



Object Oriented Programming

BE(CSE) II-Semester

Prepared by
S. Durga Devi,
Assistant Professor,
CSE,CBIT

S. Durga Devi ,CSE,CBIT



Unit-V

GUI Programming

- Simple GUI Programming with tkinter package
- Sample Graphics using Turtle
- Plotting Graphs in Python.



GUI-Graphical user interfaces

- GUI is a visual components on the computer acts as interface between user and computer.
- GUI conveys the information, represents actions taken by the user.
- Example

<https://www.jotform.com/form-templates/music-school-registration-form>



List of GUI packages or platforms for creating GUI in python

1. Tkinter
2. PyQt
3. WxPython
4. Web (Remi).
5. Turtle.

Tkinter is popular and standard library for python as it provides easy and fast way to create GUI applications.



How do you create GUI in python?

- Tkinter: stands for toolkit interface.
- Tkinter is one of the package provided by python used to create GUI applications.
- Tool kit means set of API for creating the GUI components.
- Tkinter provides several libraries to create GUI elements like button, menus, labels, textbox etc. are called widgets.
- Widgets are used to take input from the user and displays information to the user.
- By using widgets we can create GUI application.



Widget examples



Buttons



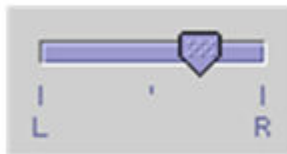
Combo Box



List



TextField



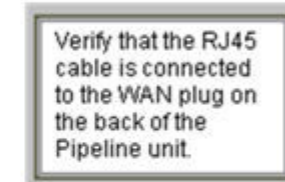
Slider



Menu



Label



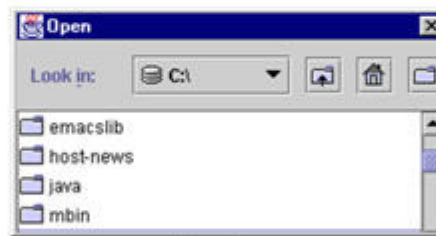
Text Area



Tool Tip



Progress Bar



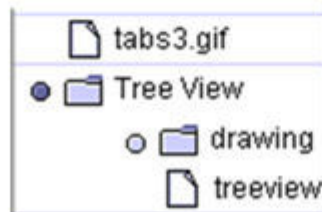
File Chooser



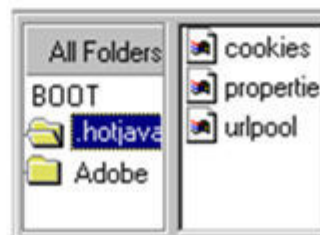
Color Chooser

First Na...	Last Name
Mark	Andrews
Tom	Ball
Alan	Chung
Jeff	Dinkins

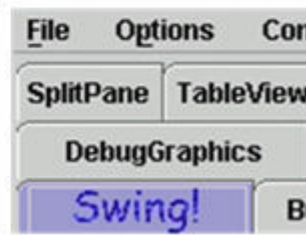
Table



Tree



Split Pane



Tabbed Pane



How do you use tkinter module?

- To create widgets a top level window is to be created by using following steps.
 1. Import tkinter module.

```
from tkinter import *
```

2. Create a window




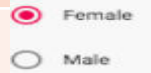
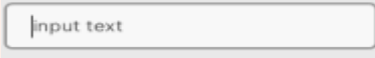
```
window=Tk()
```

3. Add widgets to window

4. Call main event loop for the event handling(take action against event triggered by the user)



Tkinter widget classes

Widget	description
Button	Creates buttons 
Label	Displays text or images 
Labelframe	Displays the border and title
Frame	As container holds other widgets
Checkbutton	Checkboxes 
Radiobutton	Radio buttons 
Canvas	To draw the canvas on window
Entry	Single line text field 
Listbox	Displays the list of options
Menu	Creates menu bar
Menubutton	Displays menu items
Message	Displays message box to the user
Scale	Creates slider
Scrollbar	Creates scrollbar to move window up and down

For more widgets go to https://www.tutorialspoint.com/python/python_gui_programming.htm

S. Durga Devi ,CSE,CBIT

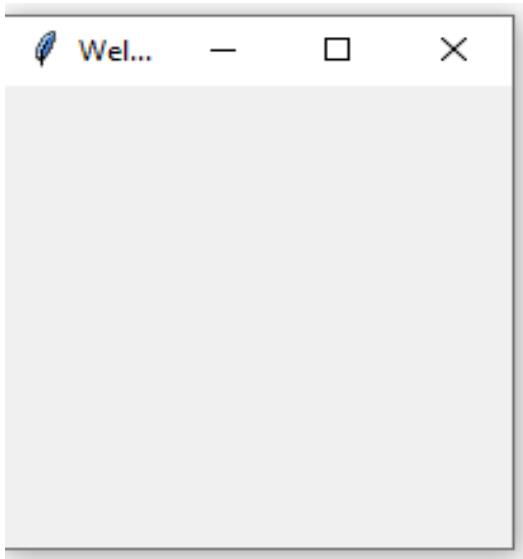


Example

- Create a Tkinter window with title

```
from tkinter import *  
window=Tk()  
window.title("Welcome to happy learning")  
window.mainloop()
```

TkEx1.py





Find the screen size

TxScreenSize.py

```
from tkinter import *  
window=Tk()  
print(window.winfo_screenwidth())    1366  
print(window.winfo_screenheight())   768  
|
```



practice

- How to make window full screen?
- How to resize window at specified location?

Refer text book.



Tkinter geometry

- Tkinter geometry provides methods to organize widgets on the parent widget area.
- Following are geometry methods
 1. `pack()`
 2. `place()`
 3. `grid()`



1. pack() method

- pack() method organizes widgets in the block.
- Syntax: **widgetname.pack(option)**
- Options are
 1. **expand:** If the expand is set to true, the widget expands to fill any space.
 2. **fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
 3. **side:** Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT

https://www.tutorialspoint.com/python/tk_pack.htm



Example on pack() method

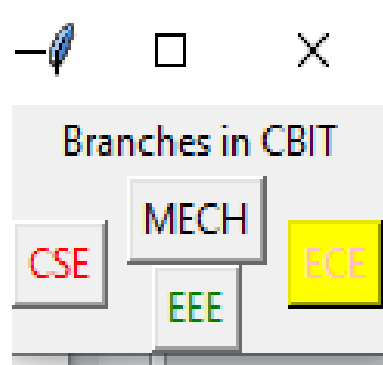
```
from tkinter import *  
window=Tk()  
l1=Label(text="student registration form")  
b1=Button(text="Login")  
b2=Button(text="cancel")  
l1.pack(side=BOTTOM)  
b1.pack(fill='y')  
b2.pack(expand='True')  
window.mainloop()
```

```
from tkinter import *  
root = Tk()  
listbox = Listbox(root)  
listbox.pack()  
for i in range(20):  
    listbox.insert(END, str(i))  
root.mainloop()
```

```
from tkinter import *  
root = Tk()  
listbox = Listbox(root)  
listbox.pack(fill=BOTH,expand=1)  
for i in range(20):  
    listbox.insert(END, str(i))  
root.mainloop()
```



- Create following gui application.
- Create a label as branches in cbit
- Create 4 branches as buttons and arrange them in all four corners



- Solution : see next slide



packEx.py

```
from tkinter import *
parent = Tk()
label=Label(text="Branches in CBIT")
label.pack(side=TOP)
cse = Button(text= "CSE", fg = "red")
cse.pack( side = LEFT)
ece = Button(text= "ECE", fg = "pink",bg="yellow")
ece.pack( side = RIGHT )
mech = Button(text= "MECH", fg = "black")
mech.pack( side = TOP )
eee= Button(text= "EEE", fg = "green")
eee.pack(side = BOTTOM)
parent.mainloop()
```




2. grid() method

- grid() method organizes the widgets in the tabular form.
- Syntax: widgetname.grid(option)
- List of options

columns	Column no in which widget is to be placed. Default 0 (leftmost column)
columnspan	No of columns widget can occupy, default is 1
row	Row no in which widget can be placed . Topmost row is default 0
rowspan	no of rows widget can occupy, default is 1
sticky	if cell is larger than the widget size, sticky is used to specify the position to align inside the cell. It may be the string concatenation of zero or more of N, E, S, W, NE, NW, SE, and SW

For more options

<https://www.javatpoint.com/python-tkinter>

S. Durga Devi ,CSE,CBIT



Create following login form

loginFormGridEx.py

```
from tkinter import *
# create a window
top=Tk()
top.title("LoginForm")
#create label name
name=Label(text="name").grid(row=0,column=0,sticky=SW)
#create textbox
t1=Entry().grid(row=0,column=1)
#create label password
password=Label(text="password").grid(row=1, column=0)
#create textbox
t2=Entry().grid(row=1,column=1)
#create submit button
submit=Button(text="Submit").grid(row=2, column=0)
#create cancel button
cancel= Button(text="Canel").grid(row=2, column=1)
```



3.place() method

- place() method is used to add the widgets at the specified position on the parent window.
- Syntax: widgetname.place(options)
- List of options

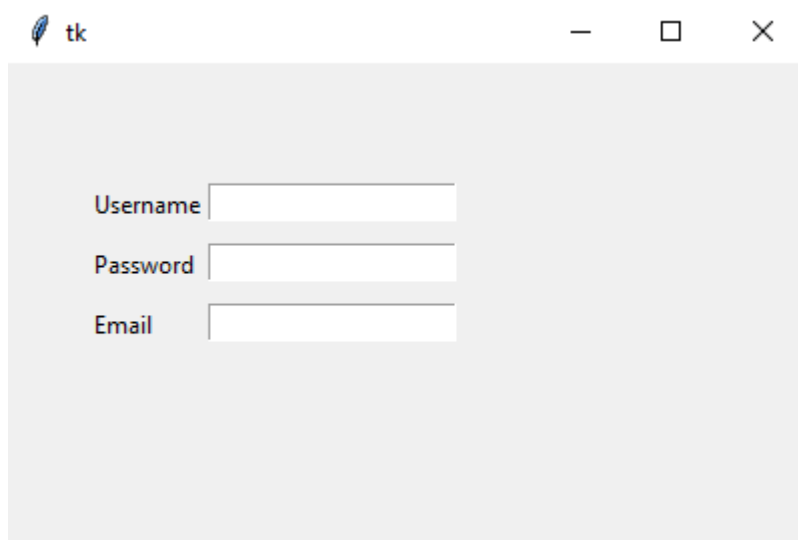
anchor	Exact position of the widget within the container. Default is upper leftmost corner(NW). N, E, S, W, NE, NW, SE, or SW
bordermode	Default border type is INSIDE. Ignore the parents inside the border.other option is OUTSIDE
height, width	Height and width in pixels
x,y	Horizontal and vertical offset in pixels
relx,rely	float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction
relheight, relwidth	float between 0.0 and 1.0 that is fraction of parents height and width.



Example on place() method

- Create the following GUI application

Place()Ex.py



```
from tkinter import *
parent=Tk()
# define size of window
parent.geometry("400x400")
name=Label(text="Username").place(x=40,y=60)
password=Label(text="Password").place(x=40,y=90)
email=Label(text="Email").place(x=40,y=120)
nm=Entry().place(x=100,y=60)
password=Entry().place(x=100,y=90)
e=Entry().place(x=100,y=120)
parent.mainloop()
```



bordermode

- bordermode=OUTSIDE means counts x and y pixels includes the borders
- bordermode=INSIDE means counts x and y pixels without borders.

```
from tkinter import *  
  
root = Tk()  
f1 = Frame(root, borderwidth=20, relief=SUNKEN, width=50, height=50)  
f1.pack()  
l1 = Label(f1, text="Hi").place(x=10, y=10, bordermode="outside")  
  
f2 = Frame(root, borderwidth=20, relief=SUNKEN, width=50, height=50)  
f2.pack()  
l2 = Label(f2, text="Hi").place(x=10, y=10, bordermode="inside")  
  
root.mainloop()
```





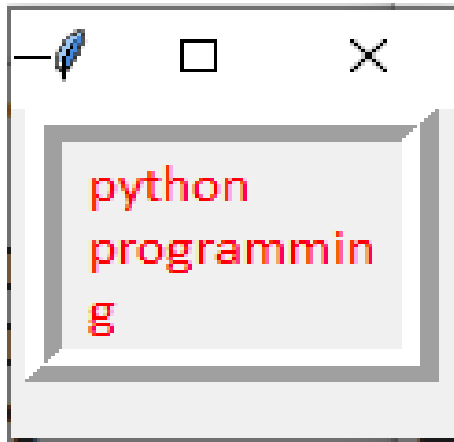
Create Message box

MessageEx.py

```
from tkinter import *

top = Tk()
top.geometry("100x100")
msg = Message( top, text = "python programming", fg='red', font='Calibri',
               |relief= RIDGE,borderwidth=10)

msg.pack()
top.mainloop()
```



relief- represents type of the borders
possible values are RIDGE,FLAT,SUNKEN,
RAISED,GROOVE. Default is FLAT



Label(parent,options)

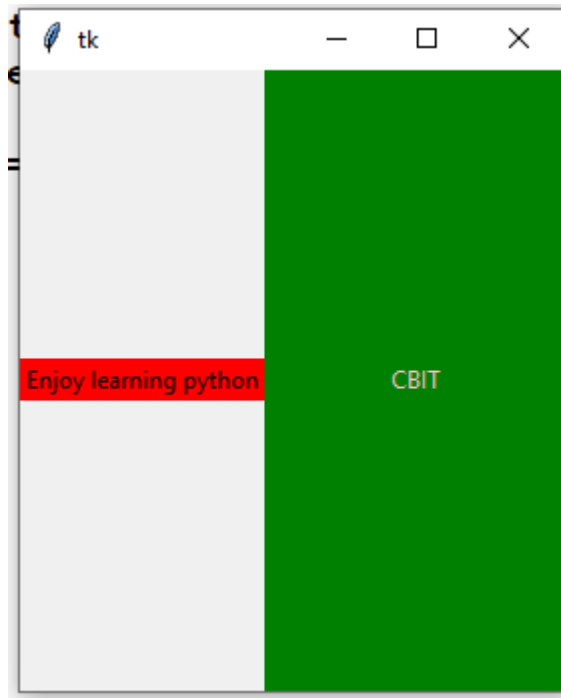
- Label() is used to create the text or image.
- syntax Label(parent, options)
- list of options

bg	Background color of the widget displays behind the widget.
bd	Width of the border default is 2
cursor	Mouse pointer changed to the arrow, dot etc.
font	Type of the font applied to the text
text	label name
fg	foreground color of the text written inside widget
image	Image is to be shown as a label
height, width	Height and width of the widget
padx	horizontally padding of the text . Default is 1
pady	vertical padding of the text. Default is 1



Write a program to create labels with different background colors

```
from tkinter import *
parent=Tk()
label1=Label(text="Enjoy learning python",bg="red").pack(side=LEFT)
label2=Label(text="CBIT",bg="green",bd=4,fg="pink",
             width=20,height=20).pack(side=RIGHT)
```



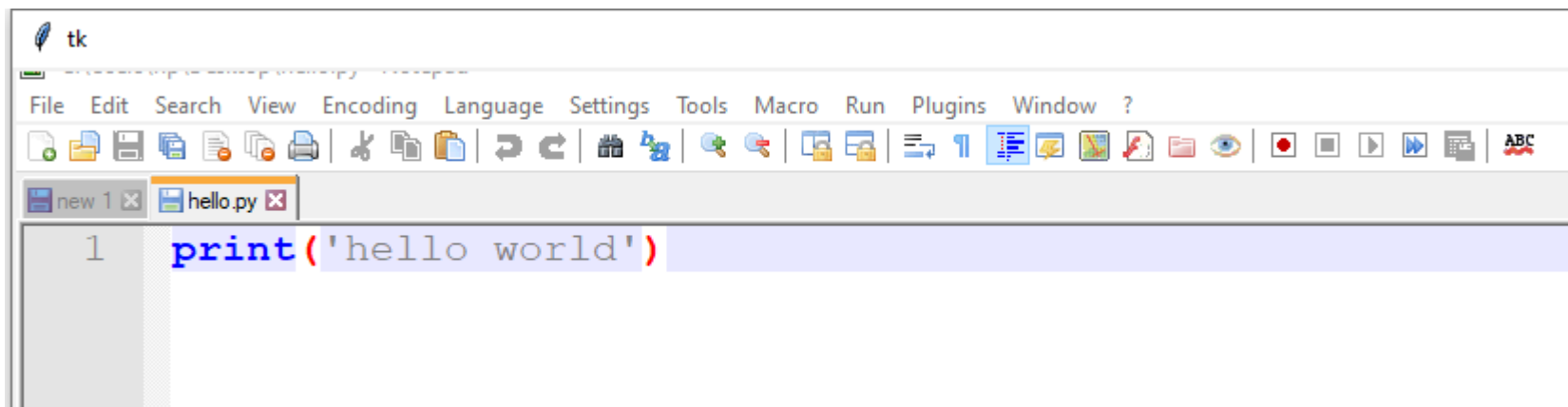
LabelEx.py



- Program to display image as label

ImageEx.py

```
from tkinter import *  
root=Tk()  
img=PhotoImage(file="textprogram.png")  
l=Label(root,image=img)  
l.pack()
```





Event handling

Eventhandling.py

```
from tkinter import *
from tkinter import messagebox
parent = Tk()
def test():
    messagebox.showinfo("messagebox","you selected CSE")
label=Label(text="Branches in CBIT")
label.pack(side=TOP)
cse = Button(text = "CSE", fg = "red",command=test)
cse.pack( side = LEFT)
ece = Button(text = "ECE", fg = "pink"|)
ece.pack( side = RIGHT )
mech = Button(text = "MECH", fg = "black")
mech.pack( side = TOP )
eee= Button(text = "EEE", fg = "green")
eee.pack(side = BOTTOM)
```



Create menu and menubars

- to create a menu

`menubar= Menu(parent)`

- To create sub menus

`file=Menu(menubar,options)`

List of options

1. `tearoff`: menu items are added at starting position 0 if `tearoff=0`
2. `activebackground`: when widget is focussed its background color will be changed.
3. `activeforeground`: changes font color of the widget



Methods in Menu

1. `add_command(options)`: add items to the menu
`file.add_command(label="Open",activebackground='red')`
2. `add_radiobutton(options)`: adds radio buttons
3. `add_checkbutton(options)`: adds check buttons
4. `add_cascade(options)`: adds menu to the parent menu.
5. `add_seperator()`: adds separator to the menu item.



- Write a program to create Menu card for the CBIT Restaurant.
- Create a Registration form of the gmail account.



Event handling

- GUI components or widgets are static in nature. To provide dynamic nature a special interface needed is called event.
- *What is an Event?*
 - an action is to be performed on a component to provide dynamic nature.
- *How Events are generated?*
 - when the user interact with GUI components events are generated.
- For Example,
 - Click on a Button
 - Moving mouse
 - Enter a character through keyboard
 - Select an item from a list
 - Scrolling the page.



- *Event handling* is a mechanism to handle the events when they are generated.
- This mechanism has a code to execute when the events are occurred is called event handler.
- Tkinter provides the mechanism to handle the events.
- Events are handled by binding them to predefined or user defined functions which allow us to run some piece of code when event take place on widget.
- To bind the event with a function we must call method name `bind()` with following Syntax:

```
Widgetname.bind(event,handlerfunction)
```

- When event occurred on widget the handler function is called with an event object



Widgetname.bind(event,handlerfunction)

- - event: is a string that contains details about which event to listen for, this must be given in the following format:
"<modifier-type-detail>". The required part in this string is only type of the event.
- - handlerfunction: this function is called when event occurred within the widget. Only name of the function we have to mention. This function takes event object as a parameter.

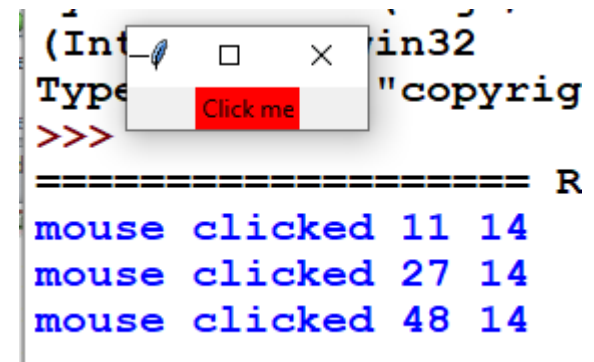


Example on mouse event handling

- This example illustrates an event handler to a mouse click event on a label that will print a message when activated with the coordinates.

Event1.py

```
from tkinter import *  
window = Tk()  
def mouseClick( event ):  
    print( "mouse clicked",event.x,event.y )  
label = Label( window, text="Click me",bg='red' )  
label.pack()  
label.bind( "<Button>", mouseClick )  
window.mainloop()  
# for output click only on the label
```



When u click on label with mouse button either it is left button or right button, it called the method mouseClick()



List of event types in tkinter

Event type	description
<Button>	Activated when mouse button has been clicked. The detail part specifies the which mouse button clicked. Ex: <Button-1> event defined for the left button clicked. <Button-2>: Middle Button clicked. <Button-3>: right Button clicked
<Motion>	Activated when the mouse cursor moves across the designated widget.
<Enter>	When mouse cursor enters the designated widget
<Leave>	When mouse cursor leaves the designated widget
<KeyPress>	When keyboard key is pressed
<KeyRelease>	When keyboard key is released
<Double-Button>	When double clicked. To specify the left, middle or right mouse button use <Double-Button-1>, <Double-Button-2>, and <Double-Button-3> respectively.

For more details: https://www.python-course.eu/tkinter_events_binds.php



Mouse double clicked

Event2.py

```
from tkinter import *
window = Tk()
def mouseclicked(event):
    print("mouse is clicked at",event.x,event.y)
def mousedoubleclick(event):
    print("mouse is double clicked at",event.x,event.y)
label = Label( window, text="Click me",bg='red' )
label.pack()
label.bind("<Button>",mouseclicked)
label.bind("<Double-Button-1>",mousedoubleclick)
window.mainloop()
# for output click only on the label
```



mousemoved

Event3.py

```
from tkinter import *
window = Tk()
def mousedoubleclick(event):
    print("mouse is double clicked at",event.x,event.y)
def moved(event):
    print("mouse moved at ",event.x,event.y)
label = Label( window, text="Click me",bg='red' )
label.pack()
label.bind("<Double-Button-1>",mousedoubleclick)
label.bind("<Motion>",moved)
window.mainloop()
# for output click only on the label
```



Canvas in tkinter

- Canvas widget allows to create graphics like lines, circles, images and even other widgets.
- Creates customized widgets, plot graphs , graphics editors etc.
- Syntax:
 - `objectname=Canvas(parent,options)`
- List of options
 - 1. bd: borderwidth default is 2
 - 2. bg: background color of the canvas
 - 3. height: height of canvas in Y dimensions
 - 4. width: width of canvas in X dimensions
 - 5. relief: type of border : SUNKEN,RAISED,FLAT,GROOVE,RIDGE



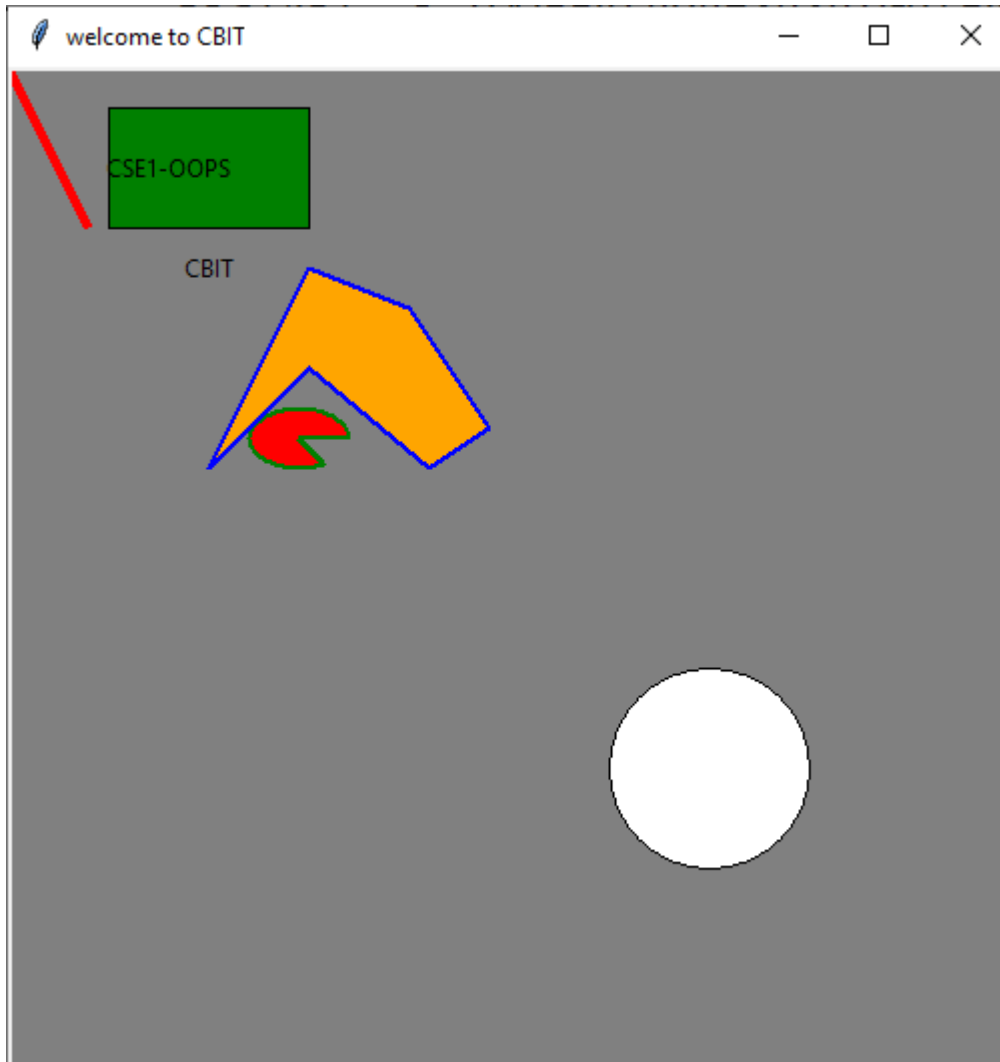
- Canvas widget supports the following graphics
 1. Line
 2. Rectangle
 3. Oval(a circle or oval)
 4. Text
 5. Arc
 6. Image
 7. Polygon
 8. window



- 1. `c.create_line(x1,y1,x2,y2,options)`: draw the line
- 2. `c.create_rectangle(x1,y1,width,height,options)` : draw the rectangle
- 3. `c.create_text(x,y,text="CBIT")`
- 4. `c.create_oval(x1,y1,x2,y2,options)`.
- 5. `c.create_polygon(points,options)`



- Write a program to display the different graphics on canvas
- CanvasEx1.py



See next slide for solution

S. Durga Devi ,CSE,CBIT



```
from tkinter import *
root=Tk()
c=Canvas(root,width=500,height=500,bg='grey')
root.title("welcome to CBIT")
c.pack()
# draw a line
c.create_line(0, 0, 40, 80, fill="red",width=5,activefill="black")
#rectangle
c.create_rectangle(50, 20, 150, 80, fill="green")
# draw the text
c.create_text(80,50,text="CSE1-OOPS")
c.create_text(100,100,text="CBIT")
#draw a circle or eclipse
c.create_oval(300,300,400,400,fill="white")
# draw a arc of 300degrees
c.create_arc(120, 200, 170, 170, start = 0,
             extent = 300, outline = "green",
             fill = "red", width = 2)
# draw a polygon
points = [150, 100, 200, 120, 240, 180,
          210, 200, 150, 150, 100, 200]
c.create_polygon(points,outline = "blue",fill = "orange", width = 2)

root.mainloop()
```



write an application to create a painting or writing on the canvas window.

```
from tkinter import *
```

```
canvas_width = 500
```

```
canvas_height = 150
```

```
def paint( event ):
```

```
    python_green = "#476042"
```

```
    x1, y1 = ( event.x - 1 ), ( event.y - 1 )
```

```
    x2, y2 = ( event.x + 1 ), ( event.y + 1 )
```

```
    w.create_oval( x1, y1, x2, y2, fill = python_green )
```

```
master = Tk()
```

```
master.title( "Painting using Ovals" )
```

```
w = Canvas(master,
```

```
    width=canvas_width,
```

```
    height=canvas_height)
```

```
w.pack(expand = YES, fill = BOTH)
```

```
w.bind( "<B1-Motion>", paint )
```

```
message = Label( master, text = "Press and Drag the mouse to draw" )
```

```
message.pack( side = BOTTOM)
```

```
mainloop()
```

CanvasPaint.py



references

- <https://docs.python.org/3/library/tkinter.html#>
- <https://realpython.com/python-gui-tkinter/>
- https://www.tutorialspoint.com/python/python_gui_programming.htm
- <https://www.javatpoint.com/python-tkinter>
- https://www.python-course.eu/tkinter_mastermind.php
- https://www.python-course.eu/tkinter_events_binds.php
- <https://instructobit.com/tutorial/51/Python-Tkinter-event-handling>