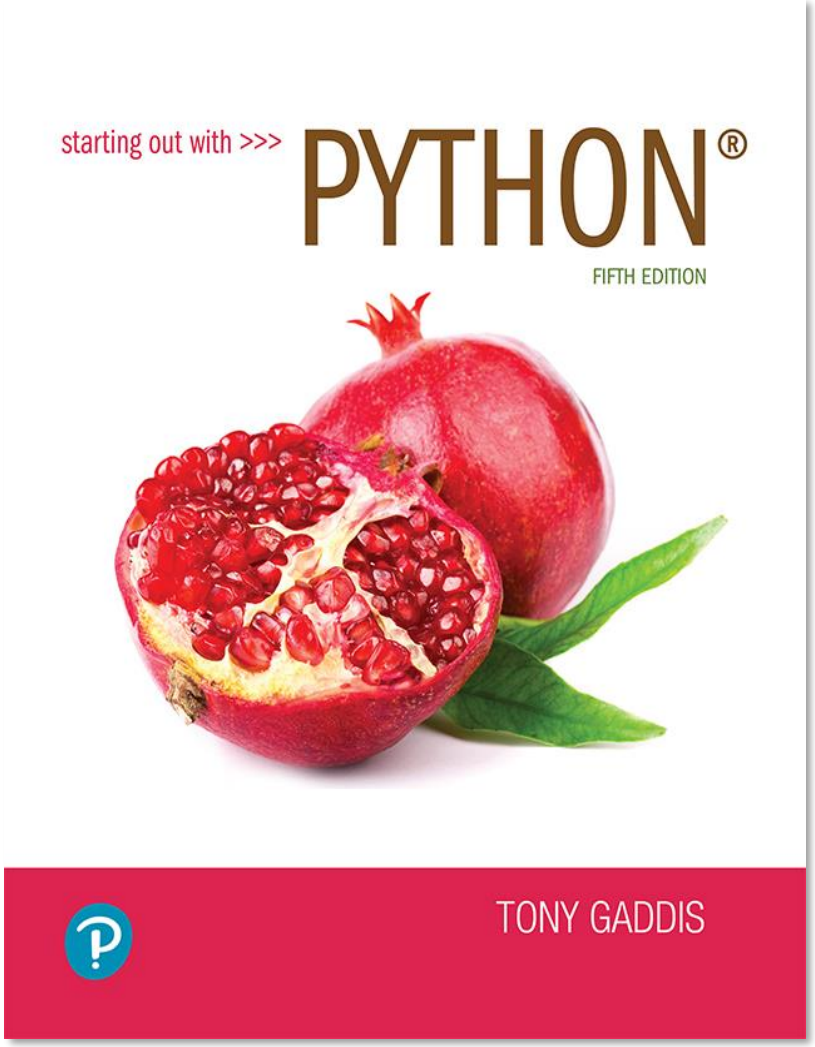
Starting out with Python

Fifth Edition



Chapter 13

GUI Programming

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Topics (1 of 2)

- Graphical User Interfaces
- Using the `tkinter` Module
- Displaying 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Topics (2 of 2)

- Radio Buttons and Check Buttons
- `ListBox` Widgets
- Drawing Shapes with the `Canvas` Widget

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Graphical User Interfaces (1 of 3)

- 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Graphical User Interfaces (2 of 3)

- Command line interfaces

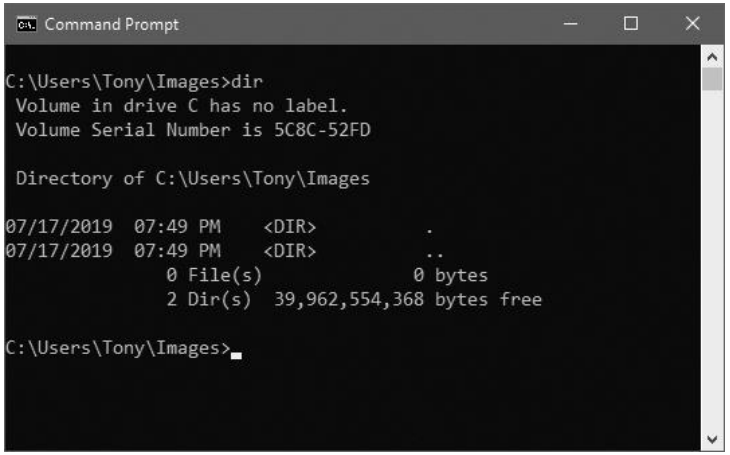
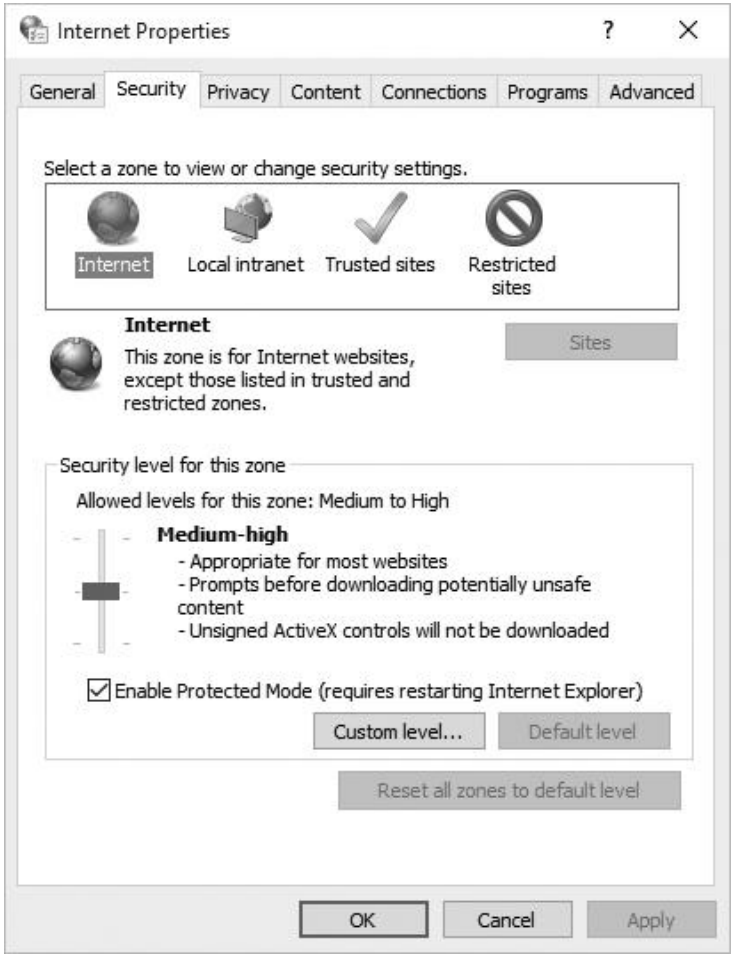


Figure 13-1 A command line interface

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Graphical User Interfaces (3 of 3)

- 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



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

GUI Programs Are Event-Driven

- In text-based environments, 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Using the `tkinter` Module (1 of 3)

- 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Using the `tkinter` Module (2 of 3)

Table 13-1 `tkinter` widgets

Widget	Description
`Button`	A button that can cause an action to occur when it is clicked.
`Canvas`	A rectangular area that can be used to display graphics.
`Checkbutton`	A button that may be in either the “on” or “off” position.
`Entry`	An area in which the user may type a single line of input from the keyboard.
`Frame`	A container that can hold other widgets.
`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.
`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.

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

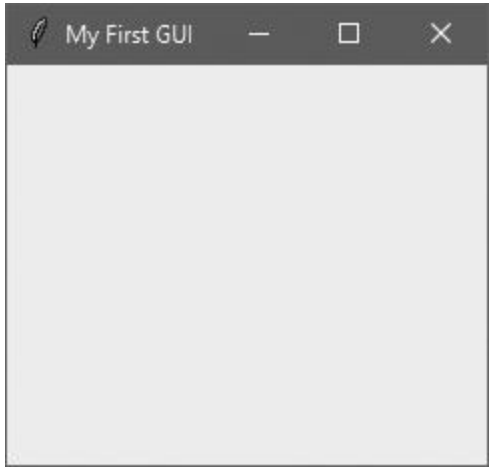
Using the `tkinter` Module

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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (1 of 6)

```
# This program displays an empty window.

import tkinter

class MyGUI:
    def __init__(self):
        # Create the main window widget.
        self.main_window = tkinter.Tk()

        # Display a title.
        self.main_window.title('My First GUI')

        # Enter the tkinter main loop.
        tkinter.mainloop()

# Create an instance of the MyGUI class.
if __name__ == '__main__':
    my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Display Text with `Label` Widgets (1 of 2)

- `Label` widget: displays a single line of text in a window
  - Made by creating an instance of `tkinter` module's `Label` class
  - Format: `tkinter.Label(self.main_window, text = 'my text')`
    - First argument references the root widget, second argument shows text that should appear in label

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Display Text with `Label` Widgets (2 of 2)

- `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'`

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (2 of 6)

```
self.label = tkinter.Label(self.main_window,
                           text='Hello World!')
self.label.pack()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

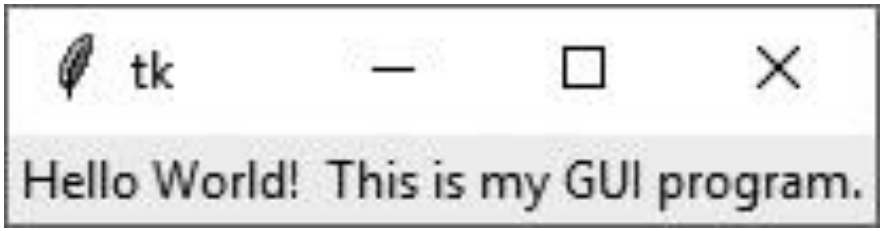
Example (3 of 6)

```
self.label1 = tkinter.Label(self.main_window,
                            text='Hello World!')
self.label2 = tkinter.Label(self.main_window,
                            text='This is my GUI program.')
self.label.pack()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (4 of 6)

```
self.label1 = tkinter.Label(self.main_window,
                            text='Hello World!')
self.label2 = tkinter.Label(self.main_window,
                            text='This is my GUI program.')
self.label1.pack(side='left')
self.label2.pack(side='left')
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

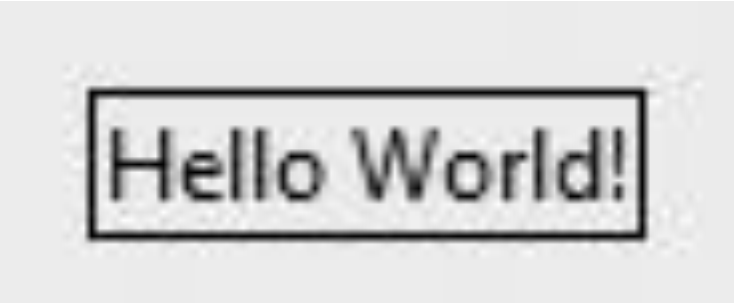
Adding Borders to Labels

- When creating a `Label` widget, you can use the `borderwidth` and `relief` arguments to display a border around the label
- The `borderwidth` argument specifies the width of the border, in pixels
- The `relief` argument specifies the border style

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (5 of 6)

```
self.label = tkinter.Label(self.main_window,
                           text='Hello World',
                           borderwidth=1,
                           relief='solid'
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (6 of 6)

```
self.label = tkinter.Label(self.main_window,
                           text='Hello World',
                           borderwidth=4,
                           relief='solid'
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Values for `relief` Argument (1 of 2)

`relief` Argument	Description
`relief='flat'`	The border is hidden and there is no 3D effect.
`relief='raised'`	The widget has a raised 3D appearance.
`relief='sunken'`	The widget has a sunken 3D appearance.
`relief='ridge'`	The border around the widget has a raised 3D appearance.
`relief='solid'`	The border appears as a solid line with no 3D effect.
`relief='groove'`	The border around the widget appears as a groove.

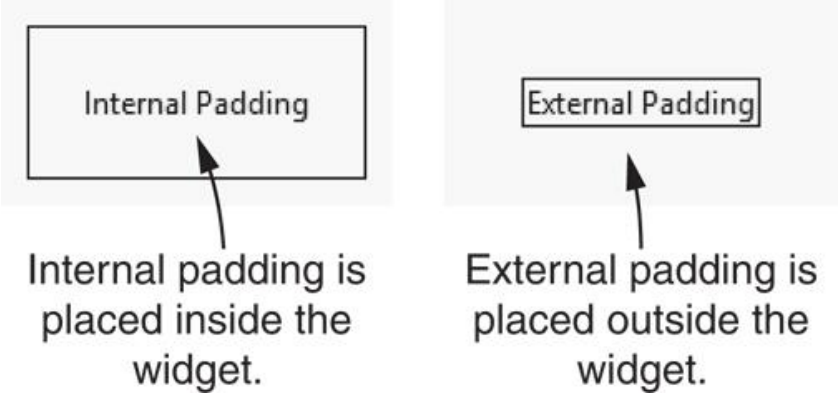
Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Values for `relief` Argument (2 of 2)

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Padding

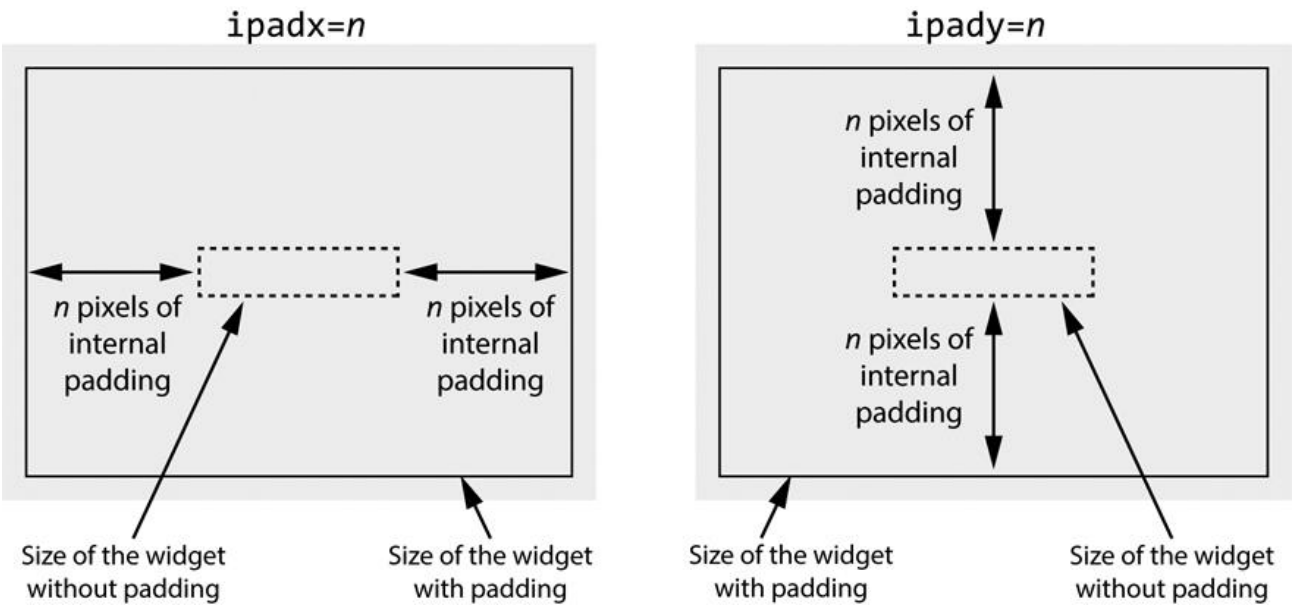
- Padding: space that appears around a widget
  - Internal padding appears around the inside edge of a widget
  - External padding appears around the outside edge of a widget



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Internal Padding (1 of 3)

- To add horizontal internal padding to a widget, pass the argument `ipadx=n` to the widget's `pack` method
- To add vertical internal padding to a widget, pass the argument `ipady=n` to the widget's `pack` method



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Internal Padding (2 of 3)

```
# This program demonstrates internal padding.
import tkinter

class MyGUI:
    def __init__(self):
        # Create the main window widget.
        self.main_window = tkinter.Tk()

        # Create two Label widgets with solid borders.
        self.label1 = tkinter.Label(self.main_window,
                                    text='Hello World!',
                                    borderwidth=1,
                                    relief='solid')

        self.label2 = tkinter.Label(self.main_window,
                                    text='This is my GUI program.',
                                    borderwidth=1,
                                    relief='solid')

```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

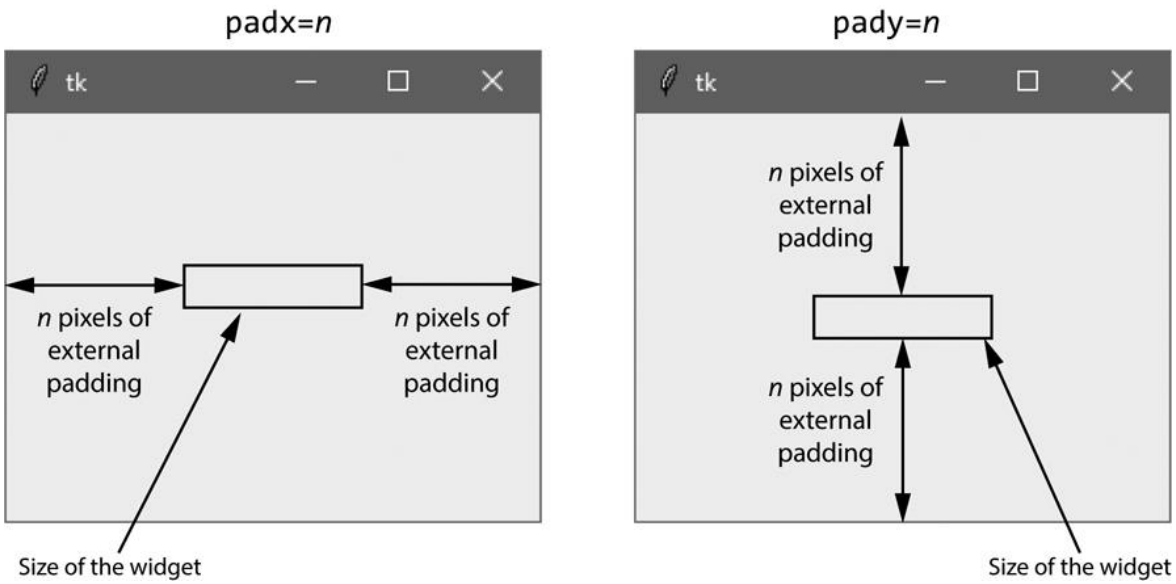
Internal Padding (3 of 3)

```
        # Display the Labels with 20 pixels of horizontal
        # and vertical internal padding.
        self.label1.pack(ipadx=20, ipady=20)
        self.label2.pack(ipadx=20, ipady=20)

        # Enter the tkinter main loop.
        tkinter.mainloop()

# Create an instance of the MyGUI class.
if __name__ == '__main__':
    my_gui = MyGUI()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

External Padding (1 of 3)

- To add horizontal external padding to a widget, pass the argument `padx=n` to the widget's `pack` method
- To add vertical external padding to a widget, pass the argument `pady=n` to the widget's `pack` method



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

External Padding (2 of 3)

```
# This program demonstrates external padding.
import tkinter

class MyGUI:
    def __init__(self):
        # Create the main window widget.
        self.main_window = tkinter.Tk()

        # Create two Label widgets with solid borders.
        self.label1 = tkinter.Label(self.main_window,
                                    text='Hello World!',
                                    borderwidth=1,
                                    relief='solid')

        self.label2 = tkinter.Label(self.main_window,
                                    text='This is my GUI program.',
                                    borderwidth=1,
                                    relief='solid')

        # Display the Labels with 20 pixels of horizontal
        # and vertical external padding.
        self.label1.pack(padx=20, pady=20)
        self.label2.pack(padx=20, pady=20)

        # Enter the tkinter main loop.
        tkinter.mainloop()

# Create an instance of the MyGUI class.
if __name__ == '__main__':
    my_gui = MyGUI()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

External Padding (3 of 3)

```
# This program demonstrates both internal and external padding.
import tkinter

class MyGUI:
    def __init__(self):
        # Create the main window widget.
        self.main_window = tkinter.Tk()

        # Create two Label widgets with solid borders.
        self.label1 = tkinter.Label(self.main_window,
                                    text='Hello World!',
                                    borderwidth=1,
                                    relief='solid')

        self.label2 = tkinter.Label(self.main_window,
                                    text='This is my GUI program.',
                                    borderwidth=1,
                                    relief='solid')

        # Display the Labels with 20 pixels of horizontal
        # and vertical external padding.
        self.label1.pack(ipadx=20, ipady=20, padx=20, pady=20)
        self.label2.pack(ipadx=20, ipady=20, padx=20, pady=20)

        # Enter the tkinter main loop.
        tkinter.mainloop()

# Create an instance of the MyGUI class.
if __name__ == '__main__':
    my_gui = MyGUI()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Organizing Widgets with Frames (1 of 2)

- `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:

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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Organizing Widgets with Frames (2 of 2)

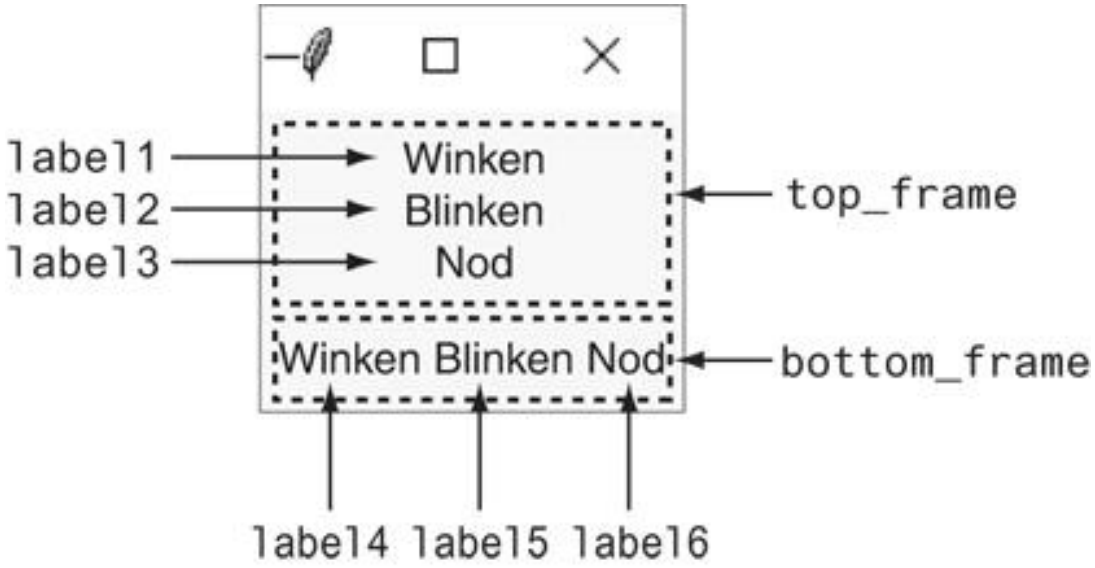


Figure 13-19 Arrangement of widgets

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

`Button` Widgets and Info Dialog Boxes

(1 of 4)

- `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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

`Button` Widgets and Info Dialog Boxes

(2 of 4)

- 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

`Button` Widgets and Info Dialog Boxes

(3 of 4)

Figure 13-20 The main window displayed by Program 13-10

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

`Button` Widgets and Info Dialog Boxes

(4 of 4)

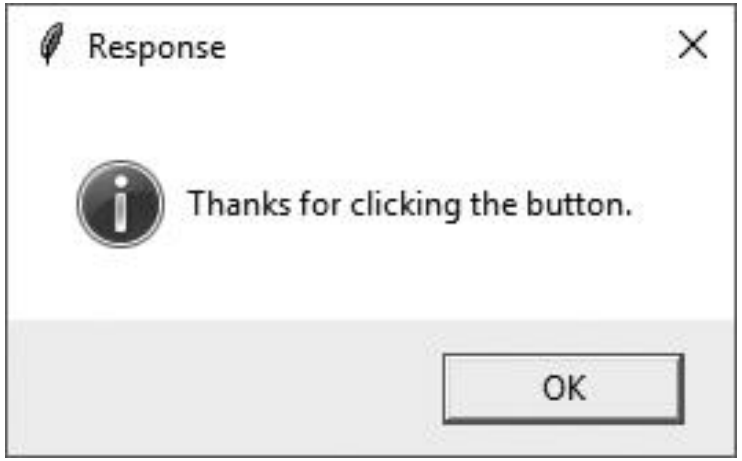


Figure 13-21 The info dialog box displayed by Program 13-10

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Getting Input with the `Entry` Widget

(1 of 2)

- `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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Getting Input with the `Entry` Widget

(2 of 2)

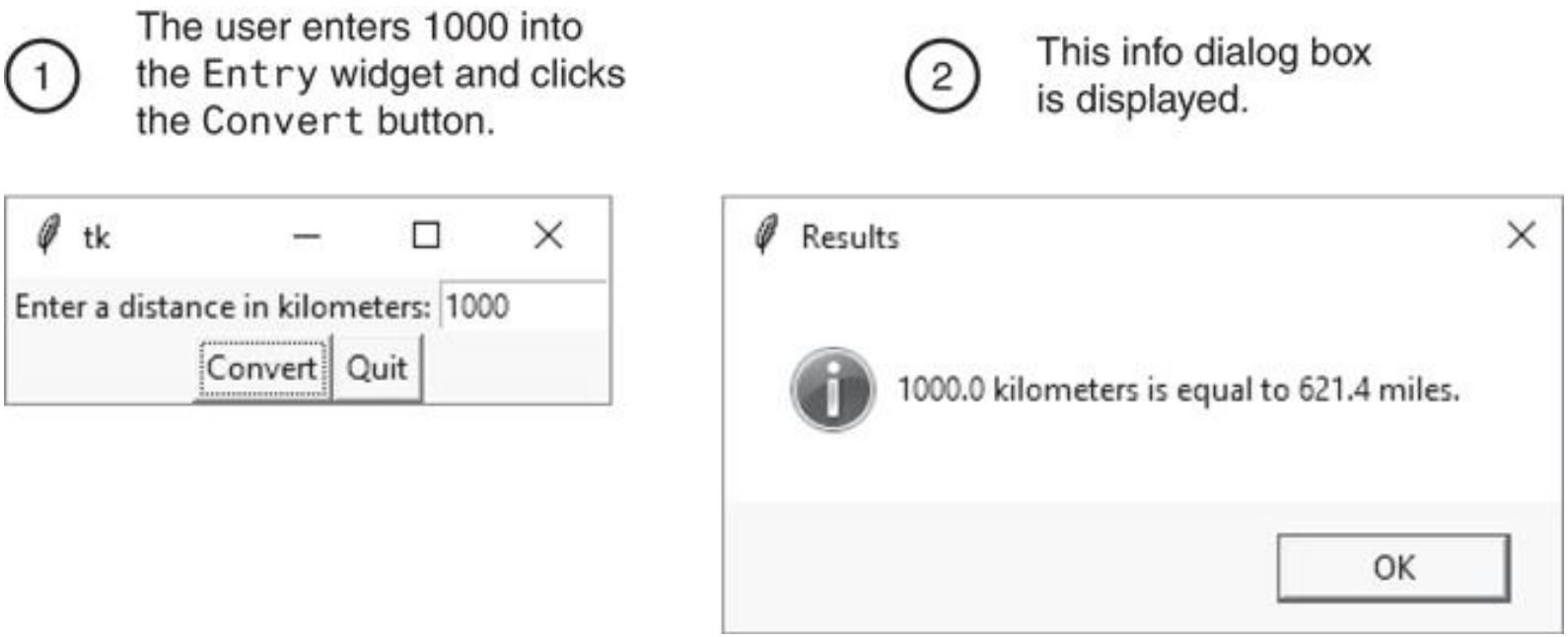


Figure 13-25 The info dialog box

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Using Labels as Output Fields (1 of 3)

- 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Using Labels as Output Fields (2 of 3)

- `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 with automatically be displayed in the `Label` widget

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Using Labels as Output Fields (3 of 3)

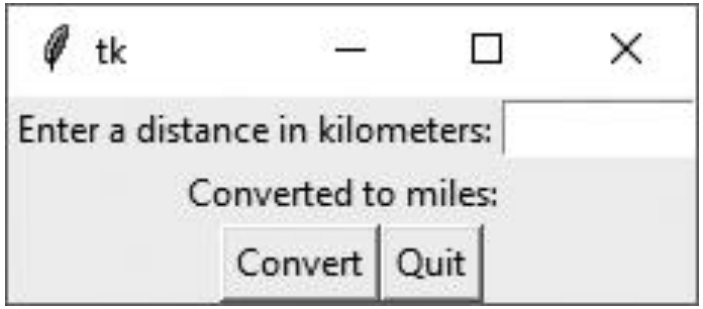


Figure 13-26 The window initially displayed

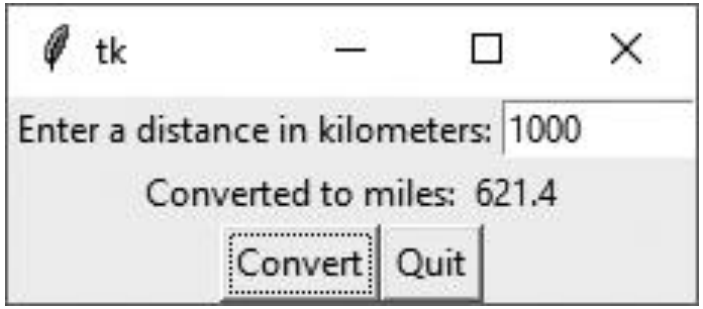


Figure 13-27 The window showing 1000 kilometers converted to miles

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Radio Buttons and Check Buttons (1 of 2)

- 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
- `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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Radio Buttons and Check Buttons (2 of 2)

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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

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
  - 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Check Buttons

- 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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

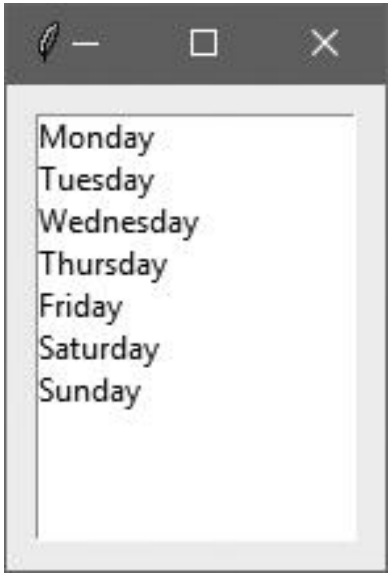
`Listbox` Widgets

- A `Listbox` widget displays a list of items and allows the user to select one or more items

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example

```
# This program demonstrates a simple Listbox.
import tkinter

class ListboxExample:
    def __init__(self):
        # Create the main window.
        self.main_window = tkinter.Tk()

        # Create a Listbox widget.
        self.listbox = tkinter.Listbox(self.main_window)
        self.listbox.pack(padx=10, pady=10)

        # Populate the Listbox with the data.
        self.listbox.insert(0, 'Monday')
        self.listbox.insert(1, 'Tuesday')
        self.listbox.insert(2, 'Wednesday')
        self.listbox.insert(3, 'Thursday')
        self.listbox.insert(4, 'Friday')
        self.listbox.insert(5, 'Saturday')
        self.listbox.insert(6, 'Sunday')

        # Start the main loop.
        tkinter.mainloop()

# Create an instance of the ListboxExample class.
if __name__ == '__main__':
    listbox_example = ListboxExample()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Specifying the Size of a `Listbox`

```
# This program demonstrates a simple Listbox.
import tkinter

class ListboxExample:
    def __init__(self):
        # Create the main window.
        self.main_window = tkinter.Tk()

        # Create a Listbox widget.
        self.listbox = tkinter.Listbox(
            self.main_window, height=0, width=0)
        self.listbox.pack(padx=10, pady=10)

        # Create a list with the days of the week.
        days = ['Monday', 'Tuesday', 'Wednesday',
                'Thursday', 'Friday', 'Saturday',
                'Sunday']

        # Populate the Listbox with the data.
        for day in days:
            self.listbox.insert(tkinter.END, day)

        # Start the main loop.
        tkinter.mainloop()

# Create an instance of the ListboxExample class.
if __name__ == '__main__':
     listbox_example = ListboxExample()
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Selection Modes

Mode	Description
`tkinter.BROWSE` ***(This is the default mode)***	The user can select one item at a time by clicking in the `Listbox`. Additionally, if the user clicks and drags the mouse inside the `Listbox`, the item that is currently under the cursor will be selected.
`tkinter.EXTENDED`	The user can select a group of adjacent items by clicking on the first item and dragging the mouse to the last item.
`tkinter.MULTIPLE`	Multiple items can be selected. When you click an unselected item, the item becomes selected. When you click a selected item, the item becomes unselected.
`tkinter.SINGLE`	The user can select one item at a time by clicking in the `Listbox`.

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Retrieving the Selected Item(s) (1 of 2)

- The `curselection` method returns a tuple containing the indexes of the items that are currently selected in the `Listbox`
  - If no item is selected, the tuple will be empty
  - If an item is selected and the selection mode is `tkinter.BROWSE` or `tkinter.SINGLE`, the tuple will contain only one element
  - If the `Listbox's` selection mode is `tkinter.EXTENDED` or `tkinter.MULTIPLE`, the tuple can contain multiple elements because those selection modes allow the user to select multiple items

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Retrieving the Selected Item(s) (2 of 2)

- Once you have the index of the selected item, you can use the `get` method to retrieve the selected item or items from the `Listbox`
- The `get` method accepts an integer index as its argument and returns the item that is located at that index in the `Listbox`

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Examples

```
self.listbox = tkinter.Listbox(self.main_window)

 (etc…)

index = self.listbox.curselection()
tkinter.messagebox.showinfo(self.listbox.get(index))
```

```
self.listbox = tkinter.Listbox(self.main_window,
                               selectmode=tkinter.EXTENDED)

 (etc…)

indexes = self.listbox.curselection()
for i in indexes:
    tkinter.messagebox.showinfo(self.listbox.get(i))
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Listboxes and Callback Functions

(1 of 3)

- You can optionally bind a callback function to a `Listbox`
- When the user clicks an item in the `Listbox`, the callback function is immediately executed

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Listboxes and Callback Functions

(2 of 3)

- Example callback function:
  - Assume `listbox` is a `Listbox` widget
  - You must provide a parameter to accept an event object as an argument

```
def show_item(self, event):
    index = self.listbox.curselection()
    item = self.listbox.get(index)
    tkinter.messagebox.showinfo('Selected Item', item)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Listboxes and Callback Functions

(3 of 3)

- To bind the `show_item` function to the `listbox` widget:

```
self.listbox.bind('<<ListboxSelect>>', self.show_item)
```

- Once done, the `show_item` function will execute whenever the user clicks an item in the `listbox` widget

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Adding a Vertical Scrollbar to a `Listbox`

1. Create a frame to hold the `Listbox` and the scrollbar
2. Pack the frame
3. Create a `Listbox` inside the frame
4. Pack the `Listbox` to the left side of the frame
5. Create the vertical scrollbar inside the frame
6. Pack the scrollbar to the right side of the frame
7. Configure the scrollbar to call the `Listbox's yview` method when the slider knob is moved
8. Configure the `Listbox` to call the scrollbar's `set` method any time the `Listbox` is updated

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (1 of 4)

```
# Create the main window.
self.main_window = tkinter.Tk()

# Create a frame for the Listbox and vertical scrollbar.
self.listbox_frame = tkinter.Frame(self.main_window)
self.listbox_frame.pack(padx=20, pady=20)

# Create a Listbox widget in the listbox_frame.
self.listbox = tkinter.Listbox(
    self.listbox_frame, height=6, width=0)
self.listbox.pack(side='left')

# Create a vertical Scrollbar in the listbox_frame.
self.scrollbar = tkinter.Scrollbar(
    self.listbox_frame, orient=tkinter.VERTICAL)
self.scrollbar.pack(side='right', fill=tkinter.Y)

# Configure the Scrollbar and Listbox to work together.
self.scrollbar.config(command=self.listbox.yview)
self.listbox.config(yscrollcommand=self.scrollbar.set)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Adding a Horizontal Scrollbar to a `Listbox`

1. Create a frame to hold the `Listbox` and the scrollbar
2. Pack the frame
3. Create a `Listbox` inside the frame
4. Pack the `Listbox` to the top of the frame
5. Create the horizontal scrollbar inside the frame
6. Pack the scrollbar to the bottom of the frame
7. Configure the scrollbar to call the `Listbox's xview` method when the slider knob is moved
8. Configure the `Listbox` to call the scrollbar's `set` method any time the `Listbox` is updated

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (2 of 4)

```
# Create the main window.
self.main_window = tkinter.Tk()

# Create a frame for the Listbox and scrollbar.
self.listbox_frame = tkinter.Frame(self.main_window)
self.listbox_frame.pack(padx=20, pady=20)

# Create a Listbox widget in the listbox_frame.
self.listbox = tkinter.Listbox(
    self.listbox_frame, height=0, width=30)
self.listbox.pack(side='top')

# Create a horizontal Scrollbar in the listbox_frame.
self.scrollbar = tkinter.Scrollbar(
    self.listbox_frame, orient=tkinter.HORIZONTAL)
self.scrollbar.pack(side='bottom', fill=tkinter.X)

# Configure the Scrollbar and Listbox to work together.
self.scrollbar.config(command=self.listbox.xview)
self.listbox.config(xscrollcommand=self.scrollbar.set)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Adding Vertical and Horizontal Scrollbars to a `Listbox` (1 of 2)

1. Create the outer frame to hold the inner frame and the horizontal scrollbar
2. Pack the outer frame
3. Create the inner frame to hold the `Listbox` and the vertical scrollbar
4. Pack the inner frame
5. Create a `Listbox` inside the inner frame
6. Pack the `Listbox` to the left side of the inner frame
7. Create a vertical scrollbar inside the inner frame
8. Pack the scrollbar to the right side of the inner frame
9. Create the horizontal scrollbar inside the outer frame

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Adding Vertical and Horizontal Scrollbars to a `Listbox` (2 of 2)

10. Pack the horizontal scrollbar to the bottom of the outer frame
11. Configure the vertical scrollbar to call the `Listbox's yview` method when the slider knob is moved
12. Configure the horizontal scrollbar to call the `Listbox's xview` method when the slider knob is moved
13. Configure the `Listbox` to call both scrollbar's `set` method any time the `Listbox` is updated

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (3 of 4)

```
# Create the main window.
self.main_window = tkinter.Tk()

# Create an outer frame to hold the inner frame
# and the horizontal scrollbar.
self.outer_frame = tkinter.Frame(self.main_window)
self.outer_frame.pack(padx=20, pady=20)

# Create an inner frame for the Listbox and vertical scrollbar.
self.inner_frame = tkinter.Frame(self.outer_frame)
self.inner_frame.pack()

# Create a Listbox widget in the inner_frame.
self.listbox = tkinter.Listbox(
    self.inner_frame, height=5, width=30)
self.listbox.pack(side='left')

# Create a vertical Scrollbar in the inner_frame.
self.v_scrollbar = tkinter.Scrollbar(
    self.inner_frame, orient=tkinter.VERTICAL)
self.v_scrollbar.pack(side='right', fill=tkinter.Y)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Example (4 of 4)

```
# Create a horizontal Scrollbar in the outer_frame.
self.h_scrollbar = tkinter.Scrollbar(
    self.outer_frame, orient=tkinter.HORIZONTAL)
self.h_scrollbar.pack(side='bottom', fill=tkinter.X)

# Configure the Scrollbars and the Listbox to work together.
self.v_scrollbar.config(command=self.listbox.yview)
self.h_scrollbar.config(command=self.listbox.xview)
self.listbox.config(yscrollcommand=self.v_scrollbar.set,
                    xscrollcommand=self.h_scrollbar.set)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing Shapes with the `Canvas` Widget (1 of 4)

- 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.

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing Shapes with the `Canvas` Widget (2 of 4)

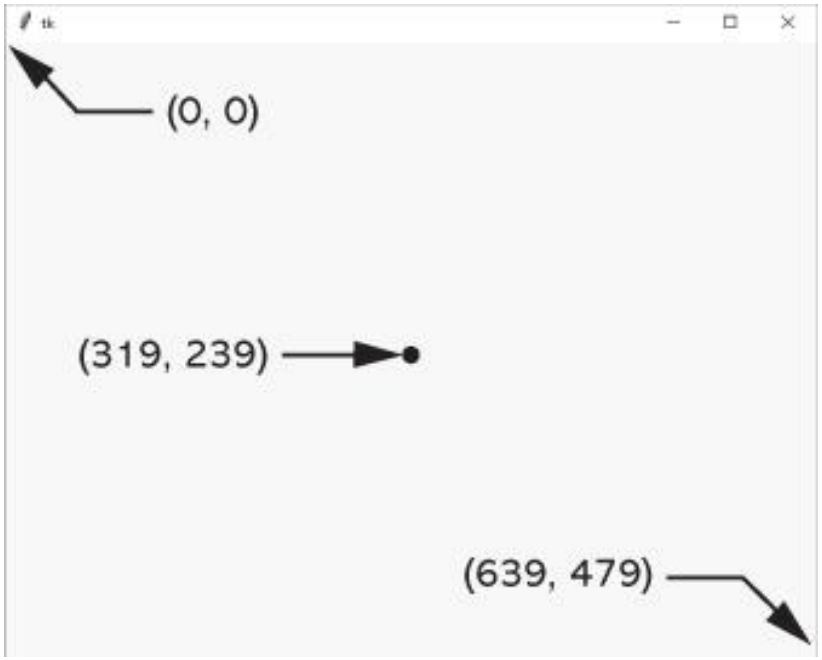


Figure 13-45 Various pixel locations in a 640 by 480 window

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing Shapes with the `Canvas` Widget (3 of 4)

- 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)
```

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing Shapes with the `Canvas` Widget (4 of 4)

- 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_oval`
  - `create_arc`
  - `create_polygon`
  - `create_text`

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Line (1 of 2)

Coordinates of the line's ending point

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

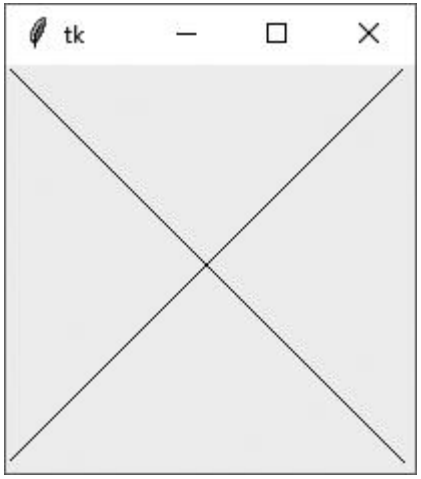
Coordinates of the line's starting point

Optional arguments (See Table 13-2)

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Line (2 of 2)

```
# This program demonstrates the Canvas widget.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

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

            # Draw two lines.
            self.canvas.create_line(0, 0, 199, 199)
            self.canvas.create_line(199, 0, 0, 199)

            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
if _ _name_ _ == '_ _main_ _':
        my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Rectangle (1 of 2)

Coordinates of the lower-right corner

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

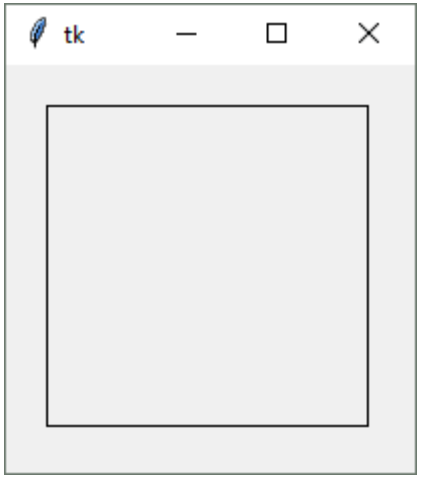
Coordinates of the upper-left corner

Optional arguments (See Table 13-3)

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Rectangle (2 of 2)

```
# This program draws a rectangle on a Canvas.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

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

            # Draw two lines.
            self.canvas.create_line(0, 0, 199, 199)
            self.canvas.create_line(199, 0, 0, 199)

            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

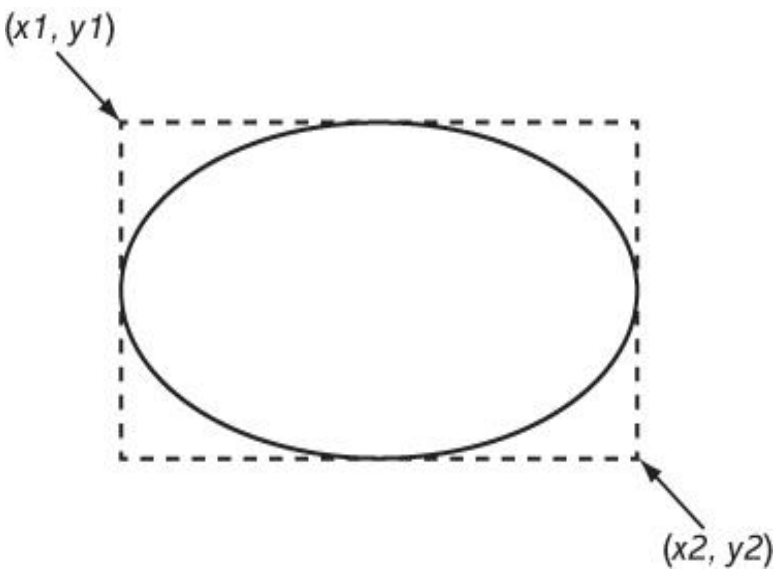
Drawing an Oval (1 of 2)

Coordinates of the lower-right corner of bounding rectangle

```
canvas_name.create_oval(x1, y1, x2, y2, options…)
```

Coordinates of the upper-left corner of bounding rectangle

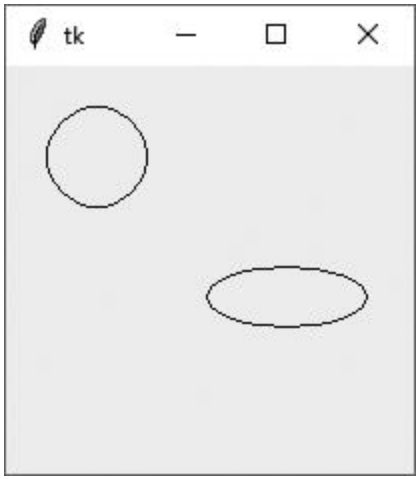
Optional arguments (See Table 13-4)



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing an Oval (2 of 2)

```
# This program draws a rectangle on a Canvas.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

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

            # Draw two ovals.
            self.canvas.create_oval(20, 20, 70, 70)
            self.canvas.create_oval(100, 100, 180, 130)

            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
if _ _name_ _ == '_ _main_ _':
my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing an Arc (1 of 2)

Coordinates of the upper-left corner of bounding rectangle

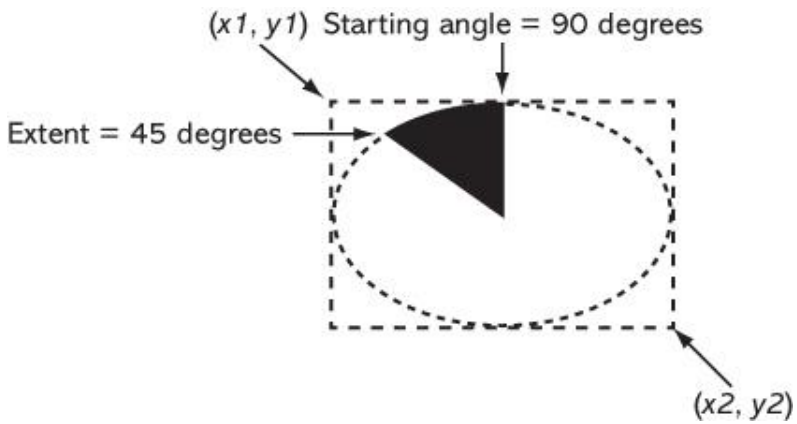
Coordinates of the lower-right corner of bounding rectangle

```
canvas_name.create_arc(x1, y1, x2, y2,
                       start=angle, extent=width,
                       options…)
```

Starting angle

Counter clockwise extent of the arc

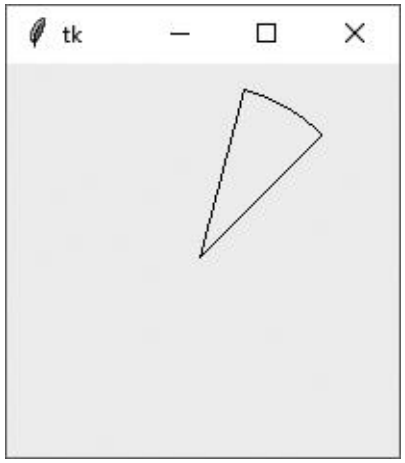
Optional arguments (See Table 13-5)



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing an Arc (2 of 2)

```
# This program draws an arc on a Canvas.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

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

            # Draw an arc.
            self.canvas.create_arc(10, 10, 190, 190, start=45, extent=30)

            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
if _ _name_ _ == '_ _main_ _':
        my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Polygon (1 of 3)

Coordinates of the second vertex

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

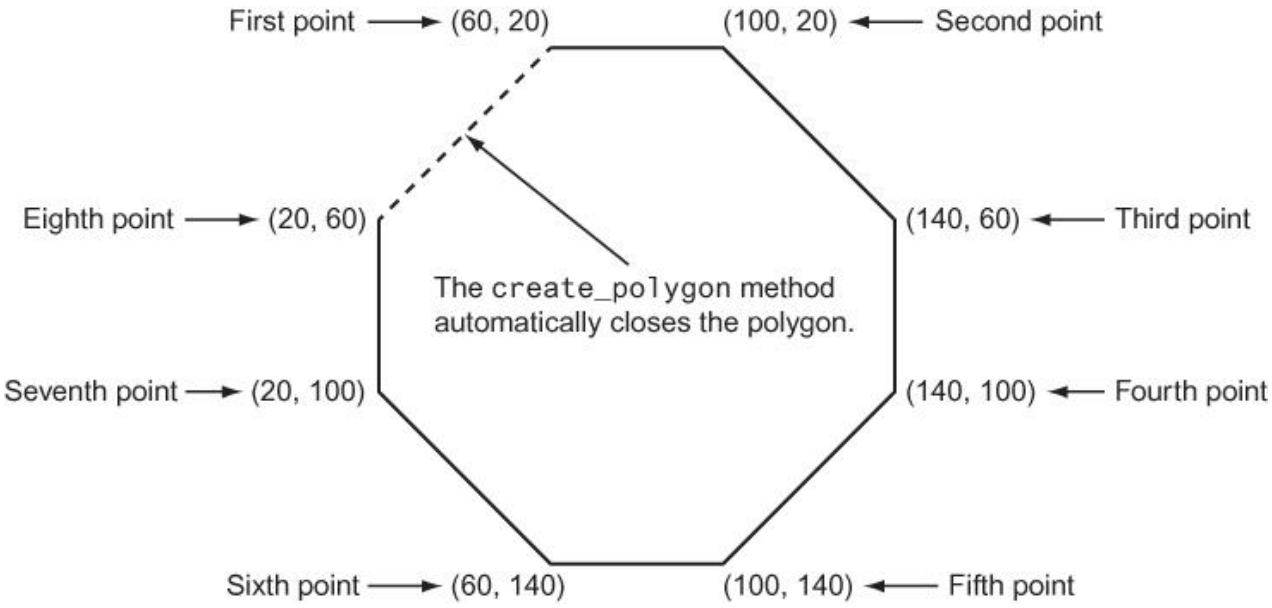
Coordinates of the first vertex

Optional arguments (See Table 13-7)

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Polygon (2 of 3)

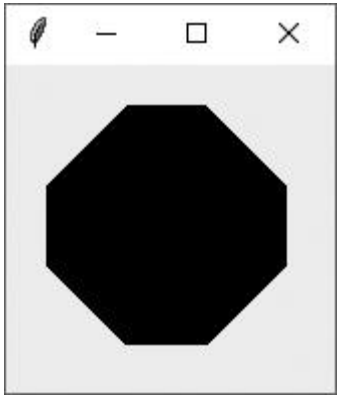
```
self.canvas.create_polygon(60, 20, 100, 20, 140, 60, 140, 100,
                           100, 140, 60, 140, 20, 100, 20, 60)
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Drawing a Polygon (3 of 3)

```
# This program draws a polygon on a Canvas.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

            # Create the Canvas widget.
            self.canvas = tkinter.Canvas(self.main_window, width=160,
            height=160)

            # Draw a polygon.
            self.canvas.create_polygon(60, 20, 100100, 140, 60, 140, 20,
                                       100, 20, 60, 20, 140, 60, 140, 100,)


            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
if _ _name_ _ == '_ _main_ _':
        my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Displaying Text on the Canvas (1 of 2)

Text to display

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

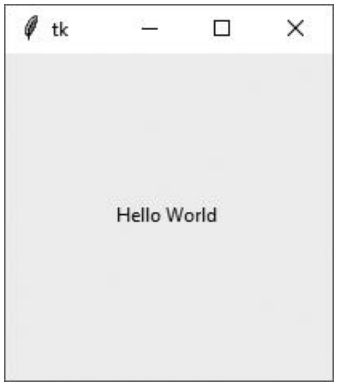
Coordinates of the text's insertion point

Optional arguments (See Table 13-8)

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

Displaying Text on the Canvas (2 of 2)

```
# This program draws a polygon on a Canvas.
import tkinter

class MyGUI:
        def _ _init_ _(self):
            # Create the main window.
            self.main_window = tkinter.Tk()

            # Create the Canvas widget.
            self.canvas = tkinter.Canvas(self.main_window, width=160,
            height=160)

            # Display text in the center of the window.
            self.canvas.create_text(100, 100, text='Hello World')

            # Pack the canvas.
            self.canvas.pack()

            # Start the mainloop.
            tkinter.mainloop()

# Create an instance of the MyGUI class.
if _ _name_ _ == '_ _main_ _':
        my_gui = MyGUI()
```



Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved

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

Pearson Copyright © 2021, 2018, 2015 Pearson Education, Inc. All Rights Reserved