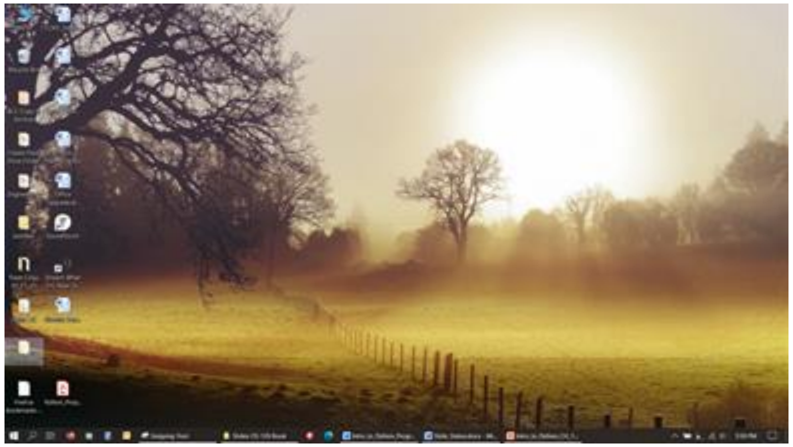
Computer Programming in Python

Chapter 10

Graphical User Interfaces (GUIs)

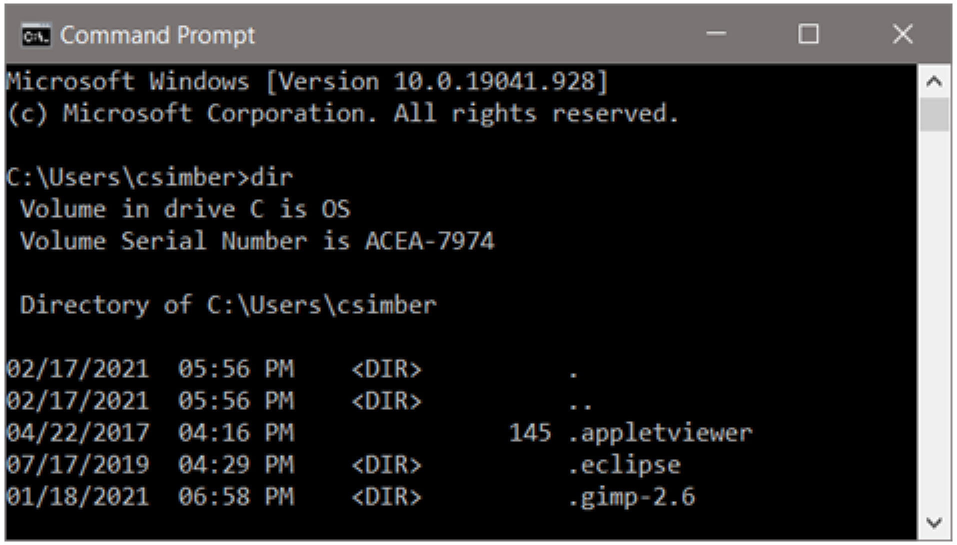
Chapter 10 Graphical User Interfaces (GUIs)

- Graphical User Interfaces
  - Originally created by researchers at the Xerox PARC
    - Palo Alto Research Center California
  - Quickly became the user-preferred choice for interfacing with computers in the 1980’s



Chapter 10 Graphical User Interfaces (GUIs)

- Graphical User Interfaces
  - Prior to GUIs, command line interfaces were used to interact with computers, and in some cases they continue to be used



```
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\csimber>dir
 Volume in drive C is OS
 Volume Serial Number is ACEA-7974

 Directory of C:\Users\csimber

02/17/2021  05:56 PM    <DIR>          .
02/17/2021  05:56 PM    <DIR>          ..
04/22/2017  04:16 PM               145 .appletviewer
07/17/2019  04:29 PM    <DIR>          .eclipse
01/18/2021  06:58 PM    <DIR>          .gimp-2.6
```

Chapter 10 Graphical User Interfaces (GUIs)

- Graphical User Interfaces
  - GUIs are event driven by user input such as clicking on a button or tab, scrolling, or resizing a window
  - The user determines the sequence of many of the events
  - Careful design is required to control orderly access to the events
    - For instance, a user may click a button to compute a result before entering data required for the computation
    - This increases the input validation aspects of a program

Chapter 10 Graphical User Interfaces (GUIs)

- Graphical User Interfaces
  - Input Validation
    - A value needed for computation must be entered by the user before allowing computation, and the value entered must be within the correct range of values to avoid issues such as division by zero
    - The graceful handling of incorrect input is required, but with the added requirement of a graphical interface

Chapter 10 Graphical User Interfaces (GUIs)

- Input Validation
  - Consider a program that computes the circumference of a circle based on an input of radius when a button is clicked
    - The radius must be input prior to computation
    - The radius input must be a positive number
  - The graceful handling of incorrect input is required for a robust and well-engineered solution

Chapter 10 Graphical User Interfaces (GUIs)

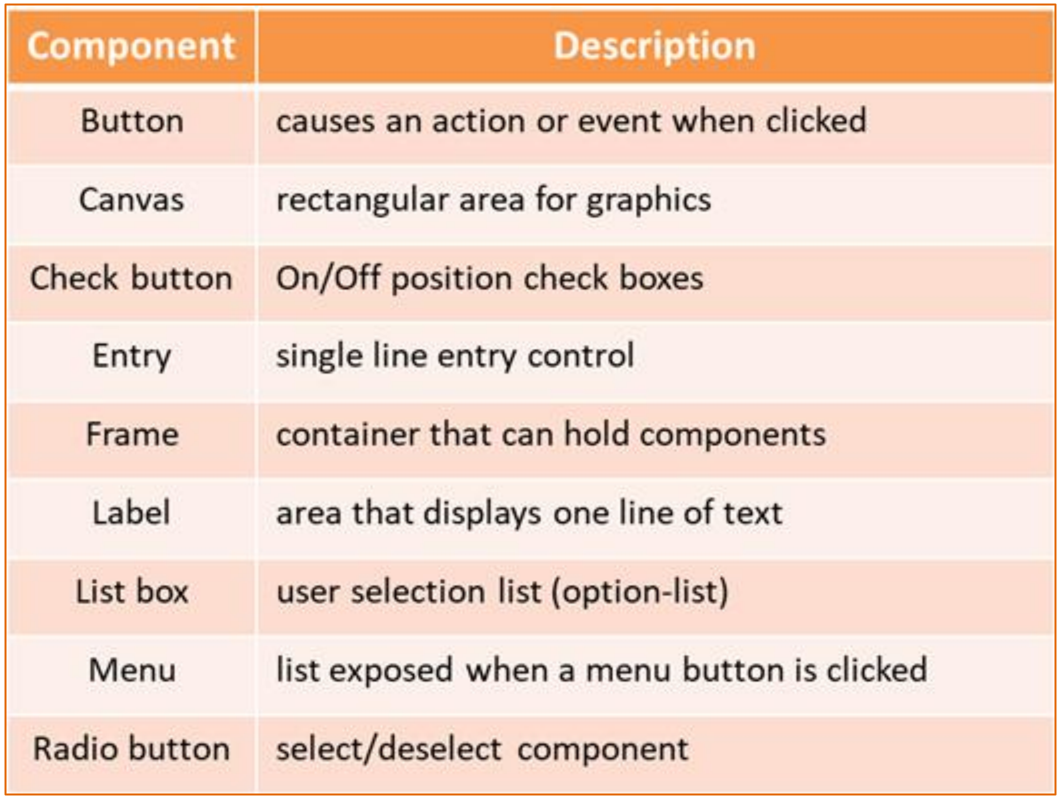
- Graphical User Interfaces
  - The main interface
  - Generating a main GUI requires generating a window
  - The user will interact with the program through ***controls*** (widgets or components) in the window
  - A control is an element of the interface that enables a user to accomplish some function or to access an area of the program

Chapter 10 Graphical User Interfaces (GUIs)

- Graphical User Interfaces
  - Python provides controls and features through the ***Tkinter*** module
    - Installed with Python and provides windows and controls that are easy to program
    - Standard Python interface into TK GUI Toolkit which is used by developers in other languages as well
      - The name Tkinter is short for TK Interface

Chapter 10 Graphical User Interfaces (GUIs)

- A few of the ***Tkinter*** controls



Component	Description
Button	causes an action or event when clicked
Canvas	rectangular area for graphics
Check button	On/Off position check boxes
Entry	single line entry control
Frame	container that can hold components
Label	area that displays one line of text
List box	user selection list (option-list)
Menu	list exposed when a menu button is clicked
Radio button	select/deselect component

Chapter 10 Graphical User Interfaces (GUIs)

- Designing GUI Programs
  - The design for a program with a graphical interface is similar to non-interface program design, but there are essentially two designs - the program design and the interface design
  - In many organization, there is a Visualization Team responsible for interfaces
    - This team has developed expertise in GUI design and development
  - The “back end” of the program is developed by another team
    - This team has developed expertise in algorithm design and development

Chapter 10 Graphical User Interfaces (GUIs)

- Designing the Interface
  - Create a preliminary design (sketch) of the layout for the GUI to determine the location for controls
  - Walk through the program operation steps as a user would
  - Determine intuitive controls and the labels for instruction
  - A user should not have to wonder how to use the program or what units to enter

Design a User Friendly Interface

Chapter 10 Graphical User Interfaces (GUIs)

- Designing GUI Programs
  - The controls depend on what the program does and the user interaction
    - A few labels and buttons may be adequate, or radio buttons or option lists
  - Deciding during the design phase will save time redesigning or reconfiguring an inadequate or problem interface

Chapter 10 Graphical User Interfaces (GUIs)

- Designing GUIs
  - Programmers often overlook essential aspects of the interface since they know what the program does, how it functions, and the inputs required
  - The Agile process typically involves stakeholder reviews and in some cases the customer
    - Provides an opportunity for people not familiar with the planned design to offer suggestions for improvement

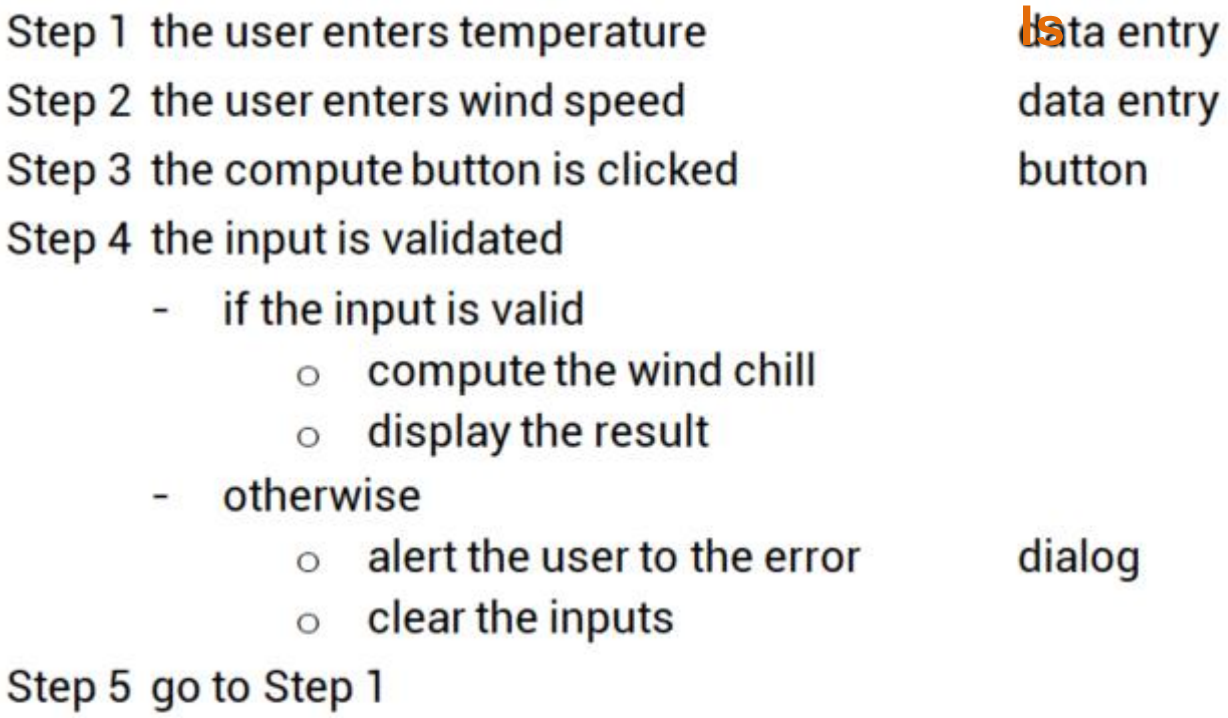
Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example – GUI Weather Program
  - Develop a GUI interface for a program that receives user input for temperature and wind speed, and computes and displays the wind chill factor when a button is clicked
    - Two input values are required
    - The computation of wind chill is the processing
    - A button will be clicked to compute the value
    - The result will be displayed

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example – GUI Weather Program

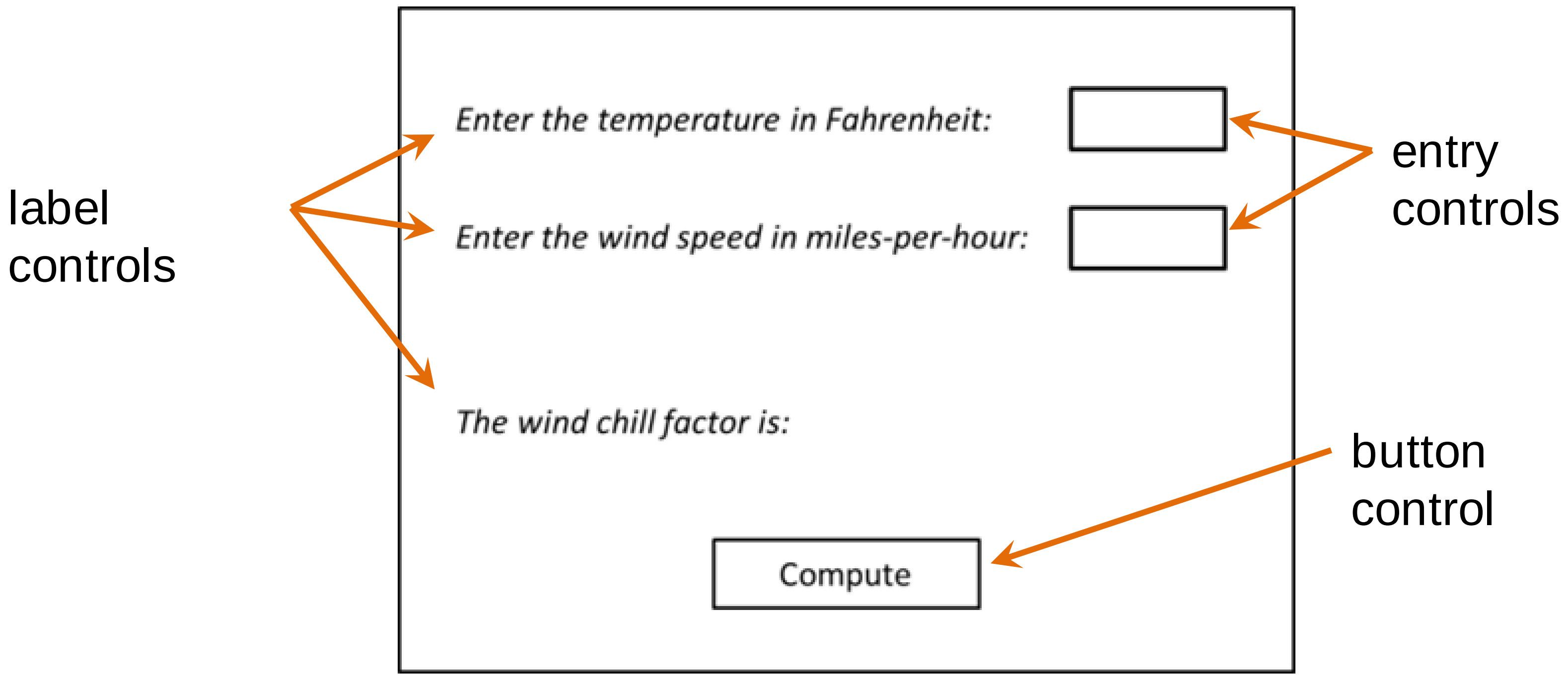
Contro ls



```
Step 1  the user enters temperature                 data entry
Step 2  the user enters wind speed                  data entry
Step 3  the compute button is clicked               button
Step 4  the input is validated
          -   if the input is valid
                o   compute the wind chill
                o   display the result
          -   otherwise
                o   alert the user to the error     dialog
                o   clear the inputs
Step 5  go to Step 1
```

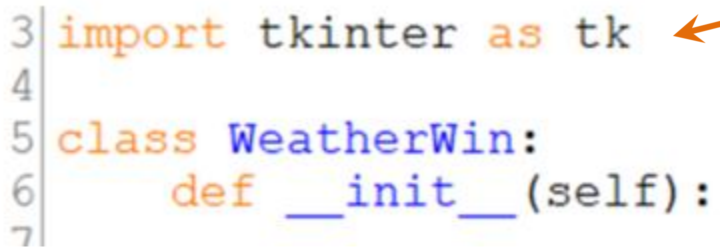
Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example – Sketch
  - Planned interface


label controls
Enter the temperature in Fahrenheit:
Enter the wind speed in miles-per-hour:
The wind chill factor is:
Compute
entry controls
button control

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example
  - Programmers use an OOP approach to GUI development
  - The interface (window) is the WeatherWin class



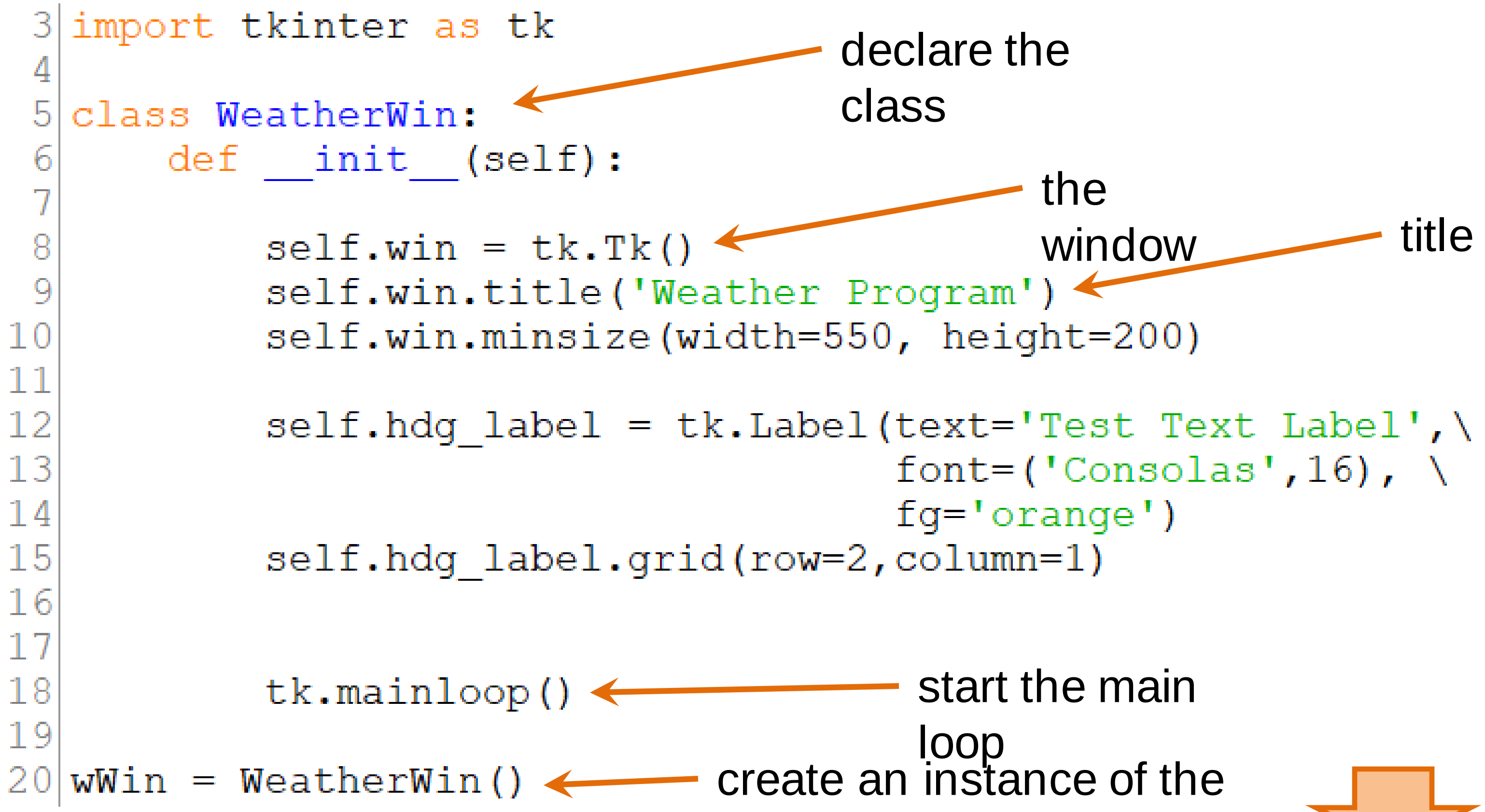
```python
import tkinter as tk

class WeatherWin:
    def __init__(self):
```

import tkinter

tkinter is imported as "tk" to lessen typing

Chapter 10 Graphical User Interfaces (GUIs)

- Example



```
import tkinter as tk

class WeatherWin:
    def __init__(self):

        self.win = tk.Tk()
        self.win.title('Weather Program')
        self.win.minsize(width=550, height=200)

        self.hdg_label = tk.Label(text='Test Text Label',\
                                  font=('Consolas',16), \
                                  fg='orange')
        self.hdg_label.grid(row=2,column=1)


        tk.mainloop()

wWin = WeatherWin()
```

declare the class

the window

title

start the main loop

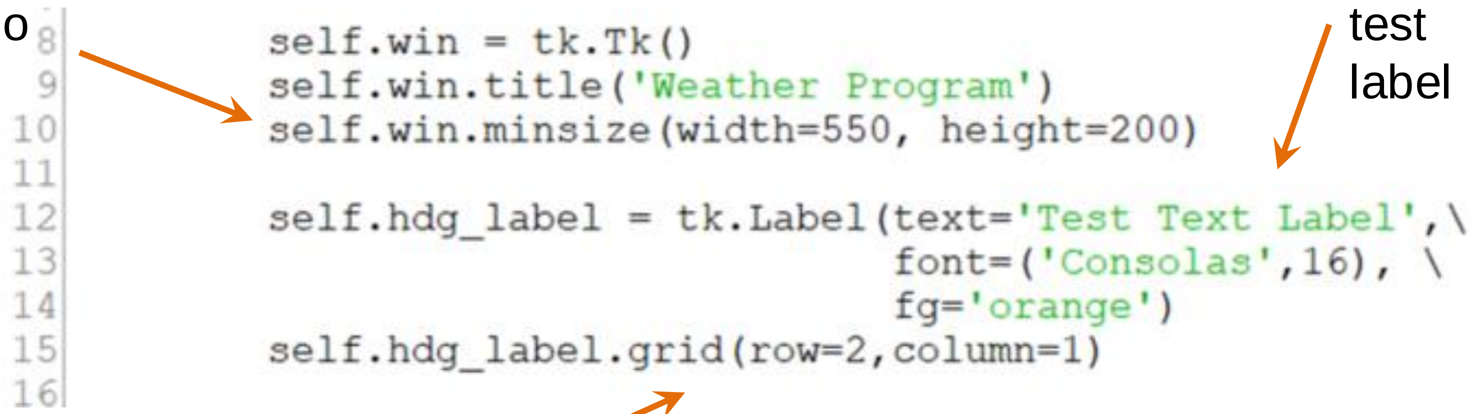
create an instance of the class

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example

window size

test label



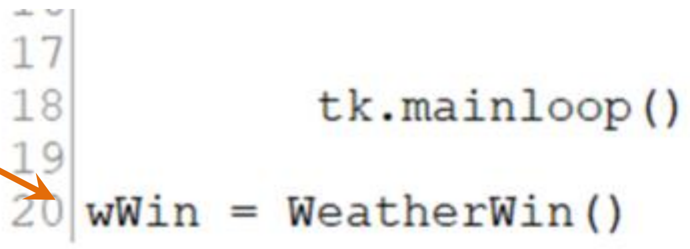
```
self.win = tk.Tk()
self.win.title('Weather Program')
self.win.minsize(width=550, height=200)

self.hdg_label = tk.Label(text='Test Text Label',\
                          font=('Consolas',16), \
                          fg='orange')
self.hdg_label.grid(row=2,column=1)
```

position the label

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example
  - The tkinter ***main loop*** in the program is used to “listen” for events like a button click when the program is running
  - The loop executes when the program starts and continues running until the user ends the program
  - The call to start it is the last line in the class

create an instance of the class

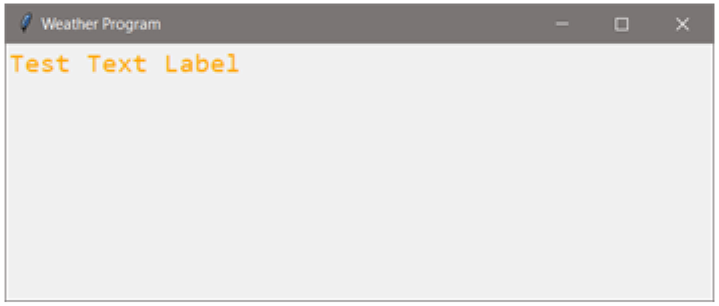


```
            tk.mainloop()

wWin = WeatherWin()
```

start the main loop

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Example
  - The program is tested in this limited form
  - Build a portion of the project and test that portion
  - It is easier to debug a few lines of code than it is to debug fifty lines of code



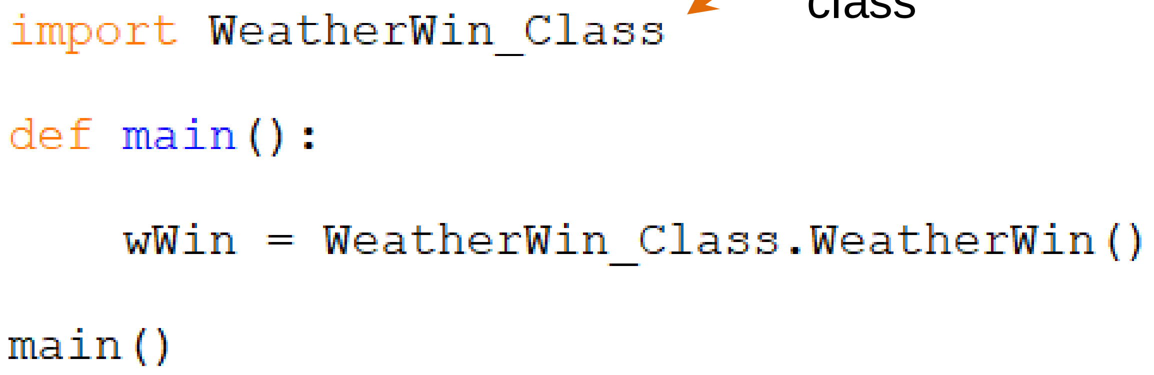
Test Text Label

“Build a little, test a little”

Chapter 10 Graphical User Interfaces (GUIs)

- Comments before continuing
  - For the example, The instance of the class is being created in the same file
  - A program would typically create an instance of the class

File containing the class



```
import WeatherWin_Class

def main():

    wWin = WeatherWin_Class.WeatherWin()

main()
```

Chapter 10 Graphical User Interfaces (GUIs)

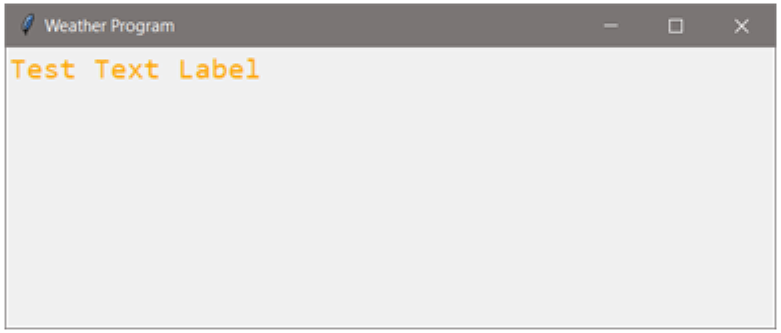
- Positioning Components
  - Tkinter provides Geometry Managers to position components
    - pack() - organizes within a block using a few available options
    - grid() - organizes using rows and columns with customizing options
    - place() - organizes using x, y coordinates

The example uses the Grid Geometry Manager

Chapter 10 Graphical User Interfaces (GUIs)

- Positioning Components
  - The label with text was positioned in row 2 and column 1
  - But, the label is displayed top-left in the window

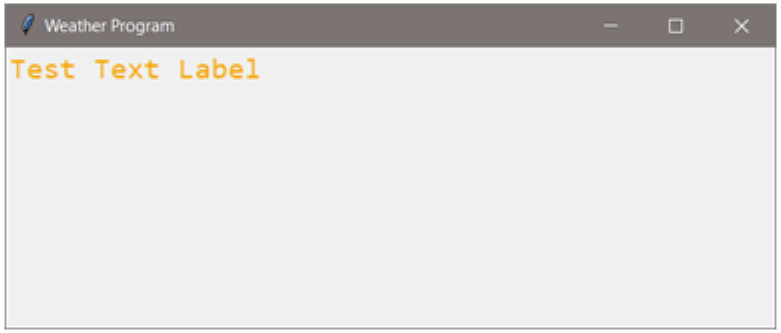
```
self.hdg_label.grid(row=2,column=1)
```



Test Text Label

Chapter 10 Graphical User Interfaces (GUIs)

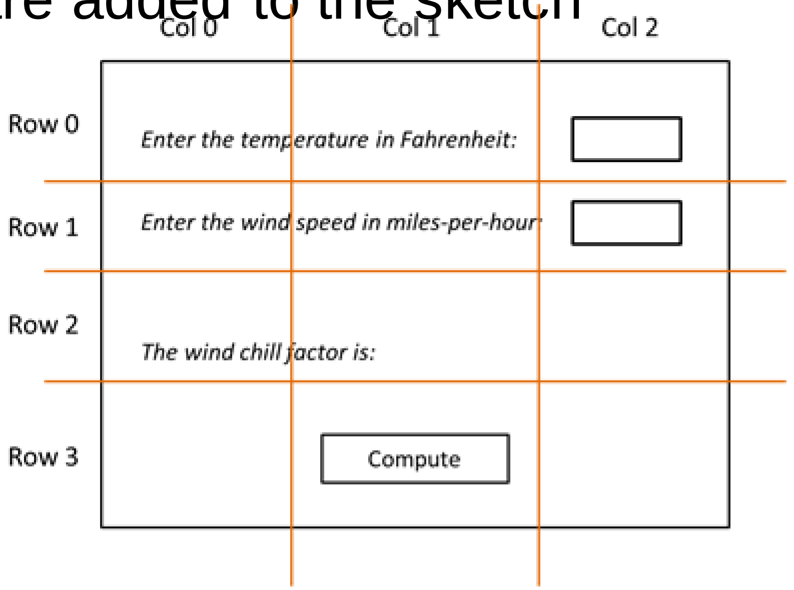
- Positioning Components
  - Grid will size each row and column to the smallest size required for the items that are in them
  - There is nothing in row 0 or 1 and nothing in column 0, so…
  - This is resolved by setting row and column sizes



Test Text Label

Chapter 10 Graphical User Interfaces (GUIs)

- Positioning Components
  - Positioning elements should be completed in the design phase
  - Grid positions elements in rows and columns
  - The lines are added to the sketch


Col 0
Col 1
Col 2
Row 0
Row 1
Row 2
Row 3
Enter the temperature in Fahrenheit:
Enter the wind speed in miles-per-hour
The wind chill factor is:
Compute

Chapter 10 Graphical User Interfaces (GUIs)

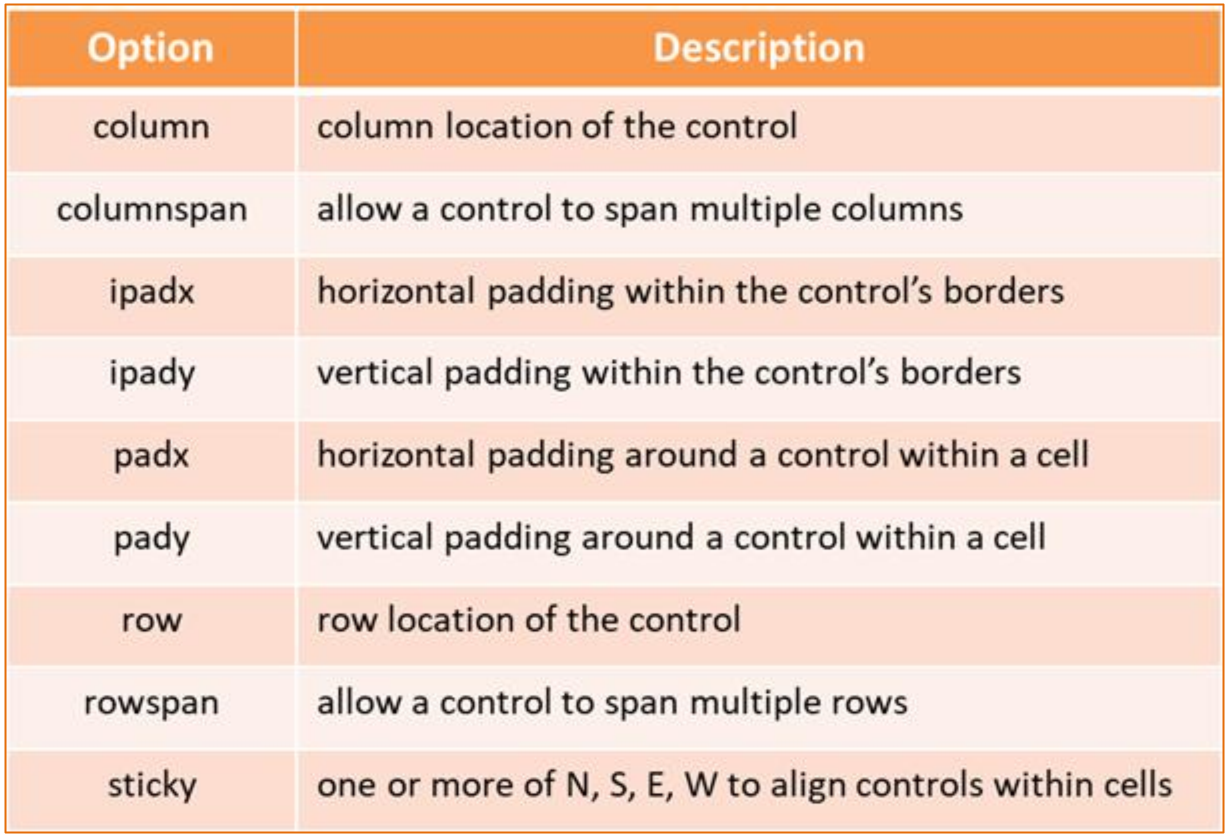
- Setting row heights and column widths
  - The ***configure*** method is used to set the row heights and column widths
    - Numbering begins at zero for rows and columns

sets the height of row 0 to 50 pixels →	`rowconfigure(0,50)`
sets the width of column 0 to 100 pixels →	`columnconfigure(0,100)`

Configure allows row heights and column widths to be fixed

Chapter 10 Graphical User Interfaces (GUIs)

- Grid Options



Option	Description
column	column location of the control
columnspan	allow a control to span multiple columns
ipadx	horizontal padding within the control's borders
ipady	vertical padding within the control's borders
padx	horizontal padding around a control within a cell
pady	vertical padding around a control within a cell
row	row location of the control
rowspan	allow a control to span multiple rows
sticky	one or more of N, S, E, W to align controls within cells

Chapter 10 Graphical User Interfaces (GUIs)

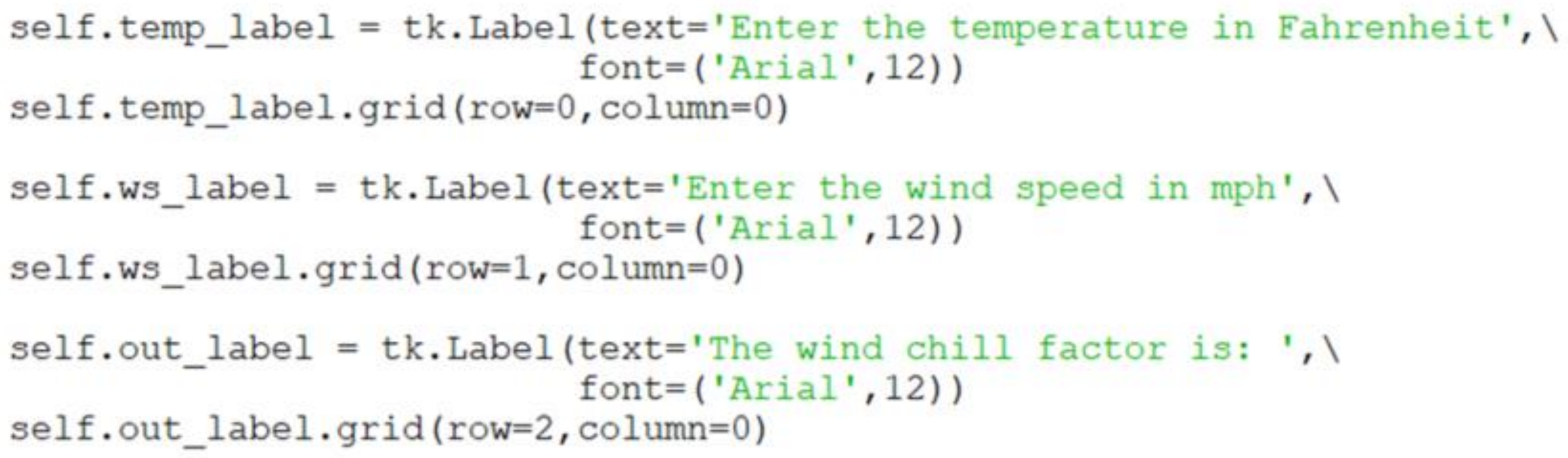
- Interface Design
  - Set row heights and column widths using the sketch

```
# configure columns and rows
for c in range(3):
    self.win.columnconfigure(c, minsize=100)

self.win.rowconfigure(0, minsize = 50)
self.win.rowconfigure(1, minsize = 50)
self.win.rowconfigure(2, minsize = 100)
self.win.rowconfigure(3, minsize = 100)
```

A loop can be used for multiple settings

Chapter 10 Graphical User Interfaces (GUIs)

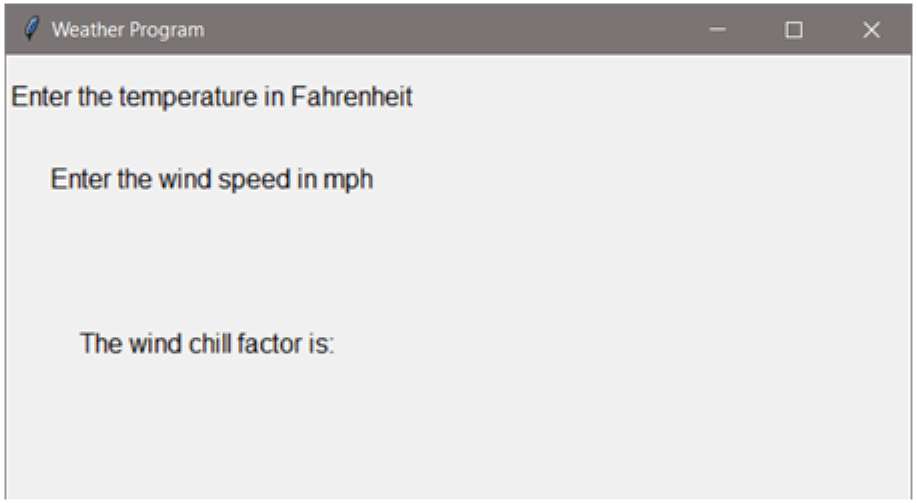
- Interface Design
  - Determine labels and their positions in the rows and columns



```
self.temp_label = tk.Label(text='Enter the temperature in Fahrenheit',\
                          font=('Arial',12))
self.temp_label.grid(row=0,column=0)

self.ws_label = tk.Label(text='Enter the wind speed in mph',\
                          font=('Arial',12))
self.ws_label.grid(row=1,column=0)

self.out_label = tk.Label(text='The wind chill factor is: ',\
                          font=('Arial',12))
self.out_label.grid(row=2,column=0)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Design
  - After each label is created with text and the font option, the location on the grid is assigned
  - The updated code centers the labels within the column



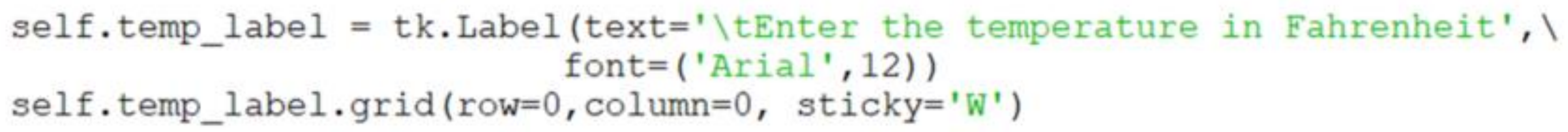
Enter the temperature in Fahrenheit

Enter the wind speed in mph

The wind chill factor is:

Chapter 10 Graphical User Interfaces (GUIs)

- Positioning Components
  - An option for alignment in grid is called ***sticky***
    - Accepts an assignment of N, S, E, or W
      - Using W for west would put the text against the left hand side of the GUI, but a tab can be added to the label text to move it to the right (as a simple solution)

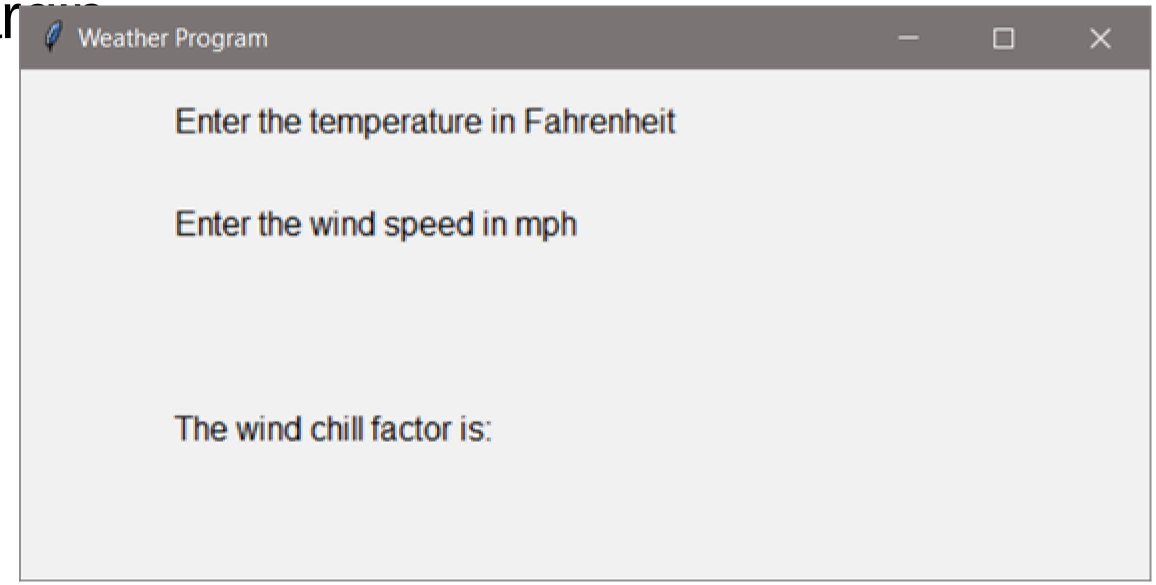


```
self.temp_label = tk.Label(text='\tEnter the temperature in Fahrenheit',\
                           font=('Arial',12))
self.temp_label.grid(row=0,column=0, sticky='W')
```

position left in column

Chapter 10 Graphical User Interfaces (GUIs)

- Interface Design
  - With the addition of the tab and sticky option, the text labels are left aligned and tabbed from the mar



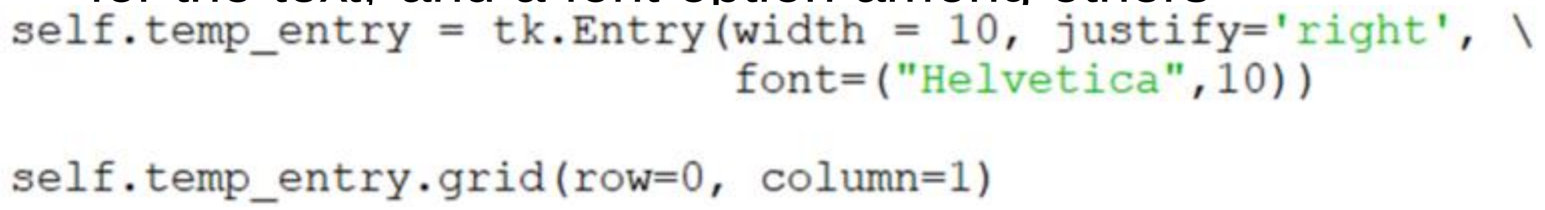
Enter the temperature in Fahrenheit

Enter the wind speed in mph

The wind chill factor is:

Chapter 10 Graphical User Interfaces (GUIs)

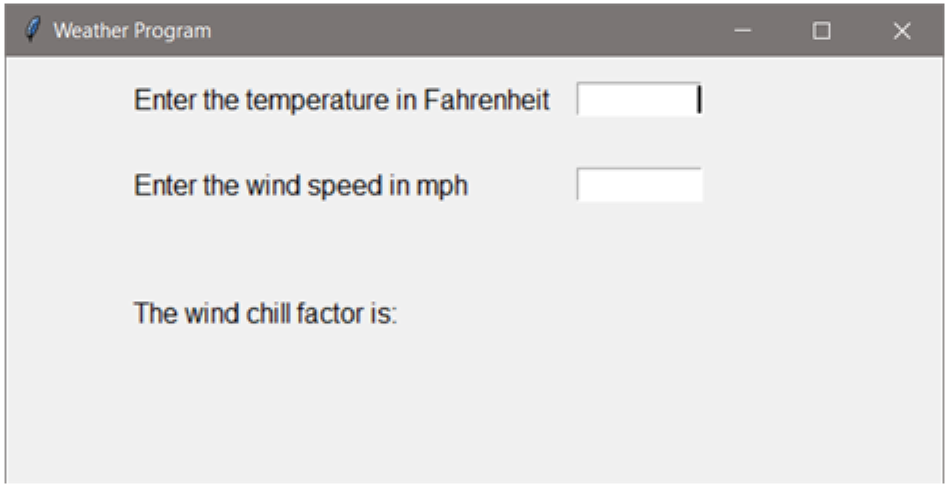
- Entry Controls
  - The entry controls for input are on the same rows as the prompts for temperature and wind speed (in the next column)
  - Entry controls accept a width in characters, justification for the text, and a font option among others



```
self.temp_entry = tk.Entry(width = 10, justify='right', \
                           font=("Helvetica",10))

self.temp_entry.grid(row=0, column=1)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Entry Controls
  - Entry control options include foreground, background, font, relief, and there is a show option to display a character such as “*” instead of what the user is typing



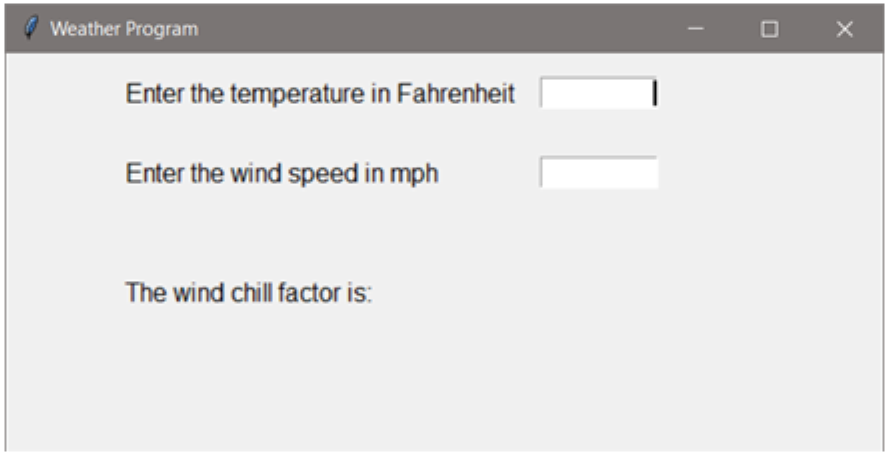
Enter the temperature in Fahrenheit

Enter the wind speed in mph

The wind chill factor is:

Chapter 10 Graphical User Interfaces (GUIs)

- Entry Controls
  - The ***get()*** will be used when the button is clicked to obtain what the user entered in the entry controls
  - This will be handled in a function called when the button is clicked



Enter the temperature in Fahrenheit

Enter the wind speed in mph

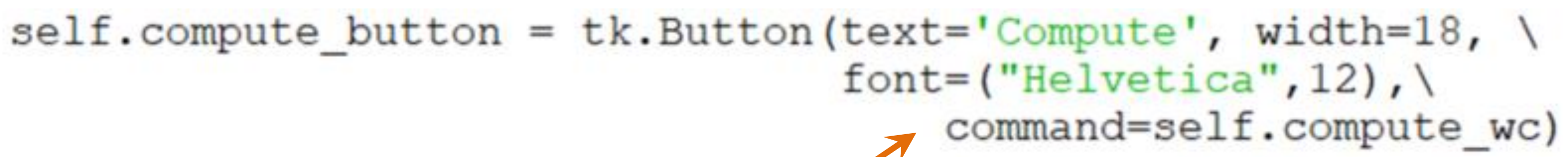
The wind chill factor is:

Chapter 10 Graphical User Interfaces (GUIs)

- Button Controls
  - The Tkinter button has many options for customization and can be positioned using grid
  - In the example, a button that reacts when it is clicked will call a function that gathers the input text, computes the wind chill factor, and displays the result
  - This is commonly referred to as a ***callback function***

Chapter 10 Graphical User Interfaces (GUIs)

- Button Controls
  - The ***command*** option for the button assigns the action to be performed when the button is clicked
  - In the code below a *compute_wc* function will be called



```
self.compute_button = tk.Button(text='Compute', width=18, \
                                font=("Helvetica",12),\
                                    command=self.compute_wc)
```

button command
assignment

callback
function

Chapter 10 Graphical User Interfaces (GUIs)

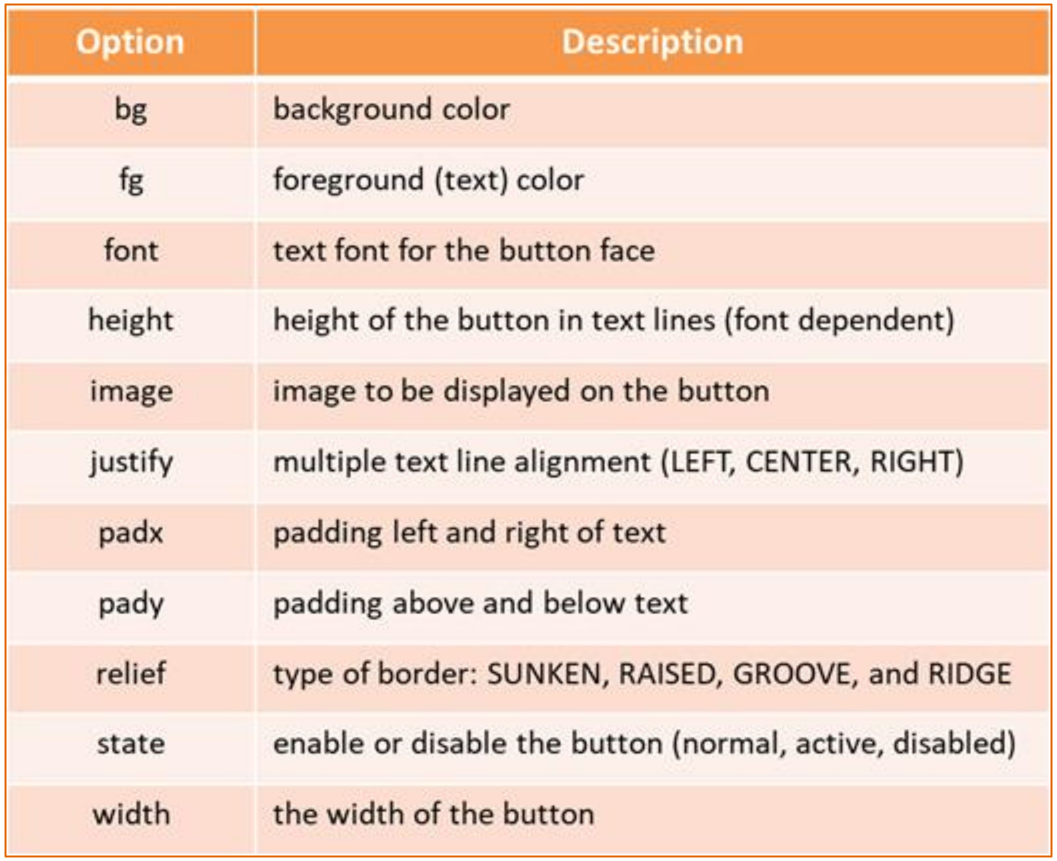
- Button Controls
  - To center the button in the interface, the *columnspan* option is used allowing it to span all three columns of the interface (centered by default)

```
self.compute_button = tk.Button(text='Compute', width=18, \
                                font=("Helvetica",12),\
                                    command=self.compute_wc)

self.compute_button.grid(row=3,column=0, columnspan=3)
```

allows the button to span all 3 columns

Chapter 10 Graphical User Interfaces (GUIs)

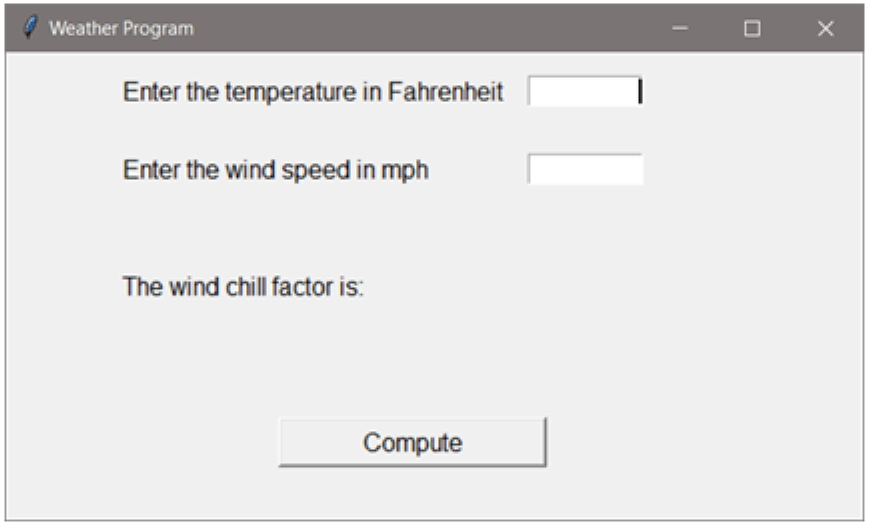
- Button Options



Option	Description
bg	background color
fg	foreground (text) color
font	text font for the button face
height	height of the button in text lines (font dependent)
image	image to be displayed on the button
justify	multiple text line alignment (LEFT, CENTER, RIGHT)
padx	padding left and right of text
pady	padding above and below text
relief	type of border: SUNKEN, RAISED, GROOVE, and RIDGE
state	enable or disable the button (normal, active, disabled)
width	the width of the button

Chapter 10 Graphical User Interfaces (GUIs)

- Weather Program Example
  - Notice that the code to actually compute the wind chill factor has not been written
  - Setting up the GUI has been the focus so far

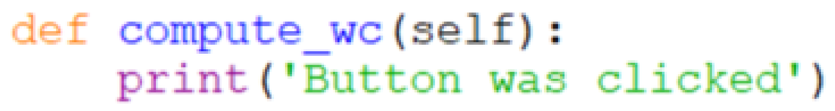

Weather Program
Enter the temperature in Fahrenheit
Enter the wind speed in mph
The wind chill factor is:
Compute

Chapter 10 Graphical User Interfaces (GUIs)

- Weather Program Example
  - The callback function for the button will be defined within the class to provide access to the interface controls
    - Later an example will show how the function can be located in another module or inline using what is called a lambda expression
  - For now, the *command* option assignment will execute a function within the class

Chapter 10 Graphical User Interfaces (GUIs)

- Weather Program Example
  - The function will include verification of the input, computing the wind chill factor if valid data was entered, and updating the label to show the result of the computation
  - A simple output statement can be used for testing purposes



```
def compute_wc(self):
    print('Button was clicked')
```

Print statements can be used to quickly test parts of the program

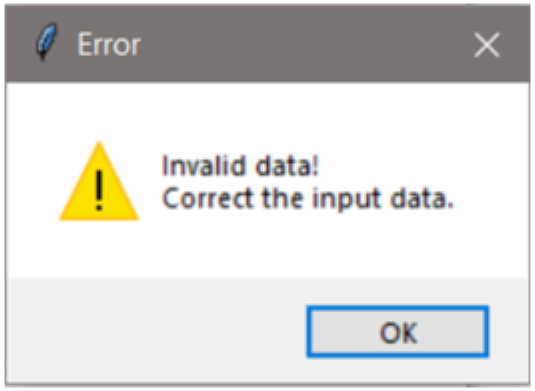
Chapter 10 Graphical User Interfaces (GUIs)

- Weather Program Example
  - The callback function will use the ***get()*** function for the entry controls to retrieve the input as a string
  - The input must be converted to floats for the calculations

```
def compute_wc(self):
    temp_string = self.temp_entry.get()
    ws_string = self.ws_entry.get()
    try:
        temp = float(temp_string)
        ws = float(ws_string)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Dialog and Information Boxes
  - If the data entered is invalid, the user must be alerted to the problem and be given the ability to correct the issue without restarting the program
  - A Dialog box can be used



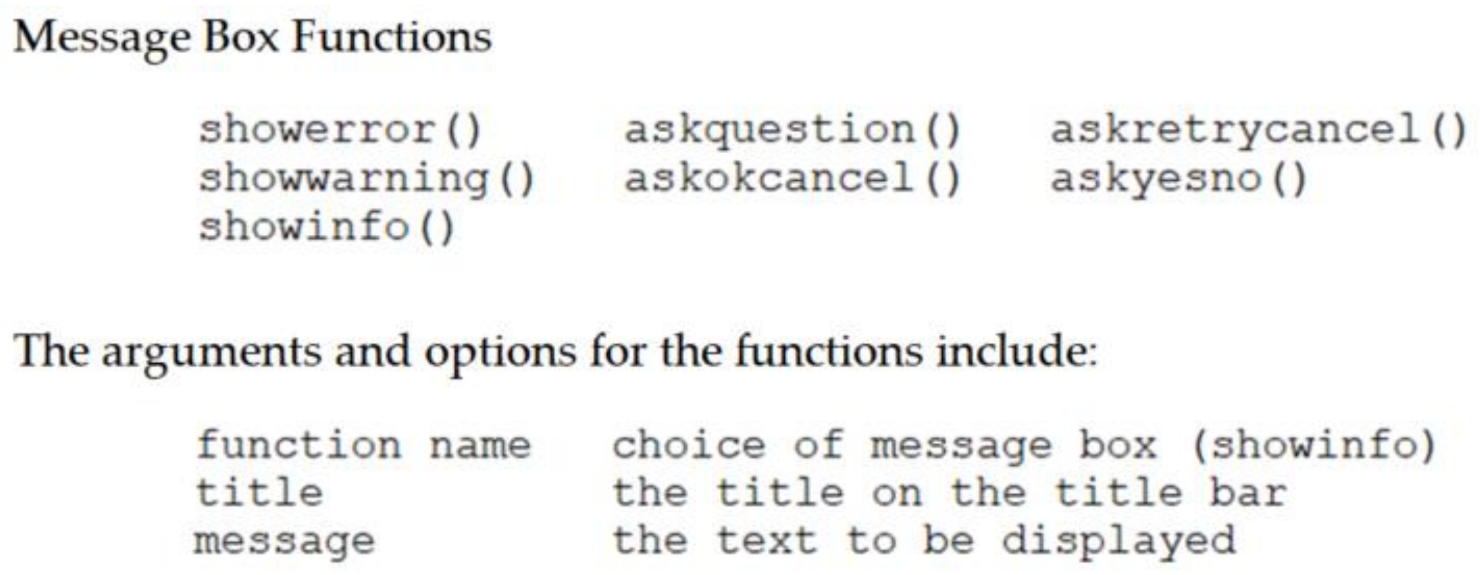
Invalid data!
Correct the input data.

Chapter 10 Graphical User Interfaces (GUIs)

- Dialog and Information Boxes
  - Issues in GUI programs are typically handled using a dialog or message box
  - The box explains the issue and has an “OK” button that must be clicked to continue
  - The ***tkinter.messagebox*** module provides information boxes
    - Requires importing that module

Chapter 10 Graphical User Interfaces (GUIs)

- Message Box Functions



Message Box Functions

```
showerror()       askquestion()     askretrycancel()
showwarning()     askokcancel()     askyesno()
showinfo()
```

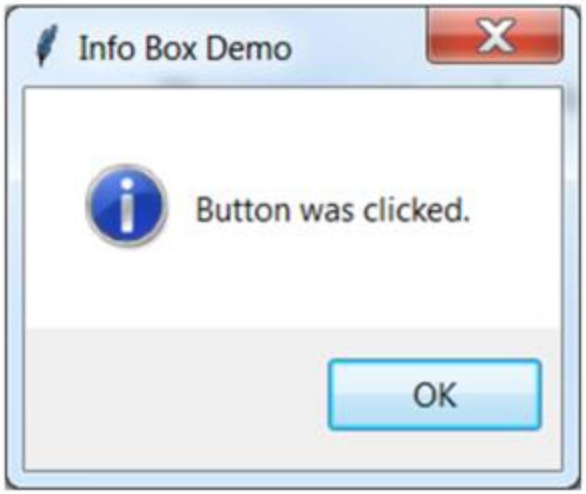
The arguments and options for the functions include:

`function name`	`choice of message box (showinfo)`
`title`	`the title on the title bar`
`message`	`the text to be displayed`

Chapter 10 Graphical User Interfaces (GUIs)

- Message Box Functions
  - Standard ***showinfo*** message box
  - Options including setting the window title and text in the

```
tk.messagebox.showinfo('Info Box Demo', 'Button was clicked.')
```

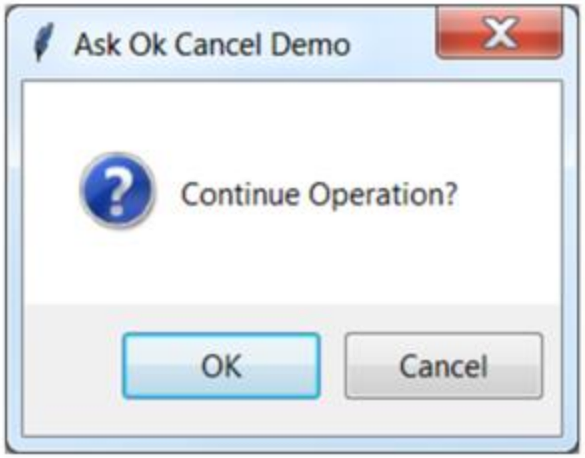


Button was clicked.

Chapter 10 Graphical User Interfaces (GUIs)

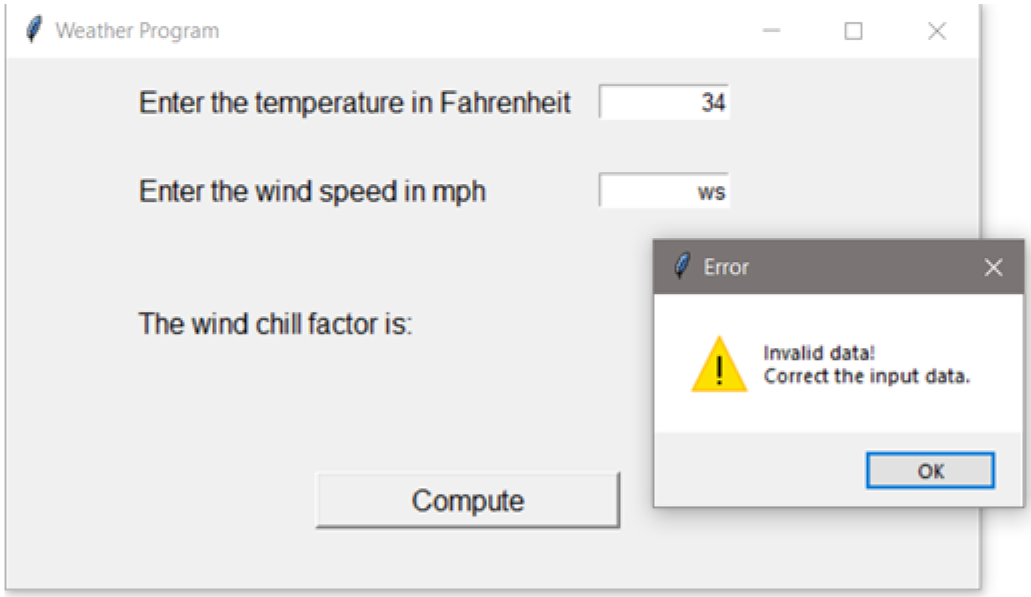
- Message Box Functions
  - The ***ask-ok-cancel*** box

```
tk.messagebox.askokcancel('Ask, OK, Cancel', 'Continue Operation?')
```



Chapter 10 Graphical User Interfaces (GUIs)

- The Example
  - The message box alerts the user and allows the program to continue after the “OK” button is clicked


Weather Program
Enter the temperature in Fahrenheit
34
Enter the wind speed in mph
ws
The wind chill factor is:
Compute
Error
OK

Invalid data!
Correct the input data.

Chapter 10 Graphical User Interfaces (GUIs)

- The Example
  - The actual computation and output of the wind chill factor has yet to be completed
  - Once the calculation is complete, the label on the GUI can be updated to include the resulting wind chill factor using the ***config*** method
  - The statement will be a copy of the original label text, with the wind chill factor appended

Chapter 10 Graphical User Interfaces (GUIs)

- The Example
  - The ***config*** method provides a way to update a label
  - The label creation code is shown here:

```
self.out_label = tk.Label(text='\tThe wind chill factor is: ',\
                          font=('Arial',12))
self.out_label.grid(row=2,column=0, sticky='W')
```

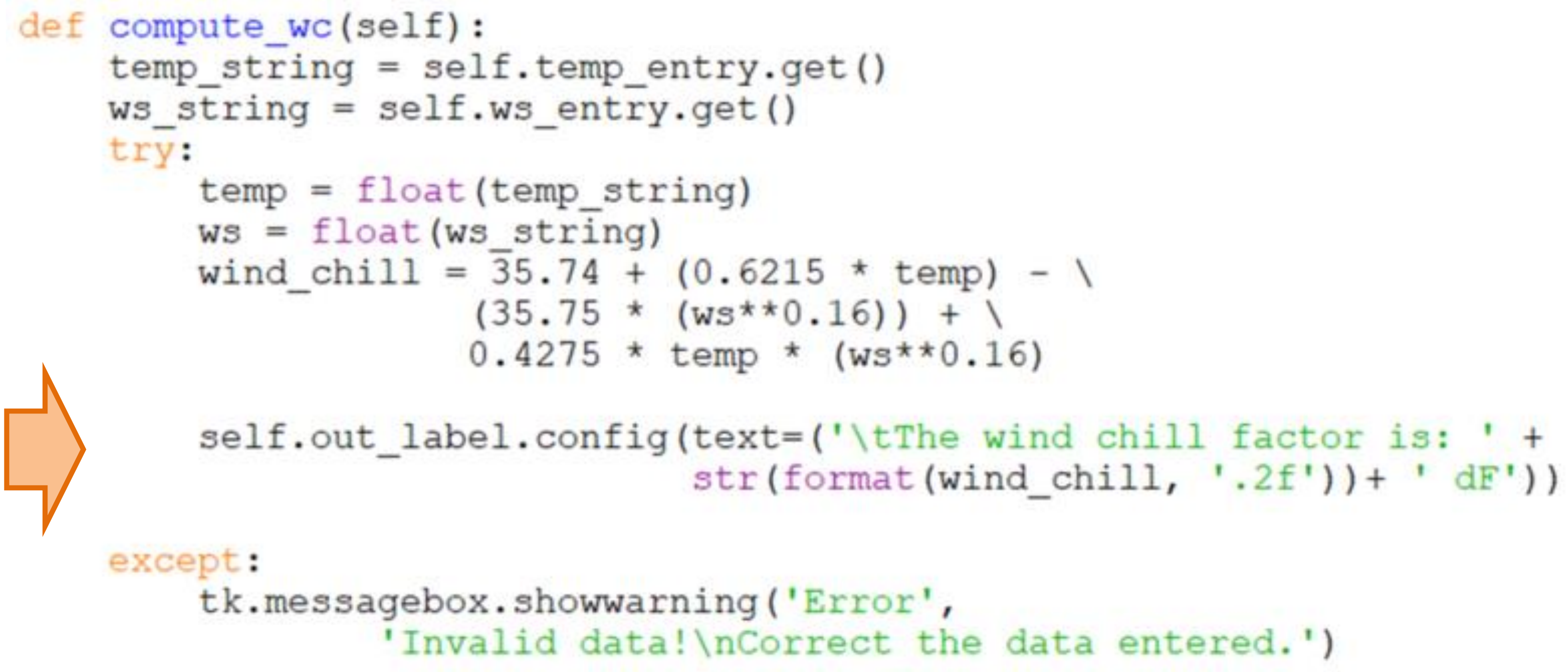
  - The label update code is shown here:

```
self.out_label.config(text=('\tThe wind chill factor is: ' +
                      str(format(wind_chill, '.2f'))+ ' dF'))
```

result add to the label

Chapter 10 Graphical User Interfaces (GUIs)

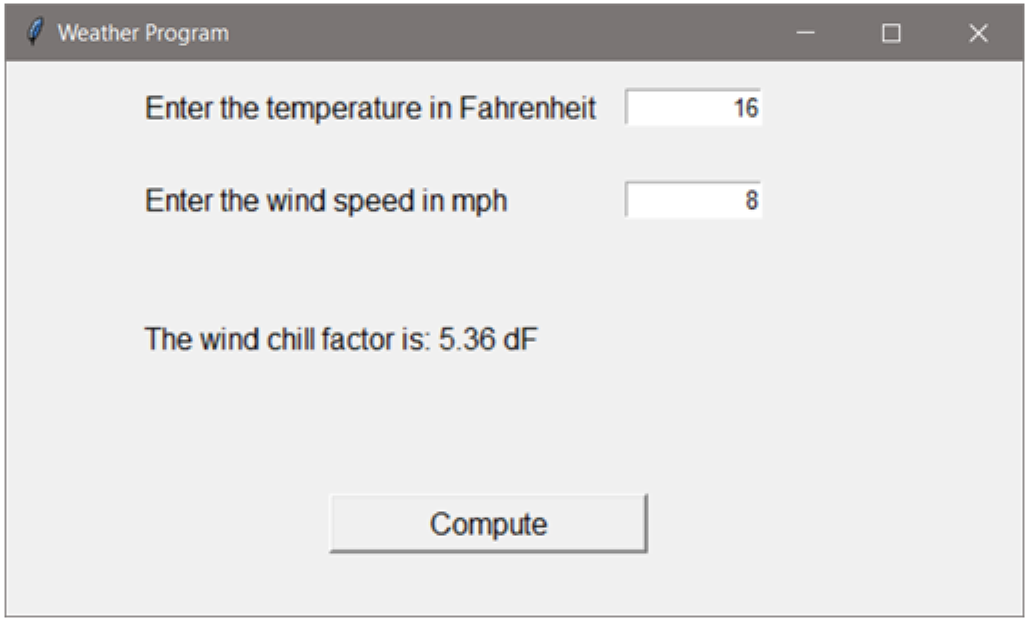
- The Example
  - The label is updated by the function after the



```
def compute_wc(self):
    temp_string = self.temp_entry.get()
    ws_string = self.ws_entry.get()
    try:
        temp = float(temp_string)
        ws = float(ws_string)
        wind_chill = 35.74 + (0.6215 * temp) - \
                    (35.75 * (ws**0.16)) + \
                    0.4275 * temp * (ws**0.16)

        self.out_label.config(text=('\tThe wind chill factor is: ' +
                                   str(format(wind_chill, '.2f'))+ ' dF'))

    except:
        tk.messagebox.showwarning('Error',
                'Invalid data!\nCorrect the data entered.')
```

Chapter 10 Graphical User Interfaces (GUIs)

- The Weather Example
  - The final GUI



Enter the temperature in Fahrenheit 16

Enter the wind speed in mph 8

The wind chill factor is: 5.36 dF

Chapter 10 Graphical User Interfaces (GUIs)

The StringVar

and

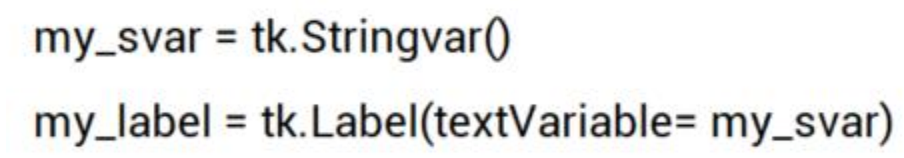
More Controls

Chapter 10 Graphical User Interfaces (GUIs)

- StringVar
  - Another way to update/change a label is to use a ***StringVar*** object
  - The StringVar modifies any control that uses it whenever the contents of the StringVar is changed
    - This provides an immediate update to a control anytime the value that is stored in the StringVar object changes

Chapter 10 Graphical User Interfaces (GUIs)

- StringVar
  - A StringVar is declared and assigned to a control using the ***textVariable*** assignment

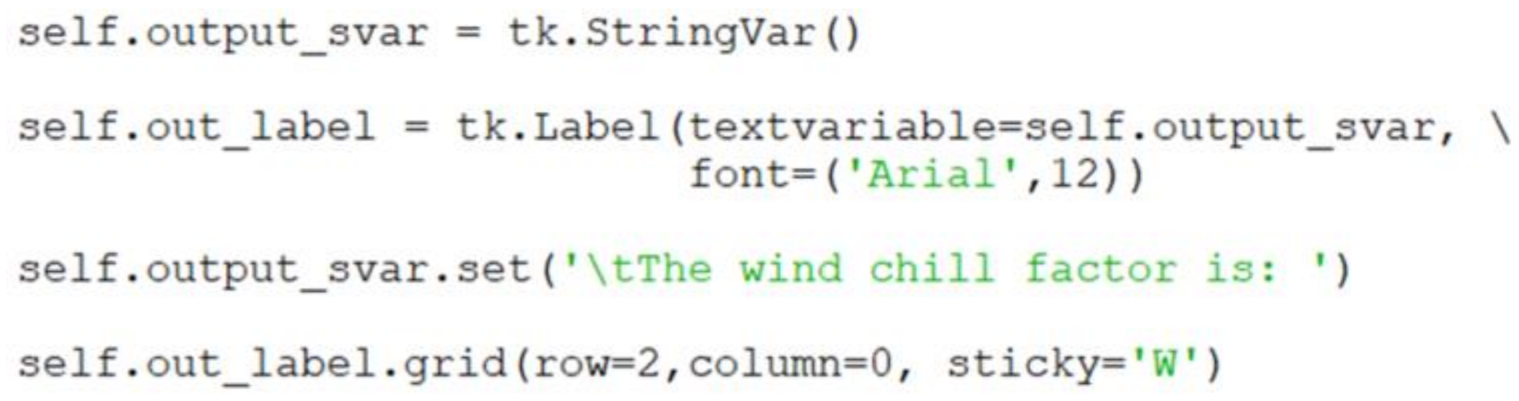


```
my_svar = tk.Stringvar()
my_label = tk.Label(textVariable= my_svar)
```

  - The ***IntVar*** is the integer version of the object

Chapter 10 Graphical User Interfaces (GUIs)

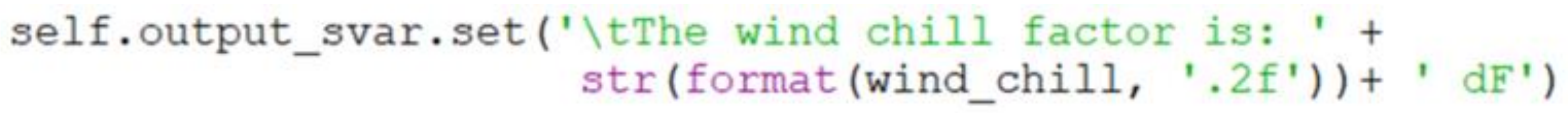
- StringVar
  - A StringVar is declared, initialized, and then updated using the **set()** method
  - The weather example code has been modified to include declaring a StringVar and assigning it to the output label



```
self.output_svar = tk.StringVar()

self.out_label = tk.Label(textvariable=self.output_svar, \
                          font=('Arial',12))

self.output_svar.set('\tThe wind chill factor is: ')

self.out_label.grid(row=2,column=0, sticky='W')
```

Chapter 10 Graphical User Interfaces (GUIs)

- StringVar
  - The compute function is modified to use ***set()*** to update the StringVar



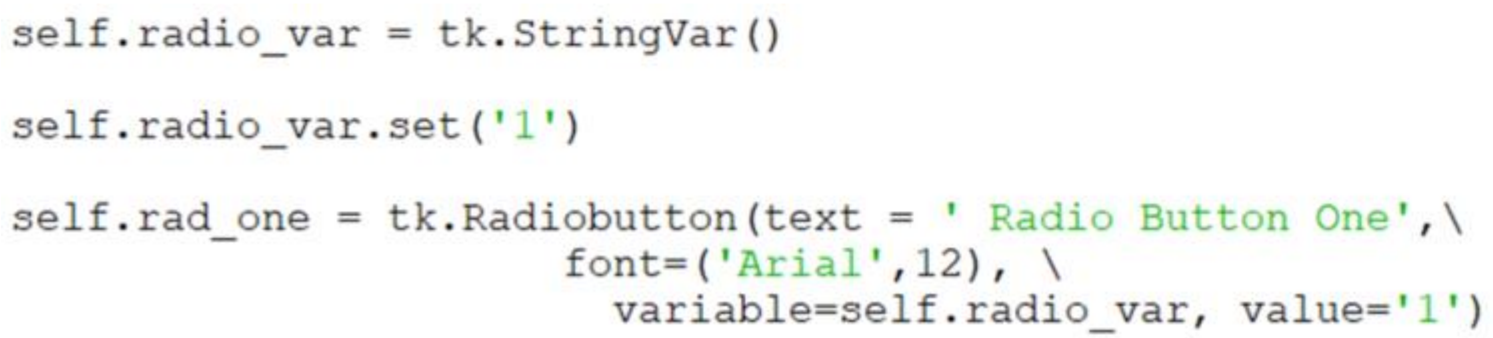
```
self.output_svar.set('\tThe wind chill factor is: ' +
                     str(format(wind_chill, '.2f'))+ ' dF')
```

Chapter 10 Graphical User Interfaces (GUIs)

- Radio Buttons
  - When operation of the program requires that only one selection be made by the user, ***radio buttons*** accommodate this because they are mutually exclusive
  - If a second button is selected, the button that was previously selected is unselected
  - Options for radio buttons are similar to other controls including text and fonts, and they can be located using the geometry managers

Chapter 10 Graphical User Interfaces (GUIs)

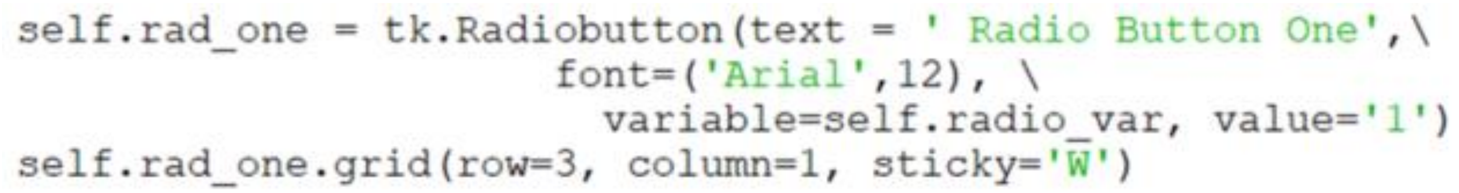
- Radio Buttons
  - Declare a StringVar to store the radio button selection
  - Set one of the buttons as a default
  - Declare a radio button with text and font options, assign the StringVar to the radio button's variable, and assign a value for the button



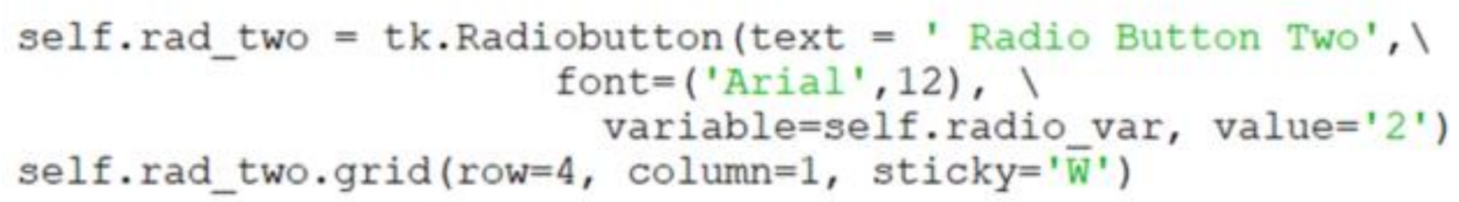
```
self.radio_var = tk.StringVar()

self.radio_var.set('1')

self.rad_one = tk.Radiobutton(text = ' Radio Button One',\
                              font=('Arial',12), \
                                variable=self.radio_var, value='1')
```

Chapter 10 Graphical User Interfaces (GUIs)

- Radio Buttons
  - When declaring multiple radio buttons, each will have a unique value assigned



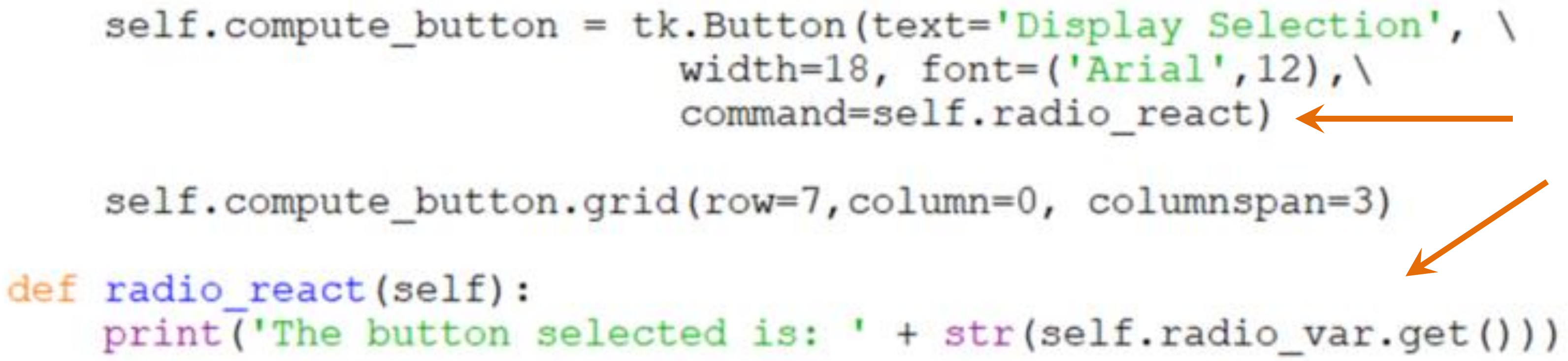
```
self.rad_one = tk.Radiobutton(text = ' Radio Button One',\
                              font=('Arial',12), \
                                variable=self.radio_var, value='1')
self.rad_one.grid(row=3, column=1, sticky='W')
```



```
self.rad_two = tk.Radiobutton(text = ' Radio Button Two',\
                              font=('Arial',12), \
                                variable=self.radio_var, value='2')
self.rad_two.grid(row=4, column=1, sticky='W')
```

Chapter 10 Graphical User Interfaces (GUIs)

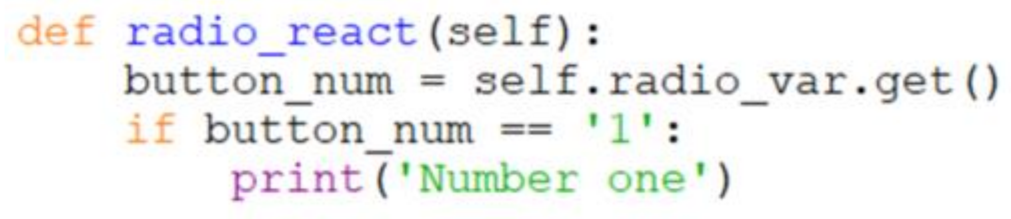
- Radio Buttons
  - A button click is typically used to obtain the radio button selected through a command assignment and function
  - The button selected is obtained through the StringVar using get()



```
    self.compute_button = tk.Button(text='Display Selection', \
                                    width=18, font=('Arial',12),\
                                    command=self.radio_react)

    self.compute_button.grid(row=7,column=0, columnspan=3)

def radio_react(self):
    print('The button selected is: ' + str(self.radio_var.get()))
```

Chapter 10 Graphical User Interfaces (GUIs)

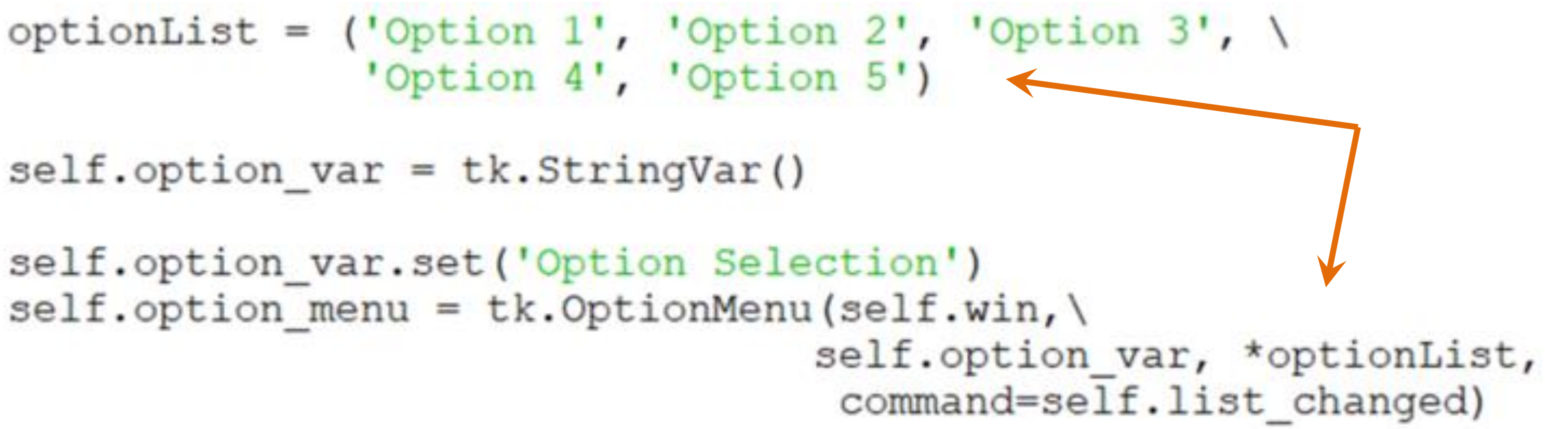
- Radio Buttons
  - After retrieving the selected button using ***get()***, conditional statements can be used to respond with specific operations



```
def radio_react(self):
    button_num = self.radio_var.get()
    if button_num == '1':
        print('Number one')
```

Chapter 10 Graphical User Interfaces (GUIs)

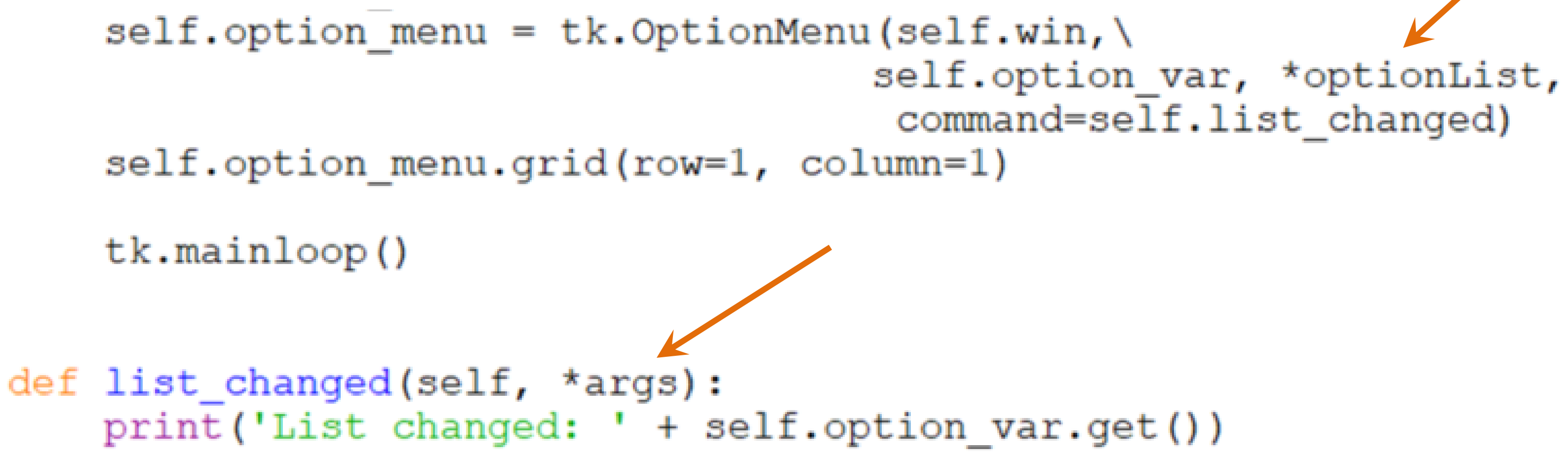
- Option Lists
  - Another mutually exclusive control that allows the user to select is an ***option list*** which operates similar to radio buttons



```
optionList = ('Option 1', 'Option 2', 'Option 3', \
              'Option 4', 'Option 5')

self.option_var = tk.StringVar()

self.option_var.set('Option Selection')
self.option_menu = tk.OptionMenu(self.win,\
                                 self.option_var, *optionList,
                                  command=self.list_changed)
```

Chapter 10 Graphical User Interfaces (GUIs)

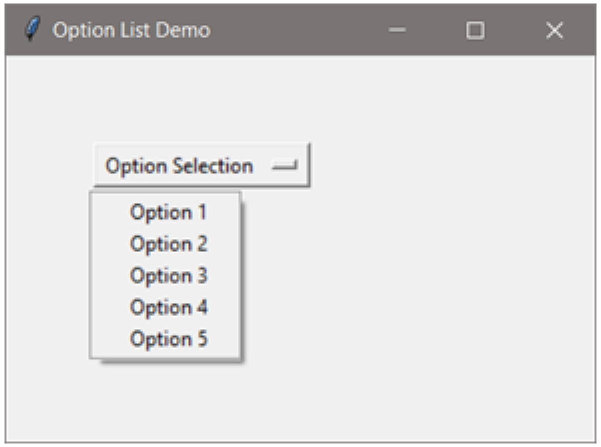
- Option Lists
  - One difference is the way the list itself is handled



```
    self.option_menu = tk.OptionMenu(self.win,\
                                     self.option_var, *optionList,
                                      command=self.list_changed)
    self.option_menu.grid(row=1, column=1)

    tk.mainloop()


def list_changed(self, *args):
    print('List changed: ' + self.option_var.get())
```

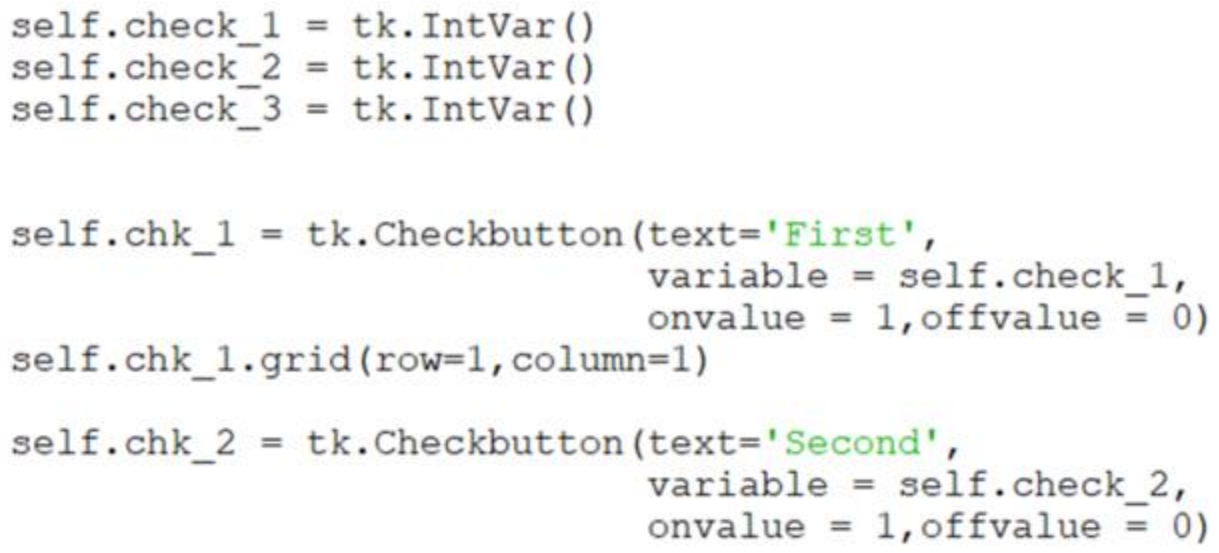
Chapter 10 Graphical User Interfaces (GUIs)

- Option Lists
  - A StringVar stores the value selected and a command is linked to the list
  - The function called needs to receive the arguments and use the ***get()*** method to obtain the selection



Chapter 10 Graphical User Interfaces (GUIs)

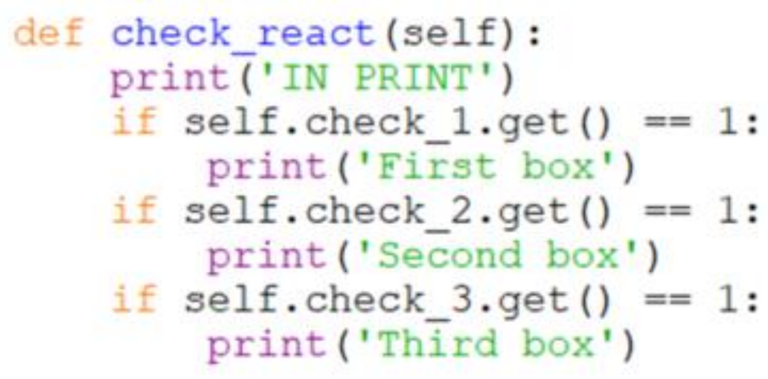
- Check Buttons (Boxes)
  - To allow multiple selections, check boxes are used
  - Here an intVar is used for each check button
    - Each can be on or off, and values are assigned for each



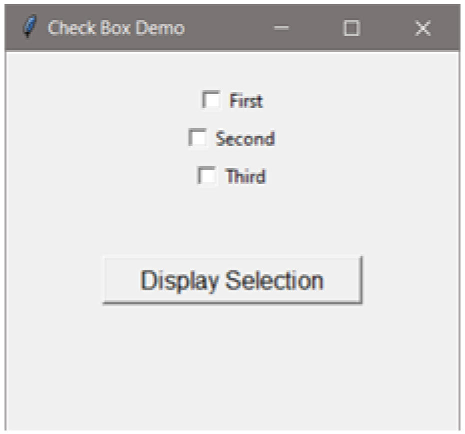
```
self.check_1 = tk.IntVar()
self.check_2 = tk.IntVar()
self.check_3 = tk.IntVar()


self.chk_1 = tk.Checkbutton(text='First',
                            variable = self.check_1,
                            onvalue = 1,offvalue = 0)
self.chk_1.grid(row=1,column=1)

self.chk_2 = tk.Checkbutton(text='Second',
                            variable = self.check_2,
                            onvalue = 1,offvalue = 0)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Check Buttons (Boxes)
  - Since multiple boxes could be checked by the user, they all need to be tested



```
def check_react(self):
    print('IN PRINT')
    if self.check_1.get() == 1:
        print('First box')
    if self.check_2.get() == 1:
        print('Second box')
    if self.check_3.get() == 1:
        print('Third box')
```

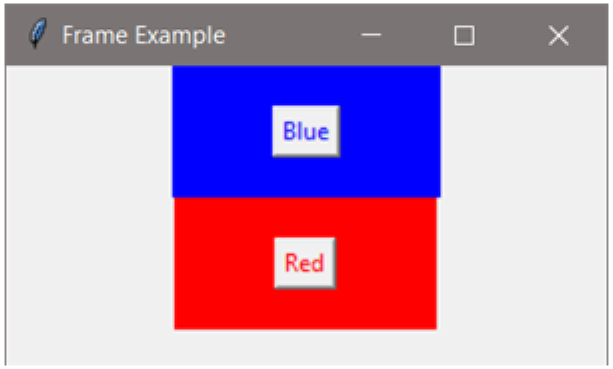


Chapter 10 Graphical User Interfaces (GUIs)

- Frames
  - The tkinter ***frame*** (panel) is a container that can hold other controls
  - It displays as a rectangle and is used to organize other controls and provide additional customization
  - Multiple frames can hold different components arranged in different ways, and the frames can be positioned in the main window using grid geometry

Chapter 10 Graphical User Interfaces (GUIs)

- Frames
  - The frame example creates two frames with a button on each, and uses padding for x and y around the buttons to expand and display the frames
    - The padding and color has been added to highlight the frames



Chapter 10 Graphical User Interfaces (GUIs)

- Frames
  - Frame options include

bd	size of the border (defaults to 2 pixels)
bg	background color
height	height of the frame
relief	type of border: flat, groove, raised, ridge,
solid,	or sunken
width	width of the frame

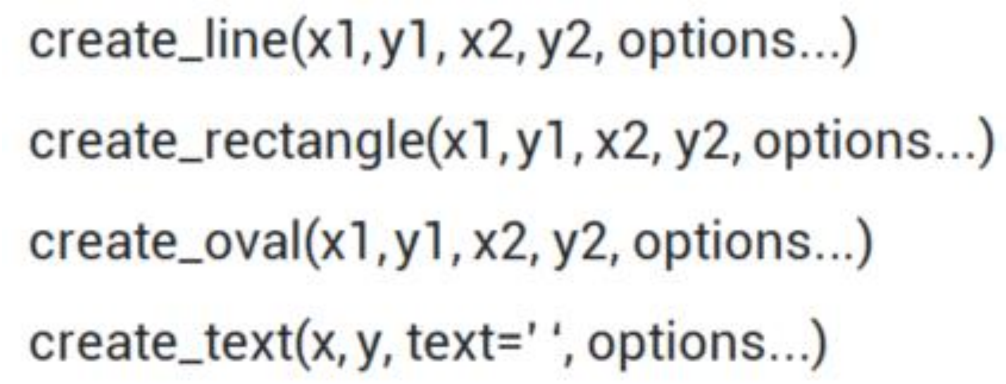
Chapter 10 Graphical User Interfaces (GUIs)

- ***Canvas***
  - Used for drawing graphics
    - Graphs, charts, plots, lines, and geometric shapes
    - Can be used with the Matplotlib module
    - Canvas x, y coordinate system locates 0, 0 at the top-left of the canvas and the units are pixels

A canvas is used for drawing graphics

Chapter 10 Graphical User Interfaces (GUIs)

- Canvas
  - General formats for drawing on a canvas

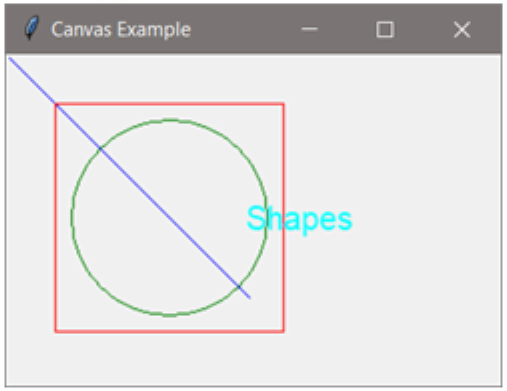


```
create_line(x1,y1, x2, y2, options...)
create_rectangle(x1,y1, x2, y2, options...)
create_oval(x1,y1, x2, y2, options...)
create_text(x, y, text=' ', options...)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Canvas Example

```
self.cv = tk.Canvas(self.win, width=300, height=200)

self.cv.create_line(0,0,150,150, fill='blue')

self.cv.create_rectangle(30,30,170,170, outline='red')

self.cv.create_oval(40,40,160,160, outline='green')

self.cv.create_text(180,100, font=('Arial,14'),
                    text='Shapes', fill='cyan')

self.cv.grid()
```

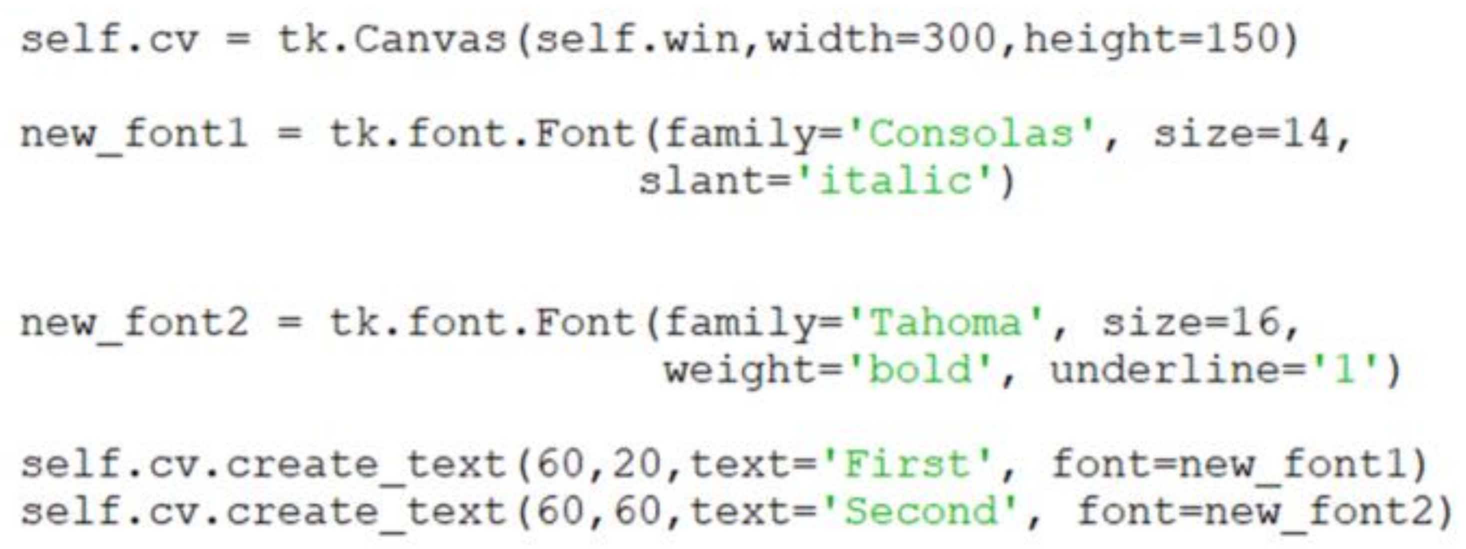


Chapter 10 Graphical User Interfaces (GUIs)

- Canvas Example
  - The font used with *create_text* can be customized using anchor and justify for positioning
  - Can use the tkinter *font* module which provides the ability to assign a font family (Courier, Consolas, etc.), a size in points, a weight (bold and normal), slant (italic), underline, and overstrike (crossed out text)

Chapter 10 Graphical User Interfaces (GUIs)

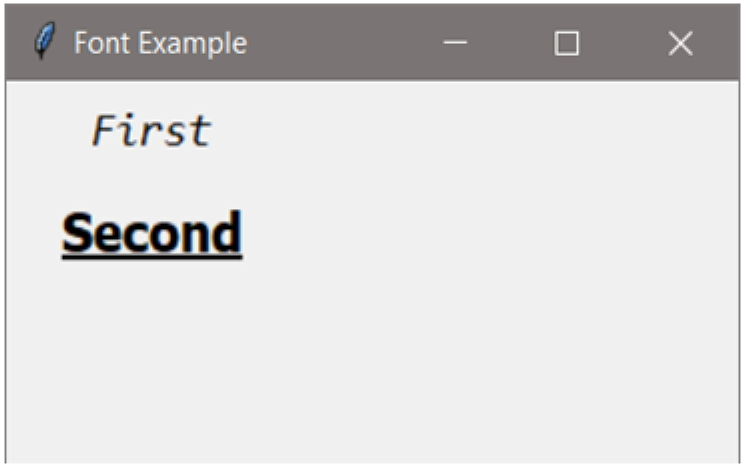
- Canvas Example
  - Font creation and *create_text* example



```
self.cv = tk.Canvas(self.win,width=300,height=150)

new_font1 = tk.font.Font(family='Consolas', size=14,
                         slant='italic')


new_font2 = tk.font.Font(family='Tahoma', size=16,
                          weight='bold', underline='1')

self.cv.create_text(60,20,text='First', font=new_font1)
self.cv.create_text(60,60,text='Second', font=new_font2)
```

Chapter 10 Graphical User Interfaces (GUIs)

- Canvas Example
  - Font creation and *create_text* example



Chapter 10 Graphical User Interfaces (GUIs)

Turtle Graphics

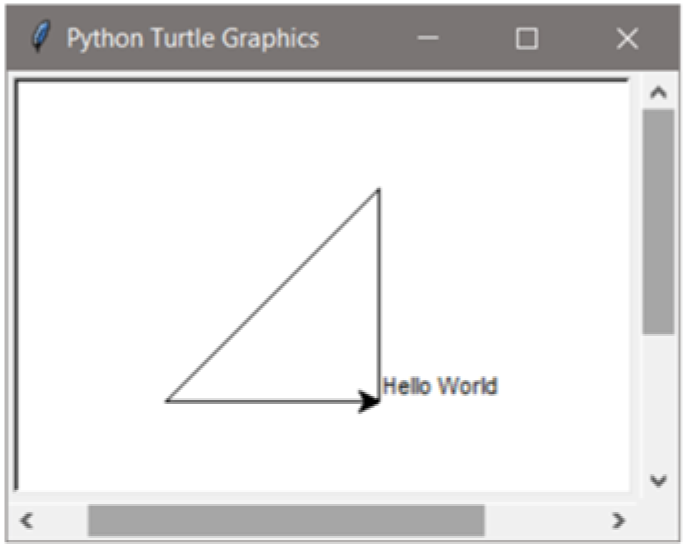
Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics
  - The turtle package is a pre-installed Python library that enables drawing pictures and shapes
  - The “pen” used for drawing is called the turtle
    - Although it is shaped like an arrow
  - Turtle graphics are mainly used to introduce children to computers and a fun way to draw shapes and learn computer commands

Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics
  - The following program imports the turtle package and draws a triangle using three lines, and adds a line of text

```
import turtle

turtle.goto(0,100)
turtle.goto(-100,0)
turtle.goto(0,0)
turtle.write(' Hello World')
```



Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics
  - Use a coordinate system with 0, 0 at the center of the turtle output window
  - The turtle begins at 0, 0 and is pointing east which is 0 degrees for the turtle, with 90 degrees being north, 180 degrees being west, and 270 degrees being south

Chapter 10 Graphical User Interfaces (GUIs)

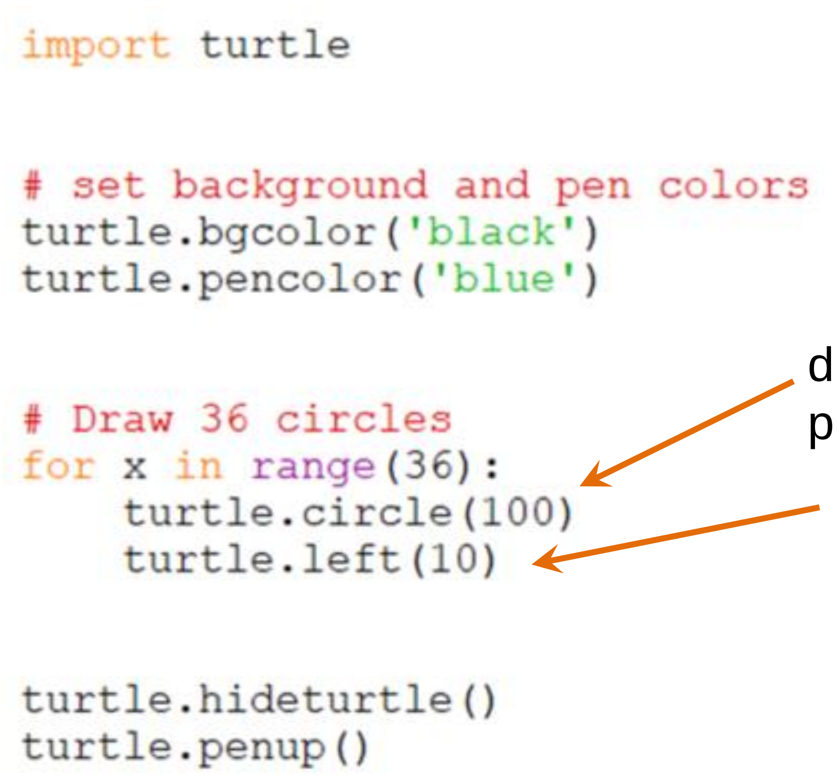
- Turtle Graphics
  - By default the turtle pen is down (writing position), but can be lifted to move it to another location without drawing a line
  - There are options for pen color, background color, changing the angle and pen size, and various shape commands
  - The animation speed can be set so that the drawing is completed slowly instead of all at once

Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics
  - By default the turtle pen is down (writing position), but can be lifted to move it to another location without drawing a line
  - There are options for pen color, background color, changing the angle and pen size, and various shape commands
  - The animation speed can be set so that the drawing is completed slowly instead of all at once

Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics Example



```
import turtle


# set background and pen colors
turtle.bgcolor('black')
turtle.pencolor('blue')


# Draw 36 circles
for x in range(36):
    turtle.circle(100)
    turtle.left(10)


turtle.hideturtle()
turtle.penup()
```

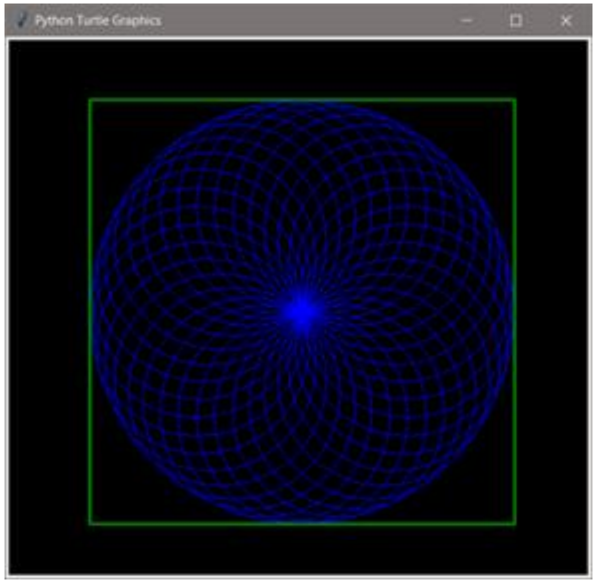
draw a circle 100 pixels

Turn left 10 degrees

Chapter 10 Graphical User Interfaces (GUIs)

- Turtle Graphics Example

```
# change the pen color and width
turtle.pencolor('green')
turtle.pensize(3)


# draw a green square around the shape
turtle.goto(-200,-200)
turtle.pendown()
turtle.goto(-200,200)
turtle.goto(200,200)
turtle.goto(200,-200)
turtle.goto(-200,-200)
```



Chapter 10 Graphical User Interfaces (GUIs)

Chapter 10 Graphical User Interfaces (GUIs)