Graphical User Interfaces
a sliding puzzle characteristics and components

The Tk GUI Toolkit in Python
Tkinter, Tk, and Tcl
hello world again using grid, Radiobutton, and Checkbutton handling mouse events

Summary + Assignments
graphical user interfaces using Tkinter

1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
A Sliding Puzzle
Elements of the GUI

The GUI for the sliding puzzle contains:

1. buttons to start, scramble, unscramble, and clear
2. a canvas widget to display the sliding puzzle
3. a numpy array stores the data of the puzzle
4. the scramble/unscramble buttons are animations
5. mouse events allow the user to solve the puzzle

Functionality:

1. the start button of the GUI shows an ordered board
2. the scramble button perturbs the current board
3. pushing unscramble shows how to solve the puzzle
4. the user can move rectangles by clicking on a rectangle that can slide to the free position

Object-oriented programming is used to implement GUIs.
Elements of the GUI

The GUI for the sliding puzzle contains:

1. buttons to start, scramble, unscramble, and clear
2. a canvas widget to display the sliding puzzle
3. a numpy array stores the data of the puzzle
4. the scramble/unscramble buttons are animations
5. mouse events allow the user to solve the puzzle

Functionality:

1. the start button of the GUI shows an ordered board
2. the scramble button perturbs the current board
3. pushing unscramble shows how to solve the puzzle
4. the user can move rectangles by clicking on a rectangle that can slide to the free position

Object-oriented programming is used to implement GUls.
1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly *event driven* and give the user control over the actions.

Command line interfaces are call non-GUI.

Specific categories of UI are

- **WUI**: *Web UI* and
- **HUI**: *Handheld UI*.

Goal: increase *usability* of software.
User Interfaces (UI)
GUI, WUI, and HUI

Interfaces to our programs so far were command line.

Consequences:
1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly event driven and give the user control over the actions.

Command line interfaces are call non-GUI.
Specific categories of UI are
WUI: Web UI and HUI: Handheld UI.

Goal: increase usability of software.
Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly event driven and give the user control over the actions.

Command line interfaces are call non-GUI. Specific categories of UI are WUI: Web UI and HUI: Handheld UI.

Goal: increase usability of software.
Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly *event driven* and give the user control over the actions.

Command line interfaces are call non-GUI.
Specific categories of UI are WUI: *Web UI* and HUI: *Handheld UI*.

Goal: increase *usability* of software.
Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly \textit{event driven} and give the user control over the actions.

Command line interfaces are call non-GUI. Specific categories of UI are WUI: \textit{Web UI} and HUI: \textit{Handheld UI}.

Goal: increase \textit{usability} of software.
User Interfaces (UI)  
GUI, WUI, and HUI

Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly event driven and give the user control over the actions.

Command line interfaces are call non-GUI. Specific categories of UI are WUI: Web UI and HUI: Handheld UI.

Goal: increase usability of software.
Characteristics of GUIs

The most significant features of a GUI are

**Window:** area of display device to view and interact with an object. Information is *viewed*.

**Icon:** pictorial representation of an object.

**Menu:** collection of choices, typically to perform actions on an object.

**Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area:** area within a window where the user can enter data, strings or numbers.

**Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*.

Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUls

The most significant features of a GUI are

**Window**: area of display device to view and interact with an object. Information is viewed.

**Icon**: pictorial representation of an object.

**Menu**: collection of choices, typically to perform actions on an object.

**Pointer**: is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area**: area within a window where the user can enter data, strings or numbers.

**Directness**: direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUls should be *user centered*. Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUIs

The most significant features of a GUI are

**Window:** area of display device to view and interact with an object. Information is *viewed*.

**Icon:** pictorial representation of an object.

**Menu:** collection of choices, typically to perform actions on an object.

**Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area:** area within a window where the user can enter data, strings or numbers.

**Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*. Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUIs

The most significant features of a GUI are

**Window:** area of display device to view and interact with an object. Information is *viewed*.

**Icon:** pictorial representation of an object.

**Menu:** collection of choices, typically to perform actions on an object.

**Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area:** area within a window where the user can enter data, strings or numbers.

**Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*. Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUIs

The most significant features of a GUI are

**Window:** area of display device to view and interact with an object. Information is *viewed*.

**Icon:** pictorial representation of an object.

**Menu:** collection of choices, typically to perform actions on an object.

**Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area:** area within a window where the user can enter data, strings or numbers.

**Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*. Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUIs

The most significant features of a GUI are

**Window:** area of display device to view and interact with an object. Information is *viewed*.

**Icon:** pictorial representation of an object.

**Menu:** collection of choices, typically to perform actions on an object.

**Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area:** area within a window where the user can enter data, strings or numbers.

**Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*.

Goals: simple, aesthetic, productive, and customizable.
Characteristics of GUIs

The most significant features of a GUI are

**Window**: area of display device to view and interact with an object. Information is *viewed*.

**Icon**: pictorial representation of an object.

**Menu**: collection of choices, typically to perform actions on an object.

**Pointer**: is similar to a typing cursor, controlled by a pointing device, typically a mouse.

**Client Area**: area within a window where the user can enter data, strings or numbers.

**Directness**: direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*. Goals: simple, aesthetic, productive, and customizable.
GUI components are *widgets* = window gadgets.

Some GUI components are

- **Button**: to trigger an event
- **Label**: to display text or icons
- **Entry**: to accept single line user input
- **Text**: to accept multiple line user input
- **Menu**: to display list of items
- **Listbox**: to display list of text options
- **Canvas**: to draw geometric figures
GUI components

GUI components are *widgets* = window gadgets. Some GUI components are

**Button**: to trigger an event

**Label**: to display text or icons

**Entry**: to accept single line user input

**Text**: to accept multiple line user input

**Menu**: to display list of items

**Listbox**: to display list of text options

**Canvas**: to draw geometric figures
GUI components are *widgets* = window gadgets. Some GUI components are

- **Button**: to trigger an event
- **Label**: to display text or icons
- **Entry**: to accept single line user input
- **Text**: to accept multiple line user input
- **Menu**: to display list of items
- **Listbox**: to display list of text options
- **Canvas**: to draw geometric figures
GUI components are *widgets* = window gadgets. Some GUI components are

**Button**: to trigger an event

**Label**: to display text or icons

**Entry**: to accept single line user input

**Text**: to accept multiple line user input

**Menu**: to display list of items

**Listbox**: to display list of text options

**Canvas**: to draw geometric figures
GUI components are *widgets* = window gadgets. Some GUI components are

**Button:** to trigger an event  
**Label:** to display text or icons  
**Entry:** to accept single line user input  
**Text:** to accept multiple line user input  
**Menu:** to display list of items  
**Listbox:** to display list of text options  
**Canvas:** to draw geometric figures
GUI components are *widgets* = window gadgets. Some GUI components are

- **Button**: to trigger an event
- **Label**: to display text or icons
- **Entry**: to accept single line user input
- **Text**: to accept multiple line user input
- **Menu**: to display list of items
- **Listbox**: to display list of text options
- **Canvas**: to draw geometric figures
GUI components are \emph{widgets} = window gadgets. Some GUI components are

- **Button**: to trigger an event
- **Label**: to display text or icons
- **Entry**: to accept single line user input
- **Text**: to accept multiple line user input
- **Menu**: to display list of items
- **Listbox**: to display list of text options
- **Canvas**: to draw geometric figures
GUI components

GUI components are *widgets* = window gadgets. Some GUI components are

- **Button**: to trigger an event
- **Label**: to display text or icons
- **Entry**: to accept single line user input
- **Text**: to accept multiple line user input
- **Menu**: to display list of items
- **Listbox**: to display list of text options
- **Canvas**: to draw geometric figures
graphical user interfaces using Tkinter

1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
The Tkinter (= Tk interface) library provides an object-oriented interface to the Tk GUI toolkit, the graphical interface development tool for Tcl, Tk = Tool Kit, Tcl = Tool Command Language.

Benefit: platform independent GUI development.
Tkinter, Tk, and Tcl
GUIs for Python programmers

The Tkinter (= Tk interface) library provides an object-oriented interface to the Tk GUI toolkit, the graphical interface development tool for Tcl, Tk = Tool Kit, Tcl = Tool Command Language.

Benefit: platform independent GUI development.
Graphical user interfaces using Tkinter

1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
Hello World!

our first GUI

Five steps:

1. from Tkinter import *
2. create a new object of the class Tk
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.
Five steps:

1. from Tkinter import *
2. create a new object of the class Tk
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.
Hello World!
our first GUI

Five steps:

1. from Tkinter import *
2. create a new object of the class Tk
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.
Hello World!
our first GUI

Five steps:

1. from Tkinter import *
2. create a new object of the class Tk
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.
Hello World!
our first GUI

Five steps:

1. `from Tkinter import *`
2. create a new object of the class `Tk`
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.
The Code for Hello World!
in the file guihello.py

```python
# L-30 MCS 260 Mon 3 Nov 2008 : guihello.py

# Hello world with a Graphical User Interface.
# The code below displays "Hello World!" in a
# new window, using the Tkinter GUI library.

from Tkinter import *

top = Tk()  # top is the new window
# Label is a widget to design the interface
Lbl = Label(top,text="Hello World!",
           width=20,height=5)
# to arrange the widget in a window we call
Lbl.pack()  # the geometry manager
top.mainloop() # enter main event loop
```
The Code for Hello World!

in the file guihello.py

# L-30 MCS 260 Mon 3 Nov 2008 : guihello.py

# Hello world with a Graphical User Interface.  
# The code below displays "Hello World!" in a  
# new window, using the Tkinter GUI library.

from Tkinter import *  

top = Tk()        # top is the new window  

# Label is a widget to design the interface  
Lbl = Label(top,text="Hello World!",
            width=20,height=5)  

# to arrange the widget in a window we call  
Lbl.pack()        # the geometry manager  
top.mainloop()    # enter main event loop
The Code for Hello World!
in the file guihello.py

from Tkinter import *  
top = Tk()    # top is the new window  
Lbl = Label(top,text="Hello World!",
    width=20,height=5)

# to arrange the widget in a window we call
Lbl.pack()    # the geometry manager

top.mainloop()  # enter main event loop
# Hello world with a Graphical User Interface.
# The code below displays "Hello World!" in a new window, using the Tkinter GUI library.

from Tkinter import *
top = Tk()  # top is the new window
# Label is a widget to design the interface
Lbl = Label(top,text="Hello World!",
         width=20,height=5)
# to arrange the widget in a window we call
Lbl.pack()  # the geometry manager
top.mainloop()  # enter main event loop
The Code for Hello World!
in the file guihello.py

```python
# L-30 MCS 260 Mon 3 Nov 2008 : guihello.py

# Hello world with a Graphical User Interface.
# The code below displays "Hello World!" in a
# new window, using the Tkinter GUI library.

from Tkinter import *
top = Tk()         # top is the new window
# Label is a widget to design the interface
Lbl = Label(top, text="Hello World!",
           width=20, height=5)
# to arrange the widget in a window we call
Lbl.pack()        # the geometry manager
top.mainloop()    # enter main event loop
```
Dialogue with User

prompting for a name

We prompt the user to enter a name:

After the user typed in a name:

Pushing the enter button pops up the window:
Dialogue with User

prompting for a name

We prompt the user to enter a name:

After the user typed in a name:

Pushing the enter button pops up the window:
Dialogue with User
prompting for a name

We prompt the user to enter a name:

After the user typed in a name:

Pushing the enter button pops up the window:
The Widget Entry

to accept user input

from Tkinter import *
top = Tk()
Label(top, text="Who's there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0, column=1)
top.mainloop()

Observe:

1. use of row and column with grid
2. the widget Entry to accept user strings
The Widget Entry

to accept user input

```
from Tkinter import *
top = Tk()
Label(top,text="Who’s there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0,column=1)
top.mainloop()
```

Observe:

1. use of row and column with grid
2. the widget Entry to accept user strings
The Widget Entry to accept user input

from Tkinter import *
top = Tk()
Label(top,text="Who’s there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0,column=1)
top.mainloop()

Observe:

① use of row and column with grid
② the widget Entry to accept user strings
The Widget Entry

to accept user input

```python
from Tkinter import *
top = Tk()
Label(top,text="Who’s there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0,column=1)
top.mainloop()
```

Observe:

1. use of row and column with grid
2. the widget Entry to accept user strings
**The Widget Button**

to enter user input

```python
import tkMessageBox
def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)
b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()
```

1. Button will call hello when pressed
2. hello calls showinfo of tkMessageBox
import tkMessageBox

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)

b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()

1 Button will call hello when pressed
2 hello calls showinfo of tkMessageBox
import tkMessageBox

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)

b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()

1. Button will call hello when pressed
2. hello calls showinfo of tkMessageBox
import tkMessageBox

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter", s)

b = Button(top, text="enter", command=hello)
b.grid(row=1, column=1)
top.mainloop()

1. Button will call hello when pressed
2. hello calls showinfo of tkMessageBox
import tkMessageBox

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)

b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()
import tkMessageBox
def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)
b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()

1. Button will call hello when pressed  
2. hello calls showinfo of tkMessageBox
The Widget Entry and Button

in the file guihello2.py

from Tkinter import *
import tkMessageBox

top = Tk()
Label(top,text="Who's there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0,column=1)

def hello():
    "opens a window to say hello"
s = 'hello ' + e.get()
tkMessageBox.showinfo("enter",s)

b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()
from Tkinter import *
import tkMessageBox

top = Tk()
Label(top,text="Who's there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0,column=1)

def hello():
    "opens a window to say hello"
s = 'hello ' + e.get()
tkMessageBox.showinfo("enter",s)

b = Button(top,text="enter",command=hello)
b.grid(row=1,column=1)
top.mainloop()
from Tkinter import *
import tkMessageBox

top = Tk()
Label(top, text="Who's there ? ").grid(row=0)
e = Entry(top)
e.grid(row=0, column=1)

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
tkMessageBox.showinfo("enter",s)

b = Button(top, text="enter", command=hello)
b.grid(row=1, column=1)
top.mainloop()
The Widget Entry and Button

in the file guiHello2.py

```python
from Tkinter import *
import tkMessageBox

root = Tk()
Label(root,text="Who's there ? ").grid(row=0)
e = Entry(root)
e.grid(row=0,column=1)

def hello():
    "opens a window to say hello"
    s = 'hello ' + e.get()
    tkMessageBox.showinfo("enter",s)

b = Button(root,text="enter",command=hello)
b.grid(row=1,column=1)
root.mainloop()
```
Graphical User Interfaces

1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
the Geometry Manager grid

To layout the geometry like this

Observe:

1. the title of the window
2. buttons spanning multiple columns
3. an entry field containing text
from Tkinter import *

import Tkinter
import Tk
import Tcl

from Tkinter import *

top = Tk()
top.title("use of grid")

l1 = Label(top,text=" text 1 ")
l1.grid(row=0,column=4)

b0 = Button(top,text=" button 0 ")
b0.grid(row=0,column=1,sticky=W+E+N+S)

b1 = Button(top,text=" button 1 ")
b1.grid(row=1,column=1,columnspan=4,sticky=W+E+N+S)

e1 = Entry(top)
e1.insert(INSERT,"entry 1 ")
e1.grid(row=2,column=0,columnspan=2)

b2 = Button(top,text = "button 2 ")
b2.grid(row=2,column=3)

import Tkinter
import Tk
import Tcl

import Tkinter
import Tk
import Tcl

Using grid

the file usegrid.py
from Tkinter import *

top = Tk()
top.title("use of grid")
t1 = Label(top,text=" text 1 ")
t1.grid(row=0,column=4)

b0 = Button(top,text=" button 0 ")
b0.grid(row=0,column=1,sticky=W+E+N+S)

b1 = Button(top,text=" button 1 ")
b1.grid(row=1,column=1,columnspan=4,sticky=W+E+N+S)

e1 = Entry(top)
e1.insert(INSERT,"entry 1 ")
e1.grid(row=2,column=0,columnspan=2)

b2 = Button(top,text = "button 2 ")
b2.grid(row=2,column=3)
top.mainloop()
Graphical User Interfaces

The Tk GUI Toolkit in Python

Tkinter, Tk, and Tcl

hello world again

using grid, Radiobutton, and Checkbutton

handling mouse events

Summary + Assignments

Using grid

the file usegrid.py

from Tkinter import *
top = Tk()
top.title("use of grid")
t1 = Label(top,text=" text 1 ")
t1.grid(row=0,column=4)
b0 = Button(top,text=" button 0 ")
b0.grid(row=0,column=1,sticky=W+E+N+S)
b1 = Button(top,text=" button 1 ")
b1.grid(row=1,column=1,columnspan=4,sticky=W+E+N+S)
e1 = Entry(top)
e1.insert(INSERT,"entry 1 ")
e1.grid(row=2,column=0,columnspan=2)
b2 = Button(top,text = "button 2 ")
b2.grid(row=2,column=3)
top.mainloop()
from Tkinter import *

# import all Tkinter widgets

top = Tk()

top.title("use of grid")

t1 = Label(top,text=" text 1 ")

t1.grid(row=0,column=4)

b0 = Button(top,text=" button 0 ")

b0.grid(row=0,column=1,sticky=W+E+N+S)

b1 = Button(top,text=" button 1 ")

b1.grid(row=1,column=1,columnspan=4,sticky=W+E+N+S)

e1 = Entry(top)
e1.insert(INSERT,"entry 1 ")
e1.grid(row=2,column=0,columnspan=2)

b2 = Button(top,text = "button 2 ")

b2.grid(row=2,column=3)

top.mainloop()
from Tkinter import *
top = Tk()
top.title("use of grid")
t1 = Label(top,text=" text 1 ")
t1.grid(row=0,column=4)
b0 = Button(top,text=" button 0 ")
b0.grid(row=0,column=1,sticky=W+E+N+S)
b1 = Button(top,text=" button 1 ")
b1.grid(row=1,column=1,columnspan=4,sticky=W+E+N+S)
e1 = Entry(top)
e1.insert(INSERT,"entry 1 ")
e1.grid(row=2,column=0,columnspan=2)
b2 = Button(top,text = "button 2 ")
b2.grid(row=2,column=3)
top.mainloop()
from Tkinter import *
top = Tk()
top.title("use of grid")
t1 = Label(top, text=" text 1 ")
t1.grid(row=0, column=4)
b0 = Button(top, text=" button 0 ")
b0.grid(row=0, column=1, sticky=W+E+N+S)
b1 = Button(top, text=" button 1 ")
b1.grid(row=1, column=1, columnspan=4, sticky=W+E+N+S)
e1 = Entry(top)
e1.insert(INSERT, "entry 1 ")
e1.grid(row=2, column=0, columnspan=2)
b2 = Button(top, text = "button 2 ")
b2.grid(row=2, column=3)
top.mainloop()
Use of Radiobuttons

The GUI should do the following:

1. the entry field starts at 0
2. when user clicks on +1: add one
3. when user clicks on -1: subtract one
Use of Radiobuttons

The GUI should do the following:

1. the entry field starts at 0
2. when user clicks on +1: add one
3. when user clicks on -1: subtract one
Use of Radiobuttons

The GUI should do the following:

1. the entry field starts at 0
2. when user clicks on +1: add one
3. when user clicks on -1: subtract one
from Tkinter import *
top = Tk()
top.title("use Radiobutton")
text = Entry(top)
text.insert(INSERT,"0") # initialization
text.grid(row=0,columnspan=2)

def plus():
    "Callback function, does +1"
    s = text.get() # data in Entry
    s = str(int(s) + 1) # add one to it
    text.delete(0,END) # clear Entry
    text.insert(INSERT,s) # insert result

add = Radiobutton(top,text="+1",\
    command=plus)
add.grid(row=1,column=0)
top.mainloop()

Code for the other Radiobutton "-1" is similar.
from Tkinter import *
top = Tk()
top.title("use Radiobutton")
text = Entry(top)
text.insert(INSERT,"0")  # initialization
text.grid(row=0,columnspan=2)
def plus():
    "Callback function, does +1"
    s = text.get()       # data in Entry
    s = str(int(s) + 1)  # add one to it
    text.delete(0,END)   # clear Entry
    text.insert(INSERT,s) # insert result

add = Radiobutton(top,text="+1",
   command=plus)
add.grid(row=1,column=0)
top.mainloop()

Code for the other Radiobutton "-1" is similar.
from Tkinter import *
top = Tk()
top.title("use Radiobutton")
text = Entry(top)
text.insert(INSERT,"0")  # initialization
text.grid(row=0,columnspan=2)

def plus():
    "Callback function, does +1"
    s = text.get()          # data in Entry
    s = str(int(s) + 1)    # add one to it
    text.delete(0,END)    # clear Entry
    text.insert(INSERT,s)  # insert result

add = Radiobutton(top,text="+1",
    command=plus)
add.grid(row=1,column=0)
top.mainloop()
Code using Radiobuttons

in the file guiradio.py

```python
from Tkinter import *
top = Tk()
top.title("use Radiobutton")
text = Entry(top)
text.insert(INSERT,"0")  # initialization
text.grid(row=0,columnspan=2)

def plus():
    "Callback function, does +1"
    s = text.get()         # data in Entry
    s = str(int(s) + 1)    # add one to it
    text.delete(0,END)     # clear Entry
    text.insert(INSERT,s)  # insert result

add = Radiobutton(top,text="+1",\
    command=plus)
add.grid(row=1,column=0)
top.mainloop()

Code for the other Radiobutton "-1" is similar.
```
Using CheckButtons

Functionality:

1. the user can check one or two boxes
2. click on the enter button
3. to see a message displayed in the Entry box
Using CheckButtons

Functionality:
1. the user can check one or two boxes
2. click on the enter button
3. to see a message displayed in the Entry box
Using CheckButtons

Functionality:

1. the user can check one or two boxes
2. click on the enter button
3. to see a message displayed in the Entry box
Code use Checkbutton
in the file guicheck.py

from Tkinter import *
top = Tk()
top.title("use Checkbutton")
H = IntVar()  # determined by hot button
C = IntVar()  # determined by cold button

hot = Checkbutton(top,text="hot",
                  variable = H,onvalue = 1,offvalue = 0)
hot.grid(row=0,column=0)

cold = Checkbutton(top,text="cold",
                   variable = C,onvalue = 1,offvalue = 0)
cold.grid(row=0,column=1)

H and C are variables, toggled on or off by the user selecting the corresponding box.
from Tkinter import *

top = Tk()
top.title("use Checkbutton")

H = IntVar() # determined by hot button
C = IntVar() # determined by cold button

hot = Checkbutton(top,text="hot",
   variable = H,onvalue = 1,offvalue = 0)
hot.grid(row=0,column=0)

cold = Checkbutton(top,text="cold",
   variable = C,onvalue = 1,offvalue = 0)
cold.grid(row=0,column=1)

H and C are variables, toggled on or off by the user selecting the corresponding box.
from Tkinter import *

top = Tk()
top.title("use Checkbutton")

H = IntVar()  # determined by hot button
C = IntVar()  # determined by cold button

hot = Checkbutton(top,text="hot",
variable = H,onvalue = 1,offvalue = 0)
hot.grid(row=0,column=0)

cold = Checkbutton(top,text="cold",
variable = C,onvalue = 1,offvalue = 0)
cold.grid(row=0,column=1)

H and C are variables, toggled on or off by the user selecting the corresponding box.
code continued...

e = Entry(top)
e.grid(row=2,columnspan=2)

def act():
    "callback function for enter button"
    e.delete(0,END)
    if H.get() == 1 and C.get() == 0:
        e.insert(INSERT,"it is hot")
    if H.get() == 0 and C.get() == 1:
        e.insert(INSERT,"it is cold")
    if H.get() == 1 and C.get() == 1:
        e.insert(INSERT,"it is hot and cold")

b = Button(top,text = "enter",command=act)
b.grid(row=1,columnspan=2,sticky=W+E+N+S)
top.mainloop()

Observe the use of H.get() and C.get().
code continued...

e = Entry(top)
e.grid(row=2,columnspan=2)

def act():
    "callback function for enter button"
    e.delete(0,END)
    if H.get() == 1 and C.get() == 0:
        e.insert(INSERT,"it is hot")
    if H.get() == 0 and C.get() == 1:
        e.insert(INSERT,"it is cold")
    if H.get() == 1 and C.get() == 1:
        e.insert(INSERT,"it is hot and cold")

b = Button(top,text = "enter",command=act)
b.grid(row=1,columnspan=2,sticky=W+E+N+S)
top.mainloop()

Observe the use of H.get() and C.get().
code continued...

e = Entry(top)
e.grid(row=2,columnspan=2)

def act():
    "callback function for enter button"
    e.delete(0,END)
    if H.get() == 1 and C.get() == 0:
        e.insert(INSERT,"it is hot")
    if H.get() == 0 and C.get() == 1:
        e.insert(INSERT,"it is cold")
    if H.get() == 1 and C.get() == 1:
        e.insert(INSERT,"it is hot and cold")

b = Button(top,text = "enter",command=act)
b.grid(row=1,columnspan=2,sticky=W+E+N+S)
top.mainloop()

Observe the use of H.get() and C.get().
code continued...

e = Entry(top)
e.grid(row=2, columnspan=2)

def act():
    "callback function for enter button"
    e.delete(0, END)
    if H.get() == 1 and C.get() == 0:
        e.insert(INSERT, "it is hot")
    if H.get() == 0 and C.get() == 1:
        e.insert(INSERT, "it is cold")
    if H.get() == 1 and C.get() == 1:
        e.insert(INSERT, "it is hot and cold")

b = Button(top, text = "enter", command=act)
b.grid(row=1, columnspan=2, sticky=W+E+N+S)
top.mainloop()

Observe the use of H.get() and C.get().
graphical user interfaces using Tkinter

1. Graphical User Interfaces
   a sliding puzzle
   characteristics and components

2. The Tk GUI Toolkit in Python
   Tkinter, Tk, and Tcl
   hello world again
   using grid, Radiobutton, and Checkbutton
   handling mouse events

3. Summary + Assignments
Handling Mouse Events

Often the amount of data we generate is too huge for an orderly in a classical terminal window.

Much more data can be store in an image on canvas and via the mouse we may interact with the data.
Filling Squares on a Grid
with the mouse

def __init__(self, wdw, r, c):
    """
    filling squares on canvas with mouse clicks
    """
    wdw.title("mark with mouse")
    self.mag = 10  # magnification factor
    self.rows = r  # number of rows on canvas
    self.cols = c  # number of columns on canvas
__init__ continued

def __init__(self, wdw, r, c):
    ....
    self.c = Canvas(wdw, 
        width=self.mag*self.cols+2*self.mag, 
        height = self.mag*self.rows+2*self.mag, 
        bg='white')
    self.c.grid(row=1, column=0, columnspan=3)

    # to display mouse position :
    self.MousePosition = StringVar()
    self.MousePosition.set("put mouse inside box to draw")
    self.PositionLabel = Label(wdw, 
        textvariable = self.MousePosition)
    self.PositionLabel.grid( 
        row=2, column=0, columnspan=3)

    # bind mouse events
    self/bindMouseEvents()
    self.filled = zeros((r, c), bool)
__init__ continued

def __init__(self, wdw, r, c):
    ....
    self.c = Canvas(wdw, 
                    width=self.mag*self.cols+2*self.mag, 
                    height = self.mag*self.rows+2*self.mag, 
                    bg='white')
    self.c.grid(row=1, column=0, columnspan=3)
# to display mouse position :
    self.MousePosition = StringVar()
    self.MousePosition.set("put mouse inside box to draw")
    self.PositionLabel = Label(wdw, 
                                textvariable = self.MousePosition)
    self.PositionLabel.grid( 
                             (row=2, column=0, columnspan=3)
# bind mouse events
    self.BindMouseEvents()
    self.filled = zeros((r,c),bool)
def __init__(self, wdw, r, c):
    ....
    self.c = Canvas(wdw, 
        width=self.mag*self.cols+2*self.mag, 
        height = self.mag*self.rows+2*self.mag, 
        bg='white')
    self.c.grid(row=1, column=0, columnspan=3)
    # to display mouse position :
    self.MousePosition = StringVar()
    self.MousePosition.set("put mouse inside box to draw")
    self.PositionLabel = Label(wdw, 
        textvariable = self.MousePosition)
    self.PositionLabel.grid(
        row=2, column=0, columnspan=3)
    # bind mouse events
    self.BindMouseEvents()
    self.filled = zeros((r,c), bool)
Binding the Mouse Events

To bind the mouse events to canvas:

```python
def BindMouseEvents(self):
    
    Binds the mouse events to the canvas.

    self.c.bind("<Button-1>", self.ButtonPressed)
    self.c.bind("<ButtonRelease-1>",
                self.ButtonReleased)
    self.c.bind("<Enter>", self.EnteredWindow)
    self.c.bind("<Leave>", self.ExitedWindow)
    self.c.bind("<B1-Motion>", self.MouseDragged)
```

The methods `ButtonPressed`, `ButtonReleased`, `EnteredWindow`, `ExitedWindow`, and `MouseDragged` are activated by the mouse.
def ButtonPressed(self, event):
    """
    Display the coordinates of the button pressed.
    """
    self.MousePosition.set("currently at [ " + \n    str(event.x) + ", " + str(event.y) + " ]" +\n    " release to fill, or drag")

def ButtonReleased(self, event):
    """
    Display coordinates of the button released.
    """
    self.MousePosition.set("drawn at [ " + \n    str(event.x) + ", " + str(event.y) + " ]" +\n    " redo to clear")
    self.DrawRectangle(event.x, event.y)
def ButtonPressed(self, event):
    
    Display the coordinates of the button pressed.
    
    self.MousePosition.set("currently at [ " + \n    str(event.x) + ", " + str(event.y) + " ]" + \n    " release to fill, or drag")

def ButtonReleased(self, event):
    
    Display coordinates of the button released.
    
    self.MousePosition.set("drawn at [ " + \n    str(event.x) + ", " + str(event.y) + " ]" + \n    " redo to clear")

    self.DrawRectangle(event.x, event.y)
The Other Methods

def EnteredWindow(self, event):
    
    Display the message that mouse entered window.
    
    self.MousePosition.set("press mouse" + " to give coordinates")

def ExitedWindow(self, event):
    
    Display the message that mouse exited window.
    
    self.MousePosition.set("put mouse " + " inside box to draw")

def MouseDragged(self, event):
    
    Display the coordinates of the moving mouse.
    
    self.MousePosition.set("dragging at [ " + " , " + str(event.x) + ", " + str(event.y) + " ]" + " release to draw")
The Other Methods

def EnteredWindow(self,event):
    ""
    Display the message that mouse entered window.
    ""
    self.MousePosition.set("press mouse" + "to give coordinates")
def ExitedWindow(self,event):
    ""
    Display the message that mouse exited window.
    ""
    self.MousePosition.set("put mouse " + "inside box to draw")
def MouseDragged(self,event):
    ""
    Display the coordinates of the moving mouse.
    ""
    self.MousePosition.set("dragging at [" + str(event.x) + ", " + str(event.y) + "]" + " release to draw")
The Other Methods

def EnteredWindow(self, event):
    """
    Display the message that mouse entered window.
    """
    self.MousePosition.set("press mouse" + \n        "to give coordinates")

def ExitedWindow(self, event):
    """
    Display the message that mouse exited window.
    """
    self.MousePosition.set("put mouse " + \n        "inside box to draw")

def MouseDragged(self, event):
    """
    Display the coordinates of the moving mouse.
    """
    self.MousePosition.set("dragging at [ " + \n        str(event.x) + ", " + str(event.y) + \n        "]" + " release to draw")
We started Chapter 12 in *Python Power!*

**Assignments:**

1. Write Python code to display:

![Calculator](image)

You should not provide any functionality.

2. Add functionality to the calculator shown above.

3. Design a GUI to convert temperatures between Fahrenheit and Celsius. Draw the layout and decide what widgets you will use.

4. Give Python code for the previous exercise.