Chapter 12

Introduction to Tkinter

Objectives

- Use Tkinter to create a simple graphical user interface
- Create a window with buttons and label widgets
- Set the size and colour of a window
- Learn how to use the Pack and Grid Geometry Managers to place widgets in a window

The Python Tkinter module

First came **Tk**, an open source toolkit which provides a library of interactive **widgets** or **Graphical User Interface (GUI)** elements such as windows, labels, buttons, menus and text entry fields.

Tkinter is a standard Python module providing an interface to the Tk toolkit.

You can use the interactive window to test that it is installed and running correctly.

Load Python's IDLE and import the Tkinter package by typing (all in lowercase):

```
>>> import tkinter
```

Then type

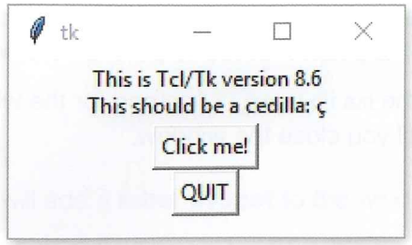
```
>>> import _tkinter
```

This will import the compiled binary associated with the Tkinter package.

Run the test routine by typing:

```
>>> tkinter._test()
```

Press **Enter** and you should see a window like this one:



You can click and drag the edges to resize the window, and drag the window around the screen.

Click on the "Click me!" button and you will see square brackets appear around the words.

Close the window by clicking the "QUIT" button. If all this runs correctly you are ready to start!

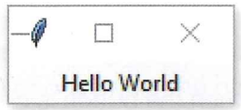
The "Hello World" program

Example 1

Open a new file in IDLE or use your usual text editor to type the following program.

```
#Program name: Ch 12 Example 1 Hello World
from  tkinter import *
root = Tk()
root.title("Greeting")
Label(root, text = "Hello World").pack()
root.mainloop()
```

Save and run the program. You should see a window like this:



This is a brief description of what each line of the program does.

- The first line after the comment imports all the functions and variables from the Tkinter package, along with their names, so that they can be called directly by these names.
- The second line causes a TK constructor method to create a new top level widget, the main window, and assign it to the variable name `root`.
- The third line sets the title of the window. (The window needs to be enlarged to see the title. Try resizing the window.)

- The fourth line creates a label with the text `Hello World` as a child of the root window, and uses the `pack` method to put it on the window.
- The final line runs the `mainloop()` method for the window. The program will continue to run until you close the window.

Widgets

Widgets are the controls, such as buttons and text entry fields, that you use to interface with the program. They can also be used to display information to the user, in the form of a label or graphic.

Adding and editing a button widget

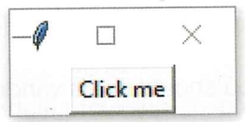
You can try out the basic process of creating and configuring a Tkinter widget in a window by typing commands in the command interface. This way, you can see the effect of each command as soon as you press **Enter**.

```
>>> from tkinter import *
>>> window = Tk() #creates the top level window called window
>>> button = Button(window, text = "Click me")
```

This displays a blank window. The last statement creates a button named `button`, but it is not yet visible because Tkinter doesn't know where to put it. To insert the button into the window, use the `pack()` method.

```
>>> button.pack()
```

As soon as this statement is executed, the button shows up:



You can move or resize the window using the mouse.

Of course, nothing happens when the button is clicked because we have not specified any action.

All widgets have specific properties associated with them, and a button has a text property which can be used to display the text on a button:

```
>>> button["text"]
'Click me'
```

You can change the text on the button by specifying the new text:

```
>>> button["text"] = "Click here"
```

You will see the window change:

Adding a label widget

The following statements will add a **label widget** to the window:

```
>>> instruction = Label(window, text = "Enter your name")
>>> instruction.pack()
```

Note that if you are never going to refer to the label by its variable name `instruction`, you could in this instance combine these two statements into one, and write

```
>>> Label(window, text = "Enter your name").pack()
```

Example 2 on the next page shows how you can also change the text on a label using the `config` method.

Placing widgets in a window

When you create a widget within a window, Tkinter needs to know where to place it. It is the job of the Geometry Manager to control the placement of objects, and there are three different Geometry Managers available.

The Pack Geometry Manager

The **Pack Geometry Manager** is well suited for windows with just a few widgets which can be placed side by side or one under the other. The label `instruction` above was placed with the Pack Geometry Manager.

The Grid Geometry Manager

The **Grid Geometry Manager** divides the window into a grid of rows and columns, and references each cell in the grid using its grid coordinates.

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

In addition, you can specify whether the widget should be placed on the left (West), right (East), top (North) or bottom (South) of each cell. You can try this in Exercise 2b at the end of the chapter.

The Place Geometry Manager

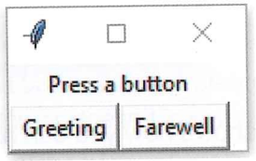
The **Place Geometry Manager** allows the program to state exactly where each widget should go, using x and y coordinates, with the top left of the window having coordinates (0, 0).

Responding to user input

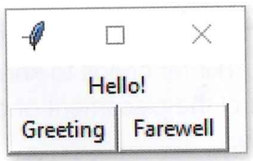
There is not much point in having a button which doesn't do anything when clicked. The example below shows how to specify what action should be taken when a button is clicked.

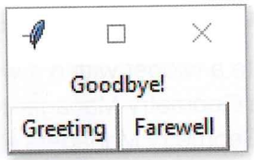
Example 2

The sample program opens a window with a label and two buttons:



When a button is pressed, the text on the label changes:






```
#Program name: Ch 12 Example 2 hello goodbye.py
from tkinter import *
window = Tk()

def greeting():
   label.config (text = "Hello!")
def farewell():
   label.config (text = "Goodbye!")

label = Label(window, text = "Press a button")
#columnspan = 2 means that the label will span two columns
label.grid(row = 0, column = 0, columnspan = 2)
button1 = Button(window,text="Greeting",width=7,command=greeting)
button1.grid(row = 1, column = 0)
button2 = Button(window,text="Farewell",width=7,command=farewell)
button2.grid(row = 1, column = 1)

#run mainloop
window.mainloop()
```

Notes:

- In the definition of `button1`, the parameter `command = greeting` tells the program to execute the function `greeting()` when the button is clicked.
- In this function, the `label` text is changed using the method `config()` and specifying the new text to be displayed.
- When `button2` is clicked, the function `farewell()` will be executed.
- `window.mainloop()` causes the program to run continuously until the window is closed.

Setting window parameters

You can set the size and colour of the window and specify whether it should be resizable. You can also put some "padding" around the button and label widgets, using the parameters `padx`, `pady`.

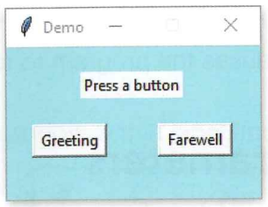
Example 3

Amend the code for Example 2, adding the extra lines as shown below:

```
#Program name: Ch 12 Example 3 hello goodbye.py
#Program places label and two buttons in a window
#Pressing a button changes the label text
#In this version the window parameters are set
from tkinter import *

def greeting():
   label.config (text = "Hello!")

def farewell():
   label.config (text = "Goodbye!")

window = Tk()
window.geometry("200x150")
window.title("Demo")

#Window is not resizable in either direction
window.resizable (False, False)
window.configure(background = "Light blue")
```

```
#Define and place labels and buttons
label = Label(window, text = "Press a button")
label.grid(row = 0,column = 0,columnspan = 2,padx = 20,pady = 20)
button1 = Button(window,text="Greeting",width=7,command=greeting)
button1.grid(row = 1, column = 0, padx = 20)
button2 = Button(window,text="Farewell",width=7,command=farewell)
button2.grid(row = 1, column = 1, padx = 20)

#run mainloop
window.mainloop()
```

The window will now appear like this:



Exercises

1. Write a program to place a button saying "Click here" in a window. When the button is clicked, a message "Hi there!" appears under the button. Give the window a title "Placing a button".

2. (a) Write a program to display, in a light green window size 200 x 120, two buttons saying "Left" and "Right". Under the buttons, display a label saying "left" when the "left" button is pressed, or "right" when the "right" button is pressed. Use the grid manager and padding to place the widgets neatly spaced in the window.

   (b) Left or right justify the labels as appropriate. You will need two functions similar to the following:

```
def leftJustify():
    label.config (text = "left")
    label.grid(row = 1, column = 0, columnspan = 2,
               padx = 30, pady = 10, sticky = W)
```


Chapter 13

Developing an application using Tkinter

Objectives

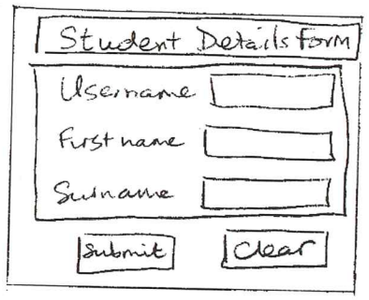
- Design and implement a data entry form for a given application
- Implement actions to be performed when a button is clicked
- Use a message window to give information to the user
- Close the Tk window and continue or end the program

Sample Application 1

This chapter describes how you might set about developing a GUI application using Tkinter. The sample application will display a data entry screen to enable a teacher or administrator to enter a user ID, first name and surname for a student.

Designing the data input window

You should start by hand drawing a rough design for your form, perhaps something like the image below.



Using frames

In the hand-drawn design above, the form has been divided into three distinct areas – the heading, the input boxes and the buttons. The diagram shows that the top two areas are each to be held in their own **frame**. The buttons will not be in a frame; they will be in the main window.

Using frames is useful because it means that moving anything in one frame will not affect the placement of anything outside the frame. You can also choose to make the background colour of the frames different from the window.

Building the form

The first thing to do is to make a list of all the widgets that are needed, their parent/child relationships, and where they will be placed in the window.

The master window is commonly given the name `root`.

The design shows just two frames, which will each be children of the master window.

The top frame contains one child, namely the form heading.

The second frame contains three text labels and three input fields (entry widgets).

The two buttons are not in a frame, so they are children of the master window.

The list of widgets to be created is as follows:

Widget name	Widget type	Parent	Description
root	Tk	(none)	Main window
frame_heading	Frame	root	Frame around the form heading
(none)	Label	frame_heading	Form heading
frame_entry	Frame	root	Frame around the text labels and input fields
(none)	Label	frame_entry	Label “Username”
(none)	Label	frame_entry	Label “First name”
(none)	Label	frame_entry	Label “Surname”
username	Entry	frame_entry	Data entry box
firstname	Entry	frame_entry	Data entry box
surname	Entry	frame_entry	Data entry box
submit_button	Button	root	Button to submit data
clear_button	Button	root	Button to clear entry boxes

In general, as you never need to change or move labels once they are placed, there is no need to name them.

When data for a student has been entered, the user will submit it by pressing a **Submit** button. This will cause the data to be printed, and the user can click a **Clear** button to clear the fields ready for the user to enter data for the next person. Once all data has been entered, the user can end the program by closing the window using the **X** icon in the top right corner.

Writing the skeleton code

The program code will follow a similar pattern for most Tkinter programs that you will write, so it is useful to start with a skeleton program which acts as the equivalent of a flowchart. Then you can fill in all the statements under each comment line.

```
#Program name: Ch 13 skeleton program.py
#program to allow entry of user ID, surname and first name
from tkinter import *

# function executed when Submit button is pressed
def submit():

# function executed when Clear button is pressed
def clear():

#create a fixed size window
root = Tk()

#place a frame to contain the form heading

#place a frame to contain labels and user entries

#place the form heading

#place the labels

#place the text entry fields

#place the Submit and Clear buttons

#run mainloop to draw the window and start the program running
root.mainloop()
```

Note: By convention, the main or root window in a Tkinter application is named `root`.

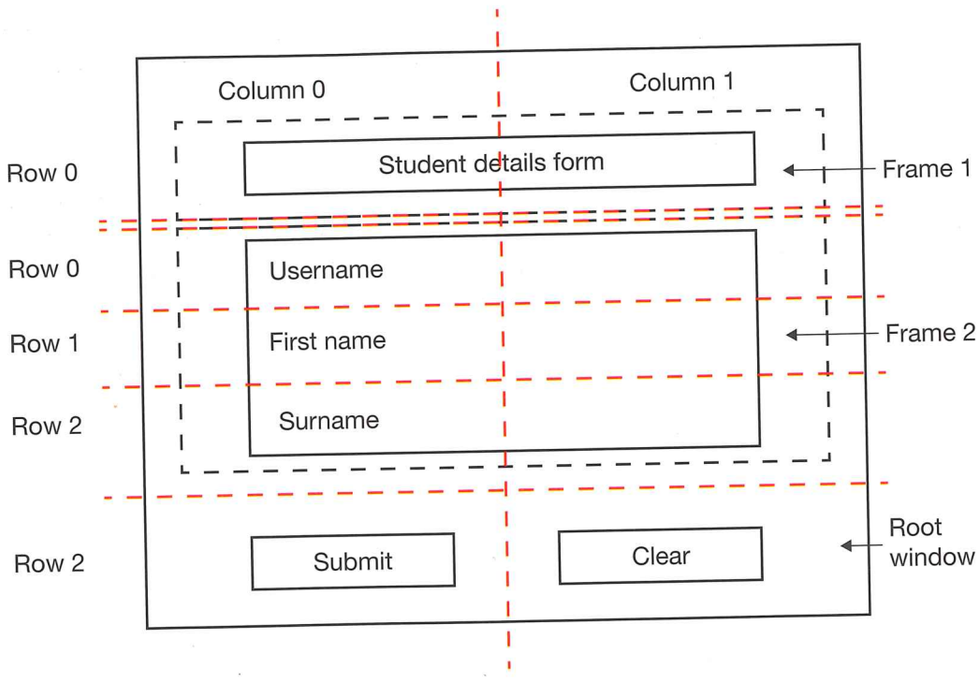
Creating the root window

The parameters of the root window will be set in the same way as in Example 3 of the previous chapter.

```
#create a fixed size window
root = Tk()
root.geometry("270x240")
root.title("Student details")
root.resizable (False, False)
root.configure(background = "Light blue")
```

Creating the frames

It's useful to draw a grid over the window design so we can easily see in which row and column within a particular frame each widget is to be placed.



Within each frame, the first row is Row 0, the second row is Row 1 and so on. The two buttons are in the main window, not in a frame, so they are in Row 2 with reference to the window, since Frame 1 is in Row 0 of the window and Frame 2 is in Row 1 of the window. Note that the rows do not have to be the same height – their height is determined by their contents.

The code to place the frames is given below:

```
#place a frame to contain the form heading
frame_heading = Frame(root)
frame_heading.grid(row=0, column=0, columnspan=2, padx=30,
                   pady=5)

#place a frame to contain labels and user entries
frame_entry = Frame(root)
frame_entry.grid(row=1, column=0, columnspan=2, padx=25,
                 pady=10)
```

Notes:

- The heading spans two columns, so the parameter `columnspan=2` is included in the statements which call the `grid()` method.
- We don't want the heading frame to be placed in the top left hand corner of the window. The parameters `padx` and `pady` cause "padding" of the specified number of characters to be included to the left of and above the frame.

Placing the form heading and the labels

The Label widgets are placed next. Notice that as there will be no need to refer to the labels, we do not need to assign them variable names. The grid statement can be included in the definition of the Label widget. (It would not matter if you did give the Label widgets their own names, but it is not necessary to do so.)

```
#place the form heading
Label(frame_heading,text="Student details form",
font=('Arial',16))\
             .grid(row=0, column=0, padx=0, pady=5)

#place the labels
Label(frame_entry, text = "Username: ")\
             .grid(row=0, column=0, padx=10, pady=5)
Label(frame_entry, text = "First name: ")\
             .grid(row=1, column=0, padx=10, pady=10)
Label(frame_entry, text ="Surname: ") \
             .grid(row=2, column=0, padx=10, pady=10)
```

Notes:

- You can specify the size and font style, as well as other optional parameters.
- The \ character indicates that the statement continues on the next line.

Placing the text entry fields

The Entry widget is used for creating a box for the user to enter text. The width is given as 15 characters in the statements below, and the background colour is specified as white.

```
#place the text entry fields
username = Entry(frame_entry, width = 15, bg = "white")
username.grid(row=0, column=1, padx=5,pady=5)

firstname = Entry(frame_entry, width = 15, bg = "white")
firstname.grid(row=1, column=1, padx=5, pady=5)

surname = Entry(frame_entry, width = 15, bg = "white")
surname.grid(row=2, column=1, padx=5, pady=5)
```

Placing the buttons

The **Submit** and **Clear** buttons are created and placed using the Button widget. The parameter `root` indicates that these are placed directly in the window named `root`. The parameter `command = submit` specifies the name of the function to be called when the **Submit** button is pressed.

```
#place the Submit and Clear buttons
submit_button = Button(root, text="Submit",width=7,
                       command=submit)
submit_button.grid(row=2, column=0, padx=0, pady=5)

clear_button = Button(root, text= "Clear", width=7,
                      command=clear)
clear_button.grid(row=2, column=1, padx=0, pady=5)
```

The Submit and Clear functions

These must be placed above the point where the functions are called, so in this example they are placed at the beginning of the program.

The `submit()` function gets the user input from the entry box and prints it. In a more realistic application, these fields could be written to a file or database for later retrieval.

The `clear()` function deletes all the characters input, from character 0 to the last character, signified by END. It then sets the focus to the first input field ready for the next username to be entered.

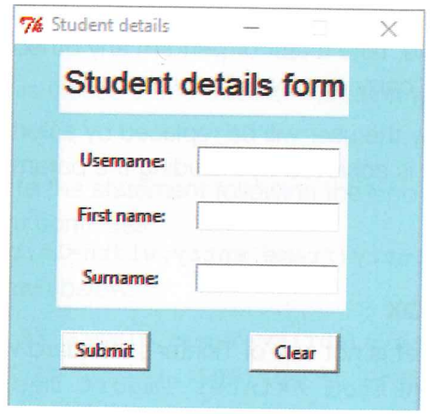

```
# functions executed when a button is pressed
def submit():
    print("Username",username.get())
    print("First name",firstname.get())
    print("Surname",surname.get())

def clear():
    username.delete(0,END)
    firstname.delete(0,END)
    surname.delete(0,END)
    username.focus_set()
```

Finally, the last statement in the program keeps the window open until you close it in the normal way with the X symbol in the top right hand corner. This will end the program.

```
#run mainloop to draw the window and start the program running
root.mainloop()
```

The final input form

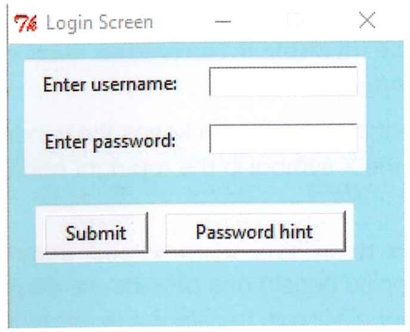
The final input form looks like this:



The complete program `Ch 13 Sample app 1 student details form.py` can be downloaded from the program folder on the website www.pgonline.co.uk.

Sample Application 2

In this application, the user will log on by typing a username and password. If the password is incorrect, an error message will be displayed, the username field and the password will be cleared and the user can press a **Password hint** button to help them get their password correct. In this example, the password is "aaaaaa". The user name is not checked. The input screen looks like this:



Once the password has been entered correctly, another message window is displayed inviting the user to continue. Once they press **OK**, the message box and the main window close and control passes back to the program, where the user could enter some data, play a game, take a test or perform any other task. In this example the program simply prints "carry on now…" and ends.

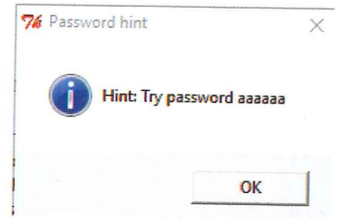
The password typed by the user will be replaced by asterisks on the screen, as a security measure. This is achieved by including the parameter `show = "*"` in the Entry widget:

```
entry_password = Entry(frame_entry,width=15,bg="white",show = "*")
```

Using a message box

The messagebox widget is not one of Tkinter's standard widgets, so you need to include the statement `from tkinter import messagebox` at the top of the program.

A message box is useful for alerting the user to an error or to give them information. In this application we will use two message boxes. The first one pops up with a message when they press a **Password hint** button if they have forgotten their password. It is common practice in many login routines to include a button to click if you have forgotten your password – normally the system will reset the password and email you a new one.



The Python code to display the message box is

```
messagebox.showinfo(title = "Password hint",
                    message = "Hint: Try password aaaaaa")
```

This generates the pop-up window, and the user must click OK to continue.

Once the user presses the **Submit** button, the program checks the password and if correct, displays a message "Password accepted" and a message box to allow the user to continue.

Note: The password and hint in this example are clearly used for testing purposes only; in a real situation, the password and the hint would be something that the user had originally supplied, such as a hint "Grandmother's birthday" to accompany a password such as GMa221155, to remind them what password they had specified.

Closing the Tkinter GUI

The window named `root` is closed in the `submit` function with the statement

```
root.destroy()
```

Control then passes back to the statement following the `root.mainloop()` statement and the program continues.

The complete listing is shown below.

```
#Program name: Ch 13 Sample app 2 validate password aaaaaa.py
#Program asks user to login, then checks password
#In this program, password is "aaaaaa"

from tkinter import *
from tkinter import messagebox

def submit():
    password = entry_password.get()
    username = entry_username.get()
    messageAlert = Label(root,width = 30)
    messageAlert.grid(row=3, column=0, columnspan=2, padx=5,
    pady=5)
```

```
    if password != "aaaaaa":
        messageAlert.config(text = "Password incorrect")
        entry_username.delete(0,"END")
        entry_password.delete(0,"END")
        entry_username.focus_set()

    else:
        messageAlert.config(text = "Password accepted")
        print("password accepted")
        print("Username: ", username)
        print("Password: ", password)
        messagebox.showinfo(title = "Password OK",
        message = "Press OK to continue")
        root.destroy()

# display a message box with a hint for password
def hint():
    messagebox.showinfo(title = "Password hint",
                        message = "Hint: Try password aaaaaa")

#create the main window
root = Tk()
root.geometry("250x180")
root.title("Login Screen")
root.resizable (False, False)
root.configure(background = "Light blue")

#place a frame round labels and user entries
frame_entry = Frame(root)
#frame_entry.pack(padx = 10, pady = 10)
frame_entry.grid(row=0, column=0, columnspan = 2,
                 padx = 10, pady = 10)

#place a frame around the buttons
frame_buttons = Frame(root)
frame_buttons.grid(row = 2, column = 0, columnspan = 3,
                   padx = 10, pady = 10)

#place the labels and text entry fields
Label(frame_entry, text = "Enter username: ").grid(row = 0,
      column = 0, padx = 5, pady = 5)
```



```
entry_username = Entry(frame_entry, width = 15, bg = "white")
entry_username.grid(row = 0, column = 1, padx = 5, pady = 5)

Label(frame_entry, text = "Enter password: ").grid(row = 1,
      column = 0, padx = 10, pady = 10)

# The parameter show = "*" will cause asterisks to appear
# instead of the characters typed by the user
entry_password = Entry(frame_entry, width=15, bg = "white",
                       show = "*")
entry_password.grid(row = 1, column = 1, padx = 5,pady = 5)

#place the submit button
submit_button = Button(frame_buttons, text = "Submit",
                       width = 8, command = submit)
submit_button.grid(row = 0, column = 0, padx = 5, pady = 5)

#place the Hint button
hint_button = Button(frame_buttons, text = "Password hint",
                     width = 15, command = hint)
hint_button.grid(row = 0, column = 1, padx = 5, pady = 5)

#run mainloop
root.mainloop()
print("carry on now...")
```