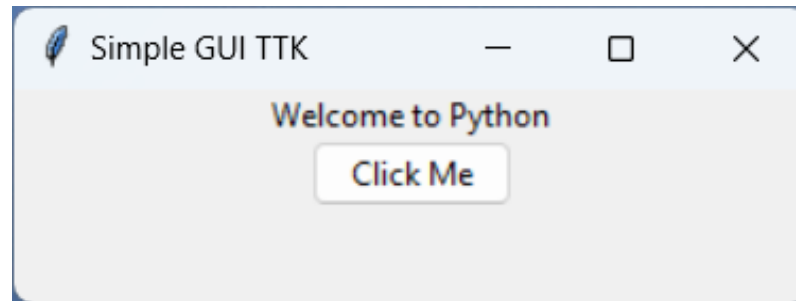
window.mainloop() # Create an event loop
```



```
# file: simple_gui_ttk.py
import tkinter as tk
from tkinter import ttk

window = tk.Tk() # Create a root window
window.title("Simple GUI TTK") # Set title
window.minsize(300,80) # Set window size to make title visible
label = ttk.Label(window, text="Welcome to Python") # Create a ttk label
button = ttk.Button(window, text="Click Me") # Create a ttk button
label.pack() # Display the label in the window
button.pack() # Display the button in the window

window.mainloop() # Create an event loop
```



# Begreper

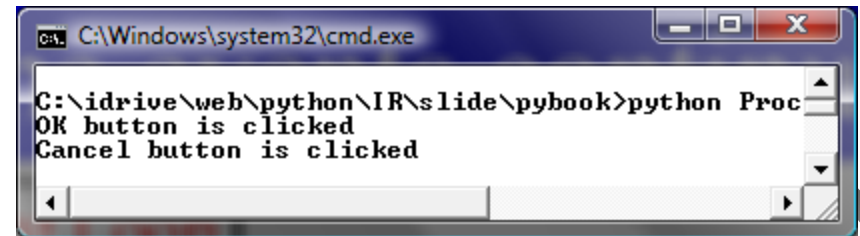
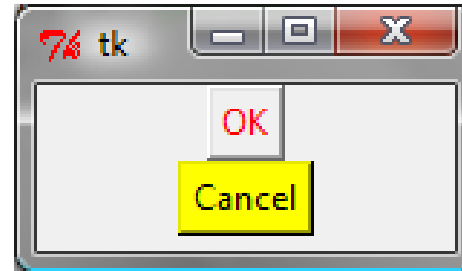
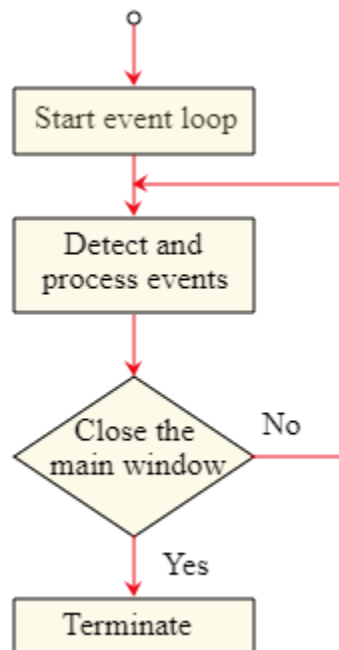
- Hovedvindu / root window
- Widgets
  - Trykk-knapp, meny, label, input felt
- Geometry managers
  - Grid
  - Pack
- Hendelser / events
- GUI programmet
  - Som klasse eller ikke?



# Processing Events

```
window.mainloop() # Create an event loop
```

The statement creates an event loop. The event loop processes the events continuously.



ProcessButtonEvent

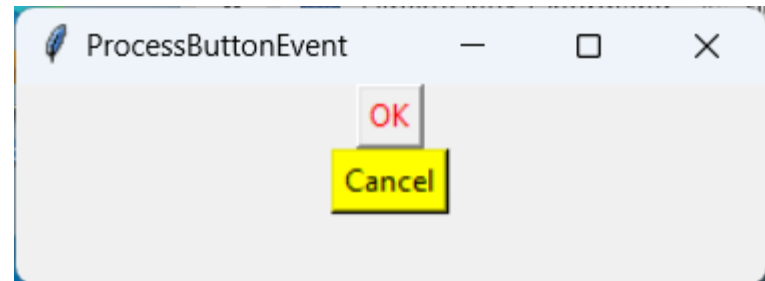
```
# file: ProcessButtonEvent.py
import tkinter as tk

def processOK():
    print("OK button is clicked")

def processCancel():
    print("Cancel button is clicked")

root = tk.Tk() # Create a root window
root.title("ProcessButtonEvent") # Set title
root.geometry("300x80") # Set window size to make title visible
bt_OK = tk.Button(root, text = "OK", fg = "red", command = processOK)
bt_Cancel = tk.Button(root, text = "Cancel", bg = "yellow",
                      command = processCancel)
bt_OK.pack() # Place the button in the window
bt_Cancel.pack() # Place the button in the window

root.mainloop() # Create an event loop
```



```
/OneDrive - UiT Office 365/Faglig/Python/Liang/Bool
eDrive - UiT Office 365/Faglig/Python/Liang/BookSo
OK button is clicked
Cancel button is clicked
```

```
# file: process_button_event_write_to_window.py
import tkinter as tk

def processOK():
    lb_message.config(text="OK button is clicked")
    # lb_message.configure(text="OK button is clicked")
    # lb_message["text"] = "OK button is clicked"

def processCancel():
    lb_message.config(text="Cancel button is clicked")

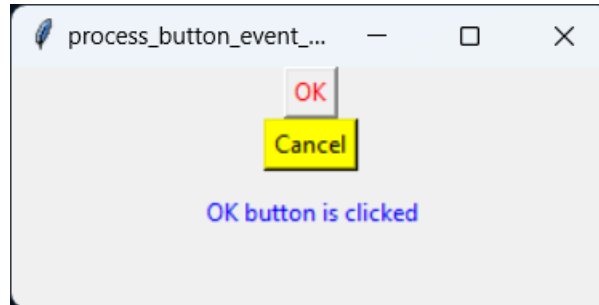
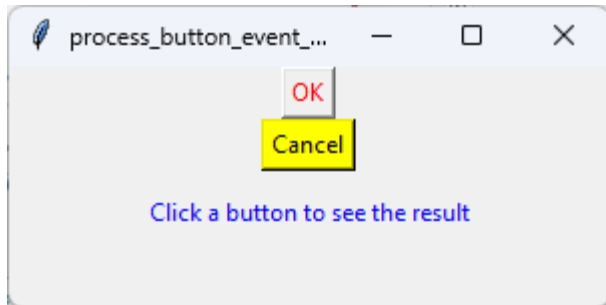
root = tk.Tk() # Create a root window
root.title("process_button_event_write_to_window") # Set title
root.geometry("300x120") # Set window size to make title visible

bt_OK = tk.Button(root, text = "OK", fg = "red", command = processOK)
bt_Cancel = tk.Button(root, text = "Cancel", bg = "yellow", command = processCancel)

lb_message = tk.Label(root, text="Click a button to see the result", fg="blue")

bt_OK.pack()
bt_Cancel.pack()
lb_message.pack(pady=10) # 10 px above and below (y-direction)

root.mainloop() # Create an event loop
```



# Widgets

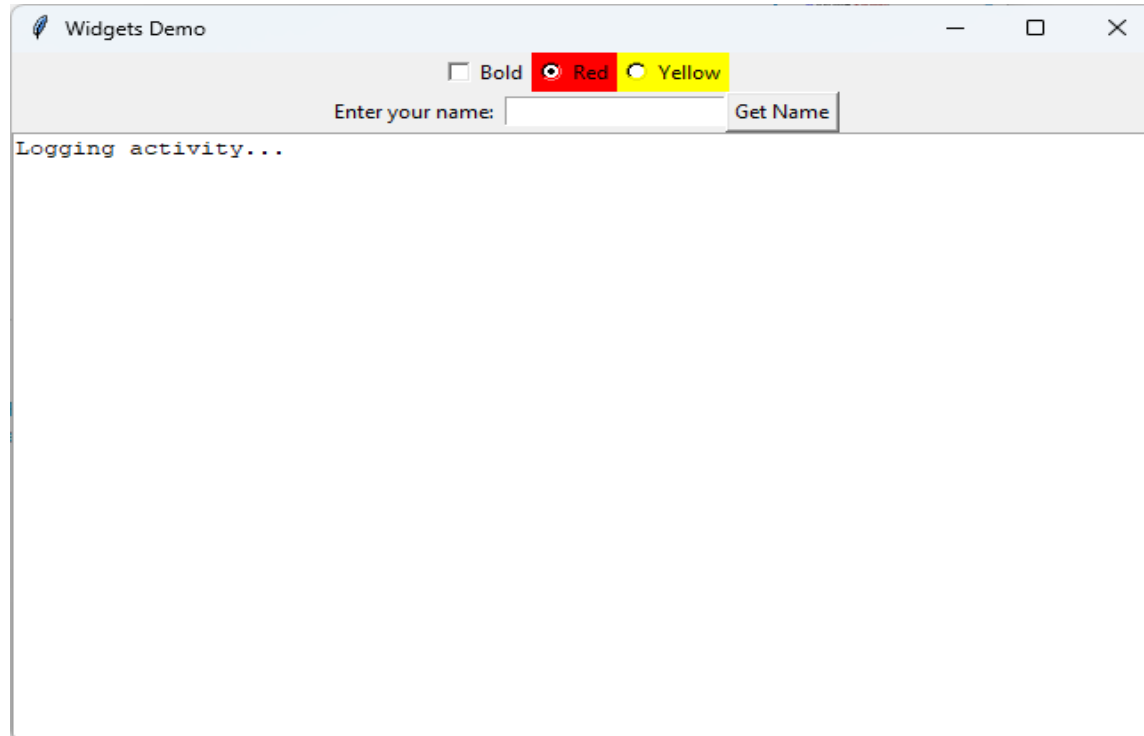
<u>Widget</u>	Beskrivelse	Typisk bruk	Prefiks
<u>Label</u>	Viser tekst i vinduet. Ikke interaktiv.	Meldinger, status, instruksjoner	<u>lb_</u>
Button	En trykknapp som kan kobles til en funksjon via <u>command</u> .	Utføre handlinger, trigge <u>callbacks</u>	<u>bt_</u>
<u>Entry</u>	Et enkelt tekstfelt der brukeren kan skrive inn én linje tekst.	Input av navn, tall, e-post osv.	en_
<u>Text</u>	Et større tekstområde for flere linjer.	Skrive lengre tekst, vise logg	<u>tx_</u>
<u>Frame</u>	En beholder for å gruppere <u>widgets</u> .	Strukturere layout, dele opp GUI	fr_



<u>Checkbox</u>	En avkrysningsboks som kan være av eller på.	Valg som kan aktiveres/deaktiveres	<u>cb_</u>
<u>Radiobutton</u>	En knapp i en gruppe der bare én kan være aktiv.	Velge én av flere alternativer	<u>rb_</u>
<u>Listbox</u>	Viser en liste med elementer	Valg fra liste, visning av data	<u>lbox_</u>



# Widget Demo



WidgetsDemo /



# Nytten av keyword arguments

Mange av funksjonene vi bruker i Tkinter har veldig mange argumenter. Slik kan det se ut hvis vi ikke utnytter keyword arguments:

```
# Uten keyword arguments (må vite rekkefølgen, 8 argumenter):  
label = tk.Label(root, "Hei", None, None, "blue", "white", None,  
("Arial", 12))
```

Dette er:

- Vanskelig å lese
- Lett å gjøre feil
- Krever at du kjenner hele parameterlisten



# Samme med keyword arguments

```
# med keyword, kun 5 argumenter  
label = tk.Label(root, text="Hei", fg="blue", bg="white",  
font=("Arial", 12))
```

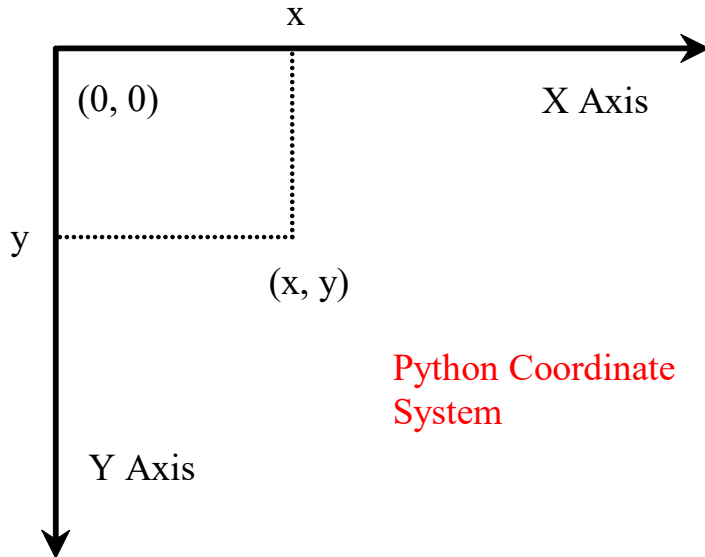
Dette er:

- Lett å lese
- Lett å vedlikeholde
- Fleksibelt – du kan endre én ting uten å røre resten

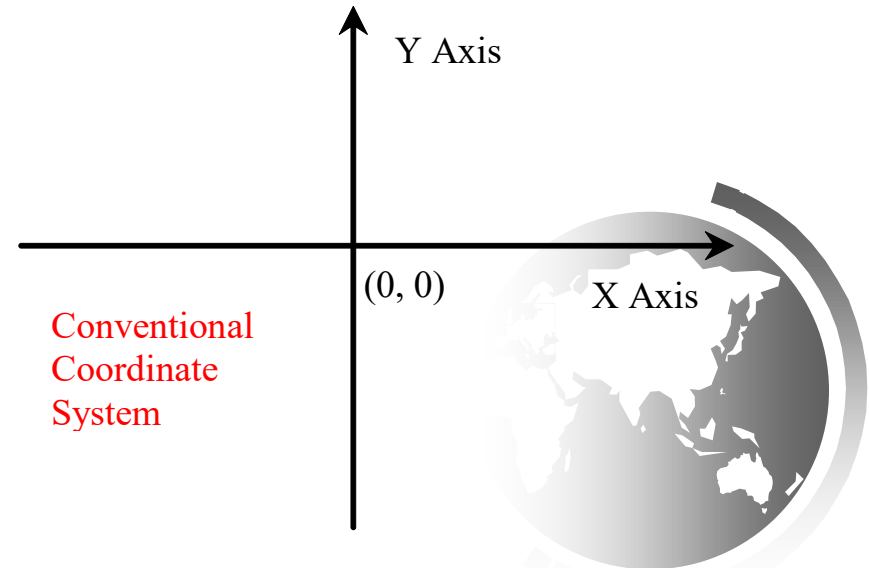


# Canvas

Canvas can be used to display shapes. You can use the method such as `create_rectangle`, `create_oval`, `create_arc`, `create_polygon`, and `create_line` to draw a rectangle, oval, arc, polygon, and line on a canvas.



Python Coordinate System



Conventional Coordinate System

# config() metoden (1)

`config()`-metoden i Tkinter er en svært nyttig måte å endre egenskaper ved en widget etter at den er opprettet. Den gir deg fleksibilitet til å oppdatere utseende eller funksjonalitet dynamisk, uten å måtte lage en ny widget.

Når vi skriver

```
lb_demo.config(text="Knappen ble trykket!")
```

...så endrer vi innholdet (teksten) som vises i Label-widgeten `lb_demo`.



# config() metoden (2)

`config()` er en metode som finnes på alle Tkinter-widgets.

Den kan brukes til å **lese eller endre** widgetens egenskaper (også kalt "options").

Du kan endre flere egenskaper samtidig, for eksempel:

```
lb_demo.config(text="Ny tekst", fg="blue",  
bg="yellow")
```



# Typiske egenskaper du kan endre med config()

Egenskap	Forklaring
text	Teksten som vises i widgeten
fg	Tekstfarge (foreground)
bg	Bakgrunnsfarge
font	Skrifttype og størrelse, f.eks. ("Arial", 12)
width	Bredde på widgeten (i tegn eller piksler, avhengig av type)
height	Høyde på widgeten
padx	Horisontal padding rundt widgeten
pady	Vertikal padding rundt widgeten
command	Funksjon som skal kalles ved interaksjon (f.eks. knappetrykk)
state	Aktiv/inaktiv ("normal" eller "disabled")
anchor	Plassering av innholdet i widgeten (f.eks. "w" for venstre)
justify	Justering av tekst (venstre, høyre, senter) i f.eks. Label
relief	Kantstil: "flat", "raised", "sunken", "groove", "ridge"

# Et alternativ til `config()` - indeksere ved hjelp av keyword

I Tkinter er hver widget bygget opp med en intern option dictionary – en slags oppslagstabell der hver egenskap (som text, fg, bg osv.) har en tilhørende verdi.

Denne kan du både lese fra og skrive til ved hjelp av indeksering, akkurat som med en vanlig Python-dictionary.

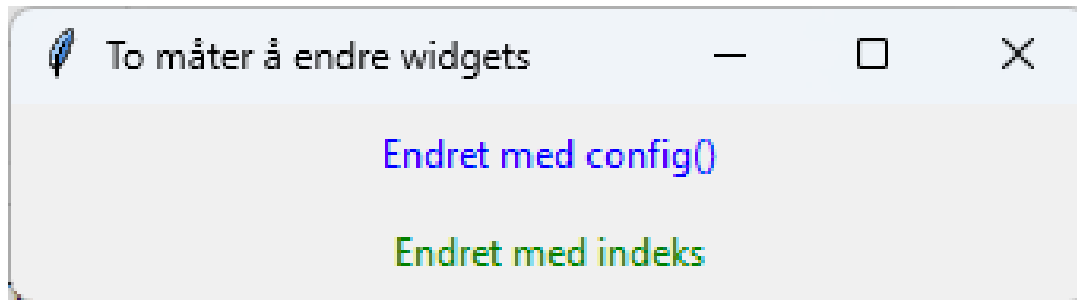


```
# Label 1 - endres med config()
lb_config = tk.Label(root, text="Label med config", fg="black")
lb_config.pack(pady=5)

# Label 2 - endres med indeks
lb_index = tk.Label(root, text="Label med indeks", fg="black")
lb_index.pack(pady=5)

# Endre Label 1 med config()
lb_config.config(text="Endret med config()", fg="blue")

# Endre Label 2 med option dictionary (indeks)
lb_index["text"] = "Endret med indeks"
lb_index["fg"] = "green"
```



# Hvilke egenskaper har en bestemt widget?

- Det er nyttig å vite hvilke egenskaper (options) en bestemt widget har – både for å kunne sette dem ved opprettelse og for å kunne endre dem senere med `config()` eller indeksbasert tilgang.
- Tkinter-widgeter har en intern **option dictionary** som inneholder alle tilgjengelige egenskaper, med tilhørende verdier. Dette gir oss mulighet til å hente ut informasjon om widgetens konfigurasjon på en systematisk måte.



# keys()

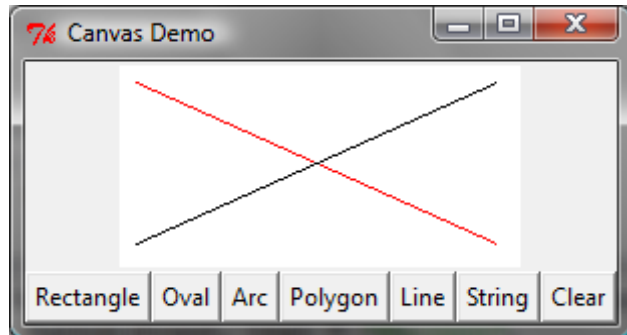
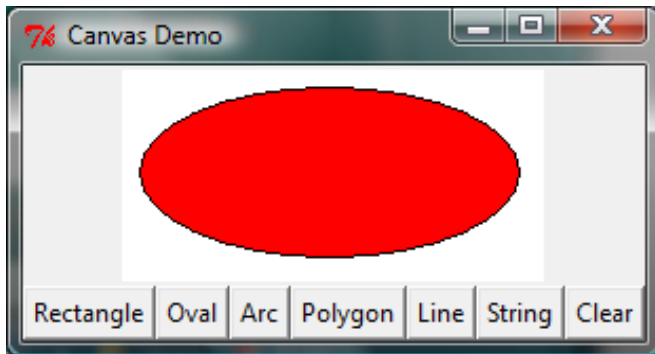
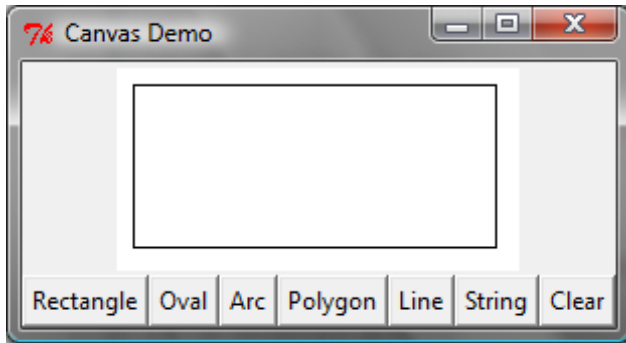
Metoden `keys()` viser alle opsjoner for en bestemt widget.

I REPL kan vi kjøre den slik:

```
>>> tk.Label().keys()
['activebackground', 'activeforeground', 'anchor',
'background', 'bd', 'bg', 'bitmap', 'borderwidth',
'compound', 'cursor', 'disabledforeground', 'fg', 'font',
'foreground', 'height', 'highlightbackground',
'highlightcolor', 'highlightthickness', 'image', 'justify',
'padx', 'pady', 'relief', 'state', 'takefocus', 'text',
'textvariable', 'underline', 'width', 'wraplength']
>>>
```



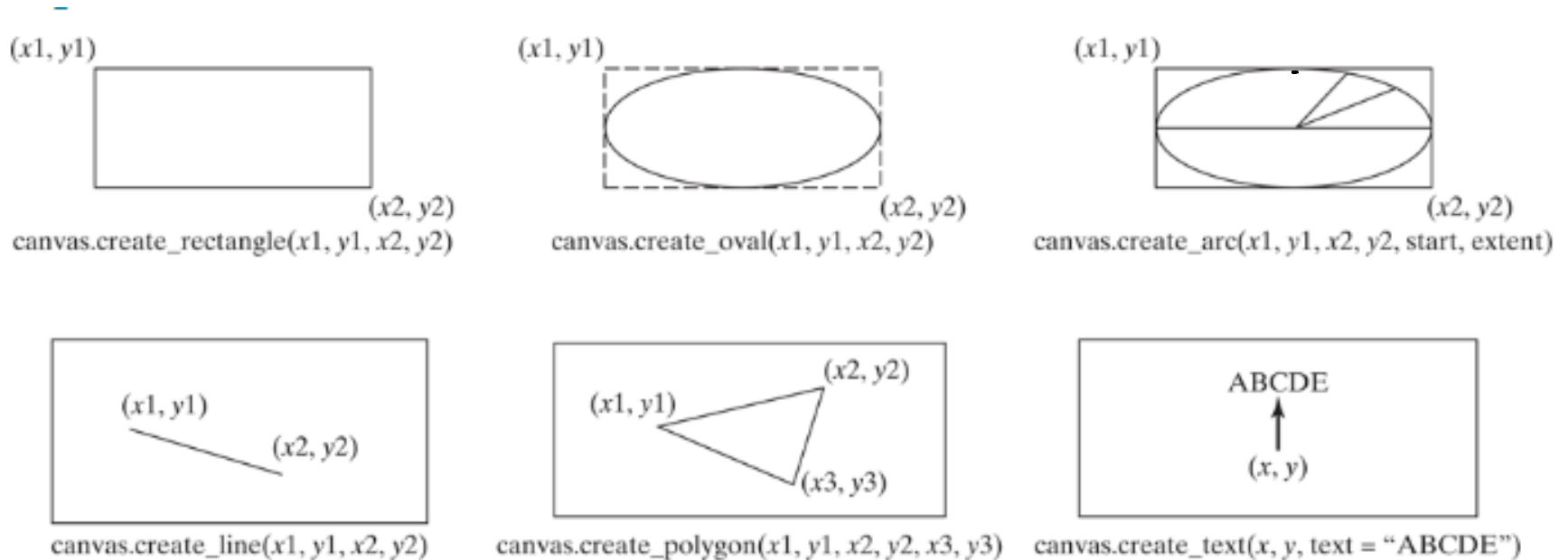
# Canvas Demo



CanvasDemo



# Drawing Methods



The Canvas class contains the methods for drawing graphics.



# create\_arc()

## **-extent degrees**

Specifies the size of the angular range occupied by the arc.

The arc's range extends for degrees degrees ***counter-clockwise from the starting angle given by the -start option.***

Degrees may be negative.

If it is greater than 360 or less than -360, then degrees modulo 360 is used as the extent.

## **-start degrees**

Specifies the beginning of the angular range occupied by the arc.

Degrees is given in units of degrees measured ***counter-clockwise from the 3-o'clock position***; it may be either positive or negative.



# Geometry Managers

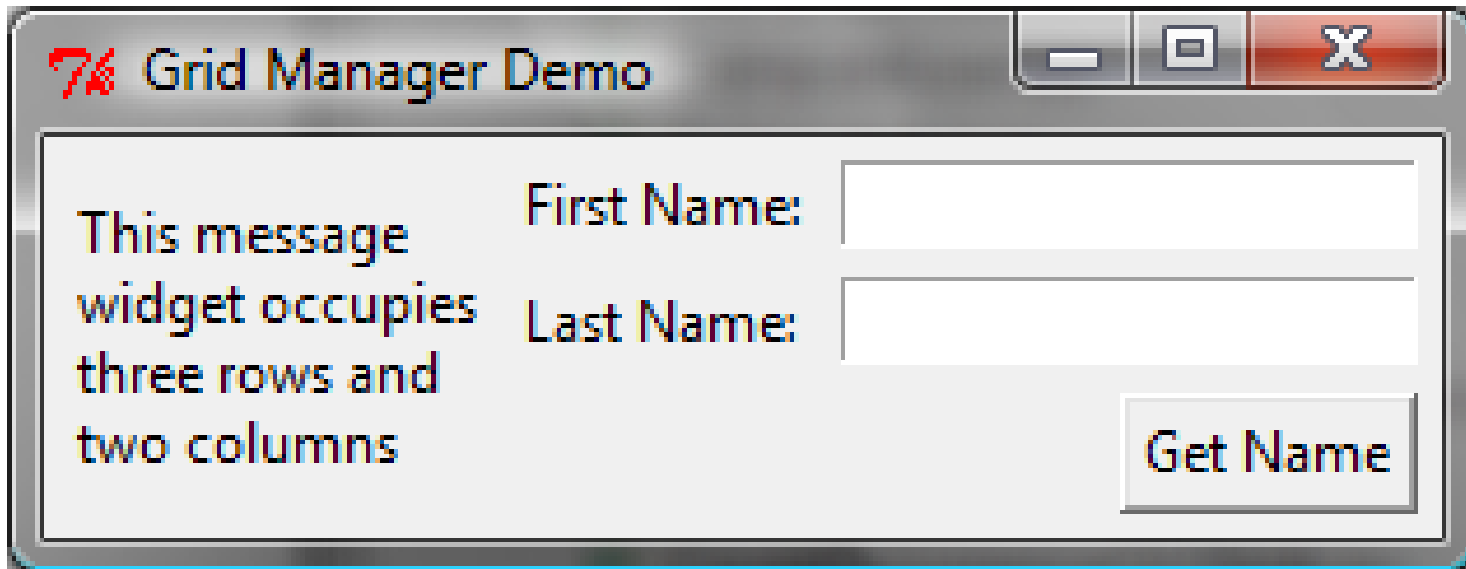
Grid Manager

Pack Manager

Place Manager

Since each manager has its own style of placing the widget, it is not a good practice to mix the managers for the widgets in the same container. You can use a frame as a subcontainer to achieve desired layout.

# Grid Managers

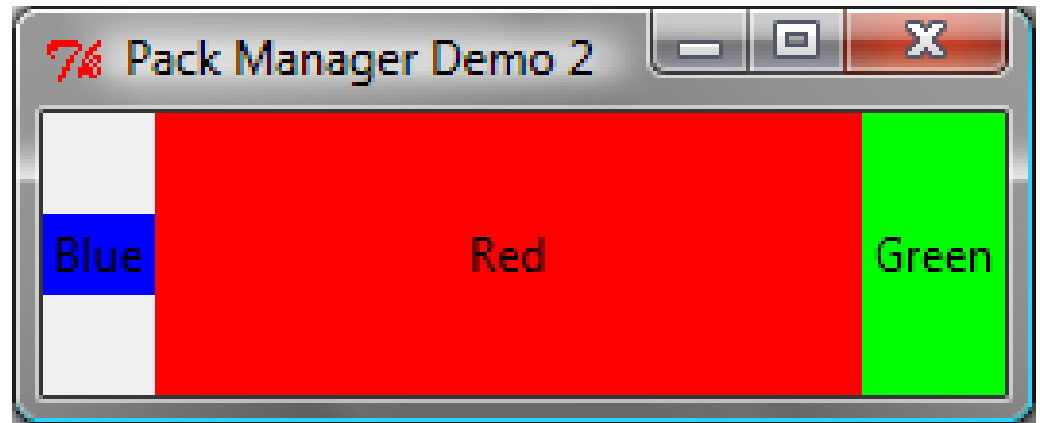
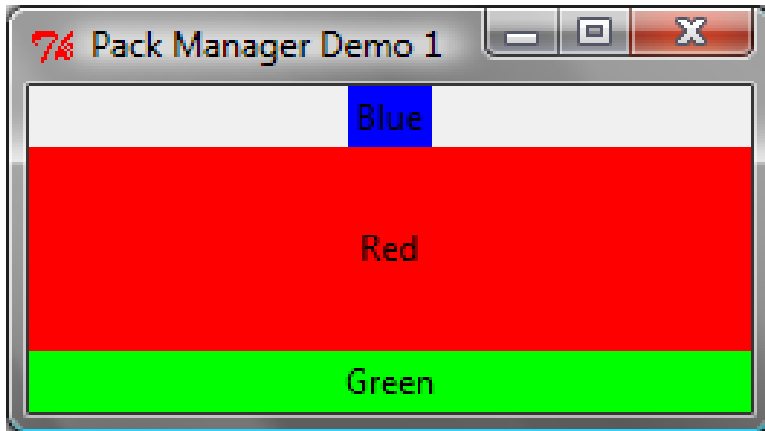


The screenshot shows a Java Swing window titled "Grid Manager Demo". The window contains a grid layout with the following components:

- A text label: "This message widget occupies three rows and two columns".
- A label: "First Name:" followed by a text input field.
- A label: "Last Name:" followed by a text input field.
- A button: "Get Name".

GridManagerDemo

# Pack Managers

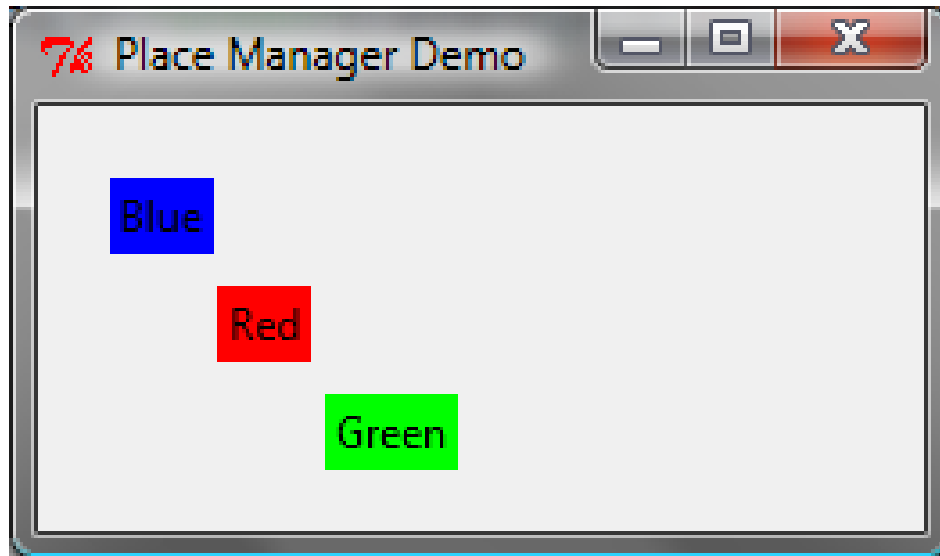


PackManagerDemo1

PackManagerDemo2



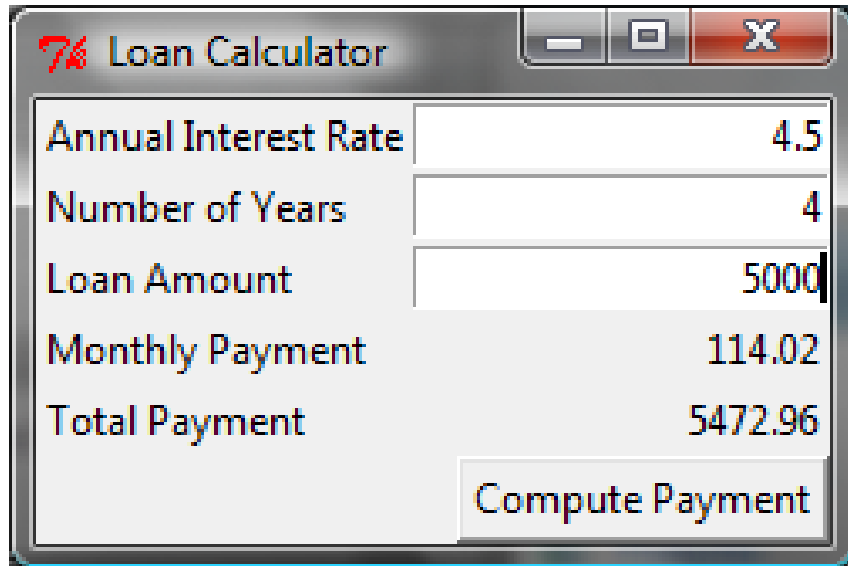
# Place Managers



PlaceManagerDemo



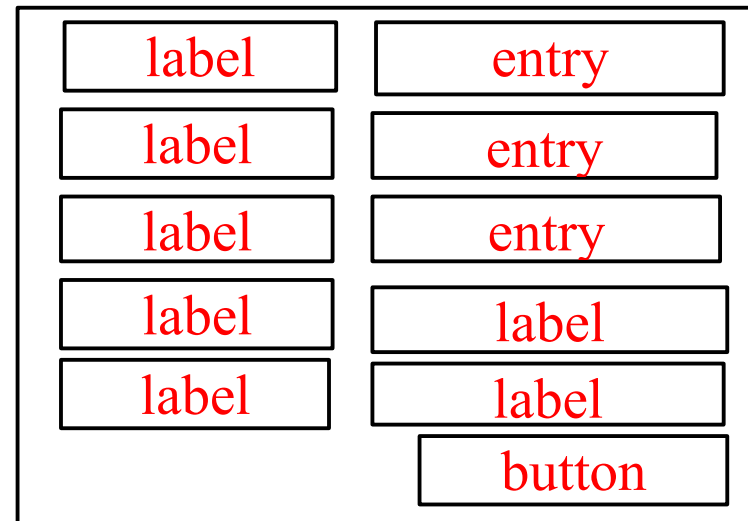
# Case Study: Loan Calculator



7% Loan Calculator

Annual Interest Rate	4.5
Number of Years	4
Loan Amount	5000
Monthly Payment	114.02
Total Payment	5472.96

Compute Payment



```
graph TD; L1[label] --- E1[entry]; L2[label] --- E2[entry]; L3[label] --- E3[entry]; L4[label] --- L5[label]; L5 --- L6[label]; L6 --- B1[button];
```

LoanCalculator



# Display Images

You can add an image in a label, button, check button, and radio button. To create an image, use the PhotoImage class as follows:

```
photo = PhotoImage(file = imagefilename)
```

The image file must be GIF or PNG. You can use a conversion utility to convert image files in other format into GIF or PNG.



# Image Example



ImageDemo

# Deck of Cards GUI



DeckOfCardsGUI