GUI Programming with PysimpleGUI[1]

Up until now, the only way our programs have been able to interact with the user is through keyboard input via the **input** statement. But most real programs use windows, buttons, scrollbars, and various other things. These *widgets* are part of what is called a *Graphical User Interface* or GUI.

We will explore PySimpleGUI – a library based on tkinter.
To install on your computer

- Go to the Command Line Interface (Windows): pip3 install PySimpleGUI
- Go to the Terminal (MacOS): pip3 install PySimpleGUI

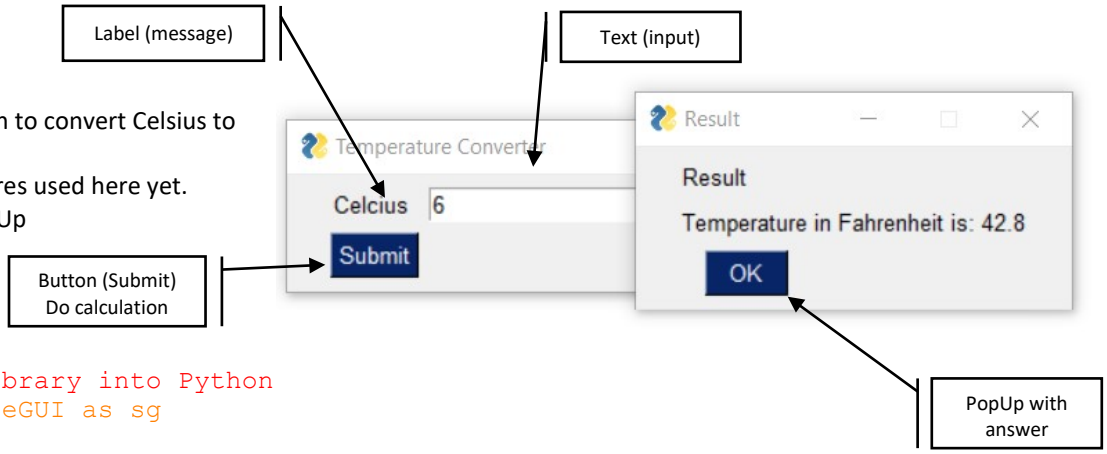
To install on a school desktop

- Go to the Terminal (MacOS): pip3 install PySimpleGUI –user

Note: All screenshots and code shown are from Windows version. There are slight variations visually that don't impact the functionality between Windows and MacOS. Code for both operating systems are in the OneDrive area.

Basics:

Create an application to convert Celsius to Fahrenheit
No formatting features used here yet.
Data entry and Pop-Up



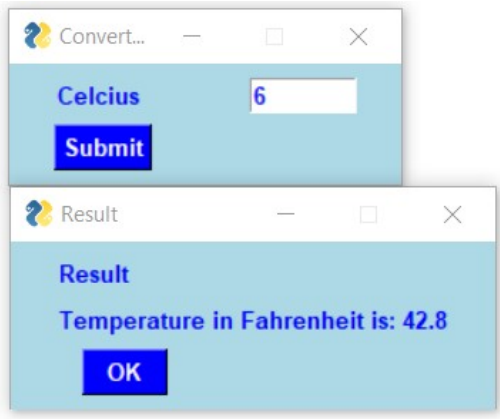

```
#Import the library into Python
import PySimpleGUI as sg

#organise your layout, Text, Input,then the button on line
below #layout organised using lists [ xxxxx , xxxxxx ]

layout = [
   [sg.Text('Celcius'), sg.InputText()],
    [sg.Submit()],
   ]

#setup window with Title
window = sg.Window('Temperature Converter').Layout(layout)

#get value(s) (part of a list)
button, value = window.Read()

if button is None:       #windows was closed without button being pressed
    exit(0)              #not essential - but programs should close properly

#value[0] get float from string, convert to fahrenheit and round
fahrenheit = round(9/5*float(value[0]) +32, 1)

#create a single string to be displayed
result =  'Temperature in Fahrenheit is: ' + str(fahrenheit)

#display in Popup
```

[1] See: https://pypi.org/project/PySimpleGUI/

```
sg.Popup('Result', result)
```

Formatting:

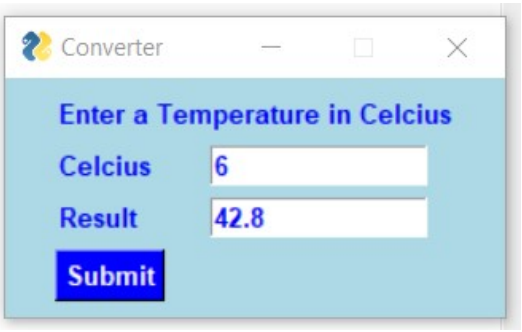
Let's organise some formatting.



Add

```
#Set formatting options for all elements
#rather than individually. Care with brackets
#add before layout

sg.SetOptions (background_color = 'LightBlue',
       element_background_color = 'LightBlue',
       text_element_background_color = 'LightBlue',
            font = ('Arial', 10, 'bold'),
            text_color = 'Blue',
            input_text_color ='Blue',
            button_color = ('White', 'Blue')


#adjust widths of labels and entry boxes
layout = [
   [sg.Text('Celcius', size =(12,1)), sg.InputText(size = (8,1))],
    [sg.Submit()]
   ]
```

Bind keys, display in text box for result and make the Form persistent



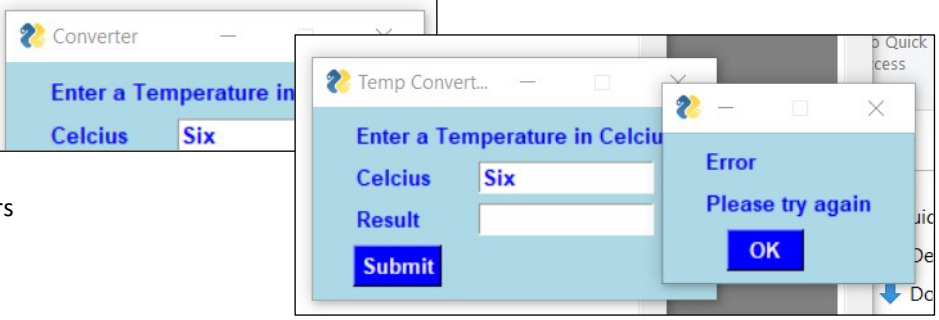
See the changes below

```
#update (via list) values and and display answers
#value[0] is celcius input, value[1] is input to place result.
#Use ReadButton with while true: - keeps window open.

layout = [ [sg.Text('Enter a Temperature in Celcius')],
    [sg.Text('Celcius', size =(8,1)), sg.InputText(size = (15,1))],
    [sg.Text('Result', size =(8,1)), sg.InputText(size = (15,1))],
    [sg.ReadButton('Submit', bind_return_key = True)]]
#Return = button press
window = sg.Window('Converter').Layout(layout)

while True:
    #get result
    button, value = window.Read()
    #break out of loop is button not pressed.
    if button is not None:
        fahrenheit = round(9/5*float(value[0]) +32, 1)
        #put result in 2nd input box - Update
        window.FindElement(1).Update(fahrenheit)
    else:
        break
```

Named input keys and catch errors

```
Traceback (most recent call last):
  File "C:\Users\tcrewe\Dropbox\01 Teaching folders\03 Year 11 I
ng\Python Stuff\Tkinter PySimpleGUI\PySimpleGUI samples\1c PSG (
and bind key).py", line 31, in <module>
    fahrenheit = round(9/5*float(value[0]) +32, 1)
ValueError: could not convert string to float: 'Six'
>>>
```

The problem is that users catch enter invalid data and cause a crash.

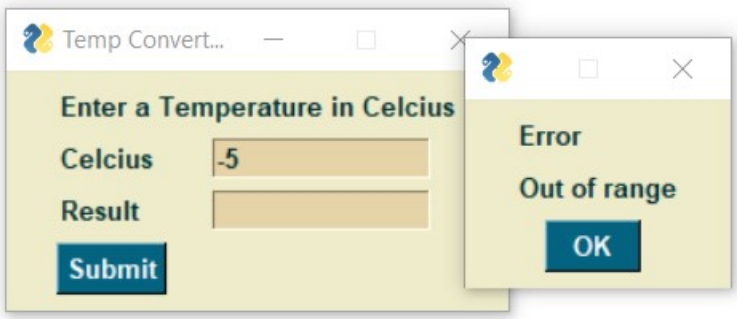


Always best to try and stop these errors

Changes below

```
#name inputs (key) uses dictionary - easy to see updating of results
#value[input] first input value is celcius input

layout = [ [sg.Text('Enter a Temperature in Celcius')],
    [sg.Text('Celcius', size =(8,1)), sg.InputText(size = (15,1),key = '_input_')],
    [sg.Text('Result', size =(8,1)), sg.InputText(size = (15,1),key = '_result_')],
    [sg.ReadButton('Submit', bind_return_key = True)]]

window = sg.FlexForm('Temp Converter').Layout(layout)

while True:
    button, value = window.Read()
    if button is not None:
        #catch program errors for text or blank entry:
        try:
            fahrenheit = round(9/5*float(value['_input_']) +32, 1)
            #put result in text box
            window.FindElement('_result_').Update(fahrenheit)
        except ValueError:
            sg.Popup('Error','Please try again')

    else:
        break
```

Validation added and Look and Feel (Windows only)

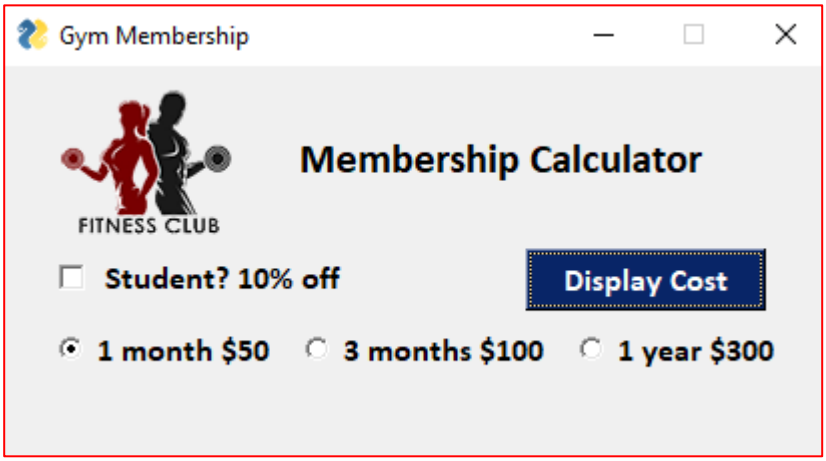


As well, as capturing errors, sometimes you want to restrict values entered, even if correct data types..

Changes below

```
#can use a variety of themes - plus individual options:  Windows only
sg.ChangeLookAndFeel('SandyBeach')
sg.SetOptions (font = ('Arial', 10, 'bold'))


while True:
    button, value = window.Read()
    if button is not None:
        #catch program errors for text, floats or blank entry:
        #also validation for range [0, 50]
        try:
            if float(value['_input_']) > 50 or float(value['_input_']) <0:
                sg.Popup('Error','Out of range')
            else:
                fahrenheit = round(9/5*int(value['_input_']) +32, 1)
                window.FindElement('_result_').Update(fahrenheit)
        except ValueError:
                sg.Popup('Error','Please try again')

    else:
        break
```

For those interested – check out another sample using radio button and checkboxes (no validation or error checking needed!!) and images



Practice Task for Part 2 of Outcome.

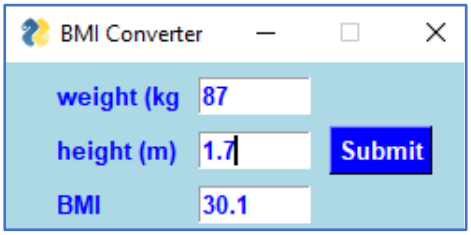
The **body mass index** (**BMI**) or **Quetelet index** is a value derived from the mass (weight) and height of an individual. The BMI is defined as the body mass divided by the square of the body height, and is universally expressed in units of kg/m² - from Body mass index, https://en.wikipedia.org/wiki/Body_mass_index April 4, 2018

Create a Graphic User Interface (GUI) using PySimpleGUI and python to calculate a person's BMI when they enter their height (in metres) and weight (in kilograms)
Use the template used earlier and adapt – Outcome (Part 2) will be completely open book and all resources will be available.

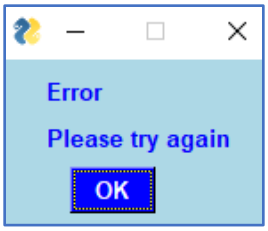
Part 1:
Program correctly allows entry of weight and height and displays BMI correctly with similar layout shown – ignore formatting ***(20 marks: structure, data entry, results, 1 decimal place etc – no formatting)***

Part 2:
Format interface to look like the sample left. ***(5 marks – formatting, font, sizes, colors)***
MacOs and Windows 'look' will be slightly different

- Title is BMI
- Background colour is light blue,
- Font for all is Arial
- Message labels, buttons and output font is bold
- Button background is blue and text color is white



Part 3:
People can enter decimal numbers, but not empty or text – if users enter these catch the error and display the following in a message box: Title Error and Message 'Please try again'
(3 marks – catching errors and validation – see previous)



Part 4 (extension/challenge)
Modify your program to display the output in green, **but** red if the BMI is greater than 30. Like that shown.
(2 marks for some extension extra stuff)
Hints:

- Change button color
- Use Text not InputText for result
- In Text (for result)
  - use property: justification = centre
  - set key = 'result'
  - set font size bigger
- use if statement to set color
- In window.FindElement('result').update() → set text_color to color)