

CHAPTER 13

GUI Programming

starting out with >>>

PYTHON®

FOURTH EDITION



TONY GADDIS

Topics

- **Graphical User Interfaces**
- **Using the `tkinter` Module**
- **Display Text with `Label` Widgets**
- **Organizing Widgets with Frames**
- **Button Widgets and Info Dialog Boxes**
- **Getting Input with the `Entry` Widget**
- **Using Labels as Output Fields**
- **Radio Buttons and Check Buttons**
- **Drawing Shapes with the `Canvas` Widget**

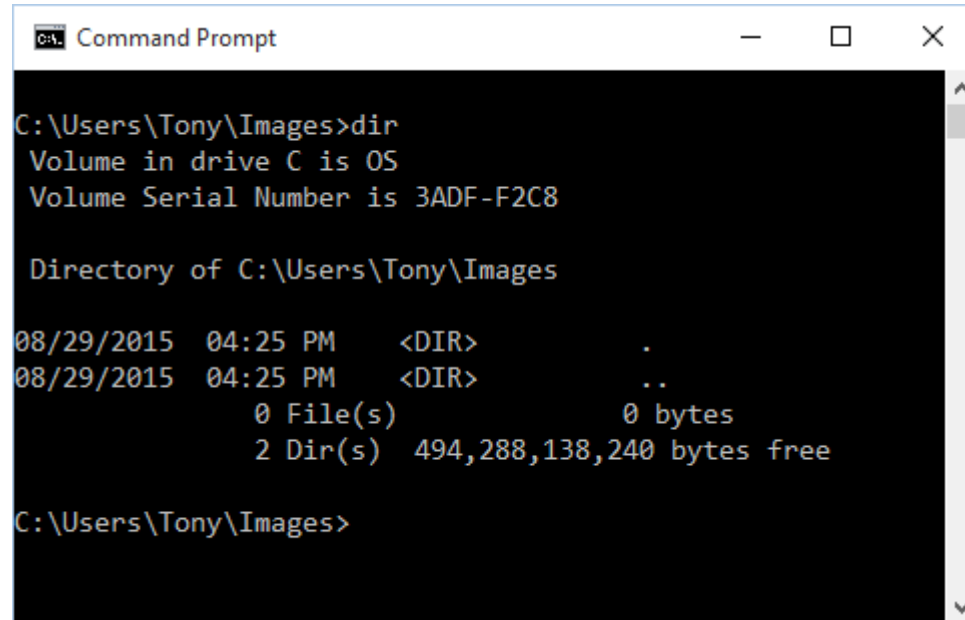


Graphical User Interfaces

- **User Interface**: the part of the computer with which the user interacts
- **Command line interface**: displays a prompt and the user types a command that is then executed
- **Graphical User Interface (GUI)**: allows users to interact with a program through graphical elements on the screen

Graphical User Interfaces (cont'd.)

- A command line interface



```
C:\Users\Tony\Images>dir
Volume in drive C is OS
Volume Serial Number is 3ADF-F2C8

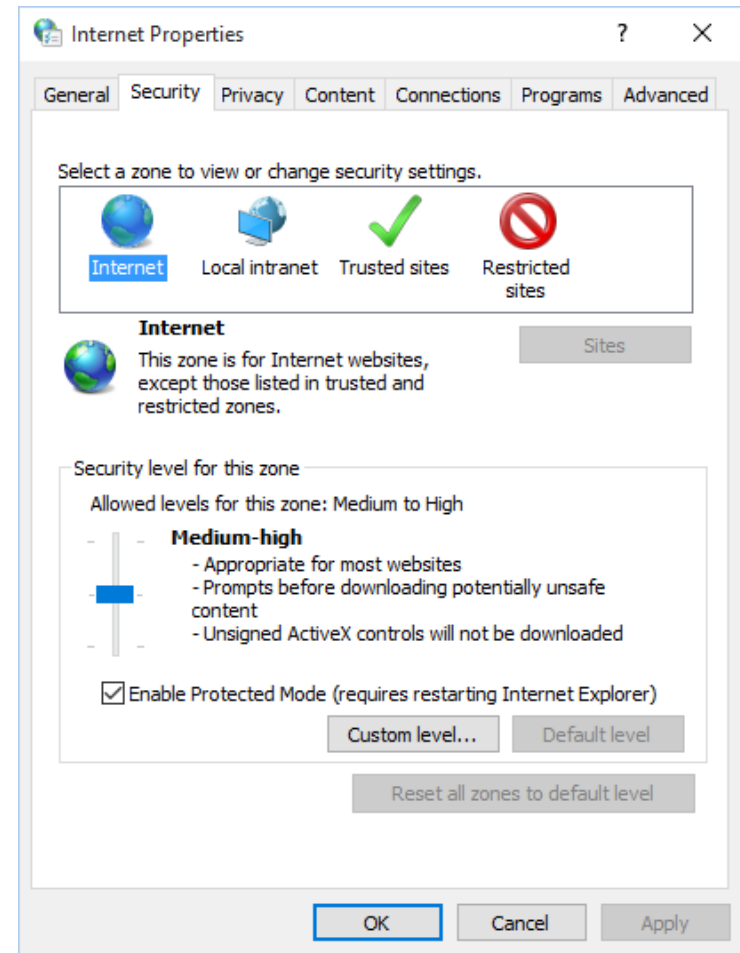
Directory of C:\Users\Tony\Images

08/29/2015  04:25 PM    <DIR>          .
08/29/2015  04:25 PM    <DIR>          ..
               0 File(s)                0 bytes
               2 Dir(s)  494,288,138,240 bytes free

C:\Users\Tony\Images>
```

Graphical User Interfaces (cont'd.)

- **Dialog boxes**: small windows that display information and allow the user to perform actions
 - Responsible for most of the interaction through GUI
 - User interacts with graphical elements such as icons, buttons, and slider bars



GUI Programs Are Event-Driven

- **In command line interfaces, programs determine the order in which things happen**
 - The user can only enter data in the order requested by the program
- **GUI environment is event-driven**
 - The user determines the order in which things happen
 - User causes events to take place and the program responds to the events

[test_averages.py](#)



Using the `tkinter` Module

- No GUI programming features built into Python
- `tkinter` module: allows you to create simple GUI programs
 - Comes with Python
- Widget: graphical element that the user can interact with or view
 - Presented by a GUI program

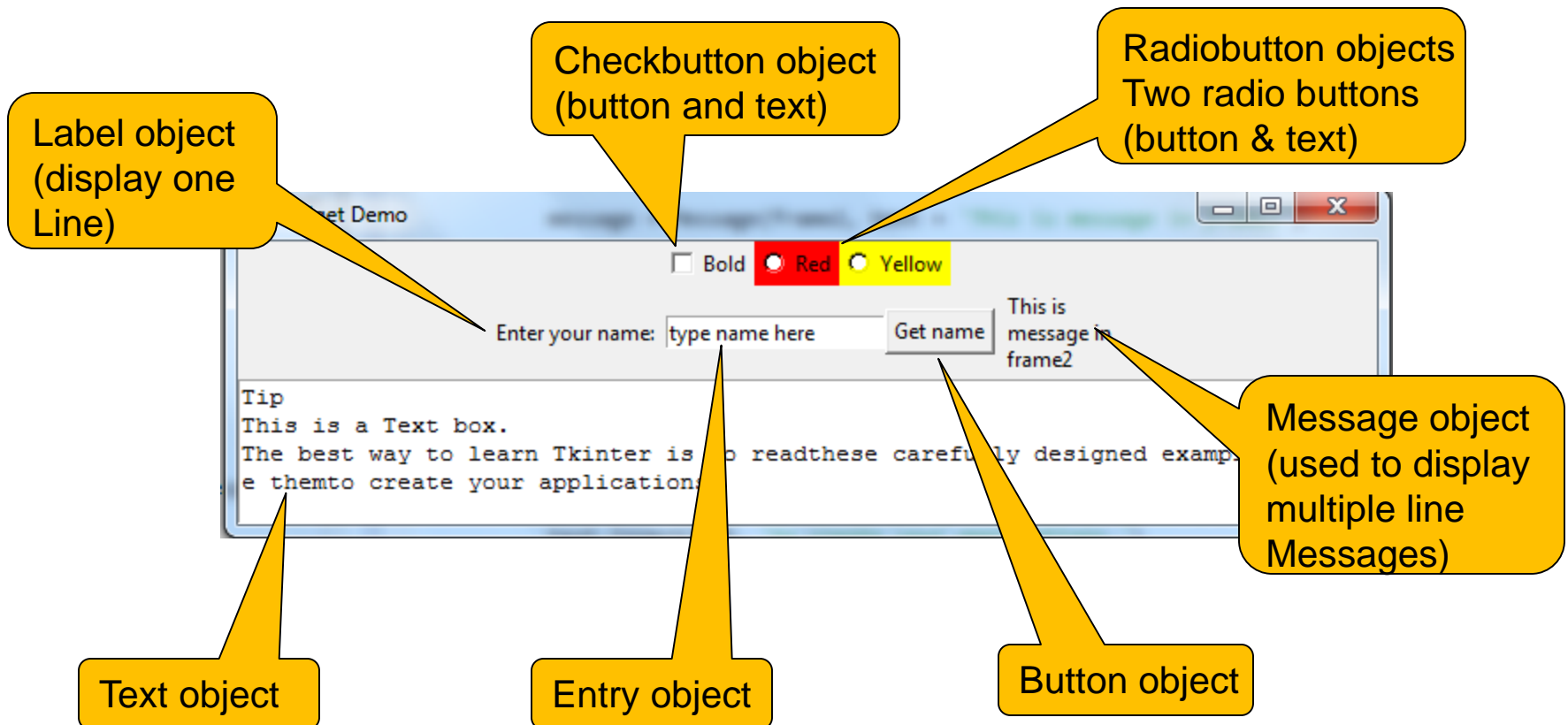
Table 13-1 tkinter Widgets

Widget	Description
<input checked="" type="checkbox"/> Button	A button that can cause an action to occur when it is clicked.
<input checked="" type="checkbox"/> Canvas	A rectangular area that can be used to display graphics.
<input checked="" type="checkbox"/> Checkbutton	A button that may be in either the “on” or “off” position.
<input checked="" type="checkbox"/> Entry	An area in which the user may type a single line of input from the keyboard.
<input checked="" type="checkbox"/> Frame	A container that can hold other widgets.
<input checked="" type="checkbox"/> Label	An area that displays one line of text or an image.
Listbox	A list from which the user may select an item
Menu	A list of menu choices that are displayed when the user clicks a Menubutton widget.
Menubutton	A menu that is displayed on the screen and may be clicked by the user
Message	Displays multiple lines of text.
<input checked="" type="checkbox"/> Radiobutton	A widget that can be either selected or deselected. Radiobutton widgets usually appear in groups and allow the user to select one of several options.
Scale	A widget that allows the user to select a value by moving a slider along a track.
Scrollbar	Can be used with some other types of widgets to provide scrolling ability.
Text	A widget that allows the user to enter multiple lines of text input.
Toplevel	A container, like a Frame, but displayed in its own window.



Label, Entry, Text, Message, Checkbutton, and Radiobutton Widgets

Widgets Demo



Using the `tkinter` Module (cont'd.)

- **Programs that use `tkinter` do not always run reliably under IDLE**
 - For best results run them from operating system command prompt `empty_window1.py`
- **Most programmers take an object-oriented approach when writing GUI programs** `empty_window2.py`
 - `__init__` method builds the GUI
 - When an instance is created the GUI appears on the screen



Display Text with Label Widgets

- **Label widget**: displays a single line of text in a window

- Made by creating an instance of `tkinter` module's `Label` class

`hello_world.py`

- Format:

```
tkinter.Label(self.main_window,  
               text = 'my text')
```

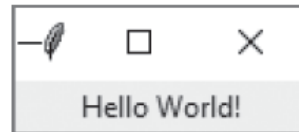
- First argument references the parent widget, second argument shows text that should appear in label

Display Text with Label Widgets (cont'd.)

- **pack method**: determines where a widget should be positioned and makes it visible when the main window is displayed
 - Called for each widget in a window
 - Receives an argument to specify positioning
 - Positioning depends on the order in which widgets were added to the main window
 - Valid arguments: `side='top'`, `side='left'`, `side='right'`

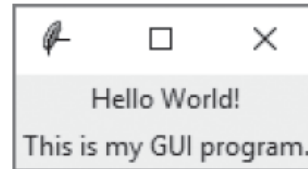
Display Text with Label Widgets (cont'd.)

Figure 13-5 Window displayed by Program 13-3



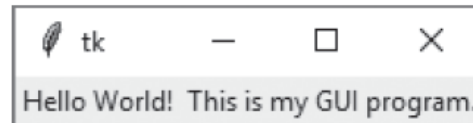
hello_world.py

Figure 13-6 Window displayed by Program 13-4



hello_world2.py

Figure 13-7 Window displayed by Program 13-5



hello_world3.py

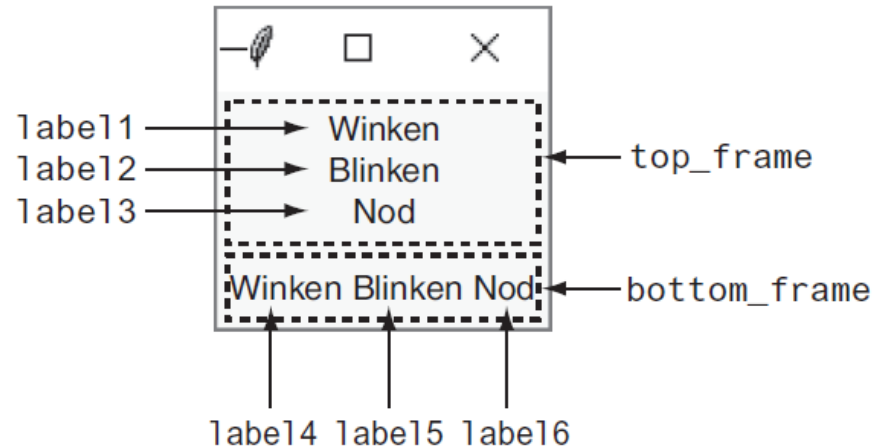
Organizing Widgets with Frames

- **Frame widget: container that holds other widgets**
 - Useful for organizing and arranging groups of widgets in a window
 - The contained widgets are added to the frame widget which contains them
 - Example Adding a label to a frame:

```
tkinter.Label(self.top_frame,  
               text = 'hi')
```

Organizing Widgets with Frames (cont'd.)

Figure 13-9 Arrangement of widgets



[frame_demo.py](#)

Adding two frames to the main window:

```
self.top_frame = tkinter.Frame(self.main_window)
self.bottom_frame = tkinter.Frame(self.main_window)
```

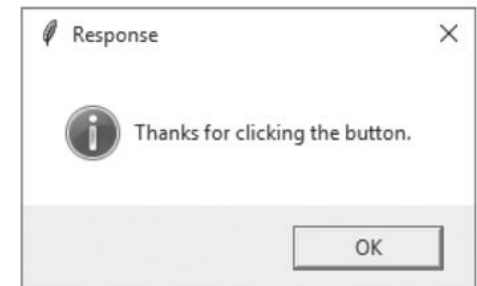
Button Widgets and Info Dialog Boxes

- **Button widget**: widget that the user can click to cause an action to take place
 - When creating a button can specify:
 - Text to appear on the face of the button
 - A callback function
- **Callback function**: function or method that executes when the user clicks the button
 - Also known as an event handler



Button Widgets and Info Dialog Boxes (cont'd.)

- **Info dialog box**: a dialog box that shows information to the user
 - Format for creating an info dialog box:
 - Import `tkinter.messagebox` module
 - `tkinter.messagebox.showinfo(title, message)`
 - `title` is displayed in dialog box's title bar
 - `message` is an informational string displayed in the main part of the dialog box

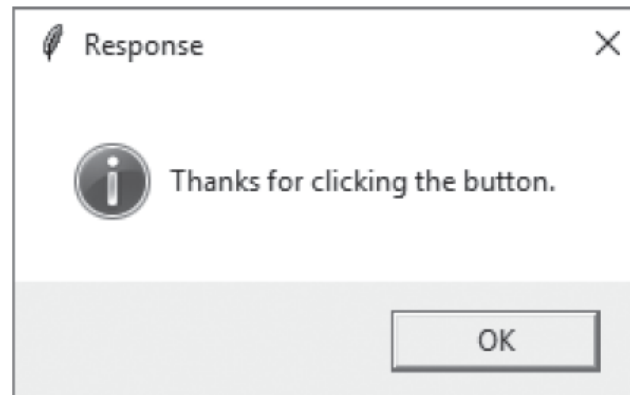


Button Widgets and Info Dialog Boxes (cont'd.)

Figure 13-10 The main window displayed by Program 13-7



Figure 13-11 The info dialog box displayed by Program 13-7



button_demo.py

Creating a Quit Button

- **Quit button**: closes the program when the user clicks it
- **To create a quit button in Python:**
 - Create a `Button` widget
 - Set the root widget's `destroy` method as the callback function
 - When the user clicks the button the `destroy` method is called and the program ends

`quit_button.py`



Getting Input with the Entry Widget

- **Entry widget**: rectangular area that the user can type text into
 - Used to gather input in a GUI program
 - Typically followed by a button for submitting the data
 - The button's callback function retrieves the data from the `Entry` widgets and processes it
 - **Entry widget's get method**: used to retrieve the data from an `Entry` widget
 - Returns a string

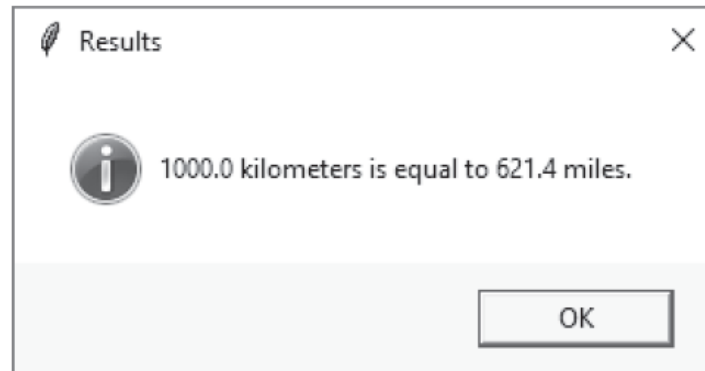
Getting Input with the Entry Widget (cont'd.)

Figure 13-15 The info dialog box

- ① The user enters 1000 into the Entry widget and clicks the Convert button.



- ② This info dialog box is displayed.



kilo_converter.py

Using Labels as Output Fields

- **Can use `Label` widgets to dynamically display output**
 - Used to replace info dialog box
 - Create empty `Label` widget in main window, and write code that displays desired data in the label when a button is clicked

Using Labels as Output Fields (cont'd.)

- **StringVar class:** `tkinter` module class that can be used along with `Label` widget to display data
 - Create `StringVar` object and then create `Label` widget and associate it with the `StringVar` object
 - Subsequently, any value stored in the `StringVar` object will automatically be displayed in the `Label` widget



Using Labels as Output Fields (cont'd.)

Figure 13-16 The window initially displayed

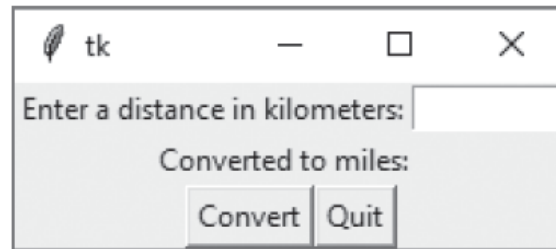
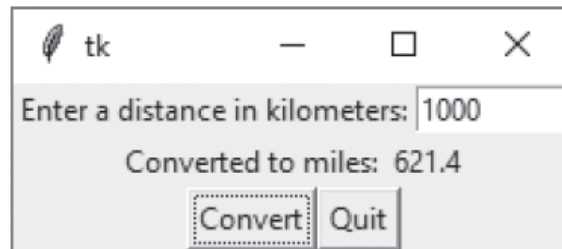


Figure 13-17 The window showing 1000 kilometers converted to miles



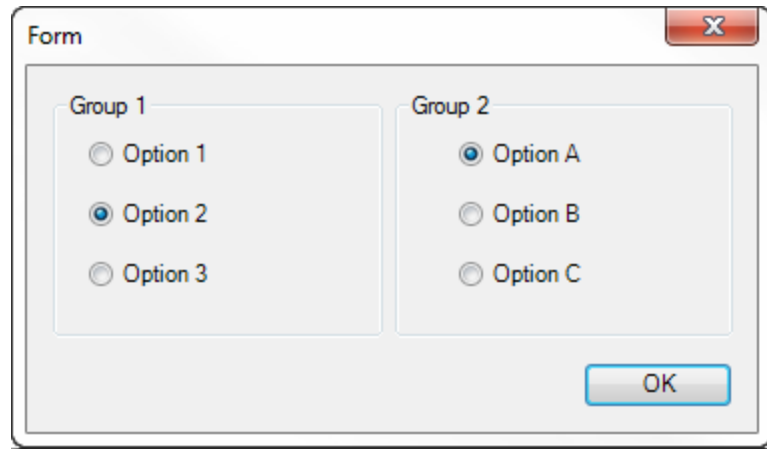
Radio Buttons and Check Buttons

- **Radio button**: small circle that appears filled when it is selected and appears empty when it is deselected
 - Useful when you want the user to select one choice from several possible options



Radio Buttons and Check Buttons

- **Radiobutton widgets:** created using **tkinter module's Radiobutton class**
 - Radiobutton widgets are mutually exclusive
 - Only one radio button in a container may be selected at any given time



Radio Buttons and Check Buttons (cont'd)

- IntVar class: a `tkinter` module class that can be used along with `Radiobutton` widgets

[radiobutton_demo.py](#)

- Steps for use:
 - Associate group of `Radiobutton` widgets with the same `IntVar` object
 - Assign unique integer to each `Radiobutton`
 - When a `Radiobutton` widget is selected, its unique integer is stored in the `IntVar` object
- Can be used to select a default radio button

Using Callback Functions with Radiobuttons

- You can specify a callback function with Radiobutton widgets
 - Provide an argument `command=self.my_method` when creating the Radiobutton widget radiobutton_demo2.py
 - The command will execute immediately when the radio button is selected
 - Replaces the need for a user to click OK or submit before determining which Radiobutton is selected



Check Buttons

checkboxbutton_demo.py

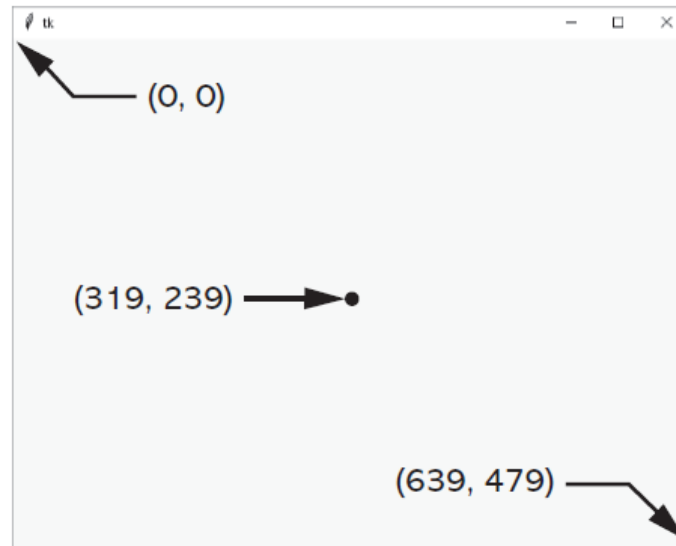
- **Check button**: small box with a label appearing next to it; check mark indicates when it is selected
 - User is allowed to select any or all of the check buttons that are displayed in a group
 - Not mutually exclusive
- **Checkbutton widgets**: created using **tkinter module's Checkbutton class**
 - Associate different `IntVar` object with each `Checkbutton` widget

Drawing Shapes with the Canvas Widget

- The Canvas widget is a blank, rectangular area that allows you to draw simple 2D shapes.
- You use the Canvas widget's *screen coordinate system* to specify the location of your graphics.
- The coordinates of the pixel in the upper-left corner of the screen are (0, 0).
 - The X coordinates increase from left to right
 - The Y coordinates increase from top to bottom.

Drawing Shapes with the Canvas Widget

Figure 13-26 Various pixel locations in a 640 by 480 window



Drawing Shapes with the Canvas Widget

- **Creating a Canvas widget:**

```
# Create the main window.  
self.main_window = tkinter.Tk()  
  
# Create the Canvas widget.  
self.canvas = tkinter.Canvas(self.main_window, width=200, height=200)
```


Drawing Shapes with the Canvas Widget

- The Canvas widget has numerous methods for drawing graphical shapes on the surface of the widget.
- The methods that we will discuss are:
 - `create_line`
 - `create_rectangle`
 - `create_polygon`
 - `create_text`
 - `create_oval`
 - `create_arc`

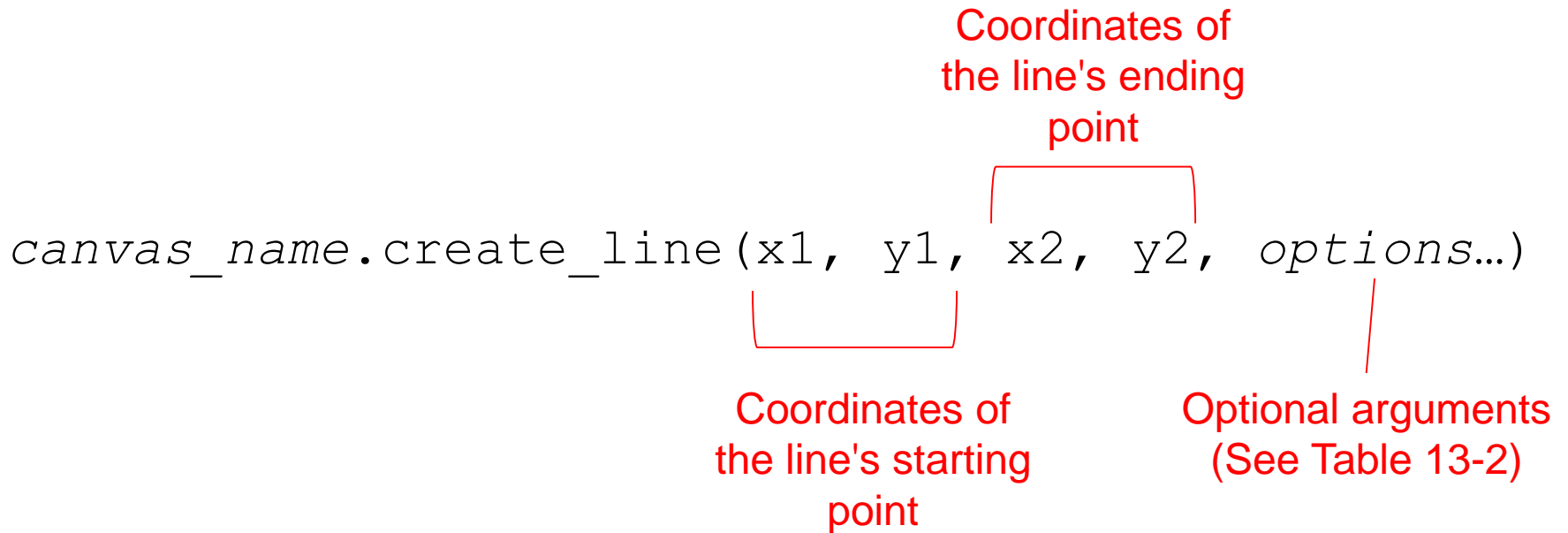
Drawing a Line

`canvas_name.create_line(x1, y1, x2, y2, options...)`

Coordinates of the line's ending point

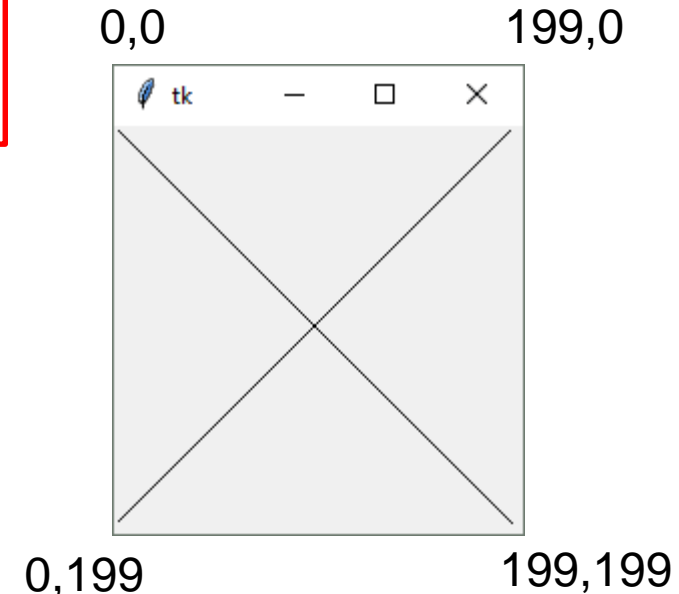
Coordinates of the line's starting point

Optional arguments (See Table 13-2)



Program 13-14 (draw_line.py)

```
1  # This program demonstrates the Canvas widget.
2  import tkinter
3
4  class MyGUI:
5      def __init__(self):
6          # Create the main window.
7          self.main_window = tkinter.Tk()
8
9          # Create the Canvas widget.
10         self.canvas = tkinter.Canvas(self.main_window, width=200,height=200)
11
12         # Draw two lines.
13         self.canvas.create_line(0, 0, 199, 199)
14         self.canvas.create_line(199, 0, 0, 199)
15
16         # Pack the canvas.
17         self.canvas.pack()
18
19         # Start the mainloop.
20         tkinter.mainloop()
21
22 # Create an instance of the MyGUI class.
23 my_gui = MyGUI()
```



Drawing a Rectangle

`canvas_name.create_rectangle(x1, y1, x2, y2, options...)`

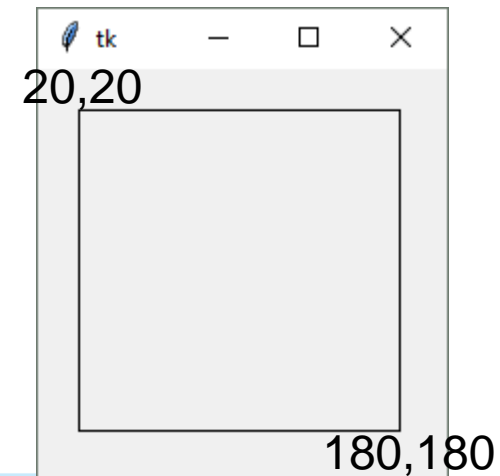
Coordinates of the lower-right corner

Coordinates of the upper-left corner

Optional arguments (See Table 13-3)

Program 13-16 (draw_square.py)

```
1  # This program draws a rectangle on a Canvas.
2  import tkinter
3
4  class MyGUI:
5      def __init__(self):
6          # Create the main window.
7          self.main_window = tkinter.Tk()
8
9          # Create the Canvas widget.
10         self.canvas = tkinter.Canvas(self.main_window, width=200, height=200)
11
12         # Draw a rectangle.
13         self.canvas.create_rectangle(20, 20, 180, 180)
14
15         # Pack the canvas.
16         self.canvas.pack()
17
18         # Start the mainloop.
19         tkinter.mainloop()
20
21 # Create an instance of the MyGUI class.
22 my_gui = MyGUI()
```



Drawing a Polygon

`canvas_name.create_polygon(x1, y1, x2, y2, ..., options...)`

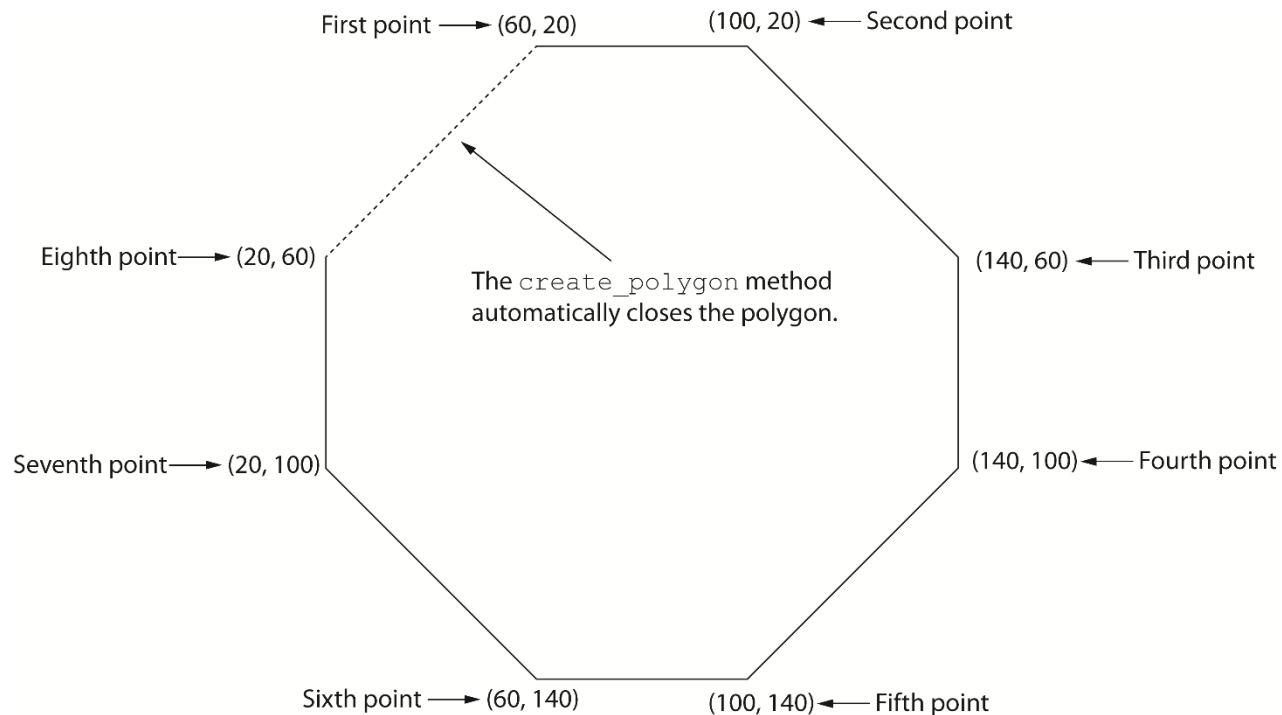
Coordinates of the second vertex

Coordinates of the first vertex

Optional arguments (See Table 13-7)

Drawing a Polygon

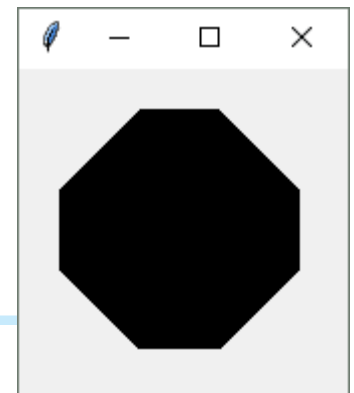
```
self.canvas.create_polygon(60, 20, 100, 20, 140, 60, 140, 100,  
                           100, 140, 60, 140, 20, 100, 20, 60)
```



Program 13-20 (draw_polygon.py)

```
1  # This program draws a polygon on a Canvas.
2  import tkinter
3
4  class MyGUI:
5      def __init__(self):
6          # Create the main window.
7          self.main_window = tkinter.Tk()
8
9          # Create the Canvas widget.
10         self.canvas = tkinter.Canvas(self.main_window, width=160, height=160)
11
12         # Draw a polygon.
13         self.canvas.create_polygon(60, 20, 100, 20, 140, 60, 140, 100,
14                                   100, 140, 60, 140, 20, 100, 20, 60)
15
16         # Pack the canvas.
17         self.canvas.pack()
18
19         # Start the mainloop.
20         tkinter.mainloop()
21
22 # Create an instance of the MyGUI class.
23 my_gui = MyGUI()
```

draw_polygon.py



Displaying Text on the Canvas

`canvas_name.create_text(x, y, text=text, options...)`

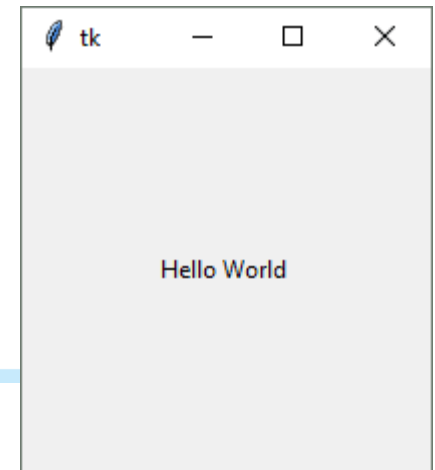
Text to display

Coordinates of the text's insertion point

Optional arguments (See Table 13-8)

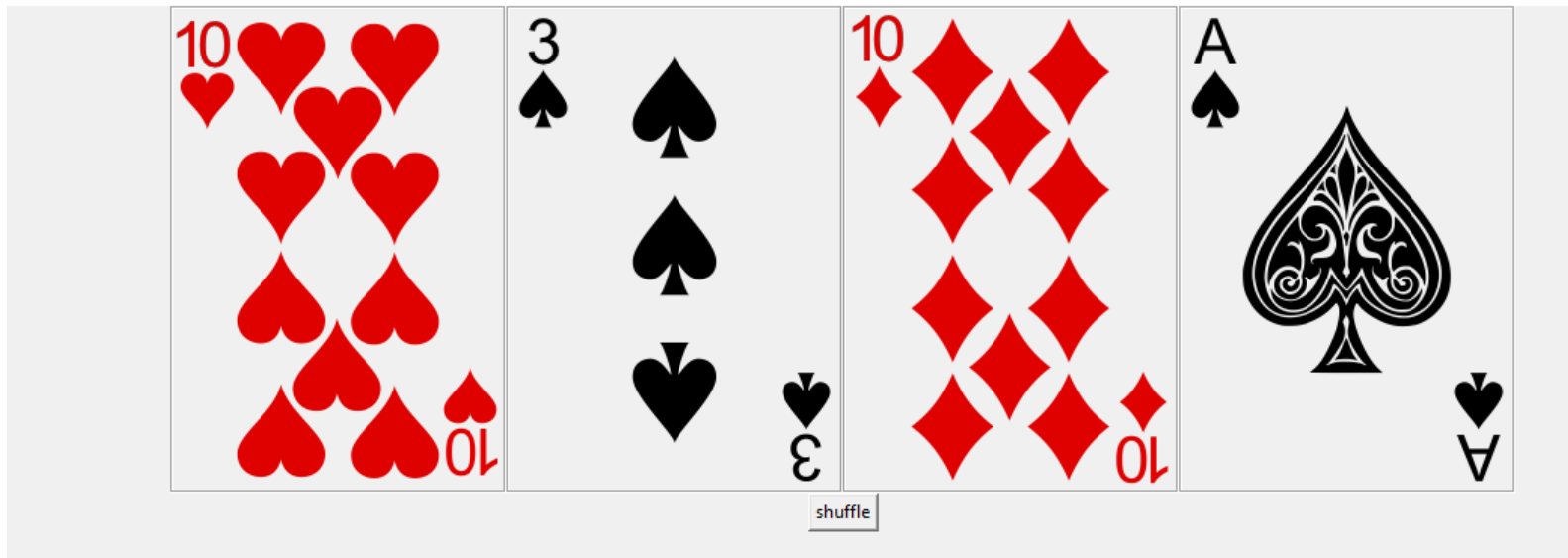
Program 13-21 (draw_text.py)

```
1  # This program draws text on a Canvas.
2  import tkinter
3
4  class MyGUI:
5      def __init__(self):
6          # Create the main window.
7          self.main_window = tkinter.Tk()
8
9          # Create the Canvas widget.
10         self.canvas = tkinter.Canvas(self.main_window, width=200, height=200)
11
12         # Display text in the center of the window.
13         self.canvas.create_text(100, 100, text='Hello World')
14
15         # Pack the canvas.
16         self.canvas.pack()
17
18         # Start the mainloop.
19         tkinter.mainloop()
20
21 # Create an instance of the MyGUI class.
22 my_gui = MyGUI()
```



Case Study

Shuffling Cards



- Each card image has a number (1→52)
- Loading default image (Jocker)
- Associating a button with an instance of PhotoImage
- Adding effects to the label



Summary

- **This chapter covered:**
 - Graphical user interfaces and their role as event-driven programs
 - The `tkinter` module, including:
 - Creating a GUI window
 - Adding widgets to a GUI window
 - Organizing widgets in frames
 - Receiving input and providing output using widgets
 - Creating buttons, check buttons, and radio buttons
 - Drawing simple shapes with the `Canvas` widget