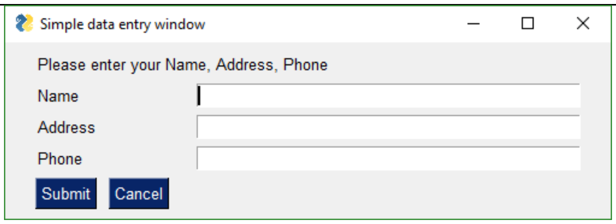
Recipe	Window Produced

Result as List

```
import PySimpleGUI as sg

# Very basic window.  Return values as a list

layout = [
            [sg.Text('Please enter your Name, Address, Phone')],
            [sg.Text('Name', size=(15, 1)), sg.InputText()],
            [sg.Text('Address', size=(15, 1)), sg.InputText()],
            [sg.Text('Phone', size=(15, 1)), sg.InputText()],
            [sg.Submit(), sg.Cancel()]
         ]

window = sg.Window('Simple data entry window').Layout(layout)
button, values = window.Read()
```


Simple data entry window

Please enter your Name, Address, Phone

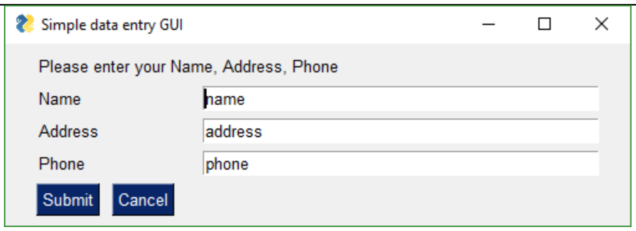
Name

Address

Phone

Result as Dictionary

```
import PySimpleGUI as sg

# Very basic window.  Return values as a dictionary

layout = [
            [sg.Text('Please enter your Name, Address, Phone')],
            [sg.Text('Name', size=(15, 1)), sg.InputText('name',
key='name')],
            [sg.Text('Address', size=(15, 1)), sg.InputText('address',
key='address')],
            [sg.Text('Phone', size=(15, 1)), sg.InputText('phone',
key='phone')],
            [sg.Submit(), sg.Cancel()]
         ]

window = sg.Window('Simple data entry GUI').Layout(layout)

button, values = window.Read()

print(button, values['name'], values['address'], values['phone'])
```


Simple data entry GUI

Please enter your Name, Address, Phone

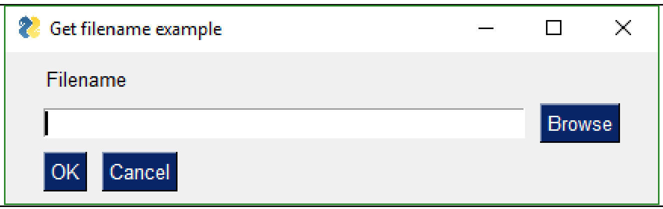
Name: name

Address: address

Phone: phone

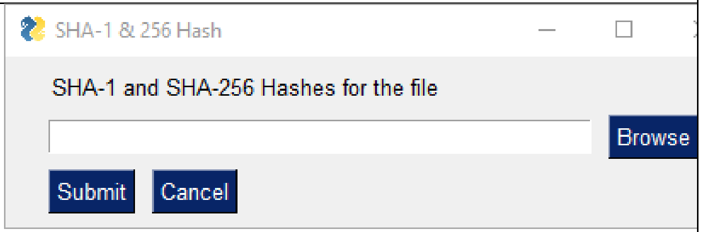
Single-Line Front-End Get Filename

```python
import PySimpleGUI as sg

button, (filename,) = sg.Window('Get filename example').Layout(
    [[sg.Text('Filename')], [sg.Input(), sg.FileBrowse()], [sg.OK(),
sg.Cancel()]]).Read()
```



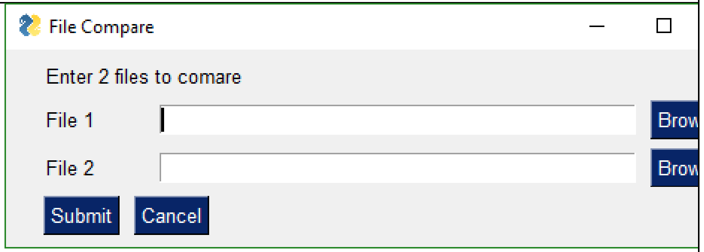
Front-End Get Filename

```python
import PySimpleGUI as sg

layout = [
            [sg.Text('SHA-1 and SHA-256 Hashes for the file')],
            [sg.InputText(), sg.FileBrowse()],
            [sg.Submit(), sg.Cancel()]
         ]

(button, (source_filename,)) = sg.Window('SHA-1 & 256
Hash').Layout(layout).Read()

print(button, source_filename)
```



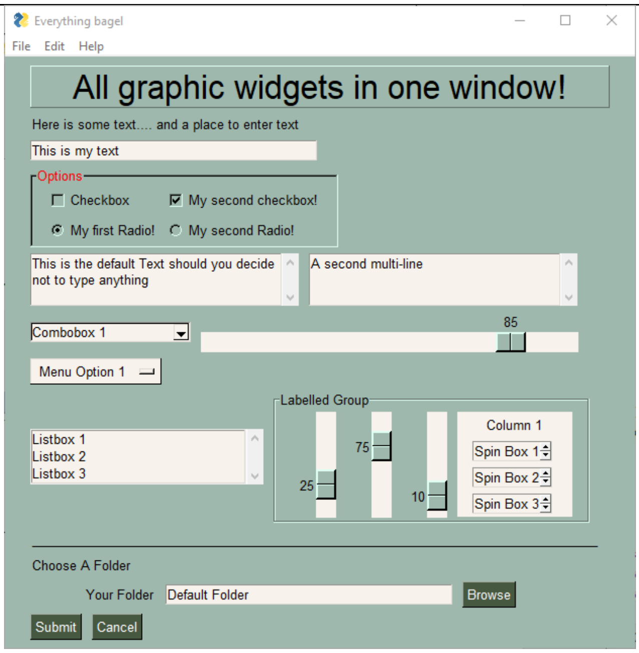
Browse for Filename

```python
import PySimpleGUI as sg

gui_rows = [
            [sg.Text('Enter 2 files to comare')],
            [sg.Text('File 1', size=(8, 1)), sg.InputText(),
sg.FileBrowse()],
            [sg.Text('File 2', size=(8, 1)), sg.InputText(),
sg.FileBrowse()],
            [sg.Submit(), sg.Cancel()]
           ]

window = sg.Window('File Compare').Layout(gui_rows)

button, values = window.Read()

print(button, values)
```



Many Elements on One Window

```
import PySimpleGUI as sg

sg.ChangeLookAndFeel('GreenTan')

# ------ Menu Definition ------ #
menu_def =  [
                ['File', ['Open', 'Save', 'Exit', 'Properties']],
                ['Edit', ['Paste', ['Special', 'Normal', ], 'Undo'], ],
                ['Help', 'About...']
            ]

# ------ Column Definition ------ #
column1 = [
            [sg.Text('Column 1', background_color='#F7F3EC',
justification='center', size=(10, 1))],
            [sg.Spin(values=('Spin Box 1', '2', '3'), initial_value='Spin Box
1')],
            [sg.Spin(values=('Spin Box 1', '2', '3'), initial_value='Spin Box
2')],
            [sg.Spin(values=('Spin Box 1', '2', '3'), initial_value='Spin Box
3')]
          ]

layout = [
    [sg.Menu(menu_def, tearoff=True)],
    [sg.Text('All graphic widgets in one window!', size=(30, 1),
justification='center', font=("Helvetica", 25), relief=sg.RELIEF_RIDGE)],
    [sg.Text('Here is some text.... and a place to enter text')],
    [sg.InputText('This is my text')],
    [sg.Frame(layout=[
        [sg.Checkbox('Checkbox', size=(10 ,1)),  sg.Checkbox('My second
checkbox!', default=True)],
        [sg.Radio('My first Radio!     ', "RADIO1", default=True, size=(10
,1)), sg.Radio('My second Radio!', "RADIO1")]], title='Options'
,title_color='red', relief=sg.RELIEF_SUNKEN, tooltip='Use these to set
flags')],
    [sg.Multiline(default_text='This is the default Text should you decide
not to type anything', size=(35, 3)),
     sg.Multiline(default_text='A second multi-line', size=(35, 3))],
    [sg.InputCombo(('Combobox 1', 'Combobox 2'), size=(20, 1)),
     sg.Slider(range=(1, 100), orientation='h', size=(34, 20),
default_value=85)],
    [sg.InputOptionMenu(('Menu Option 1', 'Menu Option 2', 'Menu Option
3'))],
    [sg.Listbox(values=('Listbox 1', 'Listbox 2', 'Listbox 3'), size=(30,
3)),
     sg.Frame('Labelled Group' ,[[
         sg.Slider(range=(1, 100), orientation='v', size=(5, 20),
default_value=25),
         sg.Slider(range=(1, 100), orientation='v', size=(5, 20),
default_value=75),
```



All graphic widgets in one window!

Here is some text.... and a place to enter text

This is my text

Options

[ ] Checkbox [✓ My second checkbox!]

[✓ My first Radio!] My second Radio!

This is the default Text should you decide not to type anything

A second multi-line

Combobox 1

85

Menu Option 1

Labelled Group

Listbox 1
Listbox 2
Listbox 3

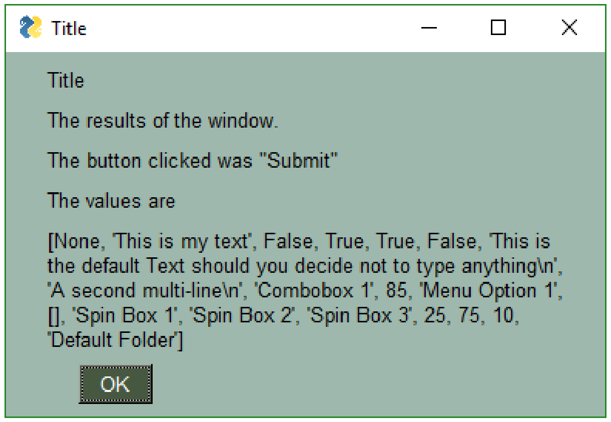
25 75 10

Column 1

Spin Box 1
Spin Box 2
Spin Box 3

Choose A Folder

Your Folder Default Folder


Title

The results of the window.

The button clicked was "Submit"

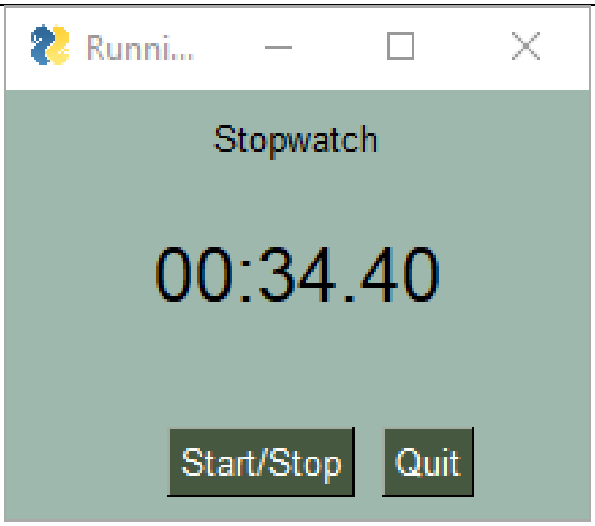
The values are

[None, 'This is my text', False, True, True, False, 'This is the default Text should you decide not to type anything\n', 'A second multi-line\n', 'Combobox 1', 85, 'Menu Option 1', [], 'Spin Box 1', 'Spin Box 2', 'Spin Box 3', 25, 75, 10, 'Default Folder']

```
         sg.Slider(range=(1, 100), orientation='v', size=(5, 20),
default_value=10),
         sg.Column(column1, background_color='#F7F3EC')]])],
    [sg.Text('_'  * 80)],
    [sg.Text('Choose A Folder', size=(35, 1))],
    [sg.Text('Your Folder', size=(15, 1), auto_size_text=False,
justification='right'),
     sg.InputText('Default Folder'), sg.FolderBrowse()],
    [sg.Submit(tooltip='Click to submit this window'), sg.Cancel()]
    ]

window = sg.Window('Everything bagel', default_element_size=(40, 1),
grab_anywhere=False).Layout(layout)

button, values = window.Read()

sg.Popup('Title',
         'The results of the window.',
         'The button clicked was "{}"'.format(button),
         'The values are', values)
```

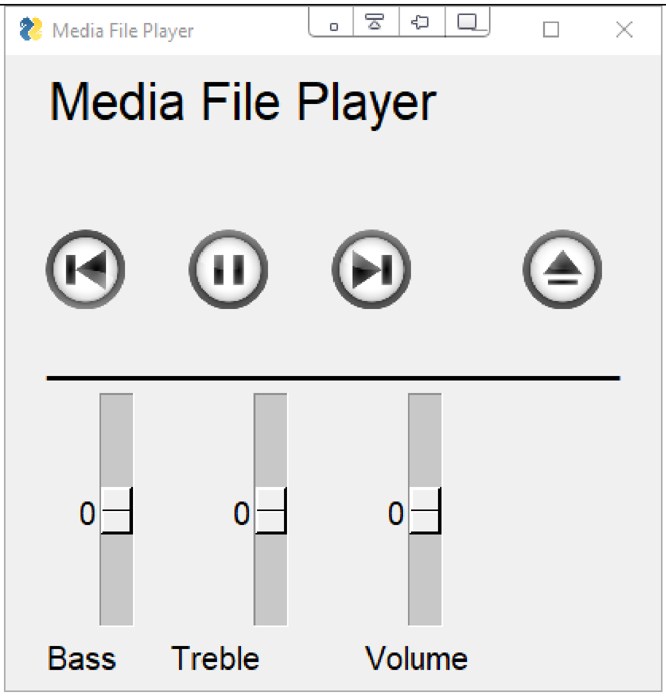
Non-Blocking Form (Async)

```
import PySimpleGUI as sg
import time

gui_rows = [[sg.Text('Stopwatch', size=(20, 2), justification='center')],
            [sg.Text('', size=(10, 2), font=('Helvetica', 20),
justification='center', key='output')],
            [sg.T(' '  * 5), sg.ReadButton('Start/Stop', focus=True),
sg.Quit()]]

window = sg.Window('Running Timer').Layout(gui_rows)

timer_running = True
i = 0
while True:           # Event Loop
    i += 1 * (timer_running is True)
    button, values = window.ReadNonBlocking()

    if values is None or button == 'Quit':  # if user closed the window using
X or clicked Quit button
        break
    elif button == 'Start/Stop':
        timer_running = not timer_running

    window.FindElement('output').Update('{:02d}:{:02d}.{:02d}'.format((i //
100) // 60, (i // 100) % 60, i % 100))
    time.sleep(.01)
```



Using Button Images

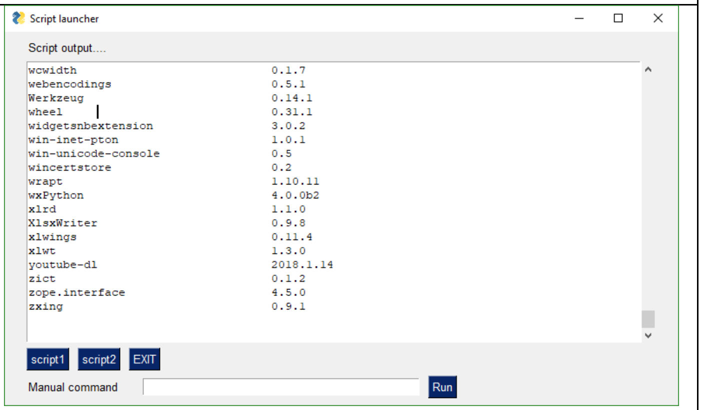
```python
import PySimpleGUI as sg

background = '#F0F0F0'
# Set the backgrounds the same as the background on the buttons
sg.SetOptions(background_color=background,
element_background_color=background)
# Images are located in a subfolder in the Demo Media Player.py folder
image_pause = './ButtonGraphics/Pause.png'
image_restart = './ButtonGraphics/Restart.png'
image_next = './ButtonGraphics/Next.png'
image_exit = './ButtonGraphics/Exit.png'

# define layout of the rows
layout = [[sg.Text('Media File Player', size=(17, 1), font=("Helvetica",
25))],
          [sg.Text('', size=(15, 2), font=("Helvetica", 14), key='output')],
          [sg.ReadButton('Restart Song', button_color=(background,
background),
                         image_filename=image_restart, image_size=(50, 50),
image_subsample=2, border_width=0),
           sg.Text(' ' * 2),
           sg.ReadButton('Pause', button_color=(background, background),
                         image_filename=image_pause, image_size=(50, 50),
image_subsample=2, border_width=0),
           sg.Text(' ' * 2),
           sg.ReadButton('Next', button_color=(background, background),
                         image_filename=image_next, image_size=(50, 50),
image_subsample=2, border_width=0),
           sg.Text(' ' * 2),
           sg.Text(' ' * 2), sg.Button('Exit', button_color=(background,
background),
                                       image_filename=image_exit,
image_size=(50, 50), image_subsample=2,
                                       border_width=0)],
          [sg.Text('_' * 30)],
          [sg.Text(' ' * 30)],
          [
              sg.Slider(range=(-10, 10), default_value=0, size=(10, 20),
orientation='vertical',
                        font=("Helvetica", 15)),
              sg.Text(' ' * 2),
              sg.Slider(range=(-10, 10), default_value=0, size=(10, 20),
orientation='vertical',
                        font=("Helvetica", 15)),
              sg.Text(' ' * 8),
              sg.Slider(range=(-10, 10), default_value=0, size=(10, 20),
orientation='vertical',
                        font=("Helvetica", 15))],
          [sg.Text('Bass', font=("Helvetica", 15), size=(6, 1)),
```



```
            sg.Text('Treble', font=("Helvetica", 15), size=(10, 1)),
            sg.Text('Volume', font=("Helvetica", 15), size=(7, 1))]
           ]

window = sg.Window('Media File Player', auto_size_text=True,
default_element_size=(20, 1),
                   font=("Helvetica", 25)).Layout(layout)
# Our event loop
while (True):
    # Read the window (this call will not block)
    button, values = window.ReadNonBlocking()
    if button == 'Exit' or values is None:
        break
        # If a button was pressed, display it on the GUI by updating the text
element
    if button:
        window.FindElement('output').Update(button)
```

Script Launcher

```
import PySimpleGUI as sg
import subprocess

# Please check Demo programs for better examples of launchers
def ExecuteCommandSubprocess(command, *args):
    try:
        sp = subprocess.Popen([command, *args], shell=True,
stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        out, err = sp.communicate()
        if out:
            print(out.decode("utf-8"))
        if err:
            print(err.decode("utf-8"))
    except:
        pass

layout = [
    [sg.Text('Script output....', size=(40, 1))],
    [sg.Output(size=(88, 20), font='Courier 10')],
    [sg.ReadButton('script1'), sg.ReadButton('script2'), sg.Button('EXIT')],
    [sg.Text('Manual command', size=(15, 1)), sg.InputText(focus=True),
sg.ReadButton('Run', bind_return_key=True)]
        ]

window = sg.Window('Script launcher').Layout(layout)

# ---===--- Loop taking in user input and using it to call scripts --- #
```

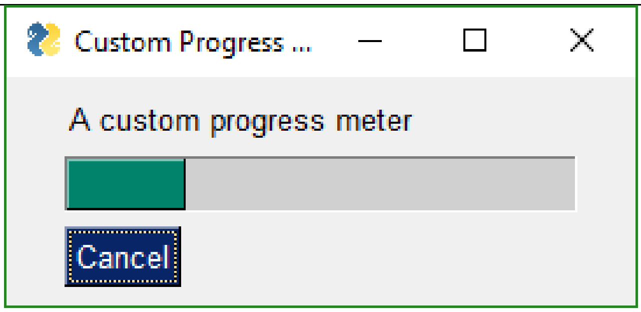


```
wcwidth                             0.1.7
webencodings                        0.5.1
Werkzeug                            0.14.1
wheel                               0.31.1
widgetsnbextension                  3.0.2
win-inet-pton                       1.0.1
win-unicode-console                 0.5
wincertstore                        0.2
wrapt                               1.10.11
wxPython                            4.0.0b2
xlrd                                1.1.0
XlsxWriter                          0.9.8
xlwings                             0.11.4
xlwt                                1.3.0
youtube-dl                          2018.1.14
zict                                0.1.2
zope.interface                      4.5.0
zxing                               0.9.1
```

```
while True:
  (button, value) = window.Read()
  if button == 'EXIT'  or button is None:
      break # exit button clicked
  if button == 'script1':
      ExecuteCommandSubprocess('pip', 'list')
  elif button == 'script2':
      ExecuteCommandSubprocess('python', '--version')
  elif button == 'Run':
      ExecuteCommandSubprocess(value[0])
```

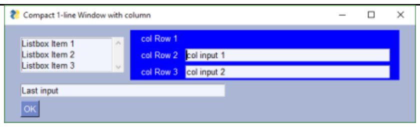
Custom Progress Meter

```
import PySimpleGUI as sg

# layout the Window
layout = [[sg.Text('A custom progress meter')],
          [sg.ProgressBar(10000, orientation='h', size=(20, 20),
key='progbar')],
          [sg.Cancel()]]

# create the Window
window = sg.Window('Custom Progress Meter').Layout(layout)
# loop that would normally do something useful
for i in range(10000):
    # check to see if the cancel button was clicked and exit loop if clicked
    button, values = window.ReadNonBlocking()
    if button == 'Cancel'  or values == None:
        break
  # update bar with loop value +1 so that bar eventually reaches the maximum
    window.FindElement('progbar').UpdateBar(i + 1)
# done with loop... need to destroy the window as it's still open
window.CloseNonBlocking()
```



Multiple Columns

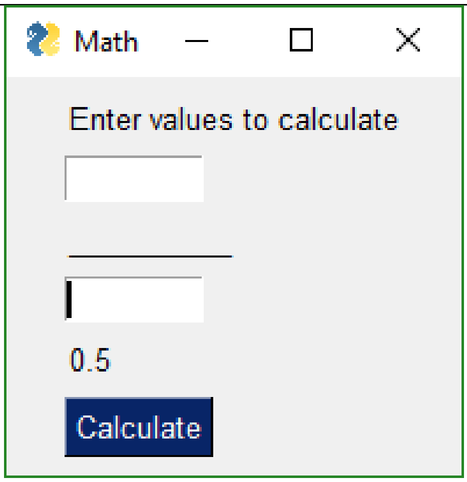
```
import PySimpleGUI as sg

# Demo of how columns work
# GUI has on row 1 a vertical slider followed by a COLUMN with 7 rows
# Prior to the Column element, this layout was not possible
# Columns layouts look identical to GUI layouts, they are a list of lists of
elements.

sg.ChangeLookAndFeel('BlueMono')

# Column layout
col = [[sg.Text('col Row 1', text_color='white', background_color='blue')],
       [sg.Text('col Row 2', text_color='white', background_color='blue'),
```



```
sg.Input('col input 1')],
       [sg.Text('col Row 3', text_color='white', background_color='blue'),
sg.Input('col input 2')]]

layout = [[sg.Listbox(values=('Listbox Item 1', 'Listbox Item 2', 'Listbox
Item 3'), select_mode=sg.LISTBOX_SELECT_MODE_MULTIPLE, size=(20,3)),
sg.Column(col, background_color='blue')],
          [sg.Input('Last input')],
          [sg.OK()]]

# Display the Window and get values

button, values = sg.Window('Compact 1-line Window with
column').Layout(layout).Read()

sg.Popup(button, values, line_width=200)
```

Updating Elements (Text Element)

```
import PySimpleGUI as sg

layout = [ [sg.Txt('Enter values to calculate')],
           [sg.In(size=(8,1), key='numerator')],
           [sg.Txt('_'  * 10)],
           [sg.In(size=(8,1), key='denominator')],
           [sg.Txt('', size=(8,1), key='output')  ],
           [sg.ReadButton('Calculate', bind_return_key=True)]]

window = sg.Window('Math').Layout(layout)

while True:
    button, values = window.Read()

    if button is not None:
        try:
            numerator = float(values['numerator'])
            denominator = float(values['denominator'])
            calc = numerator / denominator
        except:
            calc = 'Invalid'

        window.FindElement('output').Update(calc)
    else:
        break
```

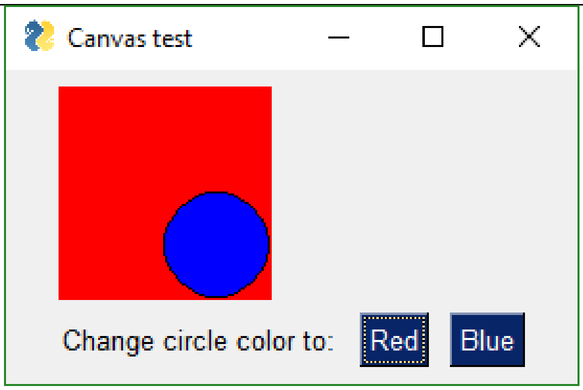


Enter values to calculate

0.5

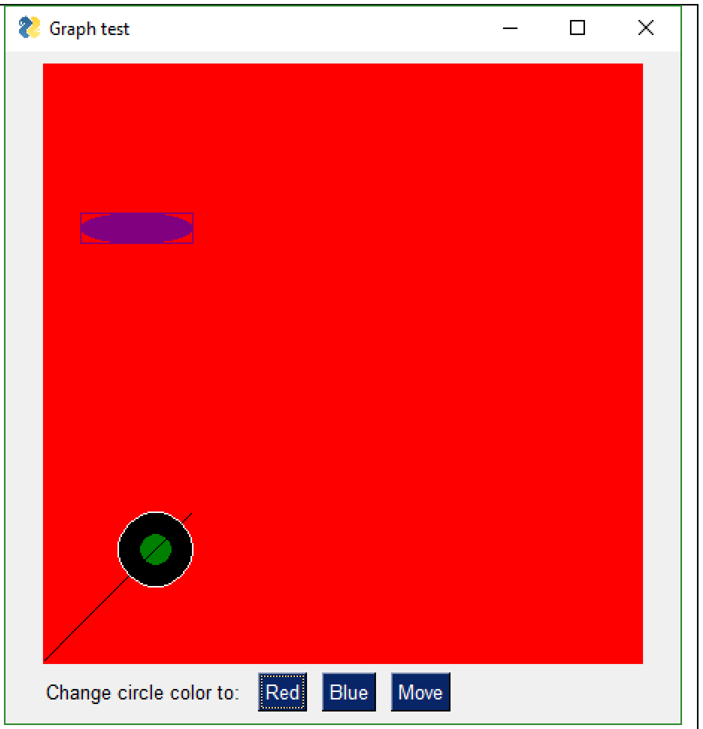
Canvas Element

```
import PySimpleGUI as sg

layout = [
    [sg.Canvas(size=(100, 100), background_color='red', key= 'canvas')],
    [sg.T('Change circle color to:'), sg.ReadButton('Red'), 
sg.ReadButton('Blue')]
]

window = sg.Window('Canvas test')
window.Layout(layout)
window.Finalize()

canvas = window.FindElement('canvas')
cir = canvas.TKCanvas.create_oval(50, 50, 100, 100)

while True:
    button, values = window.Read()
    if button is None:
        break
    if button == 'Blue':
        canvas.TKCanvas.itemconfig(cir, fill="Blue")
    elif button == 'Red':
        canvas.TKCanvas.itemconfig(cir, fill="Red")
```



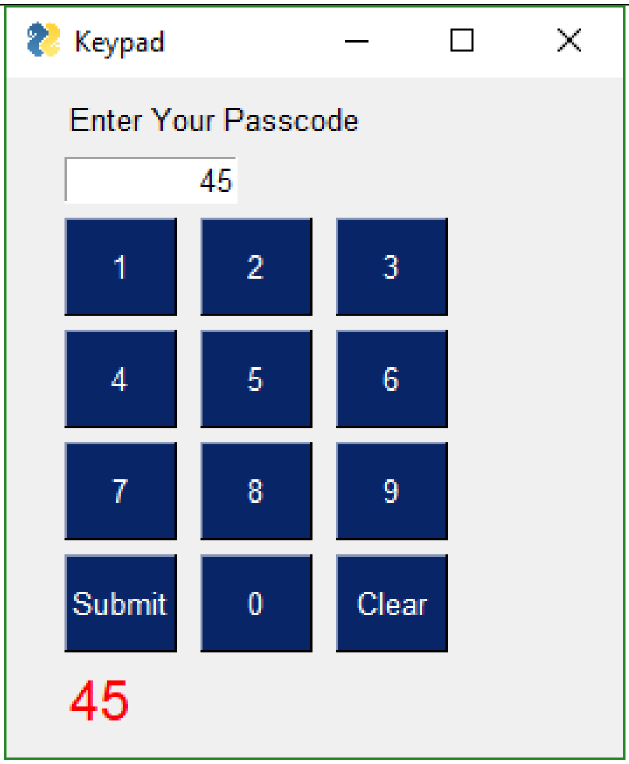
Graph Element

```
import PySimpleGUI as sg

layout = [
           [sg.Graph(canvas_size=(400, 400), graph_bottom_left=(0,0),
graph_top_right=(400, 400), background_color='red', key='graph')],
           [sg.T('Change circle color to:'), sg.ReadButton('Red'),
sg.ReadButton('Blue'), sg.ReadButton('Move')]
           ]

window = sg.Window('Graph test')
window.Layout(layout)
window.Finalize()

graph = window.FindElement('graph')
circle = graph.DrawCircle((75,75), 25, fill_color='black',line_color='white')
point = graph.DrawPoint((75,75), 10, color='green')
oval = graph.DrawOval((25,300), (100,280), fill_color='purple',
line_color='purple'  )
rectangle = graph.DrawRectangle((25,300), (100,280), line_color='purple'  )
line = graph.DrawLine((0,0), (100,100))

while True:
    button, values = window.Read()
    if button is None:
        break
    if button is 'Blue':
        graph.TKCanvas.itemconfig(circle, fill = "Blue")
    elif button is 'Red':
        graph.TKCanvas.itemconfig(circle, fill = "Red")
    elif button is 'Move':
        graph.MoveFigure(point, 10,10)
        graph.MoveFigure(circle, 10,10)
        graph.MoveFigure(oval, 10,10)
        graph.MoveFigure(rectangle, 10,10)
```



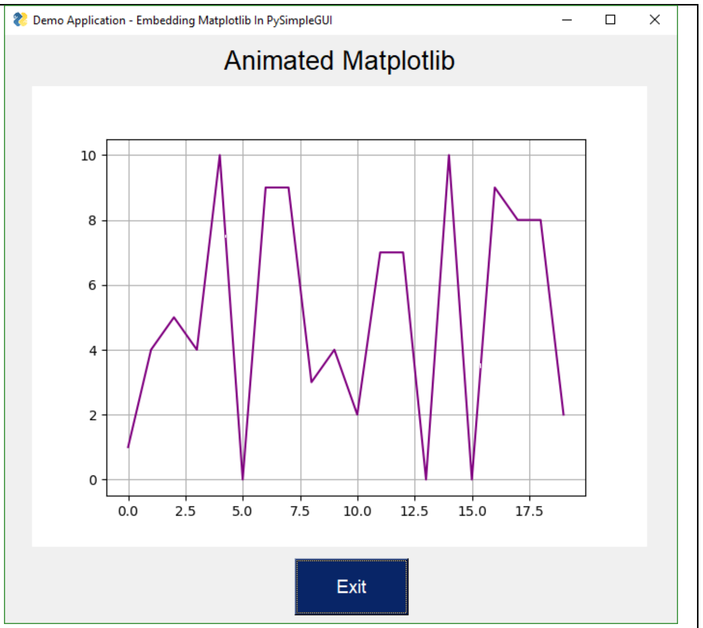
Keypad – Inserting Into Input Element

```python
import PySimpleGUI as sg

# Demonstrates a number of PySimpleGUI features including:
#   Default element size
#   auto_size_buttons
#   ReadButton
#   Dictionary return values
#   Update of elements in window (Text, Input)
#   do_not_clear of Input elements

layout = [[sg.Text('Enter Your Passcode')],
          [sg.Input(size=(10, 1), do_not_clear=True, justification='right', key='input')],
          [sg.ReadButton('1'), sg.ReadButton('2'), sg.ReadButton('3')],
          [sg.ReadButton('4'), sg.ReadButton('5'), sg.ReadButton('6')],
          [sg.ReadButton('7'), sg.ReadButton('8'), sg.ReadButton('9')],
          [sg.ReadButton('Submit'), sg.ReadButton('0'), sg.ReadButton('Clear')],
          [sg.Text('', size=(15, 1), font=('Helvetica', 18), text_color='red', key='out')],
          ]

window = sg.Window('Keypad', default_button_element_size=(5, 2), auto_size_buttons=False, grab_anywhere=False).Layout(layout)

# Loop forever reading the window's values, updating the Input field
keys_entered = ''
while True:
    button, values = window.Read()  # read the window
    if button is None:  # if the X button clicked, just exit
        break
    if button == 'Clear':  # clear keys if clear button
       keys_entered = ''
    elif button in '1234567890':
       keys_entered = values['input']  # get what's been entered so far
       keys_entered += button  # add the new digit
    elif button == 'Submit':
       keys_entered = values['input']
       window.FindElement('out').Update(keys_entered)  # output the final string

    window.FindElement('input').Update(keys_entered)  # change the window to reflect current key string
```



Animated Matplotlib Graph

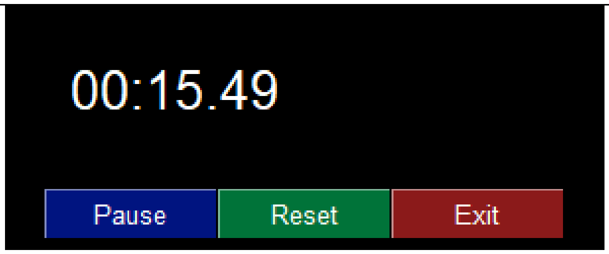
```
from random import randint
import PySimpleGUI as g
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
FigureCanvasAgg
from matplotlib.figure import Figure
import matplotlib.backends.tkagg as tkagg
import tkinter as Tk


fig = Figure()

ax = fig.add_subplot(111)
ax.set_xlabel("X axis")
ax.set_ylabel("Y axis")
ax.grid()

layout = [[g.Text('Animated Matplotlib', size=(40, 1),
justification='center', font='Helvetica 20')],
          [g.Canvas(size=(640, 480), key='canvas')],
          [g.ReadButton('Exit', size=(10, 2), pad=((280, 0), 3),
font='Helvetica 14')]]

# create the window and show it without the plot


window = g.Window('Demo Application - Embedding Matplotlib In
PySimpleGUI').Layout(layout)
window.Finalize()      # needed to access the canvas element prior to reading
the window

canvas_elem = window.FindElement('canvas')

graph = FigureCanvasTkAgg(fig, master=canvas_elem.TKCanvas)
canvas = canvas_elem.TKCanvas

dpts = [randint(0, 10) for x in range(10000)]
# Our event loop
for i in range(len(dpts)):
    button, values = window.ReadNonBlocking()
    if button == 'Exit'  or values is None:
        break

    ax.cla()
    ax.grid()

    ax.plot(range(20), dpts[i:i + 20], color='purple')
    graph.draw()
    figure_x, figure_y, figure_w, figure_h = fig.bbox.bounds
    figure_w, figure_h = int(figure_w), int(figure_h)
    photo = Tk.PhotoImage(master=canvas, width=figure_w, height=figure_h)
```



```
    canvas.create_image(640 / 2, 480 / 2, image=photo)

    figure_canvas_agg = FigureCanvasAgg(fig)
    figure_canvas_agg.draw()

    tkagg.blit(photo, figure_canvas_agg.get_renderer()._renderer,
colormode=2)
```

Floating Widget with No Border - Timer

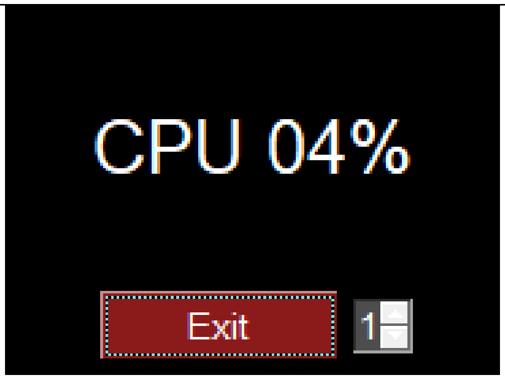
```
import PySimpleGUI as sg
import time

"""
 Timer Desktop Widget Creates a floating timer that is always on top of other
windows You move it by grabbing anywhere on the window Good example of how to
do a non-blocking, polling program using PySimpleGUI Can be used to poll
hardware when running on a Pi     NOTE - you will get a warning message
printed when you exit using exit button. It will look something like: invalid
command name \"1616802625480StopMove\"
"""


# ----------------  Create window  ----------------
sg.ChangeLookAndFeel('Black')
sg.SetOptions(element_padding=(0, 0))

layout = [[sg.Text('')],
             [sg.Text('', size=(8, 2), font=('Helvetica', 20),
justification='center', key='text')],
             [sg.ReadButton('Pause', key='button', button_color=('white',
'#001480')),
              sg.ReadButton('Reset', button_color=('white', '#007339'),
key='Reset'),
              sg.Exit(button_color=('white', 'firebrick4'), key='Exit')]]

window = sg.Window('Running Timer', no_titlebar=True,
auto_size_buttons=False, keep_on_top=True, grab_anywhere=True).Layout(layout)


# ----------------  main loop  ----------------
current_time = 0
paused = False
start_time = int(round(time.time() * 100))
while (True):
    # --------- Read and update window --------
  if not paused:
```



```
        button, values = window.ReadNonBlocking()
        current_time = int(round(time.time() * 100)) - start_time
  else:
        button, values = window.Read()
  if button == 'button':
        button = window.FindElement(button).GetText()
    # --------- Do Button Operations --------
  if values is None or button == 'Exit':
        break
  if button is 'Reset':
        start_time = int(round(time.time() * 100))
        current_time = 0
        paused_time = start_time
  elif button == 'Pause':
        paused = True
        paused_time = int(round(time.time() * 100))
        element = window.FindElement('button')
        element.Update(text='Run')
  elif button == 'Run':
        paused = False
        start_time = start_time + int(round(time.time() * 100)) - paused_time
        element = window.FindElement('button')
        element.Update(text='Pause')

    # --------- Display timer in window --------

window.FindElement('text').Update('{:02d}:{:02d}.{:02d}'.format((current_time
// 100) // 60,

(current_time // 100) % 60,

current_time % 100))
  time.sleep(.01)

# --------- After loop --------

# Broke out of main loop. Close the window.
window.CloseNonBlocking()
```

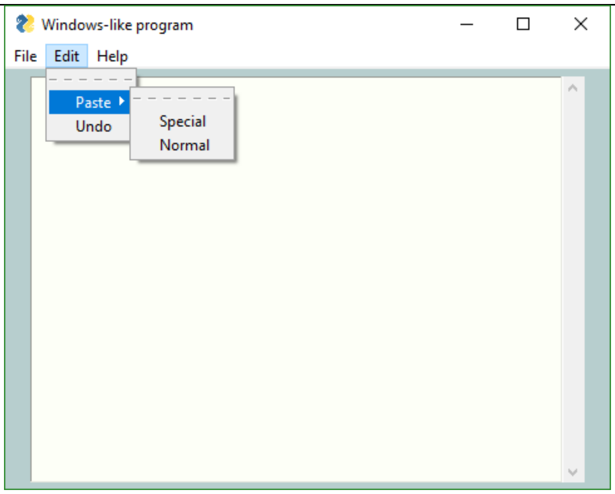
CPU Widget Using psutil

```
import PySimpleGUI as sg
import psutil

# ----------------  Create Window  ----------------
sg.ChangeLookAndFeel('Black')
layout = [[sg.Text('')],
             [sg.Text('', size=(8, 2), font=('Helvetica', 20),
justification='center', key='text')],
             [sg.Exit(button_color=('white', 'firebrick4'), pad=((15,0), 0)),
sg.Spin([x+1 for x in range(10)], 1, key='spin')]]

window = sg.Window('Running Timer', no_titlebar=True,
auto_size_buttons=False, keep_on_top=True, grab_anywhere=True).Layout(layout)


# ----------------  main loop  ----------------
while (True):
    # --------- Read and update window --------
  button, values = window.ReadNonBlocking()

    # --------- Do Button Operations --------
  if values is None or button == 'Exit':
        break
  try:
        interval = int(values['spin'])
  except:
        interval = 1

  cpu_percent = psutil.cpu_percent(interval=interval)

    # --------- Display timer in window --------

  window.FindElement('text').Update(f'CPU {cpu_percent:02.0f}%')

# Broke out of main loop. Close the window.
window.CloseNonBlocking()
```



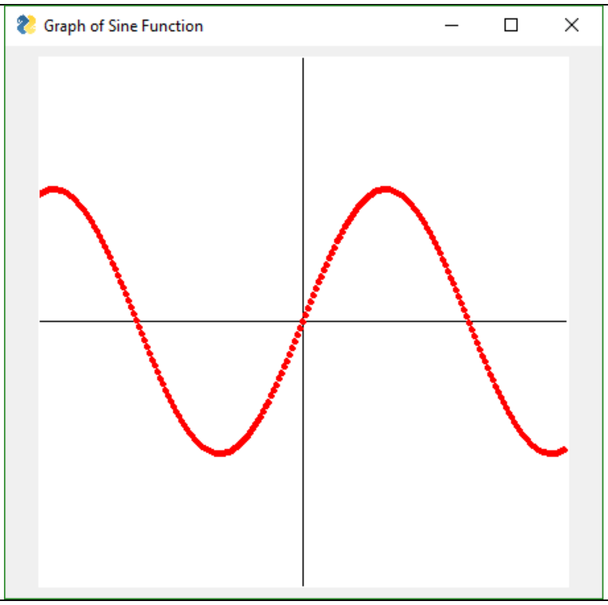
Menus in 25 Lines of Code!

```python
import PySimpleGUI as sg

sg.ChangeLookAndFeel('LightGreen')
sg.SetOptions(element_padding=(0, 0))

# ------ Menu Definition ------ #
menu_def = [['File', ['Open', 'Save', 'Exit'  ]],
            ['Edit', ['Paste', ['Special', 'Normal', ], 'Undo'], ],
            ['Help', 'About...'], ]

# ------ GUI Defintion ------ #
layout = [
    [sg.Menu(menu_def)],
    [sg.Output(size=(60, 20))]
         ]

window = sg.Window("Windows-like program", default_element_size=(12, 1),
auto_size_text=False, auto_size_buttons=False,
                   default_button_element_size=(12, 1)).Layout(layout)

# ------ Loop & Process button menu choices ------ #
while True:
    button, values = window.Read()
    if button == None or button == 'Exit':
        break
    print('Button = ', button)
    # ------ Process menu choices ------ #
    if button == 'About...':
        sg.Popup('About this program', 'Version 1.0', 'PySimpleGUI rocks...')
    elif button == 'Open':
        filename = sg.PopupGetFile('file to open', no_window=True)
        print(filename)
```



Graph Element

```
import math
import PySimpleGUI as sg

layout = [[sg.Graph(canvas_size=(400, 400), graph_bottom_left=(-100,-100),
graph_top_right=(100,100), background_color='white', key='graph')],]

window = sg.Window('Graph of Sine Function').Layout(layout)
window.Finalize()
graph = window.FindElement('graph')

graph.DrawLine((-100,0), (100,0))
graph.DrawLine((0,-100), (0,100))

for x in range(-100,100):
    y = math.sin(x/20)*50
    graph.DrawPoint((x,y), color='red')

button, values = window.Read()
```


Graph of Sine Function