

Course Name	ITD 2313 – Script Programming
Instructor	Andy Tripp
Student Name	Timothy Obinda
Due date	11/09/2025
Grade	Put grade earned here
Grading Comments	Put instructor comments here

INSTRUCTIONS FOR THE EXERCISE

You should always read the instructions in full. It is best to do a full read through before starting the assignment. Each screenshot needs to be appropriately labeled.

In these instructions, you are given information to execute specific examples in the text. You are given the Section, subsection, and a page number to identify a set of steps. On each page listed, there will be 1 or more numbered tasks to perform. These numbered tasks will be what you are to type in, execute, and then grab the screen shots of. Those screenshots will then go into your submission document.

Some advice, copy this instruction set into your submission document and then put the screen shots under each numbered task. Each individual book page in the instructions should be in a different screen shot. For any single book page, you may have all the numbered tasks on that page to be in a single screen shot.

Windows and Window Components

Windows Layout

Page 230-231

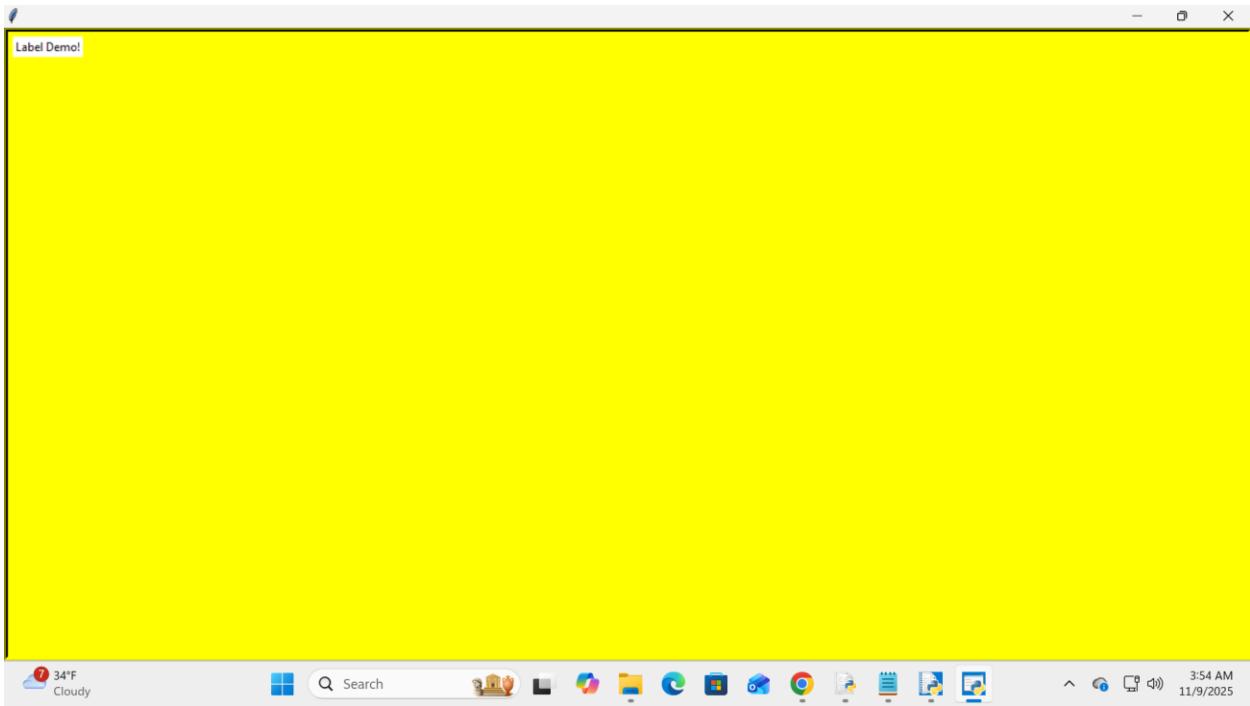
1. Use the code example block on the bottom of page 230 to start this set of instructions. You may need to review the previous pages to set up the GUI environment in Python correctly. Show in this first screenshot the grid layout results. Expand out the window to show the stuck in the corner the text mentions.

A screenshot of a Windows desktop environment. At the top is a standard window title bar for 'labeldemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\labeldemo.py (3.13.9)'. Below the title bar is a menu bar with File, Edit, Format, Run, Options, Window, and Help. The main area contains Python code for a 'LabelDemo' class using the 'breezypythongui' library. The code includes an __init__ method that creates a window and adds a label with the text 'Label Demo!', and a main method that runs the application. A conditional statement checks if the script is run directly. The code is color-coded for syntax. At the bottom of the screen is a taskbar with various pinned icons, including a weather widget showing '34°F Cloudy', a search bar, and icons for File Explorer, Edge, and other Microsoft applications. The system tray shows the date and time as '11/9/2025 3:51 AM'.

```
labeldemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\labeldemo.py (3.13.9)
File Edit Format Run Options Window Help
"""
File: labeldemo.py
"""

from breezypythongui import EasyFrame
class LabelDemo(EasyFrame):
    """Displays a greeting in a window."""
    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self, width = 300, height = 200)
        self.addLabel(text = "Label Demo!", row = 0, column = 0)
    def main():
        """Instantiates and pops up the window."""
        LabelDemo().mainloop()
if __name__ == "__main__":
    main()

Ln: 13 Col: 40
```



2. Using the code example blocks in the middle and at the bottom of page 231, change the look and feel to that of the centered look that is shown on the top of page 232. Expand out the window to show the difference between step 1 and this step output.

```
* *labeldemo.py - C:\Users\student\Downloads\Ch_09_Data_Files (1)\Ch_09_Data_Files\labeldemo.py (3.13.9)*
File Edit Format Run Options Window Help
File: labeldemo.py
"""

from breezypythongui import EasyFrame
class LabelDemo(EasyFrame):
    """Displays a greeting in a window."""
    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self)
        self.addLabel(text = "(0, 0)", row = 0, column = 0)
        self.addLabel (text = "(0, 1)", row = 0, column = 1)
        self.addLabel (text = "(1, 0)", row = 1, column = 0)
        self.addLabel (text = "(1, 1)", row = 1, column = 1)

    def main():
        """Instantiates and pops up the window."""
        LabelDemo().mainloop()

if __name__ == "__main__":
    main()

Ln: 16 Col: 60
34°F Cloudy Search File Explorer Edge File History 4:13 AM 11/9/2025
```

A screenshot of a Windows desktop environment. A code editor window is open, showing Python code for a GUI application named "labeldemo.py". The code uses the "breezypythongui" module to create a window with four labels arranged in a 2x2 grid. The application is currently running, as indicated by the window title and the fact that the code editor is displaying the running state. Below the code editor is the Windows taskbar, which includes a weather widget showing "34°F Cloudy", a search bar, and various pinned icons for applications like File Explorer, Edge, and File History. The system tray shows the date and time as "11/9/2025 4:13 AM".

The screenshot shows a Windows desktop with two code editors open side-by-side. Both editors display the same Python script, 'labeldemo.py', which uses the breezypythongui library to create a window with four labels at specific coordinates.

```
File: labeldemo.py
"""
File: labeldemo.py
"""

from breezypythongui import EasyFrame
class LabelDemo(EasyFrame):
    """Displays a greeting in a window."""
    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self)
        self.addLabel(text = "(0, 0)", row = 0, column = 0
                      sticky = "NSEW")
        self.addLabel(text = "(0, 1)", row = 0, column = 1
                      sticky = "NSEW")
        self.addLabel(text = "(1, 0)", row = 1, column = 0
                      sticky = "NSEW")
        self.addLabel(text = "(1, 1)", row = 1, column = 1
                      sticky = "NSWE")

    def main():
        """Instantiates and pops up the window."""
        LabelDemo().mainloop()

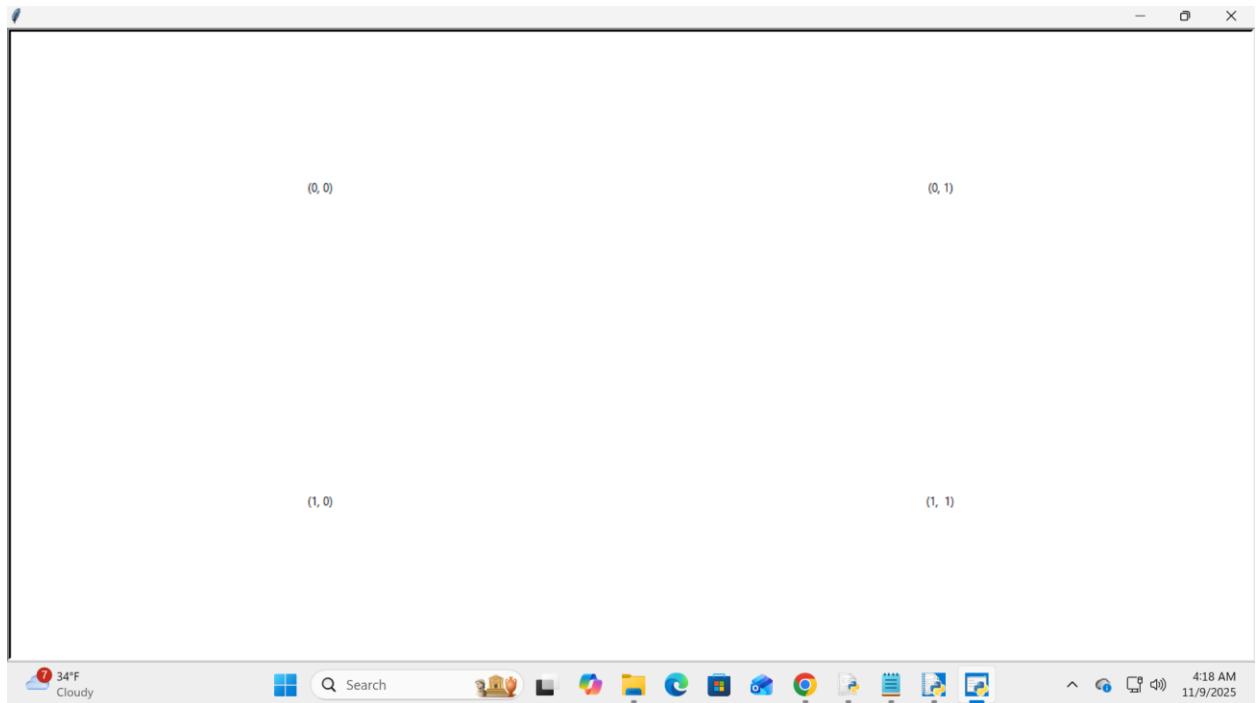
if __name__ == "__main__":
    main()
```

The script defines a class 'LabelDemo' that inherits from 'EasyFrame'. It contains a constructor '__init__' that adds four labels to the window at coordinates (0, 0), (0, 1), (1, 0), and (1, 1) respectively. The labels are sticky to all four sides ('NSEW' or 'NSWE'). The 'main' method creates an instance of 'LabelDemo' and starts the main loop. The script is run from the command line, and the resulting window is shown in both editors.

```
labeldemo.py - C:\Users\student\Downloads\Ch_09_Data_Files (1)\Ch_09_Data_Files\labeldemo.py (3.13.9)
File Edit Format Run Options Window Help
File: labeldemo.py

"""
from breezypythongui import EasyFrame
class LabelDemo(EasyFrame):
    """Displays a greeting in a window."""
    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self)
        self.addLabel(text = "(0, 0)", row = 0, column = 0,
                      sticky = "NSEW")
        self.addLabel(text = "(0, 1)", row = 0, column = 1,
                      sticky = "NSEW")
        self.addLabel(text = "(1, 0)", row = 1, column = 0,
                      sticky = "NSEW")
        self.addLabel(text = "(1, 1)", row = 1, column = 1,
                      sticky = "NSEW")
    def main():
        """Instantiates and pops up the window."""
        LabelDemo().mainloop()
if __name__ == "__main__":
    main()

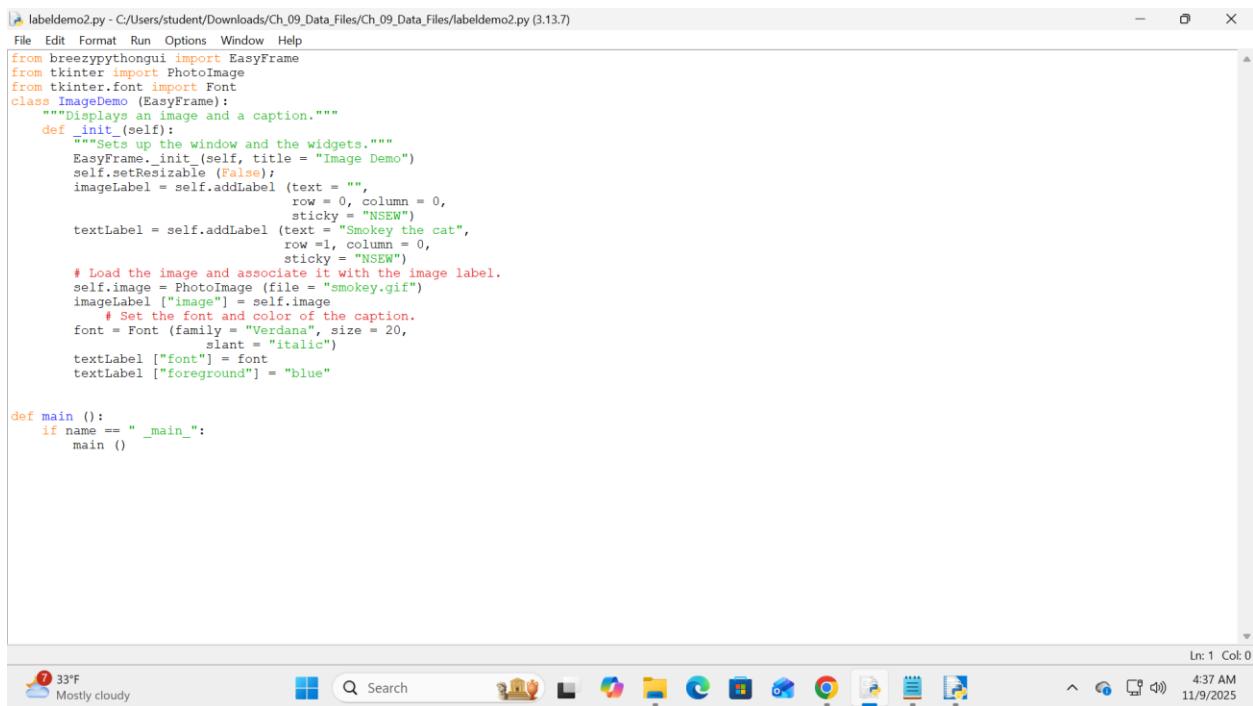
Ln: 10 Col: 0
34°F Cloudy Search 4:19 AM 11/9/2025
```



Displaying Images

Page 233 – 234

1. Smokey the Cat is back. Using the code example block on 233 and spills over to the top of page 234, get the Smokey picture to show up like it does at the top of page 233.



The screenshot shows a Windows desktop environment. In the center is a code editor window titled "labeldemo2.py - C:/Users/student/Downloads/Ch_09_Data_Files/Ch_09_Data_Files/labeldemo2.py (3.13.7)". The code is written in Python using the Tkinter library to create a window with a label and a photo. The code includes comments explaining the setup of the window, the addition of a text label with a caption, and the loading of an image (smokey.gif) which is then associated with the image label. The code also sets the font and color for the text label. A main function checks if the script is run directly and calls the main function. At the bottom of the screen, the taskbar displays various icons for system functions like weather, search, and file explorer, along with the date and time (4:37 AM, 11/9/2025).

```
labeldemo2.py - C:/Users/student/Downloads/Ch_09_Data_Files/Ch_09_Data_Files/labeldemo2.py (3.13.7)
File Edit Format Run Options Window Help
from breezypythongui import EasyFrame
from tkinter import PhotoImage
from tkinter.font import Font
class ImageDemo (EasyFrame):
    """Displays an image and a caption."""
    def __init__(self):
        """Sets up the window and the widgets."""
        EasyFrame.__init__(self, title = "Image Demo")
        self.setResizable (False)
        imageLabel = self.addLabel (text = "",
                                   row = 0, column = 0,
                                   sticky = "NSEW")
        textLabel = self.addLabel (text = "Smokey the cat",
                                   row =1, column = 0,
                                   sticky = "NSEW")
        # Load the image and associate it with the image label.
        self.image = PhotoImage (file = "smokey.gif")
        imageLabel ["image"] = self.image
        # Set the font and color of the caption.
        font = Font (family = "Verdana", size = 20,
                    slant = "italic")
        textLabel ["font"] = font
        textLabel ["foreground"] = "blue"

def main () :
    if name == "__main__":
        main ()
```

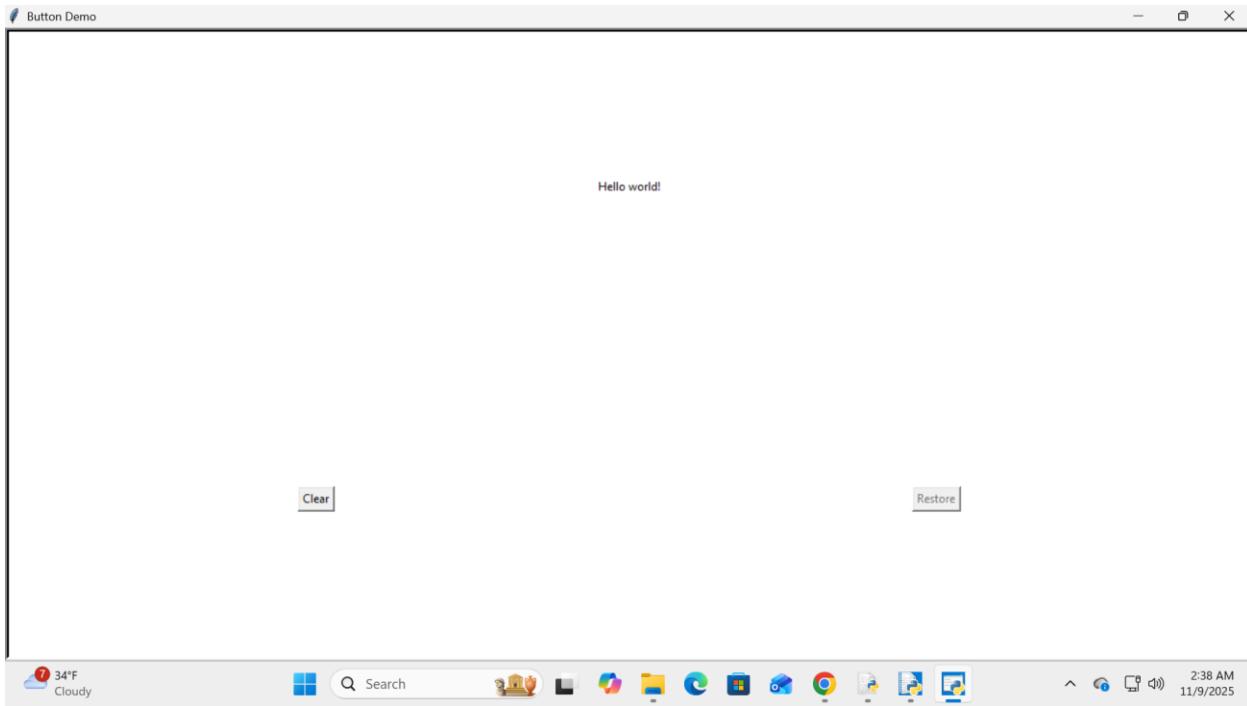
The screenshot shows a Windows desktop environment. At the top is the taskbar with various pinned icons. In the center is the "IDLE Shell 3.13.7" window, which is a Python shell interface. The window title bar says "IDLE Shell 3.13.7". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The status bar at the bottom right of the window shows "Ln: 5 Col: 0". The main pane of the window displays the following Python code:

```
>>> Python 3.13.7 (tags/v3.13.7:fbceefc3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> = RESTART: C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\labeldemo2.py
>>> |
```

Command Buttons and Responding to events

Pages 235 – 236

1. Use the code example blocks on the pages to get the Hello World program and output seen on page 235 in the middle of the page.



Input and Output with Entry Fields

Text Fields

Page 237 – 238

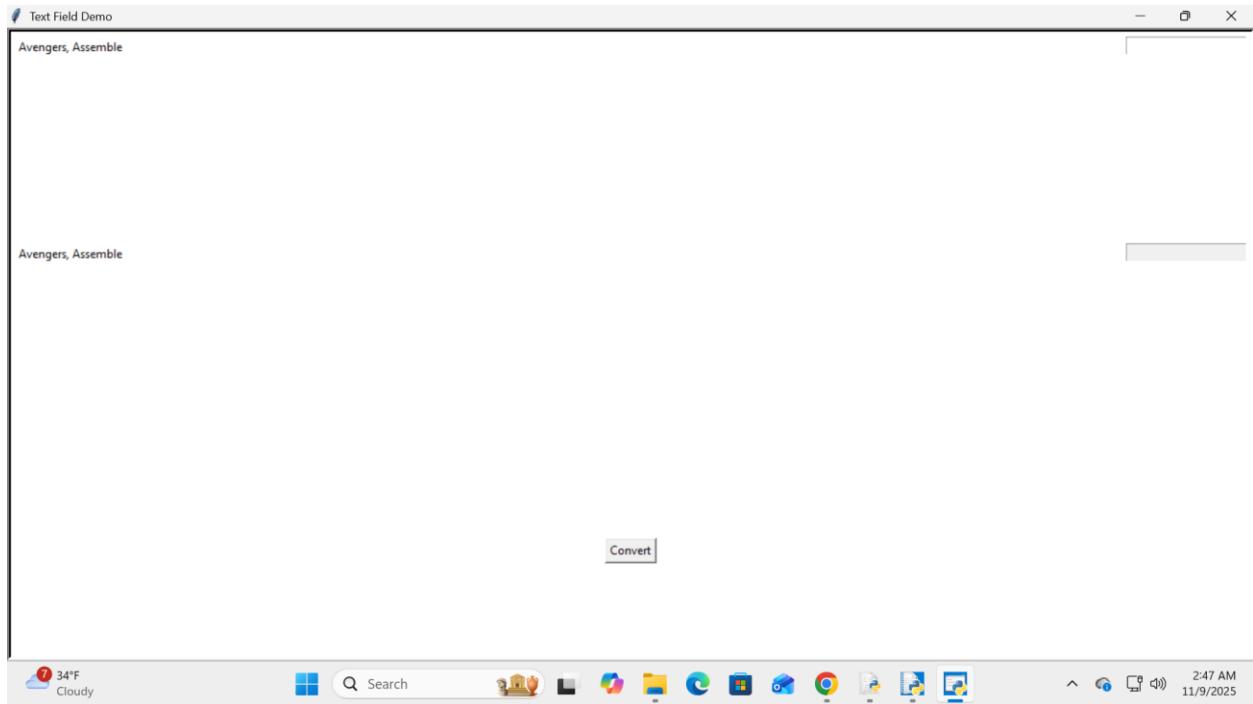
1. Using the code example blocks on these pages, produce the input/output example on the middle page 237. Grab the screenshot with a phrase that is your favorite inspirational quote of all time instead of the phrase seen in the text.

The screenshot shows a Windows desktop environment. In the center is a code editor window titled "textfielddemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\textfielddemo.py (3.13.9)". The window contains Python code for a GUI application using the breezypythongui library. The code defines a class "TextFieldDemo" that creates a window with an input field, an output field, and a button to convert input to uppercase. The code is well-commented with docstrings. At the bottom of the code editor, status bars show "Ln: 18 Col: 0" and "2:48 AM 11/9/2025". Below the code editor is the Windows taskbar, which includes icons for weather (34°F Cloudy), search, and various pinned applications like File Explorer, Edge, and others.

```
textfielddemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\textfielddemo.py (3.13.9)
File Edit Format Run Options Window Help
"""
File: textfielddemo.py
"""

from breezypythongui import EasyFrame
class TextFieldDemo(EasyFrame):
    """Converts an input string to uppercase and displays the result."""
    def __init__(self):
        """Sets up the window and widgets."""
        EasyFrame.__init__(self, title = "Text Field Demo")
        # Label and field for the input
        self.addLabel(text = "Avengers, Assemble",
                      row = 0, column = 0)
        self.inputField = self.addTextField(text = "",
                                           row = 0,
                                           column = 1)
        # Label and field for the output
        self.addLabel(text = "Avengers, Assemble",
                      row = 1, column = 0)
        self.outputField = self.addTextField(text = "",
                                           row = 1,
                                           column = 1,
                                           state = "readonly")
        # The command button
        self.button = self.addButton(text = "Convert",
                                    row = 2, column = 0,
                                    columnspan = 2,
                                    command = self.convert)
    # The event handling method for the button
    def convert(self):
        """Inputs the string, converts it to uppercase,
        and outputs the result."""
        text = self.inputField.getText()
        result = text.upper()
        self.outputField.setText(result)

Ln: 18 Col: 0
Cloudy 34°F
Search 2:48 AM
11/9/2025
```



Integer and Float Fields for Numeric Data

Pages 238 – 239

1. Using the code example blocks on these two pages, produce the output seen on the bottom of page 238 except use the number 1941.002 as the number entered and show in the program instead of what is shown in the text.

```
numberfielddemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\numberfielddemo.py (3.13.9)
File Edit Format Run Options Window Help
def __init__(self):
    """Sets up the window and widgets."""
    EasyFrame.__init__(self, title = """Number Field Demo""")

    # Label and field for the input
    self.addLabel(text = "An integer",
                  row = 0, column = 0)
    self.inputField = self.addIntegerField(value = 1941.002,
                                           row = 0,
                                           column = 1,
                                           width = 10)

    # Label and field for the output
    self.addLabel(text = "Square root",
                  row = 1, column = 0)
    self.outputField = self.addFloatField(value = 0.0,
                                          row = 1,
                                          column = 1,
                                          width = 8,
                                          precision = 2,
                                          state = "readonly")

    # The command button
    self.addButton(text = "Compute", row = 2, column = 0,
                  columnspan = 2, command = self.computeSqrt)

    # The event handling method for the button
    def computeSqrt(self):
        """Inputs the integer, computes the square root,
        and outputs the result."""
        number = self.inputField.getNumber()
        result = math.sqrt(number)
        self.outputField.setNumber(result)

def main():
    """Instantiate and pop up the window."""
    NumberFieldDemo().mainloop()

if __name__ == "__main__":
    main()
```

Ln: 18 Col: 63



```
numberfielddemo.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\numberfielddemo.py (3.13.9)
File | *IDLE Shell 3.13.9*
File Edit Shell Debug Options Window Help
Pyt .13.9:8183fa5, Oct 14 2025, 14:09:13) [MSC v.1944 64 bit (a
AMI En click "Help" above for more information.
= I ident\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\numberfi
else
    Square root 0.0
    Compute
```

Ln: 18 Col: 63



Fail-Safe Programming

Page 245 – 246

1. Using the code example blocks on the pages, produce the output shown on page 246. You may use the text book's example for this one if you so choose.

```
guessversion1.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\guessversion1.py (3.13.9)
File Edit Format Run Options Window Help
"""
File: guessversion1.py
Lays out the user interface for a GUI-based
guessing game.
"""

import random
from breezypythongui import EasyFrame

class GuessingGame(EasyFrame):
    """Plays a guessing game with the user."""

    def __init__(self):
        """Sets up the window, widgets, and data."""
        EasyFrame.__init__(self, title = "Gussing Game")
        self.myNumber = random.randint(1, 100)
        self.count = 0
        greeting = "Guess a number between 1 and 100."
        self.hintLabel = self.addLabel(text = greeting,
                                       row = 0, column = 0,
                                       sticky = "NSEW",
                                       colspan = 2)
        self.addLabel(text = "Your guess", row = 1, column = 0)
        self.guessField = self.addIntegerField(0, row = 1, column = 1)
        self.nextButton = self.addButton(text = "Next", row = 2, column = 0)
        self.newButton = self.addButton(text = "New game", row = 2, column = 1)

    def main():
        """Instantiate and pop up the window."""
        GuessingGame().mainloop()

if __name__ == "__main__":
    main()
```

guessversion1.py - C:\Users\student\Downloads\Ch_09_Data_Files\Ch_09_Data_Files\guessversion1.py (3.13.9)

IDLE Shell 3.13.9*

```
File Edit Shell Debug Options Window Help
Pyt AMI Guss... - □ X
guessversion1.py, Oct 14 2025, 14:09:13) [MSC v.1944 64 bit (AMD64)]
Ent Help" above for more information.

imp>>> from Tkinter import *
from random import randint
class GuessingGame:
    def __init__(self):
        self.myNumber = randint(1,100)
        self.count = 0
        self.guessField = Entry(self.root)
        self.guessField.pack()
        self.nextButton = Button(self.root, text="Next", command=self.nextGuess)
        self.nextButton.pack(side="left")
        self.newGameButton = Button(self.root, text="New game", command=self.newGame)
        self.newGameButton.pack(side="right")
        self.hintLabel = Label(self.root, text="Guess a number between 1 and 100.")
        self.hintLabel.pack()
        self.guessField.focus_set()

    def nextGuess(self):
        """Processes the user's next guess."""
        self.count += 1
        guess = self.guessField.getNumber()
        if guess == self.myNumber:
            self.hintLabel["text"] = "You've guessed it in " + \
                str(self.count) + " attempts!"
            self.nextButton["state"] = "disabled"
        elif guess < self.myNumber:
            self.hintLabel["text"] = "Sorry, too small!"
        else:
            self.hintLabel["text"] = "Sorry, too large!"

    def newGame(self):
        """Resets the data and GUI to their original states."""
        self.myNumber = random.randint(1,100)
        self.count = 0
        greeting = "Guess a number between 1 and 100."
        self.hintLabel["text"] = greeting
        self.guessField.setNumber(0)
        self.nextButton["state"] = "normal"

    def getNumber(self):
        return int(self.guessField.get())

    def setNumber(self, value):
        self.guessField.delete(0, "end")
        self.guessField.insert(0, value)

    def main():
        root = Tk()
        game = GuessingGame()
        root.mainloop()

if __name__ == "__main__":
    main()
```

Guess a number between 1 and 100.

Your guess: 0

Next New game

Ln: 1 Col: 0

34°F Cloudy Search 3:08 AM 11/9/2025

selfguess.py - C:\Users\student\Desktop\selfguess.py (3.13.7)

File Edit Format Run Options Window Help

```
def nextGuess (self):
    """Processes the user's next guess."""
    self.count += 1
    guess = self.guessField.getNumber ()
    if guess == self.myNumber:
        self.hintLabel ["text"] = "You've guessed it in " + \
            str (self.count) + " attempts!"
        self.nextButton ["state"] = "disabled"
    elif guess < self.myNumber:
        self.hintLabel ["text"] = "Sorry, too small!"
    else:
        self.hintLabel ["text"] = "Sorry, too large!"

def newGame (self):
    """Resets the data and GUI to their original states."""
    self.myNumber = random.randint (1,100)
    self.count = 0
    greeting = "Guess a number between 1 and 100."
    self.hintLabel ["text"] = greeting
    self.guessField.setNumber (0)
    self.nextButton ["state"] = "normal"

    def getNumber(self):
        return int(self.guessField.get())

    def setNumber(self, value):
        self.guessField.delete(0, "end")
        self.guessField.insert(0, value)

    def main():
        root = Tk()
        game = GuessingGame()
        root.mainloop()

if __name__ == "__main__":
    main()
```

Ln: 1 Col: 0

34°F Cloudy Search 3:20 AM 11/9/2025

The screenshot shows a Windows desktop environment. At the top is the taskbar with various icons for apps like File Explorer, Edge, and Mail. In the center is the Windows Start button. Below the taskbar is the desktop background. A window titled "IDLE Shell 3.13.7" is open, showing Python code. The code at the top reads:

```
File Edit Shell Debug Options Window Help
Python 3.13.7 (tags/v3.13.7-bceelc3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> ===== RESTART: C:\Users\student\Desktop\selfguess.py =====
>>>
```

The status bar at the bottom of the IDLE window shows "Ln: 5 Col: 0".

You may find the other examples of code in the chapter to be very useful to work through but no screen shots will be required of them. You are highly encouraged to experiment with the various topics that are not specifically assigned in this assignment.