UNIT-IV

- **GUI Programming: Introduction**
- **Tkinter and Python Programming**
- **Brief Tour of Other GUIs**
- **Related Modules and Other GUIs.**

Q: Explain about the Graphical User Interface (GUI)

Graphical User Interface (GUI)

Graphical User Interface (GUI) is nothing but a desktop application which helps you to interact with the computers. They are used to perform different tasks in the desktops, laptops and other electronic devices.

- **GUI** apps like **Text-Editors** are used to create, read, update and delete different types of files.
- **GUI** apps like Chess and Solitaire are games which you can play.
- **GUI** apps like **Google Chrome, Firefox and Microsoft Edge** are used to browse through the **Internet**.

They are some different types of **GUI** apps which we daily use on the laptops or desktops.

Python Libraries To Create Graphical User Interfaces:
Python has a excess of libraries and these 4 stands out mainly when it comes to GUI. There are as follows:

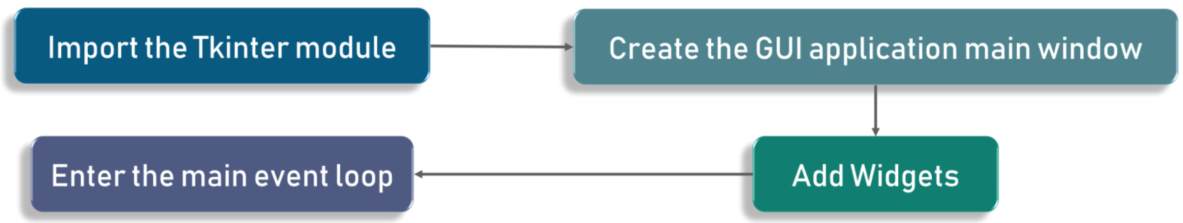
- Kivy
- Python QT
- wxPython
- Tkinter
- Jpython

Among all of these, Tkinter is preferred by learners and developers just because of how simple and easy it is.

Q: Explain about different widgets in GUI(Graphical User Interface)

Tkinter

- The primary GUI toolkit we will be using is Tk (Toolkit), Python's default GUI, We'll access Tk from its Python interface called Tkinter (i.e. **Tk interface**). It is originally designed for the **Tool Command Language** (**TCL**). Due to Tk's popularity, it has been ported to a variety of other scripting languages, including Perl (Perl/Tk), Ruby (Ruby/Tk), and Python (Tkinter).
- **Tkinter** is actually an inbuilt **Python** module used to create simple **GUI** apps. It is the most commonly used module for **GUI** apps in the **Python**.
- We no need to worry about installation of the **Tkinter** module as it comes with **Python 3.6 version** by default.
- Consider the following diagram, it shows how an application actually executes in Tkinter:


Import the Tkinter module
Create the GUI application main window
Add Widgets
Enter the main event loop

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the **Widgets** like labels, buttons, frames, etc. to the window.
4. Call the **Main event loop** so that the actions can take place on the user's computer screen.

If you noticed, there are 2 keywords here that you might not know at this point. These are the 2 keywords:

- Widgets
- Main Event Loop

An event loop is basically telling the code to keep displaying the window until we manually close it. It runs in an infinite loop in the back-end.

Example

```
from tkinter import *

#creating the application main window.
 top = Tk()

 #Entering the event main loop
top.mainloop()
```

Example

```
import tkinter
window = tkinter.Tk()
```

# to rename the title of the window window.title("aimlds")
# pack is used to show the object in the window

```
label = tkinter.Label(window, text = "Hello aimlds CSE!").pack()
window.title("GUI Window")
 window.mainloop()
```

Tkinter widgets

Widgets are something like elements in the **HTML**. You will find different types of **widgets** to the different types of elements in the **Tkinter**.

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python

		application.
2	Canvas	The canvas widget is used to draw the shapes in your GUI.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user.It is commonly used to accept user values (Input Fields)
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.
7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.
9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.
11	Radiobutton	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	Scale	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	Text	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	Toplevel	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container
18	MessageBox	This module is used to display the message-box in the desktop based applications.

These widgets are the reason that Tkinter is so popular. It makes it really easy to understand and use practically.

Python Tkinter Geometry Managers

The Tkinter geometry specifies the method by using which; the widgets are represented on display. The python Tkinter provides three geometry methods that help with positioning our widgets.

1. The pack() method
2. The grid() method
3. The place() method

Python Tkinterpack() method

The pack() widget is used to organize widget in the block, which mean it occupies the entire available width. It's a standard method to show the widgets in the window.

The syntax to use the pack() is given below.

Syntax:

widget.pack(options)

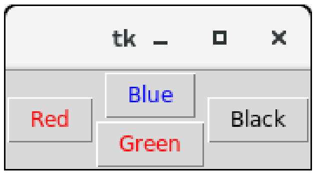
A list of possible options that can be passed in pack() is given below.

- **side:** it represents the side of the parent to which the widget is to be placed on the window.

Example

```
from tkinter import *
parent = Tk()
redbutton = Button(parent, text = "Red", fg = "red")
redbutton.pack( side = LEFT)
greenbutton = Button(parent, text = "Black", fg = "black")
greenbutton.pack( side = RIGHT )
bluebutton = Button(parent, text = "Blue", fg = "blue")
bluebutton.pack( side = TOP )
blackbutton = Button(parent, text = "Green", fg = "red")
blackbutton.pack( side = BOTTOM)
parent.mainloop()
```

Output:



Python Tkintergrid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.
This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax

widget.grid(options)

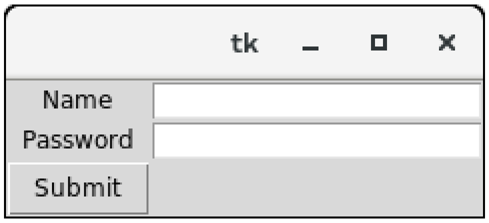
A list of possible options that can be passed inside the grid() method is given below.

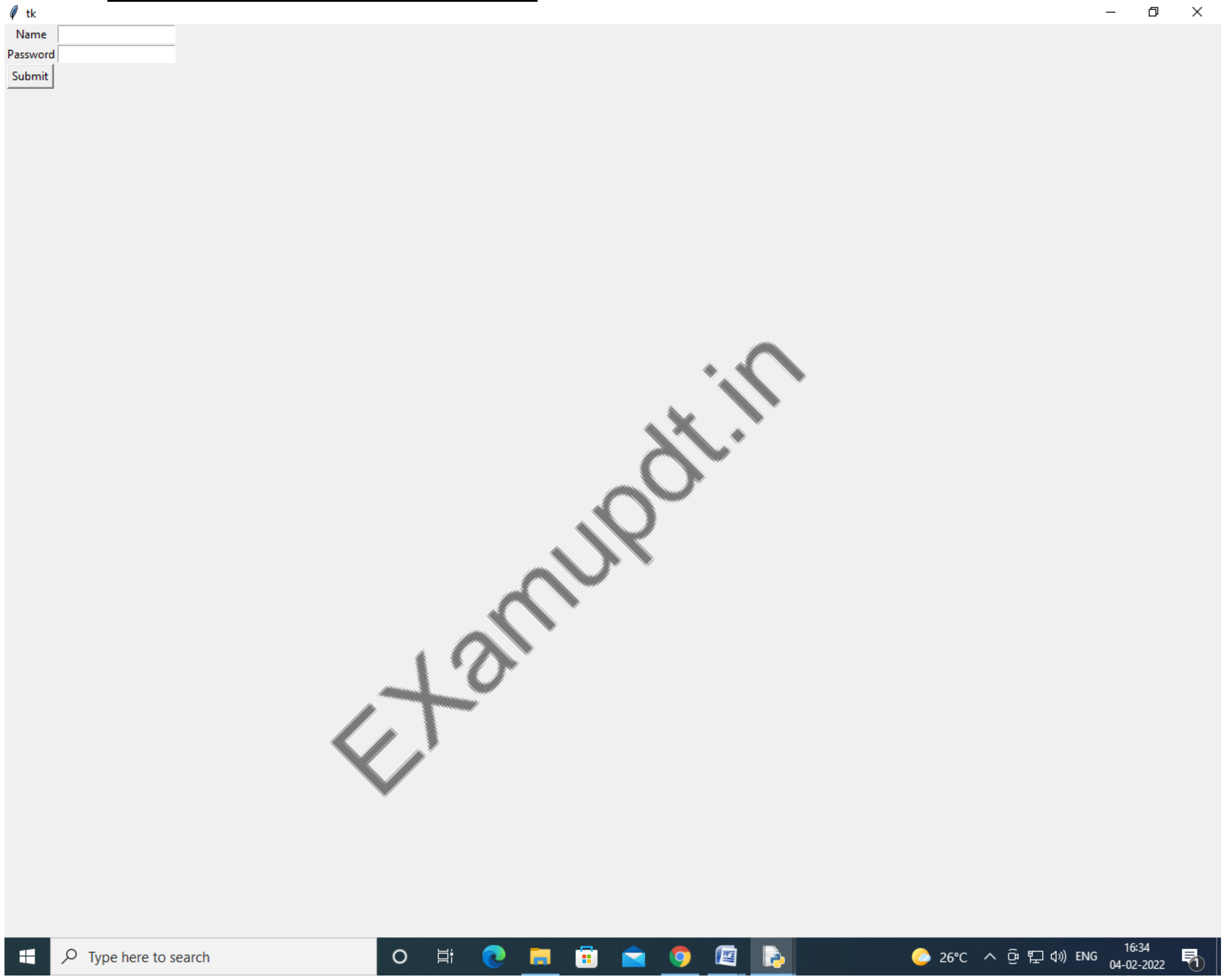
- **Column**
  The column number in which the widget is to be placed. The leftmost column is represented by 0.
- **row**
  The row number in which the widget is to be placed. The topmost row is represented by 0.

Example

```
from tkinter import *
aiml= Tk()
name = Label(aiml,text = "Name").grid(row = 0, column = 0)
e1 = Entry(aiml).grid(row = 0, column = 1)
password = Label(aiml,text = "Password").grid(row = 1, column = 0)
e2 = Entry(aiml).grid(row = 1, column = 1)
submit = Button(aiml, text = "Submit").grid(row = 4, column = 0)
aiml.mainloop()
```

Output:


tk
Name
Password
Submit


tk
Name
Password
Submit
Type here to search
26°C
ENG
16:34
04-02-2022

Python Tkinterplace() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

Syntax

widget.place(options)

A list of possible options is given below.

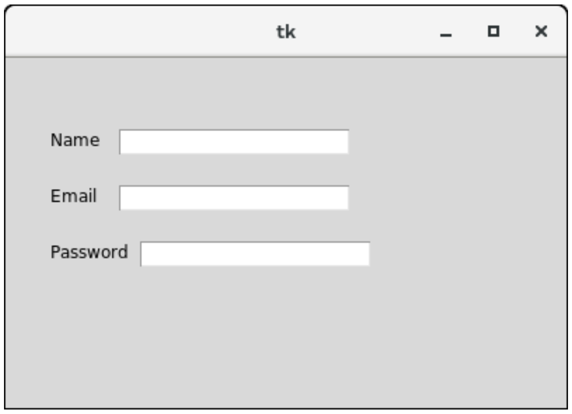
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Example

```
from tkinter import *
top = Tk()
top.geometry("400x250")
```

```
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



Q:Explian about Tkinter Button in detail

Python Tkinter Button

The button widget is used to add various types of buttons to the python application. Python allows us to configure the look of the button according to our requirements. Various options can be set or reset depending upon the requirements.

We can also associate a method or function with a button which is called when the button is pressed.

The syntax to use the button widget is given below.

Syntax

```
W = Button(parent, options)
```

A list of possible options is given below.

SN	Option	Description
1	activebackground	It represents the background of the button when the mouse hover the button.
2	activeforeground	It represents the font color of the button when the mouse hover the button.
3	Bd	It represents the border width in pixels.
4	Bg	It represents the background color of the button.

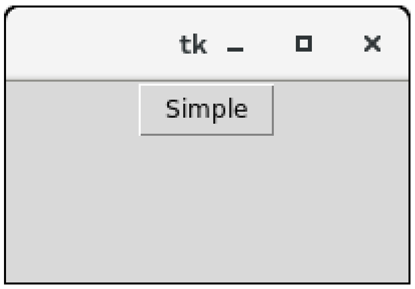
5	Command	It is set to the function call which is scheduled when the function is called.

6	Fg	Foreground color of the button.
7	Font	The font of the button text.
8	Height	The height of the button. The height is represented in the number of text lines for the textual lines or the number of pixels for the images.
10	Highlightcolor	The color of the highlight when the button has the focus.
11	Image	It is set to the image displayed on the button.
12	Justify	It illustrates the way by which the multiple text lines are represented. It is set to LEFT for left justification, RIGHT for the right justification, and CENTER for the center.
13	Padx	Additional padding to the button in the horizontal direction.
14	Pady	Additional padding to the button in the vertical direction.
15	Relief	It represents the type of the border. It can be SUNKEN, RAISED, GROOVE, and RIDGE.
17	State	This option is set to DISABLED to make the button unresponsive. The ACTIVE represents the active state of the button.
18	Underline	Set this option to make the button text underlined.
19	Width	The width of the button. It exists as a number of letters for textual buttons or pixels for image buttons.
20	Wraplength	If the value is set to a positive number, the text lines will be wrapped to fit within this length.

Example

```
#python application to create a simple button

from tkinter import *
top = Tk()
top.geometry("200x100")

b = Button(top,text = "Simple")
b.pack()
top.mainloop()
```

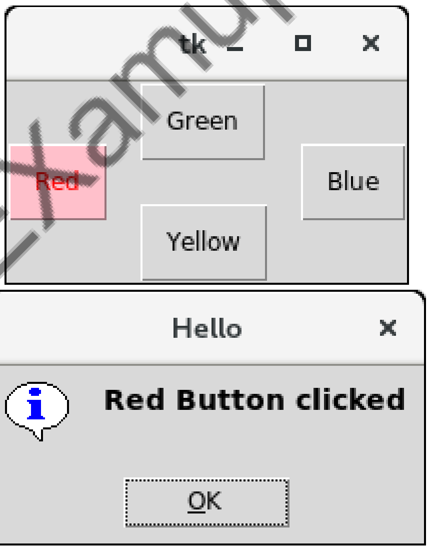
Output:



Example

```
from tkinter import *
from tkinter import messagebox

top = Tk()
top.geometry("200x100")

def fun():
   messagebox.showinfo("Hello", "Red Button clicked")

b1 = Button(top,text = "Red",command = fun,activeforeground = "red",activebackg
round = "pink",pady=10)
b2 = Button(top, text = "Blue",activeforeground = "blue",activebackground = "pink
",pady=10)
b3 = Button(top, text = "Green",activeforeground = "green",activebackground = "pi
nk",pady = 10)
b4 = Button(top, text = "Yellow",activeforeground = "yellow",activebackground = "
pink",pady = 10) b1.pack(side = LEFT)
b2.pack(side = RIGHT)
b3.pack(side = TOP)
b4.pack(side = BOTTOM)
top.mainloop()
```

Output:



Red Button clicked

Q:Explian about Tkinter *Canvas* in detail

Python Tkinter Canvas

- The canvas widget is used to add the structured graphics to the python application. It is used to draw the graph and plots to the python application. The syntax to use the canvas is given below.

Syntax

w = canvas(parent, options)

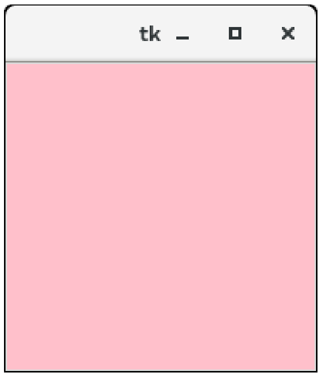
A list of possible options is given below.

SN	Option	Description
1	Bd	The represents the border width. The default width is 2.
2	Bg	It represents the background color of the canvas.
3	confine	It is set to make the canvas unscrollable outside the scroll region.
4	cursor	The cursor is used as the arrow, circle, dot, etc. on the canvas.
5	height	It represents the size of the canvas in the vertical direction.
6	highlightcolor	It represents the highlight color when the widget is focused.
7	relief	It represents the type of the border. The possible values are SUNKEN, RAISED, GROOVE, and RIDGE.
8	scrollregion	It represents the coordinates specified as the tuple containing the area of the canvas.
9	width	It represents the width of the canvas.
10	xscrollincrement	If it is set to a positive value. The canvas is placed only to the multiple of this value.
11	xscrollcommand	If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar.
12	yscrollincrement	Works like xscrollincrement, but governs vertical movement.
13	yscrollcommand	If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar.

Example

```
from tkinter import *
top = Tk()
top.geometry("200x200")
#creating a simple canvas
c = Canvas(top,bg = "pink",height = "200")
c.pack()
top.mainloop()
```

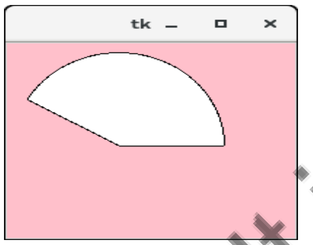
Output:



Example: Creating an arc

```
from tkinter import *

top = Tk()
top.geometry("200x200")

#creating a simple canvas
c = Canvas(top,bg = "pink",height = "200",width = 200)
arc = c.create_arc((5,10,150,200),start = 0,extent = 150, fill= "white")
c.pack()
top.mainloop()
```

Output:



Q:Explian about Tkinter *Canvas* in detail

Python TkinterCheckbutton

The Checkbutton is used to track the user's choices provided to the application. In other words, we can say that Checkbutton is used to implement the on/off selections.

The Checkbutton can contain the text or images. The Checkbutton is mostly used to provide many choices to the user among which, the user needs to choose the one. It generally implements many of many selections.

The syntax to use the checkbutton is given below.

Syntax

w = checkbutton(master, options)

A list of possible options is given below.

SN	Option	Description
1	activebackground	It represents the background color when the checkbutton is under the cursor.
2	activeforeground	It represents the foreground color of the checkbutton when the checkbutton is under the cursor.
3	Bg	The background color of the button.
4	Bitmap	It displays an image (monochrome) on the button.
5	Bd	The size of the border around the corner.
6	Command	It is associated with a function to be called when the state of the

		checkbutton is changed.
7	Cursor	The mouse pointer will be changed to the cursor name when it is over the checkbutton.
8	disableforeground	It is the color which is used to represent the text of a disabled checkbutton.
9	font	It represents the font of the checkbutton.
10	fg	The foreground color (text color) of the checkbutton.
11	height	It represents the height of the checkbutton (number of lines). The default height is 1.
12	highlightcolor	The color of the focus highlight when the checkbutton is under focus.
13	image	The image used to represent the checkbutton.
14	justify	This specifies the justification of the text if the text contains multiple lines.
15	offvalue	The associated control variable is set to 0 by default if the button is unchecked. We can change the state of an unchecked variable to some other one.
16	onvalue	The associated control variable is set to 1 by default if the button is checked. We can change the state of the checked variable to some other one.
17	padx	The horizontal padding of the checkbutton
18	pady	The vertical padding of the checkbutton.
19	relief	The type of the border of the checkbutton. By default, it is set to FLAT.
20	selectcolor	The color of the checkbutton when it is set. By default, it is red.
21	selectimage	The image is shown on the checkbutton when it is set.
22	state	It represents the state of the checkbutton. By default, it is set to normal. We can change it to DISABLED to make the checkbutton unresponsive. The state of the checkbutton is ACTIVE when it is under focus.
24	underline	It represents the index of the character in the text which is to be underlined. The indexing starts with zero in the text.
25	variable	It represents the associated variable that tracks the state of the checkbutton.
26	width	It represents the width of the checkbutton. It is represented in the number of characters that are represented in the form of texts.

27	wraplength	If this option is set to an integer number, the text will be broken intothe number of pieces.

Methods

The methods that can be called with the Checkbuttons are described in the following table.

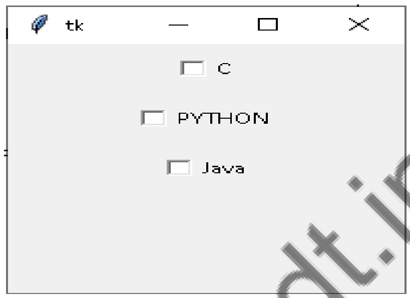
SN	Method	Description
1	deselect()	It is called to turn off the checkbutton.
2	flash()	The checkbutton is flashed between the active and normal colors.
3	invoke()	This will invoke the method associated with the checkbutton.
4	select()	It is called to turn on the checkbutton.
5	toggle()	It is used to toggle between the different Checkbuttons.

Example

```
from tkinter import

*top = Tk()

top.geometry("200x200")

chkbtn1 = Checkbutton(top, text = "aiml", )

chkbtn2 = Checkbutton(top, text = "cs", )

chkbtn3 = Checkbutton(top, text = "se", )

chkbtn4 = Checkbutton(top, text = "iot,cy",

)

chkbtn5 = Checkbutton(top, text = "ds", )

chkbtn1.pack()

chkbtn2.pack()

chkbtn3.pack

()
```

```
chkbtn4.pack

()

chkbtn3.pack

()

top.mainloop(

)
```

<u>Output:</u>



Q:Explian about Tkinter Entry in detail

Python Tkinter Entry

The Entry widget is used to provide the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user. It can only be used for one line of text from the user. For multiple lines of text, we must use the text widget.

The syntax to use the Entry widget is given below.

Syntax

```
w = Entry (parent, options)
```

A list of possible options is given below.

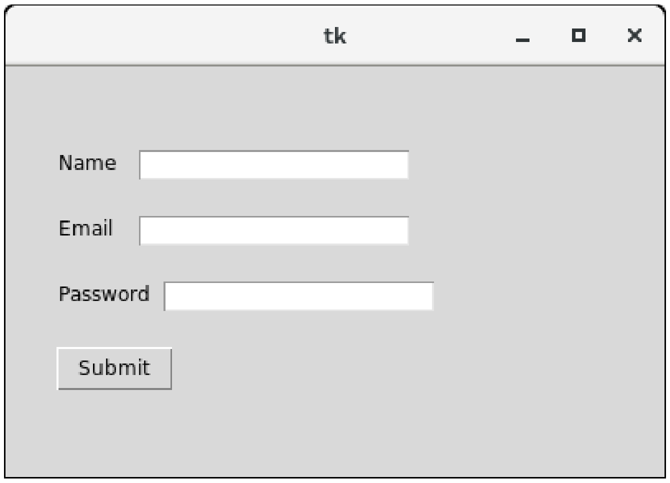
SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	The border width of the widget in pixels.
3	Cursor	The mouse pointer will be changed to the cursor type set to the arrow, dot, etc.
4	Exportselection	The text written inside the entry box will be automatically copied to the clipboard by default. We can set the exportselection to 0 to not copy this.
5	Fg	It represents the color of the text.
6	Font	It represents the font type of the text.
7	Highlightbackground	It represents the color to display in the traversal highlight region when the widget does not have the input focus.
8	Highlightcolor	It represents the color to use for the traversal highlight rectangle that is drawn around the widget when it has the input focus.
9	Highlightthickness	It represents a non-negative value indicating the width of the highlight rectangle to draw around the outside of the widget when it has the input focus.
10	Insertbackground	It represents the color to use as background in the area covered by the insertion cursor. This color will normally override either the normal background for the widget.
11	Insertborderwidth	It represents a non-negative value indicating the width of the 3-D border to draw around the insertion cursor. The value may have any of the forms acceptable to Tk_GetPixels.
12	Insertofftime	It represents a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "off" in each blink cycle. If this option is zero, then the cursor doesn't blink: it is on all the time.

13	Insertontime	Specifies a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "on" in each blink cycle.
14	Insertwidth	It represents the value indicating the total width of the insertion cursor. The value may have any of the forms acceptable to Tk_GetPixels.
15	Justify	It specifies how the text is organized if the text contains multiple lines.
16	Relief	It specifies the type of the border. Its default value is FLAT.
17	Selectbackground	The background color of the selected text.
18	Selectborderwidth	The width of the border to display around the selected task.
19	Selectforeground	The font color of the selected task.
20	Show	It is used to show the entry text of some other type instead of the string. For example, the password is typed using stars (*).
21	Textvariable	It is set to the instance of the StringVar to retrieve the text from the entry.
22	Width	The width of the displayed text or image.
23	Xscrollcommand	The entry widget can be linked to the horizontal scrollbar if we want the user to enter more text then the actual width of the widget.

Example

```
from tkinter import *
top = Tk()
top.geometry("400x250")
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
submitbtn = Button(top, text = "Submit",activebackground = "pink",activeforeground = "blue").place(x
= 30, y = 170)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



Entry widget methods

Python provides various methods to configure the data written inside the widget. There are the following methods provided by the Entry widget.

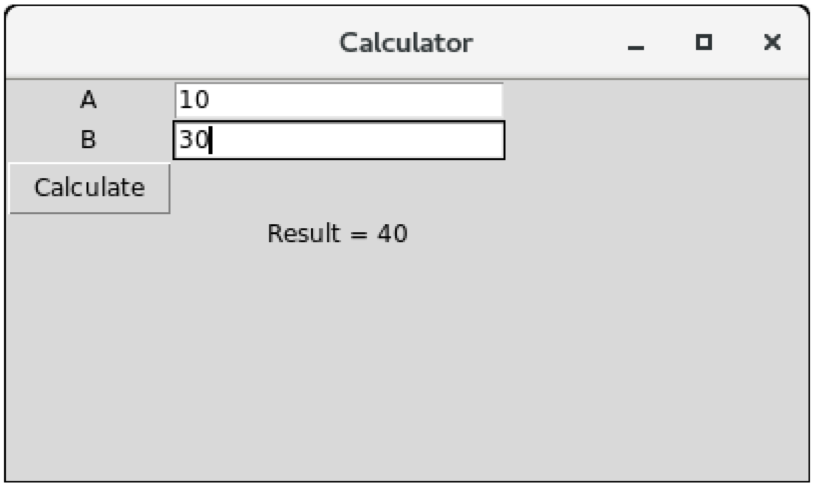
SN	Method	Description
1	delete(first, last = none)	It is used to delete the specified characters inside the widget.
2	get()	It is used to get the text written inside the widget.
3	icursor(index)	It is used to change the insertion cursor position. We can specify the index of the character before which, the cursor to be placed.
4	index(index)	It is used to place the cursor to the left of the character written at the specified index.
5	insert(index,s)	It is used to insert the specified string before the character placed at the specified index.
6	select_adjust(index)	It includes the selection of the character present at the specified index.
7	select_clear()	It clears the selection if some selection has been done.
8	select_form(index)	It sets the anchor index position to the character specified by the index.
9	select_present()	It returns true if some text in the Entry is selected otherwise returns false.
10	select_range(start,end)	It selects the characters to exist between the specified range.
11	select_to(index)	It selects all the characters from the beginning to the specified index.
12	xview(index)	It is used to link the entry widget to a horizontal scrollbar.

13	xview_scroll(number,what)	It is used to make the entry scrollable horizontally.

Example: A simple calculator

```
import tkinter as tk
from functools import partial

def call_result(label_result, n1, n2):
   num1 = (n1.get())
   num2 = (n2.get())
   result = int(num1)+int(num2)
   label_result.config(text="Result = %d" % result)
   return

root = tk.Tk()
root.geometry('400x200+100+200')

root.title('Calculator')

number1 = tk.StringVar()
number2 = tk.StringVar()

labelNum1 = tk.Label(root, text="A").grid(row=1, column=0)
labelNum2 = tk.Label(root, text="B").grid(row=2, column=0)

labelResult = tk.Label(root)

labelResult.grid(row=7, column=2)
entryNum1 = tk.Entry(root, textvariable=number1).grid(row=1, column=2)
entryNum2 = tk.Entry(root, textvariable=number2).grid(row=2, column=2)

call_result = partial(call_result, labelResult, number1, number2)

buttonCal = tk.Button(root, text="Calculate", command=call_result).grid(row=3,column=0)
root.mainloop()
```

Output:



```
A    10
B    30
Result = 40
```

Python Tkinter Frame

Python Tkinter Frame widget is used to organize the group of widgets. It acts like a container which can be used to hold the other widgets. The rectangular areas of the screen are used to organize the widgets to the python application.

The syntax to use the Frame widget is given below.

Syntax

```
w = Frame(parent, options)
```

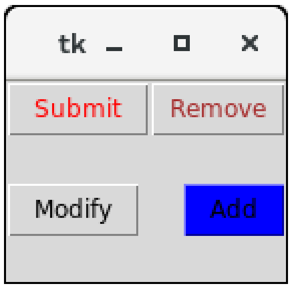
A list of possible options is given below.

SN	Option	Description
1	Bd	It represents the border width.
2	Bg	The background color of the widget.
3	Cursor	The mouse pointer is changed to the cursor type set to different values like an arrow, dot, etc.
4	Height	The height of the frame.
5	highlightbackground	The color of the background color when it is under focus.
6	Highlightcolor	The text color when the widget is under focus.
7	highlightthickness	It specifies the thickness around the border when the widget is under the focus.
8	Relief	It specifies the type of the border. The default value if FLAT.
9	Width	It represents the width of the widget.

Example

```
from tkinter import *

top = Tk()
top.geometry("140x100")
frame = Frame(top)
frame.pack()

leftframe = Frame(top)
leftframe.pack(side = LEFT)

rightframe = Frame(top)
rightframe.pack(side = RIGHT)

btn1 = Button(frame, text="Submit", fg="red",activebackground = "red")
btn1.pack(side = LEFT)

btn2 = Button(frame, text="Remove", fg="brown", activebackground = "brown")
btn2.pack(side = RIGHT)

btn3 = Button(rightframe, text="Add", fg="blue", activebackground = "blue")
btn3.pack(side = LEFT)

btn4 = Button(leftframe, text="Modify", fg="black", activebackground = "white")
btn4.pack(side = RIGHT)

top.mainloop()
```

Output:



Python Tkinter Label

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

There are the various options which can be specified to configure the text or the part of the text shown in the Label.

The syntax to use the Label is given below.

Syntax

w = Label (master, options)

A list of possible options is given below.

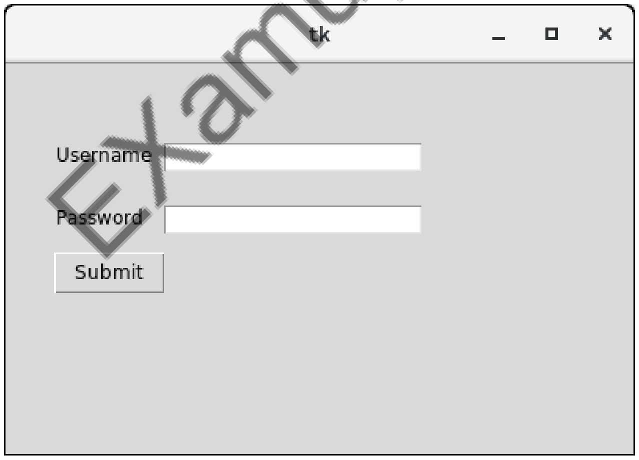
SN	Option	Description
1	Anchor	It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
2	Bg	The background color displayed behind the widget.
3	Bitmap	It is used to set the bitmap to the graphical object specified so that, thelabel can represent the graphics instead of text.
4	Bd	It represents the width of the border. The default is 2 pixels.
5	Cursor	The mouse pointer will be changed to the type of the cursor specified, i.e.,arrow, dot, etc.
6	Font	The font type of the text written inside the widget.
7	Fg	The foreground color of the text written inside the widget.
8	Height	The height of the widget.
9	Image	The image that is to be shown as the label.
10	Justify	It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for rightjustification, and CENTER for center justification.
11	Padx	The horizontal padding of the text. The default value is 1.
12	Pady	The vertical padding of the text. The default value is 1.
13	Relief	The type of the border. The default value is FLAT.
14	Text	This is set to the string variable which may contain one or more line oftext.
15	textvariable	The text written inside the widget is set to the control variable StringVar sothat it can be accessed and changed accordingly.
16	underline	We can display a line under the specified letter of the text. Set this optionto the number of the letter under which the line will be displayed.
17	Width	The width of the widget. It is specified as the number of characters.
18	wraplength	Instead of having only one line as the label text, we can break it to thenumber of lines where each line has the number of characters specified to

		this option.

Example 1

```
from tkinter import *

top = Tk()

top.geometry("400x250")

#creating label
uname = Label(top, text = "Username").place(x = 30,y = 50)

#creating label
password = Label(top, text = "Password").place(x = 30, y = 90)

sbmitbtn = Button(top, text = "Submit",activebackground = "pink", activeforegroun
d = "blue").place(x = 30, y = 120)

e1 = Entry(top,width = 20).place(x = 100, y = 50)
e2 = Entry(top, width = 20).place(x = 100, y = 90)
top.mainloop()
```

Output:



Python TkinterListbox

The Listbox widget is used to display the list items to the user. We can place only textitems in the Listbox and all text items contain the same font and color.
The user can choose one or more items from the list depending upon the configuration.

The syntax to use the Listbox is given below.
w = Listbox(parent, options)

A list of possible options is given below.

SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	It represents the size of the border. Default value is 2 pixel.
3	Cursor	The mouse pointer will look like the cursor type like dot, arrow, etc.
4	Font	The font type of the Listbox items.
5	Fg	The color of the text.
6	Height	It represents the count of the lines shown in the Listbox. The defaultvalue is 10.
7	highlightcolor	The color of the Listbox items when the widget is under focus.
8	highlightthickness	The thickness of the highlight.
9	Relief	The type of the border. The default is SUNKEN.
10	selectbackground	The background color that is used to display the selected text.
11	selectmode	It is used to determine the number of items that can be selected from the list. It can set to BROWSE, SINGLE, MULTIPLE, EXTENDED.
12	Width	It represents the width of the widget in characters.
13	xscrollcommand	It is used to let the user scroll the Listbox horizontally.
14	yscrollcommand	It is used to let the user scroll the Listbox vertically.

Methods

There are the following methods associated with the Listbox.

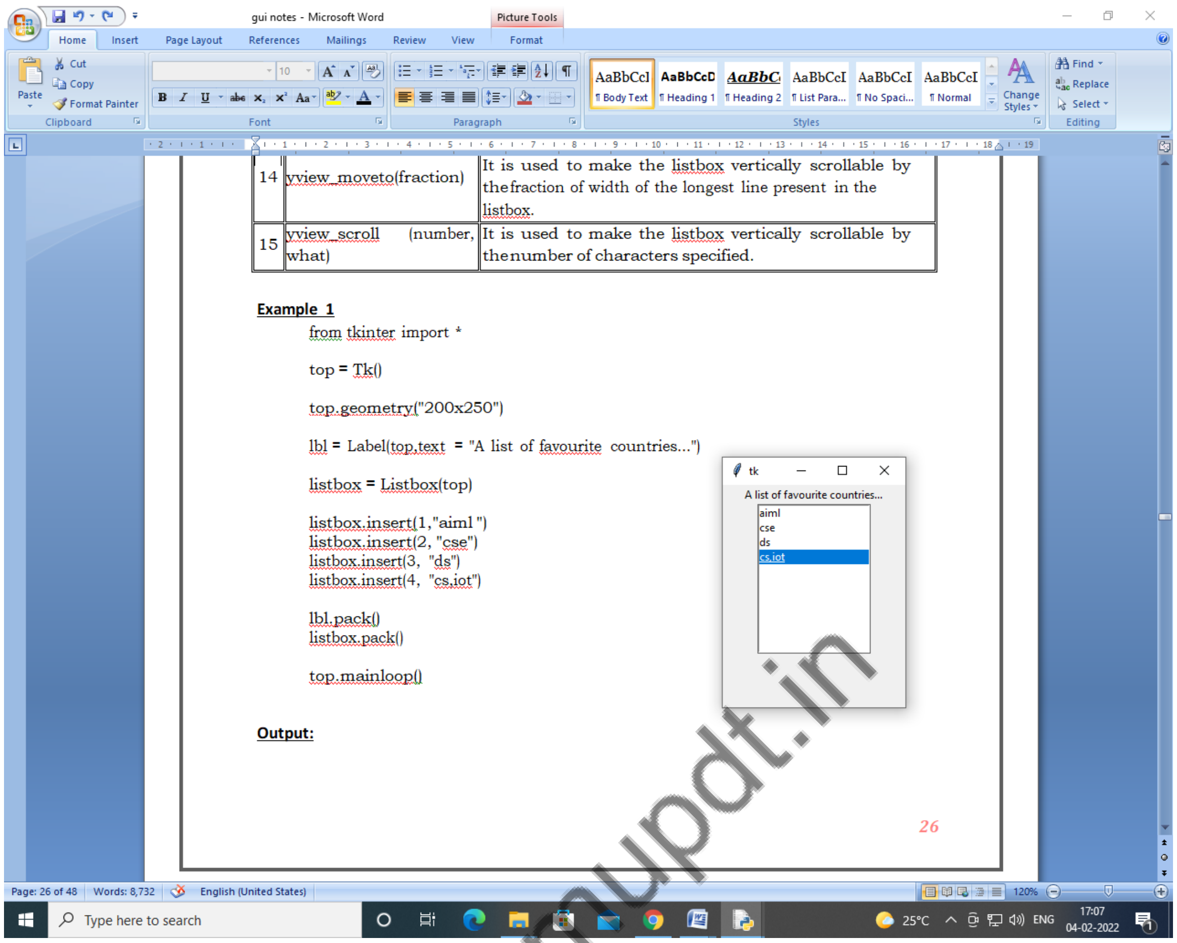
SN	Method	Description
1	activate(index)	It is used to select the lines at the specified index.
2	curselection()	It returns a tuple containing the line numbers of the selected element or elements, counting from 0. If nothing is selected, returns an empty tuple.
3	delete(first, last = None)	It is used to delete the lines which exist in the given range.
4	get(first, last = None)	It is used to get the list items that exist in the given range.
5	index(i)	It is used to place the line with the specified index at the topof the widget.
6	insert(index, *elements)	It is used to insert the new lines with the specified number ofelements before the specified index.
7	nearest(y)	It returns the index of the nearest line to the y coordinate of

		the Listbox widget.
8	see(index)	It is used to adjust the position of the listbox to make thelines specified by the index visible.
9	size()	It returns the number of lines that are present in the Listboxwidget.
10	xview()	This is used to make the widget horizontally scrollable.
11	xview_moveto(fraction)	It is used to make the listbox horizontally scrollable by thefraction of width of the longest line present in the listbox.
12	xview_scroll(number, what)	It is used to make the listbox horizontally scrollable by thenumber of characters specified.
13	yview()	It allows the Listbox to be vertically scrollable.
14	yview_moveto(fraction)	It is used to make the listbox vertically scrollable by thefraction of width of the longest line present in the listbox.
15	yview_scroll (number, what)	It is used to make the listbox vertically scrollable by thenumber of characters specified.

Example 1

```
from tkinter import *

top = Tk()

top.geometry("200x250")

lbl = Label(top,text = "A list of favourite countries...")

listbox = Listbox(top)

listbox.insert(1,"aiml ")
listbox.insert(2, "cse")
listbox.insert(3, "ds")
listbox.insert(4, "cs,iot")

lbl.pack()
listbox.pack()

top.mainloop()
```

Output:



14	yview_moveto(fraction)	It is used to make the listbox vertically scrollable by the fraction of width of the longest line present in the listbox.
15	yview_scroll (number, what)	It is used to make the listbox vertically scrollable by the number of characters specified.

Example 1

```
from tkinter import *

top = Tk()

top.geometry("200x250")

lbl = Label(top,text = "A list of favourite countries...")

listbox = Listbox(top)

listbox.insert(1,"aiml ")
listbox.insert(2, "cse")
listbox.insert(3, "ds")
listbox.insert(4, "cs,iot")

lbl.pack()
listbox.pack()

top.mainloop()
```

Output:

```
A list of favourite countries...
aiml
cse
ds
cs,iot
```

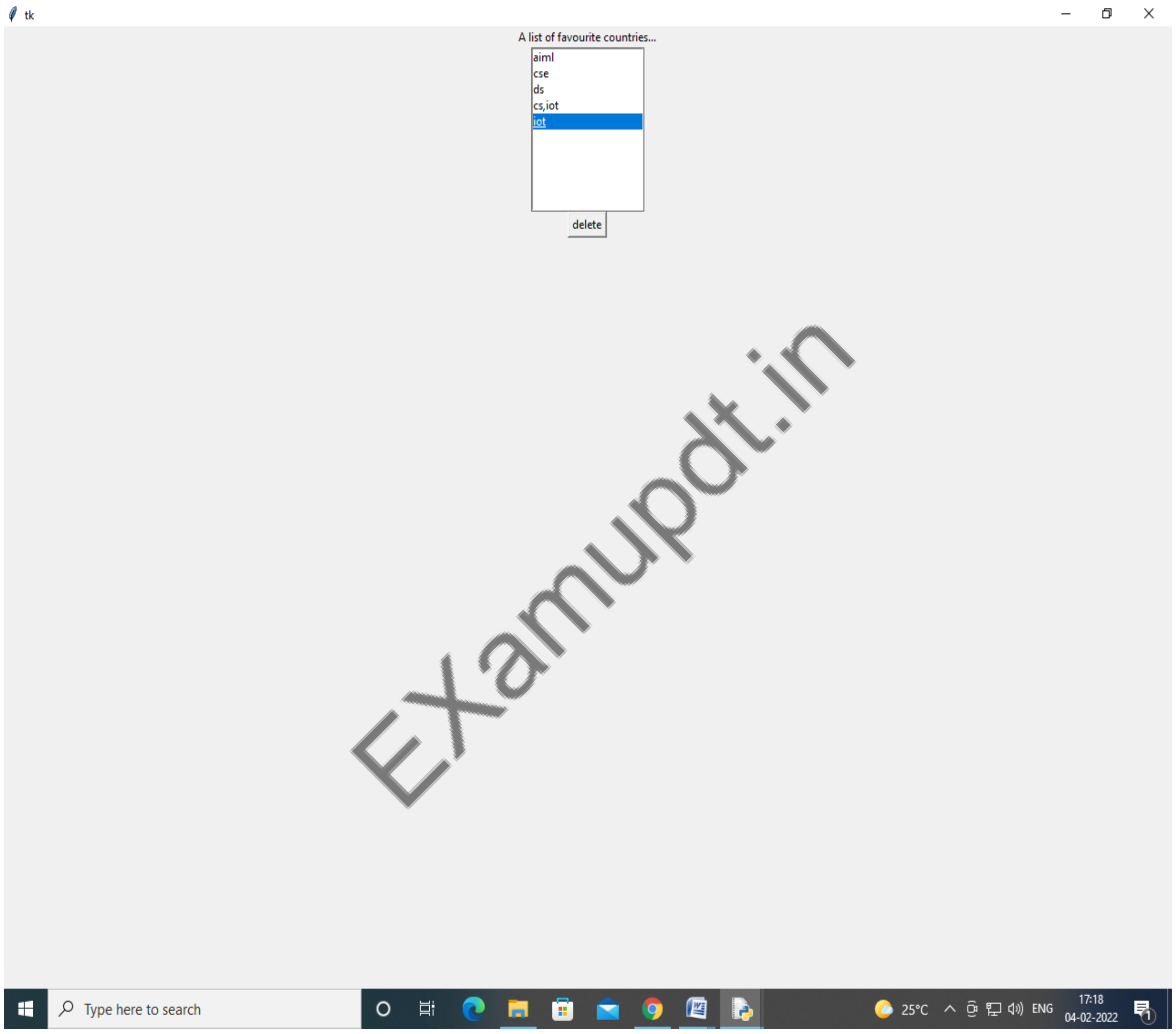
26

Example 2: Deleting the active items from the list

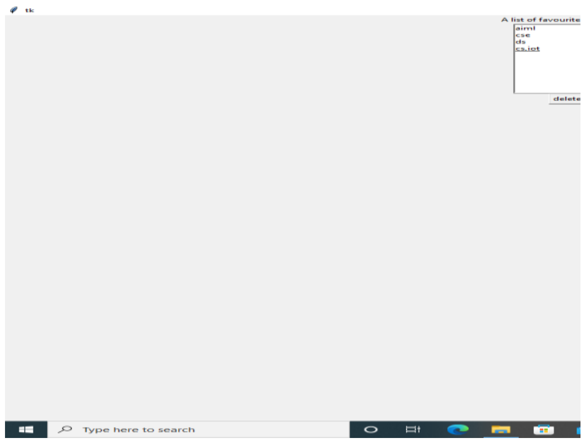
```
from tkinter import *

top = Tk()
top.geometry("200x250")

lbl = Label(top,text = "A list of favourite countries...")
listbox = Listbox(top)
listbox.insert(1,"aiml ")
listbox.insert(2, "cse")
listbox.insert(3, "ds")
listbox.insert(4, "cs,iot")
listbox.insert(5, "iot")


#this button will delete the selected item from the list

btn = Button(top, text = "delete", command = lambda listbox=listbox: listbox.delete
(ANCHOR))
```

```
lbl.pack()
listbox.pack()
btn.pack()
top.mainloop()
```

Output:


```
A list of favourite countries...
aiml
cse
ds
cs,iot
iot
```

After pressing the delete button.



Q:Explian about Tkinter Menu in detail

Python Tkinter Menu

The Menu widget is used to create various types of menus (top level, pull down, and pop up) in the python application.

The top-level menus are the one which is displayed just under the title bar of the parent window. We need to create a new instance of the Menu widget and add various commands to it by using the add() method.

The syntax to use the Menu widget is given below.

Syntax

```
w = Menu(top, options)
```

A list of possible options is given below.

SN	Option	Description
1	activebackground	The background color of the widget when the widget is under the focus.

2	activeborderwidth	The width of the border of the widget when it is under the mouse. The default is 1 pixel.
3	activeforeground	The font color of the widget when the widget has the focus.
4	Bg	The background color of the widget.
5	Bd	The border width of the widget.
6	cursor	The mouse pointer is changed to the cursor type when it hovers the widget. The cursor type can be set to arrow or dot.
7	disabledforeground	The font color of the widget when it is disabled.
8	Font	The font type of the text of the widget.
9	Fg	The foreground color of the widget.
10	postcommand	The postcommand can be set to any of the function which is called when the mourse hovers the menu.
11	relief	The type of the border of the widget. The default type is RAISED.
12	image	It is used to display an image on the menu.
13	selectcolor	The color used to display the checkbutton or radiobutton when they are selected.
14	tearoff	By default, the choices in the menu start taking place from position 1. If we set the tearoff = 1, then it will start taking place from 0th position.
15	Title	Set this option to the title of the window if you want to change the title of the window.

Methods

The Menu widget contains the following methods.

SN	Option	Description
1	add_command(options)	It is used to add the Menu items to the menu.
2	add_radiobutton(options)	This method adds the radiobutton to the menu.
3	add_checkbutton(options)	This method is used to add the checkbuttons to the menu.
4	add_cascade(options)	It is used to create a hierarchical menu to the parent menu by associating the given menu to the parent menu.
5	add_seperator()	It is used to add the seperator line to the menu.
6	add(type, options)	It is used to add the specific menu item to the menu.
7	delete(startindex, endindex)	It is used to delete the menu items exist in the specified range.
8	entryconfig(index,	It is used to configure a menu item identified by the given

	options)	index.
9	index(item)	It is used to get the index of the specified menu item.
10	insert_seperator(index)	It is used to insert a seperator at the specified index.
11	invoke(index)	It is used to invoke the associated with the choice given at the specified index.
12	type(index)	It is used to get the type of the choice specified by the index.

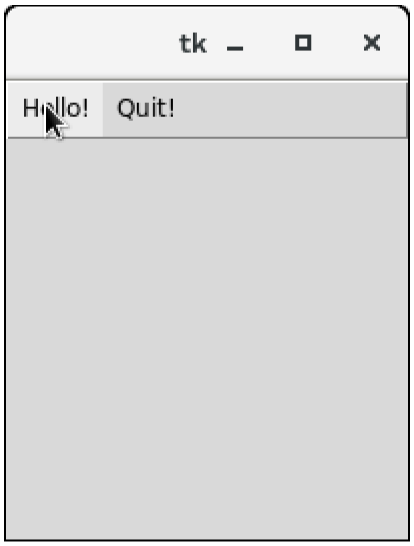
Creating a top level menu

A top-level menu can be created by instantiating the Menu widget and adding the menu items to the menu.

Example 1

```
from tkinter import *

top = Tk()

def hello():
    print("hello!")

# create a toplevel menu
menubar = Menu(top)
menubar.add_command(label="Hello!", command=hello)
menubar.add_command(label="Quit!", command=top.quit)

# display the menu
top.config(menu=menubar)

top.mainloop()
```

Output:



Clicking the hello Menubutton will print the hello on the console while clicking the Quit Menubutton will make an exit from the python application.

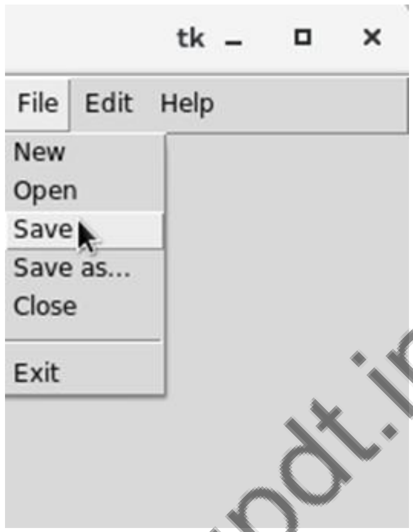
Example 2

```
from tkinter import Toplevel, Button, Tk, Menu

top = Tk()
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label="New")
file.add_command(label="Open")
file.add_command(label="Save")
file.add_command(label="Save as...")
file.add_command(label="Close")

file.add_separator()

file.add_command(label="Exit", command=top.quit)

menubar.add_cascade(label="File", menu=file)
edit = Menu(menubar, tearoff=0)
edit.add_command(label="Undo")

edit.add_separator()

edit.add_command(label="Cut")
edit.add_command(label="Copy")
edit.add_command(label="Paste")
edit.add_command(label="Delete")
edit.add_command(label="Select All")
```

```
menubar.add_cascade(label="Edit", menu=edit)
```

```
help = Menu(menubar, tearoff=0)
help.add_command(label="About")
menubar.add_cascade(label="Help", menu=help)

top.config(menu=menubar)
top.mainloop()
```

Output:



Python Tkinter Message

The Message widget is used to show the message to the user regarding the behaviour of the python application. The message widget shows the text messages to the user which can not be edited.
The message text contains more than one line. However, the message can only be shown in the single font.
The syntax to use the Message widget is given below.
Syntax

w = Message(parent, options)

A list of possible options is given below.

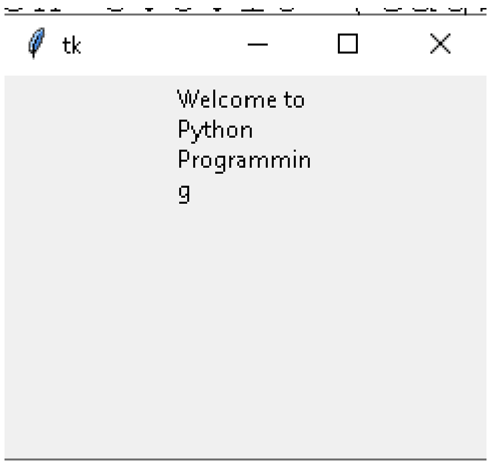
SN	Option	Description
1	Anchor	It is used to decide the exact position of the text within the space provided to the widget if the widget contains more space than the need of the text. The default is CENTER.
2	Bg	The background color of the widget.
3	Bitmap	It is used to display the graphics on the widget. It can be set to any graphical or image object.
4	Bd	It represents the size of the border in the pixel. The default size is 2 pixel.
5	Cursor	The mouse pointer is changed to the specified cursor type. The cursor

		type can be an arrow, dot, etc.
6	Font	The font type of the widget text.
7	Fg	The font color of the widget text.
8	Height	The vertical dimension of the message.
9	Image	We can set this option to a static image to show that onto the widget.
10	Justify	This option is used to specify the alignment of multiple line of code with respect to each other. The possible values can be LEFT (left alignment), CENTER (default), and RIGHT (right alignment).
11	Padx	The horizontal padding of the widget.
12	Pady	The vertical padding of the widget.
13	Relief	It represents the type of the border. The default type is FLAT.
14	Text	We can set this option to the string so that the widget can represent the specified text.
15	Textvariable	This is used to control the text represented by the widget. The textvariable can be set to the text that is shown in the widget.
16	Underline	The default value of this option is -1 that represents no underline. We can set this option to an existing number to specify that nth letter of the string will be underlined.
17	Width	It specifies the horizontal dimension of the widget in the number of characters (not pixel).
18	Wraplength	We can wrap the text to the number of lines by setting this option to the desired number so that each line contains only that number of characters.

Example

```
from tkinter import *

top = Tk()
top.geometry("100x100")

msg = Message( top, text = "Welcome to Python Programming")

msg.pack()
top.mainloop()
```

Output:



```
Welcome to
Python
Programmin
g
```

Python TkinterRadiobutton

The Radiobutton widget is used to implement one-of-many selection in the python application. It shows multiple choices to the user out of which, the user can select only one out of them. We can associate different methods with each of the radiobutton.

We can display the multiple line text or images on the radiobuttons. To keep track the user's selection the radiobutton, it is associated with a single variable. Each button displays a single value for that particular variable.

The syntax to use the Radiobutton is given below.

Syntax

```
w = Radiobutton(top, options)
```

SN	Option	Description
1	activebackground	The background color of the widget when it has the focus.
2	activeforeground	The font color of the widget text when it has the focus.
3	anchor	It represents the exact position of the text within the widget if the widget contains more space than the requirement of the text. The default value is CENTER.
4	Bg	The background color of the widget.
5	bitmap	It is used to display the graphics on the widget. It can be set to any graphical or image object.
6	borderwidth	It represents the size of the border.
7	command	This option is set to the procedure which must be called every-time when the state of the radiobutton is changed.
8	cursor	The mouse pointer is changed to the specified cursor type. It can be set to the arrow, dot, etc.
9	Font	It represents the font type of the widget text.
10	Fg	The normal foreground color of the widget text.

11	height	The vertical dimension of the widget. It is specified as the number of lines (not pixel).
12	highlightcolor	It represents the color of the focus highlight when the widget has the focus.
13	highlightbackground	The color of the focus highlight when the widget is not having the focus.
14	image	It can be set to an image object if we want to display an image on the radiobutton instead the text.
15	justify	It represents the justification of the multi-line text. It can be set to CENTER(default), LEFT, or RIGHT.
16	padx	The horizontal padding of the widget.
17	pady	The vertical padding of the widget.
18	relief	The type of the border. The default value is FLAT.
19	selectcolor	The color of the radio button when it is selected.
20	selectimage	The image to be displayed on the radiobutton when it is selected.
21	state	It represents the state of the radio button. The default state of the Radiobutton is NORMAL. However, we can set this to DISABLED to make the radiobutton unresponsive.
22	Text	The text to be displayed on the radiobutton.
23	textvariable	It is of String type that represents the text displayed by the widget.
24	underline	The default value of this option is -1, however, we can set this option to the number of character which is to be underlined.
25	value	The value of each radiobutton is assigned to the control variable when it is turned on by the user.
26	variable	It is the control variable which is used to keep track of the user's choices. It is shared among all the radiobuttons.
27	width	The horizontal dimension of the widget. It is represented as the number of characters.

28	wraplength	We can wrap the text to the number of lines by setting this option to the desired number so that each line contains only that number of characters.

Methods

The radiobutton widget provides the following methods.

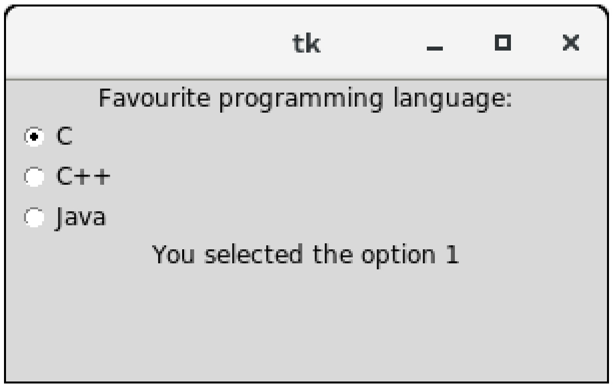
SN	Method	Description
1	deselect()	It is used to turn of the radiobutton.
2	flash()	It is used to flash the radiobutton between its active and normal colors few times.
3	invoke()	It is used to call any procedure associated when the state of a Radiobutton is changed.
4	select()	It is used to select the radiobutton.

Example

```
from tkinter import *

def selection():
  selection = "You selected the option " + str(radio.get())
  label.config(text = selection)

top = Tk()
top.geometry("300x150")
radio = IntVar()
lbl = Label(text = "Favourite programming language:")
lbl.pack()
R1 = Radiobutton(top, text="C", variable=radio, value=1,command=selection)
R1.pack( anchor = W )

R2 = Radiobutton(top, text="C++", variable=radio, value=2, command=selection)

R2.pack( anchor = W )

R3 = Radiobutton(top, text="Java", variable=radio, value=3, command=selection)

R3.pack( anchor = W)
```

```
label = Label(top)
label.pack()
top.mainloop()
```

Output:



```
Favourite programming language:
● C
○ C++
○ Java
You selected the option 1
```

Python Tkinter Scrollbar

The scrollbar widget is used to scroll down the content of the other widgets like listbox, text, and canvas. However, we can also create the horizontal scrollbars to the Entry widget.

The syntax to use the Scrollbar widget is given below.

Syntax

```
w = Scrollbar(top, options)
```

A list of possible options is given below.

SN	Option	Description
1	activebackground	The background color of the widget when it has the focus.
2	Bg	The background color of the widget.
3	Bd	The border width of the widget.
4	command	It can be set to the procedure associated with the list which can be called each time when the scrollbar is moved.
5	cursor	The mouse pointer is changed to the cursor type set to this option which can be an arrow, dot, etc.
6	elementborderwidth	It represents the border width around the arrow heads and slider. The default value is -1.
7	Highlightbackground	The focus highlighcolor when the widget doesn't have the focus.
8	highlighcolor	The focus highlighcolor when the widget has the focus.
9	highlightthickness	It represents the thickness of the focus highlight.
10	jump	It is used to control the behavior of the scroll jump. If it set to 1,

		then the callback is called when the user releases the mouse button.
11	orient	It can be set to HORIZONTAL or VERTICAL depending upon the orientation of the scrollbar.
12	repeatdelay	This option tells the duration up to which the button is to be pressed before the slider starts moving in that direction repeatedly. The default is 300 ms.
13	repeatinterval	The default value of the repeat interval is 100.
14	takefocus	We can tab the focus through this widget by default. We can set this option to 0 if we don't want this behavior.
15	troughcolor	It represents the color of the trough.
16	width	It represents the width of the scrollbar.

Methods

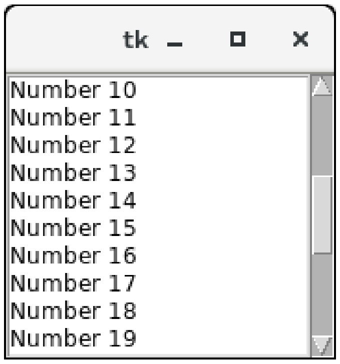
The widget provides the following methods.

SN	Method	Description
1	get()	It returns the two numbers a and b which represents the current position of the scrollbar.
2	set(first, last)	It is used to connect the scrollbar to the other widget w. The yscrollcommand or xscrollcommand of the other widget to this method.

Example

```
from tkinter import *

top = Tk()
sb = Scrollbar(top)
sb.pack(side = RIGHT, fill = Y)

mylist = Listbox(top, yscrollcommand = sb.set )

for line in range(30):
   mylist.insert(END, "Number " + str(line))

mylist.pack( side = LEFT )
sb.config( command = mylist.yview )

mainloop()
```

Output:



```
Number 10
Number 11
Number 12
Number 13
Number 14
Number 15
Number 16
Number 17
Number 18
Number 19
```

Python Tkinter Text

The Text widget is used to show the text data on the Python application. However, Tkinter provides us the Entry widget which is used to implement the single line text box.

The Text widget is used to display the multi-line formatted text with various styles and attributes. The Text widget is mostly used to provide the text editor to the user.

The Text widget also facilitates us to use the marks and tabs to locate the specific sections of the Text. We can also use the windows and images with the Text as it can also be used to display the formatted text.
The syntax to use the Text widget is given below.

Syntax

w = Text(top, options)

A list of possible options that can be used with the Text widget is given below.

SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	It represents the border width of the widget.
3	Cursor	The mouse pointer is changed to the specified cursor type, i.e. arrow, dot, etc.
4	Exportselection	The selected text is exported to the selection in the window manager. We can set this to 0 if we don't want the text to be exported.
5	Font	The font type of the text.
6	Fg	The text color of the widget.
7	Height	The vertical dimension of the widget in lines.
8	highlightbackground	The highlightcolor when the widget doesn't has the focus.
9	highlightthickness	The thickness of the focus highlight. The default value is 1.
10	Highlighcolor	The color of the focus highlight when the widget has the focus.

11	Insertbackground	It represents the color of the insertion cursor.
12	insertborderwidth	It represents the width of the border around the cursor. The default is 0.
13	Insertofftime	The time amount in Milliseconds during which the insertion cursor is off in the blink cycle.
14	Insertontime	The time amount in Milliseconds during which the insertion cursor is on in the blink cycle.
15	Insertwidth	It represents the width of the insertion cursor.
16	Padx	The horizontal padding of the widget.
17	Pady	The vertical padding of the widget.
18	Relief	The type of the border. The default is SUNKEN.
19	Selectbackground	The background color of the selected text.
20	selectborderwidth	The width of the border around the selected text.
21	spacing1	It specifies the amount of vertical space given above each line of the text. The default is 0.
22	spacing2	This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. The default is 0.
23	spacing3	It specifies the amount of vertical space to insert below each line of the text.
24	State	It the state is set to DISABLED, the widget becomes unresponsive to the mouse and keyboard unresponsive.
25	Tabs	This option controls how the tab character is used to position the text.
26	Width	It represents the width of the widget in characters.
27	Wrap	This option is used to wrap the wider lines into multiple lines. Set this option to the WORD to wrap the lines after the word that fit into the available space. The default value is CHAR which breaks the line which gets too wider at any character.
28	Xscrollcommand	To make the Text widget horizontally scrollable, we can set this option to the set() method of Scrollbar widget.
29	Yscrollcommand	To make the Text widget vertically scrollable, we can set this option to the set() method of Scrollbar widget.

Methods

We can use the following methods with the Text widget.

SN	Method	Description
1	delete(startindex, endindex)	This method is used to delete the characters of the specified range.
2	get(startindex, endindex)	It returns the characters present in the specified range.
3	index(index)	It is used to get the absolute index of the specified index.
4	insert(index, string)	It is used to insert the specified string at the given index.
5	see(index)	It returns a boolean value true or false depending upon whether the text at the specified index is visible or not.

Mark handling methods

Marks are used to bookmark the specified position between the characters of the associated text.

SN	Method	Description
1	index(mark)	It is used to get the index of the specified mark.
2	mark_gravity(mark, gravity)	It is used to get the gravity of the given mark.
3	mark_names()	It is used to get all the marks present in the Text widget.
4	mark_set(mark, index)	It is used to inform a new position of the given mark.
5	mark_unset(mark)	It is used to remove the given mark from the text.

Tag handling methods

The tags are the names given to the separate areas of the text. The tags are used to configure the different areas of the text separately. The list of tag-handling methods along with the description is given below.

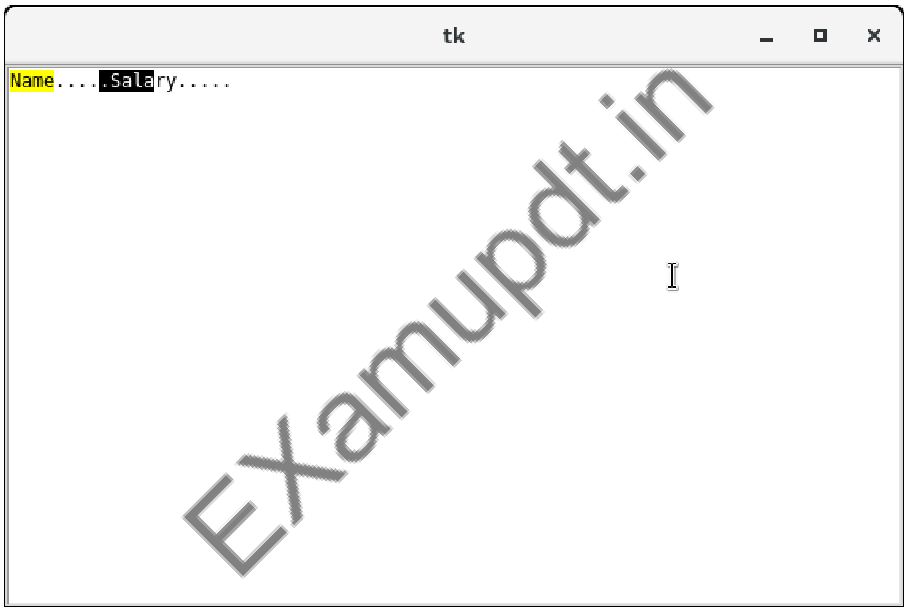
SN	Method	Description
1	tag_add(tagname, startindex, endindex)	This method is used to tag the string present in the specified range.
2	tag_config	This method is used to configure the tag properties.
3	tag_delete(tagname)	This method is used to delete a given tag.
4	tag_remove(tagname, startindex, endindex)	This method is used to remove a tag from the specified range.

Example

```
        from tkinter import *
```

```
top = Tk()
text = Text(top)
text.insert(INSERT, "Name. .. ")
text.insert(END, "Salary.... ")

text.pack()

text.tag_add("Write Here", "1.0", "1.4")
text.tag_add("Click Here", "1.8", "1.13")

text.tag_config("Write Here", background="yellow", foreground="black")
text.tag_config("Click Here", background="black", foreground="white")

top.mainloop()
```

Output:



```
Name.....Salary.....
```

Other GUIs

General GUI development using many of the abundant number of graphical toolkits that exist under Python, but alas, that is for the future. As a proxy, we would like to present a single simple GUI application written using four of the more popular and available toolkits out there: Tix (Tk Interface eXtensions), Pmw (Python MegaWidgetsTkinter extension), wxPython (Python binding to wxWidgets), and PyGTK (Python binding to GTK+).

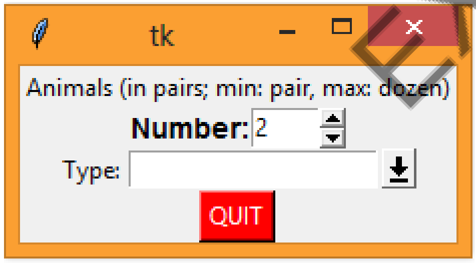
Tix, the Tk Interface eXtension, is a powerful set of user interface components that expands the capabilities of your Tcl/Tk and Python applications. Using Tix together with Tk will greatly enhance the appearance and functionality of your application.

It uses the Tix module. Tix comes with Python!

<u>Example:</u>

```
from tkinter import *
  from tkinter.tix import Control, ComboBox

  top = Tk()
  top.tk.eval('package require Tix')

  lb = Label(top,text='Animals (in pairs; min: pair, max: dozen)')
  lb.pack()

  ct = Control(top, label='Number:',integer=True, max=12, min=2, value=2, step=2)
  ct.label.config(font='Helvetica -14 bold')
  ct.pack()

  cb = ComboBox(top, label='Type:', editable=True)
  for animal in ('dog', 'cat', 'hamster', 'python'):
  cb.insert(END, animal)
  cb.pack()

  qb = Button(top, text='QUIT',command=top.quit, bg='red', fg='white')
  qb.pack()

  top.mainloop()
```

Output:



Python Mega Widgets

PMW (Python Mega Widgets) is a toolkit for building high -level widgets in Python using the Tkinter module. This toolkit provides a frame work that contains a variety of widgets richer than the one provided by Tkinter.

It basically helps the extend its longevity by adding more modern widgets to the GUI Palette.This package is 100% written in Python, which turns out to be a cross -platform widget library. Being highly configurable allows it to create additional widget collections by extending the basic Tkinter widget core set.

PMW provides many interesting and complex widgets, including: About Dialog, Balloon, Button Box, Combo Box, Combo Box Dialog, Counter, CounterDialog, Dialog, Entry Field, Group, Labeled Widget, MenuBar, Message Bar, Message Dialog, Note BookR, Note BookS, Note Book, Option Menu, Paned Widget, Prompt Dialog, RadioSelect, Scrolled Canvas, Scrolled Field, ScrolledFrame, Scrolled Listbox, ScrolledText, Selection Dialog, Text Dialog, and Time Counter.

Example:

```
from Tkinter import Button, END, Label, W
from Pmw import initialise, ComboBox, Counter

top = initialise()

lb = Label(top,
text='Animals (in pairs; min: pair, max: dozen)')
lb.pack()

ct = Counter(top, labelpos=W, label_text='Number:',
datatype='integer', entryfield_value=2,
increment=2, entryfield_validate={'validator':'integer', 'min': 2, 'max': 12})
ct.pack()
cb = ComboBox(top, labelpos=W, label_text='Type:')
for animal in ('dog', 'cat', 'hamster', 'python'):
cb.insert(end, animal)
cb.pack()

qb = Button(top, text='QUIT',
command=top.quit, bg='red', fg='white')
qb.pack()
```

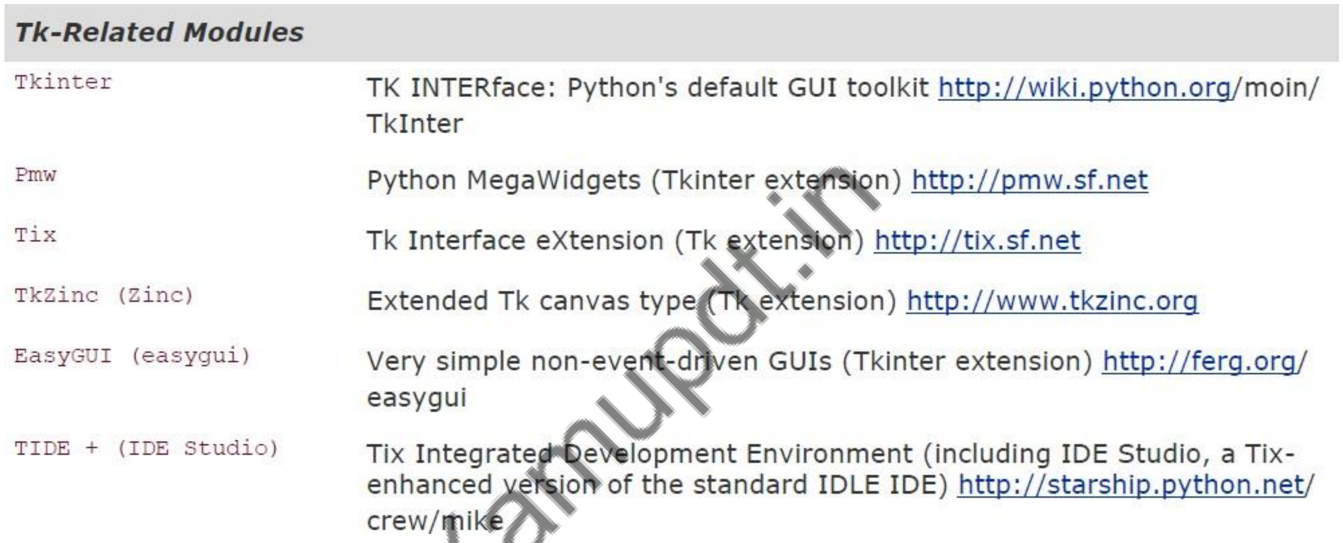
wxWidgets and wxPython

WxWidgets (formerly known as wxWindows) is a cross-platform toolkit used to build graphical user applications. It is implemented using C++ and is available on a wide number of platforms to which wxWidgets defines a consistent and common API. The best part of all is that wxWidgets uses the native GUI on each platform, so your program will have the same ook-and-feel as all the other applications on your desktop. Another feature is that you are not restricted to developing wxWidgets applications in C++. There are interfaces to both Python and Perl.

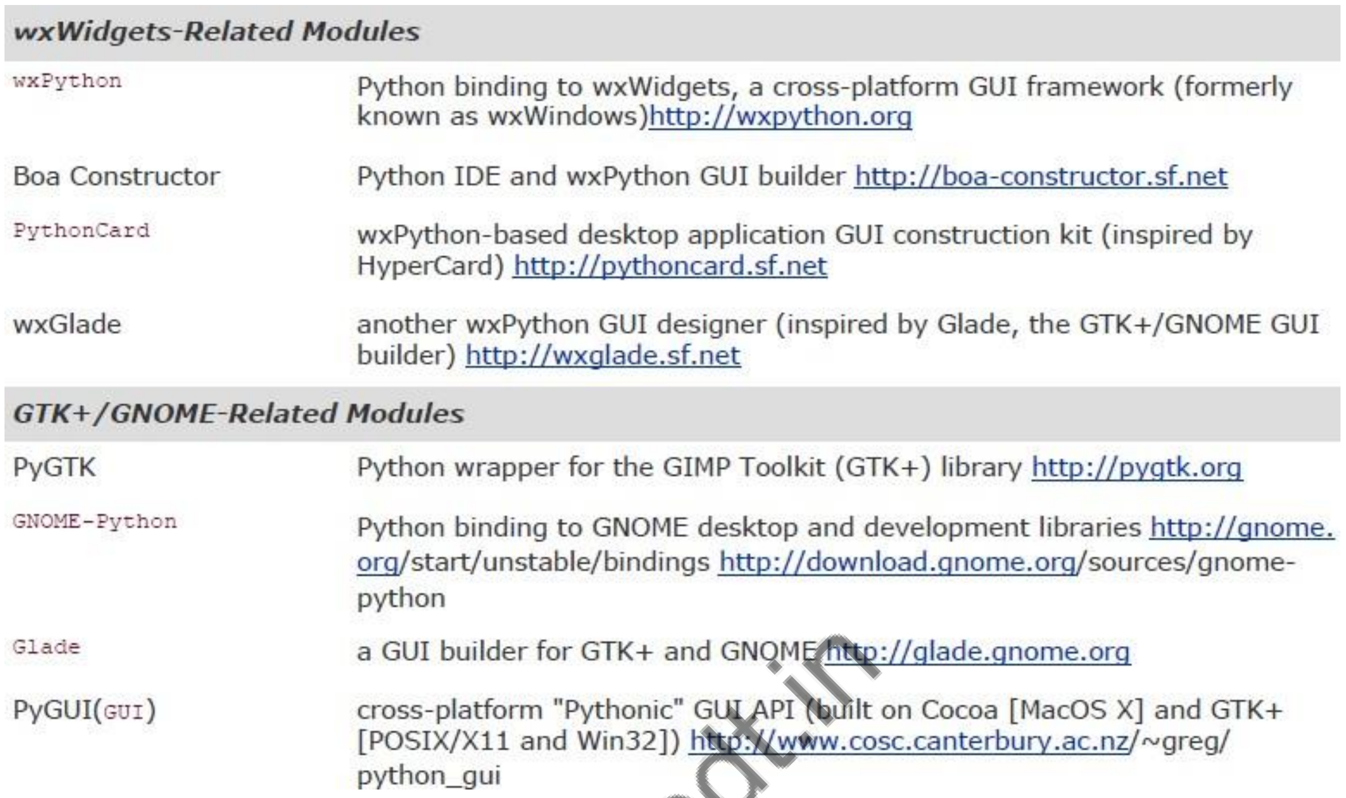
Related Modules and Other GUIs

There are other GUI development systems that can be used with Python. We present the appropriate modules along with their corresponding window systems. Table represents the GUI Systems Available for Python

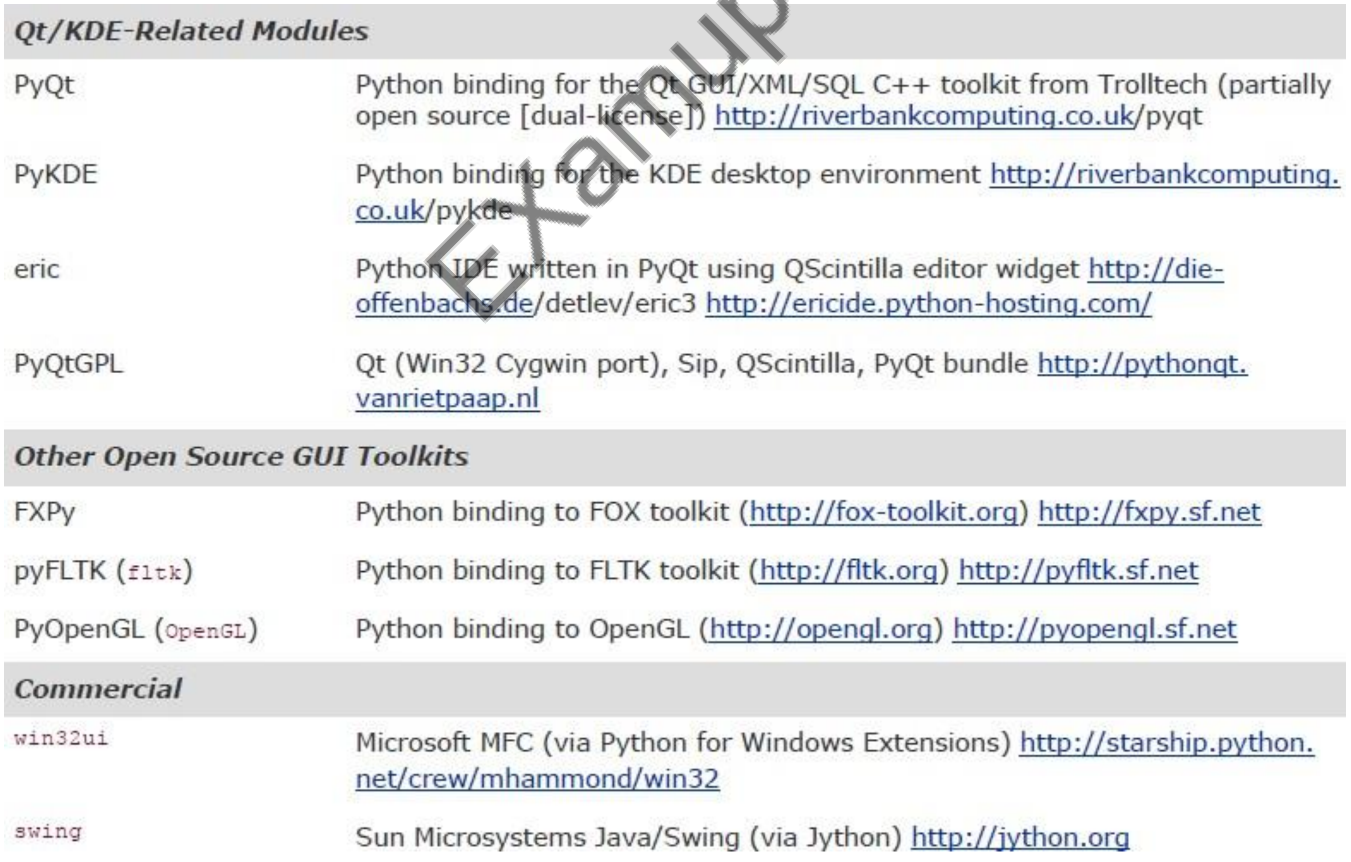
GUI Module or System Description



Tk-Related Modules	
`Tkinter`	TK INTERface: Python's default GUI toolkit http://wiki.python.org/moin/TkInter
`Pmw`	Python MegaWidgets (Tkinter extension) http://pmw.sf.net
`Tix`	Tk Interface eXtension (Tk extension) http://tix.sf.net
`TkZinc (Zinc)`	Extended Tk canvas type (Tk extension) http://www.tkzinc.org
`EasyGUI (easygui)`	Very simple non-event-driven GUIs (Tkinter extension) http://ferg.org/easygui
`TIDE + (IDE Studio)`	Tix Integrated Development Environment (including IDE Studio, a Tix-enhanced version of the standard IDLE IDE) http://starship.python.net/crew/mike



wxWidgets-Related Modules	
wxPython	Python binding to wxWidgets, a cross-platform GUI framework (formerly known as wxWindows)http://wxpython.org
Boa Constructor	Python IDE and wxPython GUI builder http://boa-constructor.sf.net
PythonCard	wxPython-based desktop application GUI construction kit (inspired by HyperCard) http://pythoncard.sf.net
wxGlade	another wxPython GUI designer (inspired by Glade, the GTK+/GNOME GUI builder) http://wxglade.sf.net
GTK+/GNOME-Related Modules	
PyGTK	Python wrapper for the GIMP Toolkit (GTK+) library http://pygtk.org
GNOME-Python	Python binding to GNOME desktop and development libraries http://gnome.org/start/unstable/bindings http://download.gnome.org/sources/gnome-python
Glade	a GUI builder for GTK+ and GNOME http://glade.gnome.org
PyGUI(GUI)	cross-platform "Pythonic" GUI API (built on Cocoa [MacOS X] and GTK+ [POSIX/X11 and Win32]) http://www.cosc.canterbury.ac.nz/~greg/python_gui



Qt/KDE-Related Modules	
PyQt	Python binding for the Qt GUI/XML/SQL C++ toolkit from Trolltech (partially open source [dual-license]) http://riverbankcomputing.co.uk/pyqt
PyKDE	Python binding for the KDE desktop environment http://riverbankcomputing.co.uk/pykde
eric	Python IDE written in PyQt using QScintilla editor widget http://die-offenbachs.de/detlev/eric3 http://ericide.python-hosting.com/
PyQtGPL	Qt (Win32 Cygwin port), Sip, QScintilla, PyQt bundle http://pythonqt.vanrietpaap.nl
Other Open Source GUI Toolkits	
FXPy	Python binding to FOX toolkit (http://fox-toolkit.org) http://fxpy.sf.net
pyFLTK (fltk)	Python binding to FLTK toolkit (http://fltk.org) http://pyfltk.sf.net
PyOpenGL (OpenGL)	Python binding to OpenGL (http://opengl.org) http://pyopengl.sf.net
Commercial	
win32ui	Microsoft MFC (via Python for Windows Extensions) http://starship.python.net/crew/mhammond/win32
swing	Sun Microsystems Java/Swing (via Jython) http://jython.org

WEB Programming:

Introduction
Wed Surfing with Python
Creating Simple Web Clients
Advanced Web Clients
CGI-Helping Servers Process Client Data
Building CGI Application
Advanced CGI
Web (HTTP) Servers

WEB PROGRAMMING

What is web development?

Web development is term for conceptualizing, creating, deploying and operating web applications and application programming interfaces for the Web.

web development involves a frontend, everything that interacts with the client, and a backend, which contains business logic and interacts with a database.

Advantages of developing web applications in Python

- **Easy to learn:** Python is the most popular language for first-time learners for a reason. The language relies on common expressions and whitespace, which allows you to write significantly less code compared to some other languages like Java or C++. Not only that, but it has a lower barrier of entry because it's comparatively more similar to your everyday language so you can easily understand the code.

- **Rich ecosystem and libraries:** Python offers a vast range of library tools and packages, which allows you to access much pre-written code, streamlining your application development time. For example, you have access to Numpy and Pandas for mathematical analysis, Pygal for charting, and SLQALchemy for composable queries. Python also offers amazing web frameworks like Django and Flask, which we'll dive into later in the article.

- **Fast prototyping:** Because Python takes significantly less time to build your projects compared to other programming languages, your ideas come to life a lot faster, allowing you to gain feedback and iterate quickly. This quick development time makes Python especially great for startups.

- **Wide-spread popularity:** Python is one of the most popular languages in the world, with communities from all over the world. Because of how popular the language is, Python is continuously updated with new features and libraries, while also providing excellent documentation and community support. Especially for new developers, Python provides extensive support and framework for one to begin their developer journey.

Q: Explain about different frameworks using in python

Python web frameworks

What are web frameworks, and why are they important?

A **web framework** is a collection of packages and modules made up of pre-written, standardized code that supports the development of web applications, making development faster and easier, and your programs more reliable and scalable.

Python web frameworks are only utilized in the backend for server-side technology, aiding in URL routing, HTTP requests and responses, accessing databases, and web security.

Frameworks automate the common implementation of common solutions which gives the flexibility to the users to focus on the application logic instead of the basic routine processes.

Lets take a look at a few operations involved in a web application using a web framework:

- **URL Routing** – Routing is the mechanism of mapping the URL directly to the code that creates the web page.
- **Input form handling and validation** – Suppose you have a form which takes some input, the idea is to validate the data and then save it.
- **Output formats with template engine** – A template engine allows the developers to generate desired content types like HTML, XML, JSON.
- **Database connection** – Database connection configuration and persistent data manipulation through an ORM.
- **Web security** – Frameworks give web security against cross-site request forgery aka CSRF, sql injection, cross-site scripting and other common malicious attacks.
- **Session storage and retrieval** – Data stored in the session storage gets cleared when the page session ends.

Advantages Of Frameworks

1. Open-source
2. Good documentation
3. Efficient
4. Secure
5. Integration

Why Use A Framework?

Frameworks make it easier to reuse the code for common HTTP operations. They structure the projects in a way so that the other developers with the knowledge of the framework can easily maintain and build the application.

What are some popular Python web frameworks?

- **Django**
- **Flask**
- **Web2Py**
- **Bottle**
- **TurboGears**
- **Pyramid**
- CubicWeb
- CherryPy
- TurboGears
- Quixote

Django

Django, an open-source framework, is a popular high-level web framework in Python which supports rapid web development and design.

Some of its features are-

- Django helps developers avoid various common security mistakes.
- With this framework, developers can take web applications from concept to launch within hours.
- The user authentication system of this framework provides a secure way to manage user accounts and passwords.
- URL routing
- Authentication
- Database schema migrations
- ORM (Object-relational mapper)

- Template engine

MVC-MVT architecture:
MVT is slightly different from MVC, Although Django takes care of the controller part which is the code that controls the interactions between the model and the view. And the template is HTML file mixed with Django template language. Developer provides the model, view and the template. User then maps it to the url and then the rest is done by django to serve it to the user.

The following using the Django framework

Instagram
Mozilla Firefox
NASA

CherryPy

CherryPy is a popular object-oriented web framework in Python. The framework allows building web applications in a much simpler way.

CherryPy allows us to use any type of technology for creating templates and data access. It is still able to handle sessions, cookies, statics, file uploads and everything else a web framework typically can.

Some of its features are-

- A powerful configuration system for developers and deployers alike.
- Built-in profiling, coverage, and testing support.
- Built-in tools for caching, encoding, sessions, authentication, static content etc.
- A flexible plug in system.
- An HTTP WSGI compliant thread pooled web server
- It has simplicity of running multiple HTTP servers at once
- A flexible plug in system
- Caching
- Encoding
- Authentication
- Built-in support for profiling, coverage and testing
- Ability to run on different platforms

TurboGears

TurboGears is a Python web application framework. The next version, TurboGears 2, is built on top of several web frameworks, including TurboGears 1, Rails and Django.

Some of its features are:

- It is designed to be a web application framework suitable for solving complex industrial-strength problems.
- It has a transaction manager to help with multi-database deployments.
- It officially supports MongoDB as one of the primary storage backends.
- It provides support for multiple template engines.

Flask

Flask is a popular Python web framework used for developing complex web applications. The framework offers suggestions but doesn’t enforce any dependencies or project layout.

Some of its features are-

- Flask is flexible.
- The framework aims to keep the core simple but extensible.
- It includes many hooks to customize its behavior.
- Integrated support for unit testing Restful request dispatching
- Contains development server and debugger
- Uses Jinja2 templating
- Support for secure cookies
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

- RESTful request dispatching
- Secure cookies support
- Ability to plug any ORM
- HTTP request handling

Web2Py

Written in Python, Web2Py is a free, open-source web framework for agile development of secure database-driven web applications. It is a full-stack framework.

Some of its features are-

- It is designed to guide a web developer to follow good software engineering practices, such as using the Model View Controller (MVC) pattern.
- Web2Py automatically addresses various issues that can lead to security vulnerabilities by following well-established practices.
- The framework includes a Database Abstraction Layer (DAL) that writes SQL dynamically.
- With no installation and configuration, it is easy to run.
- Supports almost all major operating system, like Windows, Unix/Linux, Mac, Google
- Easy to communicate with MySQL, MSSQL, IBM DB2, Informix, Ingres, MongoDB,
- SQLite, PostgreSQL, Sybase, Oracle and Google App Engine.
- It prevents the most common types of vulnerabilities including Cross Site Scripting,
- Injection Flaws and Malicious File Execution.
- Supports error tracking and internationalization.
- Multiple protocols readability.
- Employs successful software engineering practices that makes code easy to read and maintain. Ensure user-oriented advancements through backward compatibility

Bottle

Bottle is a fast, simple and lightweight WSGI micro web framework for Python web applications. The framework has no other dependencies than the standard Python library.

Some of its features are-

- Bottle runs with Python 2.7 and 3.6+.
- It has a fast and Python ic *built-in template engine* and support for mako, jinja2 and cheetah templates.
- The framework has convenient access to form data, headers, file uploads, cookies, and other HTTP-related metadata.
- Built-in HTTP development server as well as support for bjoern, Google App Engine, fapws3, cherrypy or any other WSGI capable HTTP server.
- Routing

- Templating
- Access to form data, file uploads, cookies, headers etc.
- Abstraction layer over the WSGI standard
- A built-in development server that supports any other WSGI-capable HTTP server.

Pyramid

Pyramid is a lightweight and open-source Python web framework. The framework provides only the core tools needed for nearly all web applications: mapping URLs to code, security, and serving static assets (files like JavaScript and CSS).

Some of its features are-

- New security APIs to support a massive overhaul of the authentication and authorization system.
- **Simplicity** - Anyone can start to work with it without any prior knowledge about it.
- **Minimalism** - Quite out of the box, Pyramid comes with only some important tools, which are needed for almost every web application, may it be security or serving static assets like JavaScript and CSS or attaching URLs to code.
- **Documentation** - Includes exclusive and up to date documentation.
- **Speed** - Very fast and accurate.
- **Reliability** - It is developed, keeping in mind that it is conservative and tested exhaustively. If not tested properly, it will be considered as broke.

Q: Explain about libraries using in web development of python

Python libraries for web development

Some useful Python libraries for web development to keep note of:

- If you ever need a web crawler to extract data for your application, **Scrapy** is great for that. It’s a widely used library for scraping, data mining, automated testing, and more.

- **Zappa** is a powerful library for developing a serverless application on AWS Lambda.

- **Requests** is a library that allows you to send HTTP requests easily, which is used to communicate with an application, allowing you to get HTML pages or data

Web programming refers to the writing, markup and coding involved in Web development, which includes Web content, Web client and server scripting and network security. The most common languages used for Web programming are XML, HTML, JavaScript, Perl 5 and PHP.

The work associated in website design contain multiple areas: web programming, database management, web design, web publishing, etc. Web development includes all the codes that influence a website to run. We can separate the whole process of web development into two categories:

- Front-end
- Back-end

Though frontend and backend web development are certainly distinct from each other, they are also like two sides of the same coin. A complete website relies on each side communicating and operating effectively with the other as a single unit. Both front-end and back-end are equally important in web development.

The front-end or client-side of an application is the code responsible for everything the user directly experiences on screen from text colors to buttons, images and navigation menus. Some of the common skills and tools which are used by front-end developers are listed below:

- HTML/CSS/JavaScript
- CSS preprocessors
- Frameworks
- Libraries
- Git and Github

The back-end/server-side of an application is responsible for managing information within the database and serving that information to the front-end. The backend of a website consists of a server, application, and database. In general, it involves everything that happens before hitting your browser. The tools required in back-end web development are:

.•Programming language – Ruby, PHP, Python, etc.
- Database – MySQL, PostgreSQL, MongoDB, Oracle, etc.

Web Surfing with Python: Creating Simple Web Clients

This time, Web *clients* are browsers, applications that allow users to seek documents on the World Wide Web. On the other side are Web *servers*, processes that run on an information provider's host computers. These servers wait for clients and their document requests, process them, and return the requested data. As with most servers in a client/server system, Web servers are designed to run "forever."

A user runs a Web client program such as a browser and makes a connection to a Web server elsewhere on the Internet to obtain information.

Clients may issue a variety of requests to Web servers. Such requests may include obtaining a Web page for viewing or submitting a form with data for processing. The request is then serviced by the Web server, and the reply comes back to the client in a special format for display purposes.

The "language" that is spoken by Web clients and servers, the standard protocol used for Web communication, is called *HTTP*, which stands for Hyper Text Transfer Protocol. HTTP is written "on top of" the TCP and IP protocol suite, meaning that it relies on TCP and IP to carry out its lower-level communication functionality. Its responsibility is not to route or deliver messages TCP and IP handle that but to respond to client requests (by sending and receiving HTTP messages).

HTTP is known as a "stateless" protocol because it does not keep track of information from one client request to the next, similar to the client/server architecture we have seen so far. The server stays running, but client interactions are singular events structured in such a way that once a client request is serviced, it quits. New requests can always be sent, but they are considered separate service requests.

Because of the lack of context per request, you may notice that some URLs have a long set of variables and values chained as part of the request to provide some sort of state information. Another alternative is the use of "cookies" static data stored on the client side which generally contain state information as well. In later parts of this chapter, we will look at how to use both long URLs and cookies to maintain state information.

The Internet

The Internet is a moving and fluctuating "cloud" or "pond" of interconnected clients and servers scattered around the globe. Communication between client and server consists of a series of connections from one lily pad on the pond to another, with the last step connecting to the server.

The abstraction is to have a direct connection between you the client and the server you are "visiting," but the underlying HTTP, TCP, and IP protocols are hidden underneath, doing all of the dirty work. Information regarding the intermediate "nodes" is of no concern or consequence to the general user anyway, so it's good that the implementation is hidden.

Creating Simple Web Clients

Browser is only one type of Web client. Any application that makes a request for data from a Web server is considered a "client." Yes, it is possible to create other clients that retrieve documents or data off the Internet. One important reason to do this is that a browser provides only limited capacity, i.e., it is used primarily for viewing and interacting with Web sites.

Applications that use the **urllib module** to download or access information on the Web [using either **urllib.urlopen()** or **urllib.urlretrieve()**] can be considered a simple Web client. All you need to do is provide a valid Web address.

Q: Explain about URL (Uniform Resource Locators) in detail

Uniform Resource Locators

Simple Web surfing involves using Web addresses called *URLs* (Uniform Resource Locators). Such addresses are used to locate a document on the Web or to call a CGI program to generate a document for your client.
URLs are part of a larger set of identifiers known as *URIs* (Uniform Resource Identifiers). This superset was created in anticipation of other naming conventions that have yet to be developed.

A URL is simply a URI which uses an existing protocol or scheme (i.e., http, ftp, etc.) as part of its addressing.
To complete this picture, we'll add that non-URL URIs are sometimes known as *URNs* (Uniform Resource Names), but because URLs are the only URIs in use today, you really don't hear much about URIs or URNs, save perhaps as XML identifiers.

A URL uses the format:

prot_sch://net_loc/path;params?query#frag

Web Address Components

URL Component	**Description**
prot_sch	*Network protocol or download scheme*
net_loc	*Location of server (and perhaps user information)*
path	*Slash (/) delimited path to file or CGI application*
params	*Optional parameters*
query	*Ampersand (&) delimited set of "key=value" pairs*
frag	*Fragment to a specific anchor within document*

net_loc can be broken down into several more components, some required, others optional. The *net_loc* string looks like this:

user:passwd@host:port

Network Location Components

net_loc *Component*	*Description*
user	*User name or login*
passwd	*User password*
host	*Name or address of machine running Web server [required]*
port	*Port number (if not 80, the default)*

The **host** name is the most important. The **port** number is necessary only if the Web server is running on a different port number from the default.

Q: Explain about different modules which are using in web development in python

Modules

Python supplies two different modules, each dealing with URLs in completely different functionality and capacities.

1. **urlparse**
2. **urllib**

1. urlparse Module

The urlparse module provides basic functionality with which to manipulate URL strings. These functions include **urlparse()**, **urlunparse()**, and **urljoin()**

urlparse.urlparse()

urlparse() breaks up a URL string into some of the major components described above. It has the following syntax:

urlparse(*urlstr*, *defProtSch*=None, *allowFrag*=None)

urlparse() parses *urlstr* into a 6-tuple (prot_sch, net_loc, path, params, query, frag). Each of these components has been described above.

defProtSch indicates a default network protocol or download scheme in case one is not provided in urlstr.*allowFrag* is a flag that signals whether or not a fragment part of a URL is allowed.
Here is what urlparse() outputs when given a URL:

```
>>> urlparse.urlparse('http://www.python.org/doc/FAQ.html')
('http', 'www.python.org', '/doc/FAQ.html', '', '', '')
```

urlparse.urlunparse()

urlunparse() does the exact opposite of urlparse()it merges a 6-tuple (prot_sch, net_loc, path, params, query, frag)*urltup*, which could be the output of urlparse(), into a single URL string and returns it.
Accordingly, we state the following equivalence:

urlunparse(urlparse(*urlstr*)) *urlstr*

You may have already surmised that the syntax of urlunparse() is as follows:
urlunparse(*urltup*)

urlparse.urljoin()

The urljoin() function is useful in cases where many related URLs are needed, for example, the URLs for a set of pages to be generated for a Web site. The syntax for urljoin() is:

urljoin(*baseurl*, *newurl*, *allowFrag*=None)

urljoin() takes *baseurl* and joins its base path (*net_loc* plus the full path up to, but not including, a file at the end) with *newurl*. For example:

```
>>> urlparse.urljoin('http://www.python.org/doc/FAQ.html', \
... 'current/lib/lib.htm')
'http://www.python.org/doc/current/lib/lib.html'
```

Core urlparse Module Functions

urlparse Functions	**Description**
urlparse(urlstr, defProtSch=None, allowFrag=None)	Parses urlstr into separate components, using defProtSch if the protocol or scheme is not given in urlstr; allowFrag determines whether a URL fragment is allowed
urlunparse(urltup)	Unparses a tuple of URL data (urltup) into a single URL string
urljoin(baseurl,newurl,allowFrag=None)	*Merges the base part of the* baseurl *URL with* newurl *to form a complete URL;* allowFrag *is the same as for* urlparse()

urllib Module

urllib

provides a high-level Web communication library, supporting the basic Web protocols, HTTP, FTP, and Gopher, as well as providing access to

local files.
Specifically, the functions of the urllib module are
designed to download data (from the Internet, local network, or local
host) using the aforementioned protocols.
Use of this module generally obviates the need for using the **httplib**, **ftplib**, and **gopherlib** modules unless you desire their lower-level functionality.

The **urllib** module provides functions to download data from given URLs as well as encoding and decoding strings to make them suitable for including as part of valid URL strings.

Q: Explain about the different functions in urllib module of web development

The following functions we are using in **urllib** module.

1.urlopen(),
2.urlretrieve(),
3.quote(),
4.unquote(),
5.quote_plus(),
6.unquote_plus(), and
7.urlencode().

urllib.urlopen()

urlopen() opens a Web connection to the given URL string and returns a file-like object. It has the following syntax:

urlopen(*urlstr*, *postQueryData*=None)

urlopen() opens the URL pointed to by *urlstr*. If no protocol or download scheme is given, or if a "file" scheme is passed in, urlopen() will open a local file.

For all HTTP requests, the normal request type is "GET." In these cases, the query string provided to the Web server (key-value pairs encoded or "quoted," such as the string output of the urlencode() function[see below]), should be given as part of *urlstr*.

If the "POST" request method is desired, then the query string (again encoded) should be placed in the **postQueryData** variable.

GET and POST requests are the two ways to "upload" data to a Web server.

Finally, a geturl() method exists to obtain the true URL of the final opened destination, taking into consideration any redirection that may have occurred.

urllib.urlopen() File-like Object

urlopen() ***Object Methods Description***	***Description***
f.read([bytes])	Reads all or bytes bytes from f
f.readline()	Reads a single line from f
f.readlines()	Reads a all lines from f into a list
f.close()	Closes URL connection for f
f.fileno()	Returns file number of f
f.info()	Gets MIME headers of f
f.geturl()	Returns true URL opened for f

urllib.urlretrieve()

urlretrieve() will do some quick and dirty work for you if you are interested in working with a URL document as a whole. Here is the syntax for urlretrieve():

urlretrieve(*urlstr*, *localfile*=None, *downloadStatusHook*=None)

Rather than reading from the URL like **urlopen()** does, **urlretrieve()** will simply download the entire HTML file located at ***urlstr*** to your local disk.

It will store the downloaded data into *local file* if given or a temporary file if not.

If the file has already been copied from the Internet or if the file is local, no subsequent downloading will occur.

urllib.quote() and urllib.quote_plus()

The quote*() functions take URL data and "encodes" them so that they are "fit" for inclusion as part of a URL string.
In particular, certain special characters that are unprintable or cannot be part of valid URLs acceptable to a Web server must be converted.
This is what the quote*() functions do for you. Both quote*() functions have the following syntax:

quote(*urldata*, *safe*='/')

Characters that are never converted include commas, underscores, periods, and dashes, as well as alphanumeric.

When calling **quote*()**, the **urldata** string is converted to an equivalent string that can be part of a URL string.
The *safe* string should contain a set of characters which should also *not* be converted. The default is the slash (/).

urllib.unquote() and urllib.unquote_plus()

The unquote*() functions do the exact opposite of the quote*() functionsthey convert all characters encoded in the "%xx" fashion to their ASCII equivalents. The syntax of unquote*() is as follows:

unquote*(*urldata*)

Calling **unquote()** will decode all URL-encoded characters in **urldata** and return the resulting string.
unquote_plus() will also convert plus signs back to space characters.

urllib.urlencode()

The pairs are in "key=value" format and are delimited by ampersands (&). Furthermore, the keys and their values are sent to quote_plus() for proper encoding. Here is an example output from urlencode():

```
>>> aDict = { 'name': 'Georgina Garcia', 'hmdir': '~ggarcia' }
>>> urllib.urlencode(aDict)
'name=Georgina+Garcia&hmdir=%7eggarcia'
```

Advanced Web Clients

Web browsers are basic Web clients. They are used primarily for searching and downloading documents from the Web. Advanced clients of the Web are those applications that do more than download single documents from the Internet.
One example of an advanced Web client is a *crawler* (aka *spider*, *robot*). These are programs that explore and download pages from the Internet for different reasons, some of which include:
● Indexing into a large search engine such as Google or Yahoo!
● Offline browsing downloading documents onto a local hard disk and rearranging hyperlinks to create almost a mirror image for local browsing
● Downloading and storing for historical or archival purposes, or
● Web page caching to save superfluous downloading time on Web site revisits.

Q: Explain about CGI (Common Gateway Interface) in python

CGI: Helping Web Servers Process Client Data

Introduction to CGI

CGI means the "**Common Gateway Interface".** CGI is one of the essential parts of HTTP (Hyper-Text Transfer Protocol).

CGI is a set of standards that defines a standard way of passing information or web-user requests to an application program and getting data back to forward it to users.

It is the exchange of information between the web server and a custom script. When the users requested the web-page, the server sends the requested web-page.

The web server usually passes the information to all application programs that process data and sends back an acknowledged message; this technique of passing data back-and-forth between server and application is the Common Gateway Interface.

The main feature of HTML is in its hypertext capability, text that is in one way or another highlighted to point to another document in a related context to the original.

Such a document can be accessed by a mouse click or other user selection mechanism. These (static) HTML documents live on the Web server and are sent to clients when and if requested.

Browsing

What happens when a user clicks a hyperlink to browse a particular web-page or URL (Uniform Resource Locator).

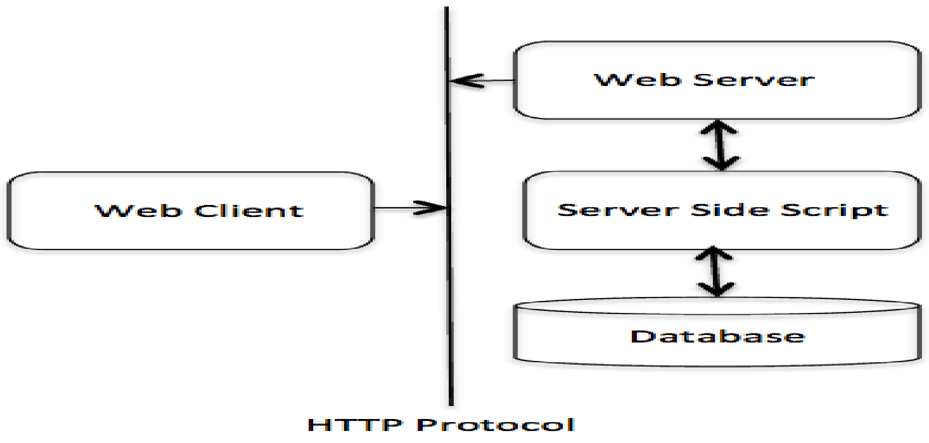
The steps are:

- Browser contacts the HTTP web server for demanding the URL
- Parsing the URL
- Look for the filename
- If it finds that file, a request is sent back
- Web browser takes a response from the web server
- As the server response, it either shows the received file or an error message.

Configuring CGI

The steps are:

1. Find out which user is running the Web-server
2. Check for server configuration to see if you can run the scripts in a particular directory
3. Check for file's permission
4. Make a clear assurance that scripts you made are readable and executable by the web server user
5. Make sure the Python-Script's first line refers to a webserver that the interpreter can run



CGI Environment variables

Environment Variables	Description
CONTENT_TYPE	describes the data-type of the content
HTTP_COOKIE	returns the visitor's cookie if one is set
CONTENT_LENGTH	It is available for a POST request to define the length of query information
HTTP_USER_AGENT	defines the browser type of the visitor
PATH_INFO	defines the path of a CGI script
REMOTE_HOST	defines the host-name of the visitor
REMOTE_ADDR	defines the IP Address of the visitor
REQUEST_METHOD	used to make request & the most common methods are - GET and POST
QUERY_STRING	The URL-encoded information that is sent with GET method request.
SCRIPT_FILENAME	The full path to the CGI script.
SCRIPT_NAME	The name of the CGI script.
SERVER_NAME	The server's hostname or IP Address
SERVER_SOFTWARE	The name and version of the software the server is running.

HTML Header

In the above program, the line **Content-type:text/html\r\n\r\n** is a portion of the HTTP, which we will use in our CGI programs.

1. HTTP Field Name: Field Content

2. For Example
3. Content-type: text/html\r\n\r\n

Sr.	Header	Description
1.	Content-type	It is a MIME string that is used to define the format of the file being returned.
2.	Expires: Date	It displays the valid date information.
3.	Location: URL	The URL that is returned by the server.
4.	Last-modified: Date	It displays the date of the last modification of the resource.
5.	Content-length: N	This information is used to report the estimated download time for a file.
6.	Set-Cookies: String	It is used to set the cookies by using string.

Functions of Python CGI Programming

The CGI module provides the many functions to work with the cgi. We are defining a few important functions as follows.

parse(fp = None, environ = os.environ, keep_blanks_values = False, strict_parsing = False) –

It is used to parse a query in the environment. We can also parse it using a file, the default for which is **sys.stdin.**

parse_qs(qs, keep_blank_values = False, strict _parsing = False) - While this is denigrated, Python uses it for urllib.parse.parse_qs() instead.

parse_qsl(qs, keep_blank_value = False, strict_parsing = False) - This is also denigrated,and maintains of for backward-compatibility.

parse_multipart(fb, pdict) - It is used to parse input of type multipart/form-data

for file uploads.The first argument is the **input file,** and the second argument is

a dictionary holding in the other parameters in the content-type header.

parse_header(string) - It is used to parse the header. It permits the MIME header into the main value and a dictionary of parameters.

test() - It is used to test a CGI script, and we can use it in our program. It will generally write minimal HTTP headers.

print_form(form) - It formats a form in HTML.

print_directory() - It formats the current directory in HTML.

escape(s, quote = False) - The **escape()** function is used to convert characters

'<', '>', and '&' in the string's to HTML safe sequence.

First CGI Program

Here is a simple link, which is linked to a CGI script called hello.py. This file is kept in /var/www/cgi-bin directory and it has following content.

Before running your CGI program, make sure you have change mode of file using **chmod 755 hello.py** UNIX command to make file executable.

```
#!/usr/bin/python


print("Content-type:text/html\r\n\r\n")

print('<html>')

print('<head>')

print('<title>Hello World - First CGI Program</title>')

print('</head>')

print('<body>')
print('<h2>Hello World! This is my first CGI program</h2>')

print('</body>')

print('</html>')
```

GET and POST Methods

You must have come across many situations when you need to pass some information from your browser to web server and ultimately to your CGI Program.

Most frequently, browser uses two methods two pass this information to web server. These methods are GET Method and POST Method.

Passing Information using GET method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the?character as follows −

http://www.test.com/cgi-bin/hello.py?key1=value1&key2=value2

The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location: box.

Never use GET method if you have password or other sensitive information to pass to the server. The GET method has size limitation: only 1024 characters can be sent in a request string. The GET method sends information using QUERY_STRING header and will be accessible in your CGI Program through QUERY_STRING environment variable.

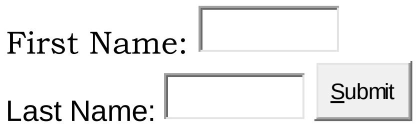
You can pass information by simply concatenating key and value pairs along with any URL or you can use HTML <FORM> tags to pass information using GET method.

Simple FORM Example: GET Method

This example passes two values using HTML FORM and submit button. We use same CGI script hello_get.py to handle this input.

```
<form action = "/cgi-bin/hello_get.py" method = "get">
First Name: <input type = "text" name = "first_name">  <br />

Last Name: <input type = "text" name = "last_name" />
<input type = "submit" value = "Submit" />
</form>
```

Here is the actual output of the above form, you enter First and Last Name and then click submit button to see the result.



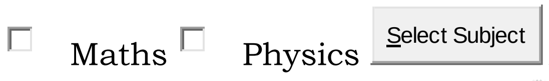
Passing Checkbox Data to CGI Program

Checkboxes are used when more than one option is required to be selected.

Here is example HTML code for a form with two checkboxes −

```
<form action = "/cgi-bin/checkbox.cgi" method = "POST" target = "_blank">
<input type = "checkbox" name = "maths" value = "on" /> Maths
<input type = "checkbox" name = "physics" value = "on" /> Physics
<input type = "submit" value = "Select Subject" />
</form>
```

The result of this code is the following form −



CGI Applications

A CGI application is slightly different from a typical program. The primary differences are in the input, output, and user interaction aspects of a computer program. When a CGI script starts, it needs to retrieve the user-supplied form data, but it has to obtain this data from the Web client, not a user on the server machine nor a disk file.

The output differs in that any data sent to standard output will be sent back to the connected Web client rather than to the screen, GUI window, or disk file. The data sent back must be a set of valid headers followed by HTML.

If it is not and the Web client is a browser, an error(specifically, an Internal Server Error) will occur because Web clients such as browsers understand only valid HTTP data (i.e., MIME headers and HTML).

cgi Module

Field Storage class

This class should be instantiated when a Python CGI script begins, as it will read in all the pertinent user information from the Web client (via the Web server).

Once this object has been instantiated, it will consist of a dictionary-like object that has a set of key-value pairs. The keys are the names of the form items that were passed in through the form while the values contain the corresponding data.

These values themselves can be one of three objects.

They can be Field Storage objects (instances) as well as instances of a similar class called Mini Field Storage, which is used in cases where no file uploads or multiple-part form data is involved.

Mini Field Storage instances contain only the key-value pair of the name and the data. Lastly, they can be a list of such objects. This occurs when a form contains more than one input item with the same field name.

Building CGI (Common Gateway Interface) Applications

Setting Up a Web Server

In order to play around with CGI development in Python, you need to first install a Web server, configure it for handling Python CGI requests, and then give the Web server access to your CGI scripts. Some of these tasks may require assistance from your system administrator.

Just start up the most basic Web server, just execute it directly with Python:
$ python -m CGIHTTPServer

This will start a Web server on port 8000 on your current machine from the current directory.
Then you can just create a Cgi-bin right underneath the directory from which you started the server and put your Python CGI scripts in there.
Put some HTML files in that directory and perhaps some .py CGI scripts in Cgi-bin, and you are ready to "surf" directly to this Web site with addresses looking something

likethese:

http://localhost:8000/friends.htm

http://localhost:8000/cgi-bin/friends2.py

Creating the Form Page

Static Form Web Page (friends.htm)

```
<HTML><HEAD><TITLE>
 Friends CGI Demo (static screen)
</TITLE></HEAD>
<BODY><H3>Friends list for: <I>NEW USER</I></H3>
<FORM ACTION="/cgi-bin/friends1.py">
<B>Enter your Name:</B>
<INPUT TYPE=text NAME=person VALUE="NEW USER" SIZE=15>
<P><B>How many friends do you have?</B>
<INPUT TYPE=radio NAME=howmany VALUE="0" CHECKED> 0
<INPUT TYPE=radio NAME=howmany VALUE="10"> 10
<INPUT TYPE=radio NAME=howmany VALUE="25"> 25
<INPUT TYPE=radio NAME=howmany VALUE="50"> 50
<INPUT TYPE=radio NAME=howmany VALUE="100"> 100
<P><INPUT TYPE=submit></FORM></BODY></HTML>
```

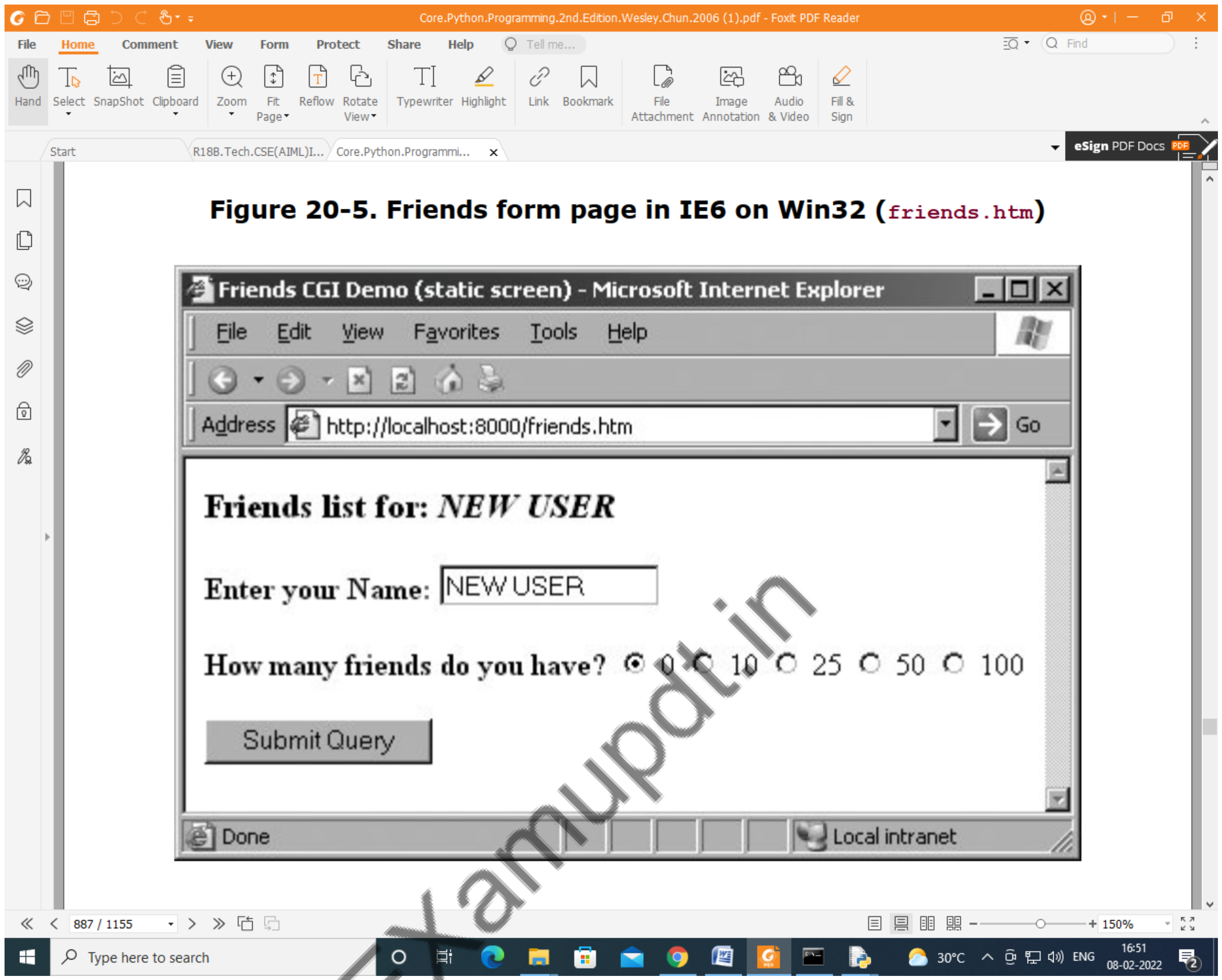

Core.Python.Programming.2nd.Edition.Wesley.Chun.2006 (1).pdf - Foxit PDF Reader
File Home Comment View Form Protect Share Help
Hand Select SnapShot Clipboard Zoom Fit Page Reflow Rotate View Typewriter Highlight Link Bookmark File Attachment Image Annotation Audio & Video Fill & Sign
Start R18B.Tech.CSE(AIML)I... Core.Python.Programmi...
Friends CGI Demo (static screen) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Done
Local intranet
887 / 1155
150%
Type here to search
30°C ENG 16:51 08-02-2022

Figure 20-5. Friends form page in IE6 on Win32 (friends.htm)

http://localhost:8000/friends.htm

Friends list for: *NEW USER*

Enter your Name: NEW USER

How many friends do you have? [✓ 0] 10 25 50 100

Submit Query

Generating the Results Page

Results Screen CGI code (friends1.py)

```
#!/usr/bin/env python

 import cgi
 reshtml = '''Content-Type: text/html\n
 <HTML><HEAD><TITLE>
 Friends CGI Demo (dynamic screen)
 </TITLE></HEAD>
 <BODY><H3>Friends list for: <I>%s</I></H3>
 Your name is: <B>%s</B><P>
 You have <B>%s</B> friends.
 </BODY></HTML>'''
 form = cgi.FieldStorage()
who = form['person'].value
howmany = form['howmany'].value
print reshtml % (who, who, howmany)
```

This script contains all the programming power to read the form input and process it, as well as return the resulting HTML page back to the user.

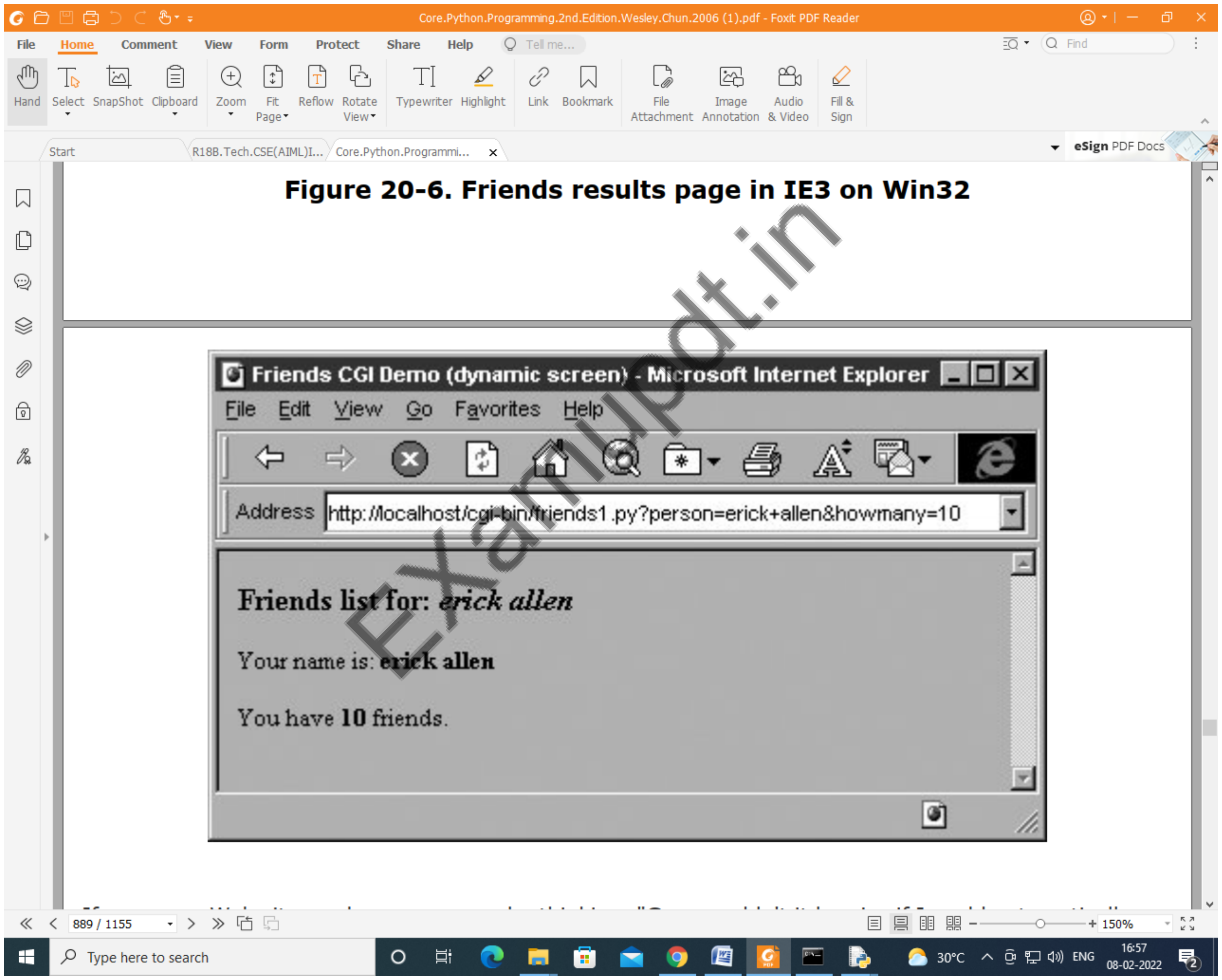


Figure 20-6. Friends results page in IE3 on Win32

Friends CGI Demo (dynamic screen) - Microsoft Internet Explorer

Address http://localhost/cgi-bin/friends1.py?person=erick+allen&howmany=10

Friends list for: *erick allen*

Your name is: **erick allen**

You have **10** friends.

Using Unicode with CGI

First of all we define the message in a Unicode string. We assume your text editor can only enter ASCII. Therefore the non-ASCII characters are input using the \u escape. In practice the message can also be read from a file or from database.

```
# Greeting in English, Spanish,
# Chinese and Japanese.
UNICODE_HELLO = u"""
Hello!
\u00A1Hola!
\u4F60\u597D!
\u3053\u3093\u306B\u3061\u306F!
"""
```

The first output the CGI generates is the content-type HTTP header. It is very important to declare here that the content is transmitted in the UTF-8 encoding so that the browser can correctly interpret it.

```
print 'Content-type: text/html; charset=UTF-8\r'
print '\r'
```

Then output the actual message. Use the string's encode() method to translate the string into UTF-8 sequences first.

```
print UNICODE_HELLO.encode('UTF-8')
```

Simple Unicode CGI Example (uniCGI.py)

```
#!/usr/bin/env python

CODEC = 'UTF-8'
UNICODE_HELLO = u'''
Hello!
\u00A1Hola!
\u4F60\u597D!
\u3053\u3093\u306B\u3061\u306F!
'''

 print 'Content-Type: text/html; charset=%s\r' % CODEC
 print '\r'13print '<HTML><HEAD><TITLE>Unicode CGI Demo</TITLE></HEAD>'
 print '<BODY>'
 print UNICODE_HELLO.encode(CODEC)
 print '</BODY></HTML>'
```

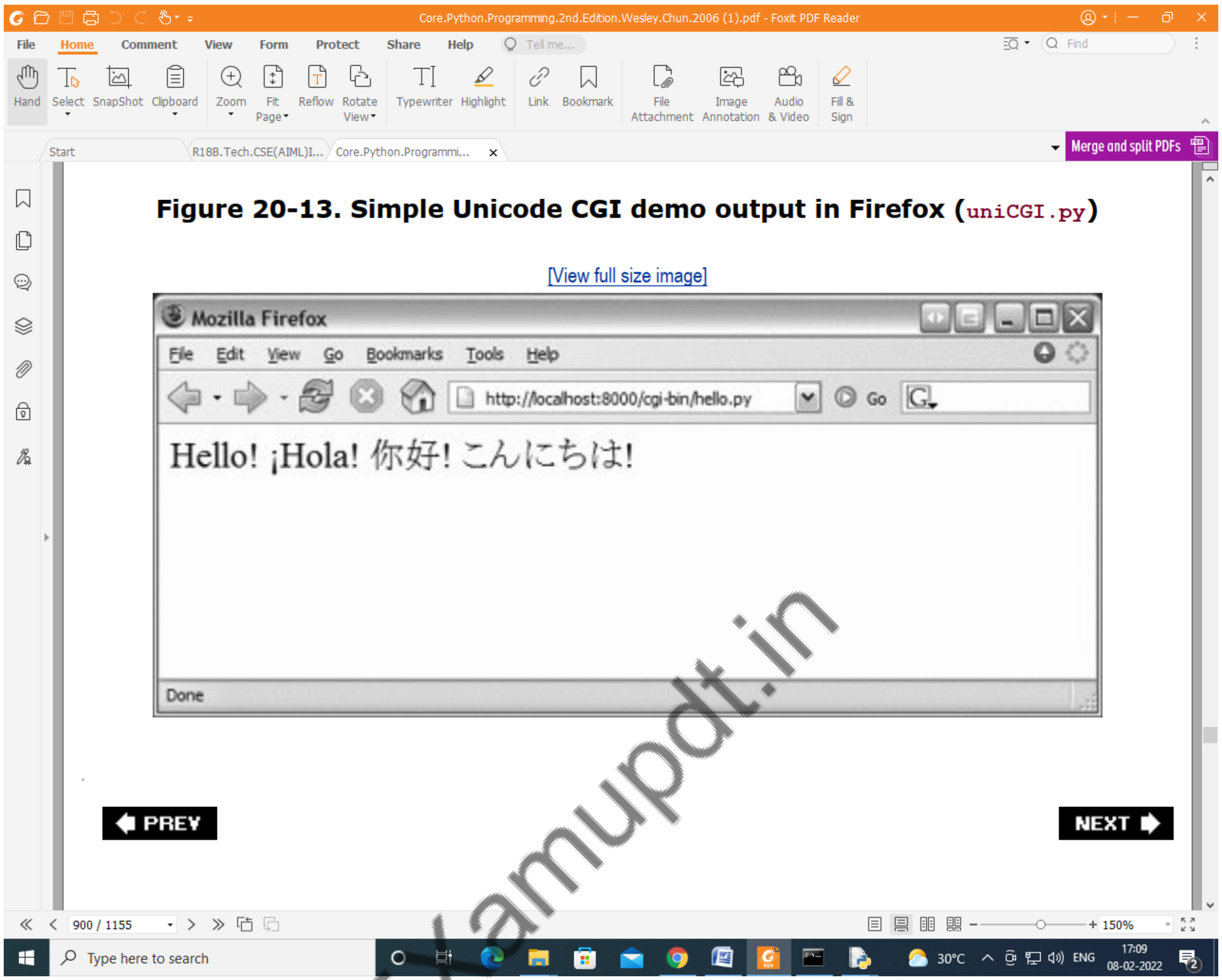


Figure 20-13. Simple Unicode CGI demo output in Firefox (uniCGI.py)

[View full size image]

Hello! ¡Hola! 你好! こんにちは!

Core.Python.Programming.2nd.Edition.Wesley.Chun.2006 (1).pdf - Foxit PDF Reader
Mozilla Firefox
http://localhost:8000/cgi-bin/hello.py
Done
PREV
NEXT
900 / 1155
150%

Q: Explain advantages of CGI Programming in python

Advantages of CGI Programming

There are various advantages of using CGI programming. Below are some of its advantages.

- The CGI programs can work on almost any web server and the portable.
- They are portable.
- The CGIs can increase the dynamic communication in the web applications.
- The CGIs can also be profitable, if we use them in development; they reduce the development costs and maintenance costs.
- The CGIs takes less time to process the requests.

Advanced CGI

These include the use of *cookies* cached data saved on the client side multiple values for the same CGI field and file upload using multipart form submissions.

1.Multipart Form Submission and File Uploading

the CGI specifications only allow two types of form encodings, "**application/x-www-formurlencoded**" and "**multipart/form-data**." Because the former is the default, there is never a need to state the encoding in the FORM tag like this:

<FORM enctype="application/x-www-form-urlencoded" ...>

But for multipart forms, you must explicitly give the encoding as:

<FORM enctype="multipart/form-data" ...>

File uploads are accomplished using the file input type:

<INPUT type=file name=...>

This directive presents an empty text field with a button on the side which allows you to browse your file directory structure for a file to upload.
When using multipart, your Web client's form submission to the server will look amazingly like (multipart) e-mail messages with attachments.

File Upload Example

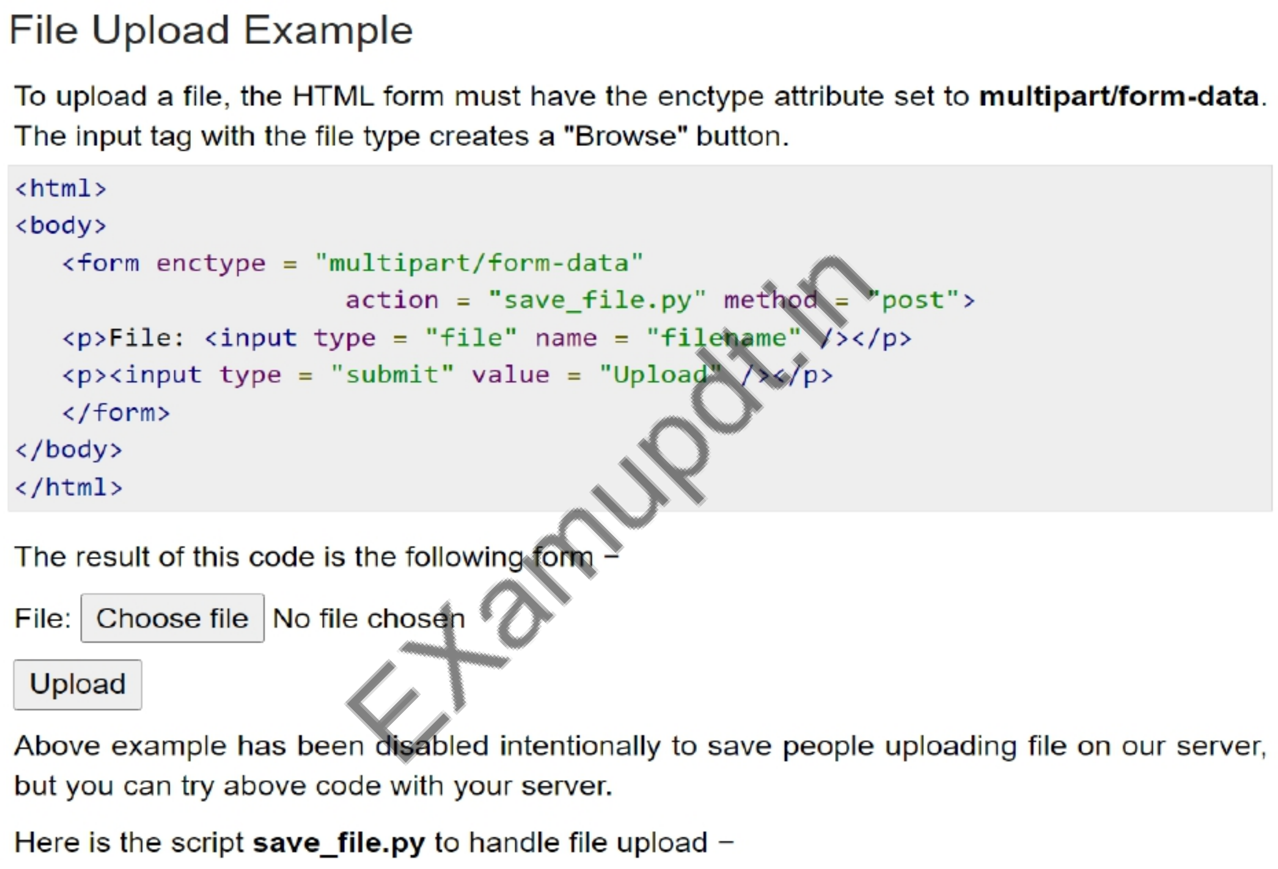
To upload a file, the HTML form must have the enctype attribute set to **multipart/form-data**. The input tag with the file type creates a "Browse" button.

```
<html>
<body>
   <form enctype = "multipart/form-data"
                     action = "save_file.py" method = "post">
   <p>File: <input type = "file" name = "filename" /></p>
   <p><input type = "submit" value = "Upload" /></p>
   </form>
</body>
</html>
```

The result of this code is the following form −

File:

Upload



File Upload Example

To upload a file, the HTML form must have the enctype attribute set to **multipart/form-data**. The input tag with the file type creates a "Browse" button.

```
<html>
<body>
   <form enctype = "multipart/form-data"
                     action = "save_file.py" method = "post">
   <p>File: <input type = "file" name = "filename" /></p>
   <p><input type = "submit" value = "Upload" /></p>
   </form>
</body>
</html>
```

The result of this code is the following form −

File: Choose file No file chosen

Upload

Above example has been disabled intentionally to save people uploading file on our server, but you can try above code with your server.

Here is the script **save_file.py** to handle file upload −

Passing Text Area Data to CGI Program

TEXTAREA element is used when multiline text has to be passed to the CGI Program.

Here is example HTML code for a form with a TEXTAREA box −

```
<form action = "/cgi-bin/textarea.py" method = "post" target = "_blank">
<textarea name = "textcontent" cols = "40" rows = "4">
Type your text here...
</textarea>
<input type = "submit" value = "Submit" />
```

```
</form>
```

The result of this code is the following form −

Below is textarea.cgi script to handle input given by web browser −

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('textcontent'):
   text_content = form.getvalue('textcontent')
else:
   text_content = "Not entered"

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>";
print "<title>Text Area - Fifth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Entered Text Content is %s</h2>" % text_content
print "</body>"
```

2. Multivalued Fields

The most common case is when you have a set of checkboxes allowing a user to select from various choices.
Each of the checkboxes is labeled with the same field name, but to differentiate them, each will have a different value associated with a particular checkbox.

When more than one checkbox is submitted, you will have multiple values associated with the same key. In these cases, rather than being given a single **MiniFieldStorage** instance for your data, the **cgi module** will create a list of such instances that you will iterate over to obtain the different values.

3. Cookies

HTTP protocol is a stateless protocol. For a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. How to maintain user's session information across all the web pages?

In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

How It Works?

Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the cookie is available for retrieval. Once retrieved, your server knows/remembers what was stored.

Cookies are a plain text data record of 5 variable-length fields −

- **Expires** − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- **Domain** − The domain name of your site.
- **Path** − The path to the directory or web page that sets the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- **Secure** − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **Name=Value** − Cookies are set and retrieved in the form of key and value pairs.

Setting up Cookies

It is very easy to send cookies to browser. These cookies are sent along with HTTP Header before to Content-type field. Assuming you want to set UserID and Password as cookies. Setting the cookies is done as follows −

```
#!/usr/bin/python

print "Set-Cookie:UserID = XYZ;\r\n"
print "Set-Cookie:Password = XYZ123;\r\n"
print "Set-Cookie:Expires = Tuesday, 31-Dec-2007 23:12:40 GMT";\r\n"
print "Set-Cookie:Domain = www.tutorialspoint.com;\r\n"
print "Set-Cookie:Path = /perl;\n"
print "Content-type:text/html\r\n\r\n"
..........Rest of the HTML Content....
```

From this example, you must have understood how to set cookies. We use **Set-Cookie** HTTP header to set cookies.

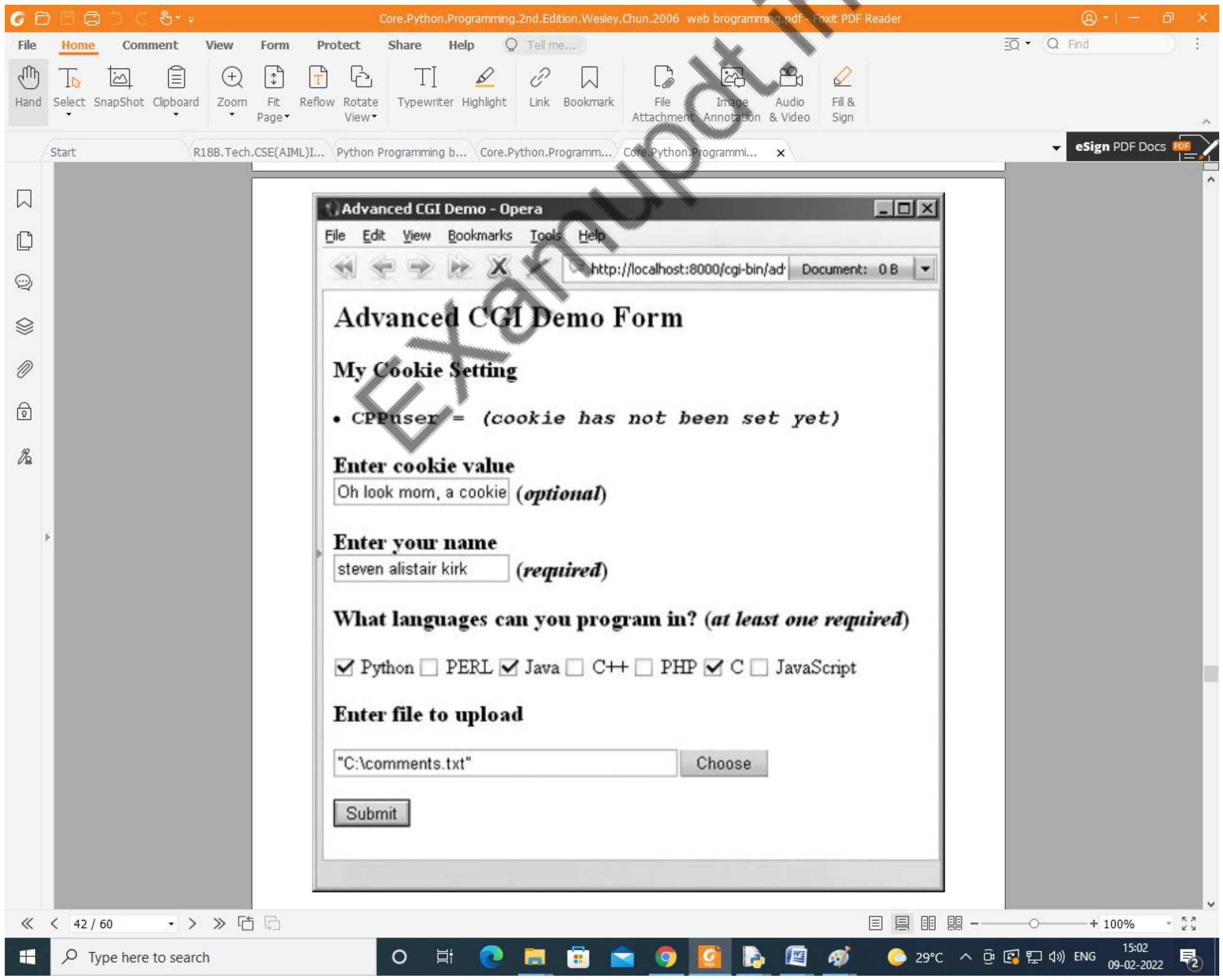
It is optional to set cookies attributes like Expires, Domain, and Path. It is notable that cookies are set before sending magic line "**Content-type:text/html\r\n\r\n**.

Retrieving Cookies

It is very easy to retrieve all the set cookies. Cookies are stored in CGI environment variable HTTP_COOKIE and they will have following form −

key1 = value1;key2 = value2;key3 = value3....

Using Advanced CGI


Core.Python.Programming.2nd.Edition.Wesley.Chun.2006 web programming.pdf - Foxit PDF Reader
File Home Comment View Form Protect Share Help
Hand Select SnapShot Clipboard Zoom Fit Page Reflow Rotate View Typewriter Highlight Link Bookmark File Attachment Image Annotation Audio & Video Fill & Sign
Advanced CGI Demo - Opera
File Edit View Bookmarks Tools Help
http://localhost:8000/cgi-bin/ad Document: 0 B
42 / 60
100%
Type here to search
29°C ENG 15:02 09-02-2022

Advanced CGI Demo Form

My Cookie Setting

- CPPuser = (cookie has not been set yet)

Enter cookie value
Oh look mom, a cookie (*optional*)

Enter your name
steven alistair kirk (*required*)

What languages can you program in? (*at least one required*)

✓ Python ☐ PERL ✓ Java ☐ C++ ☐ PHP ✓ C ☐ JavaScript

Enter file to upload

"C:\comments.txt" Choose

Submit

The data are submitted to the server using multipart encoding and retrieved in the same manner on the server side using the **FieldStorage instance.**

The only tricky part is in retrieving the uploaded file. In our application, we choose to iterate over the file, reading it line by line.

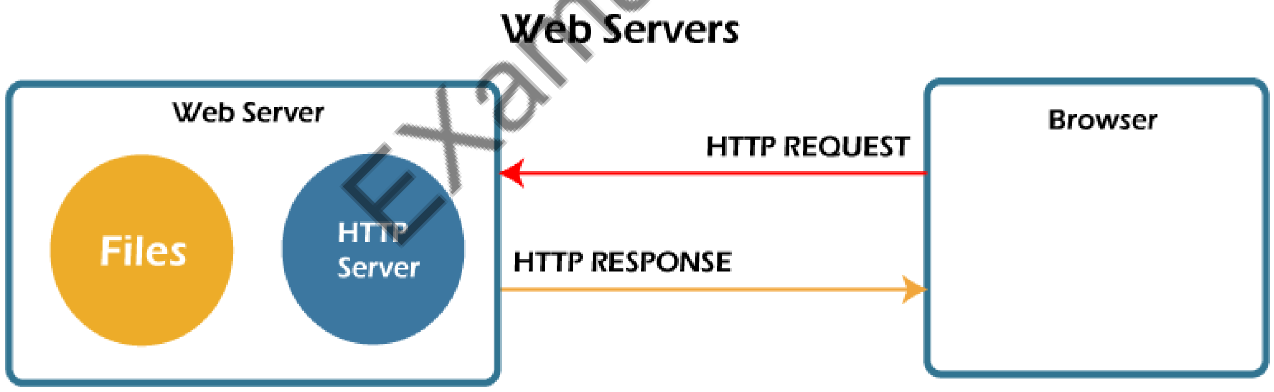
It is also possible to read in the entire contents of the file if you are not wary of its size.

Q: Explain about web servers in python

Web (HTTP) Servers

Web pages are a collection of data, including images, text files, hyperlinks, database files etc., all located on some computer (also known as server space) on the Internet. A web server is dedicated software that runs on the server-side.

When any user requests their web browser to run any web page, the web server places all the data materials together into an organized web page and forwards them back to the web browser with the help of the Internet.



This intercommunication of a web server with a web browser is done with the help of a protocol named **HTTP (Hypertext Transfer Protocol).**
These stored web pages mostly use static content, containing HTML documents, images, style sheets, text files, etc. However, **web servers can serve static as well as dynamic contents**.
Web Servers also assists in emailing services and storing files. Therefore it also uses SMTP (Simple Mail Transfer Protocol)
and FTP (File Transfer Protocol)

protocols to support the respective services. Web servers are mainly used in web hosting or hosting the website's data and running web-based applications.

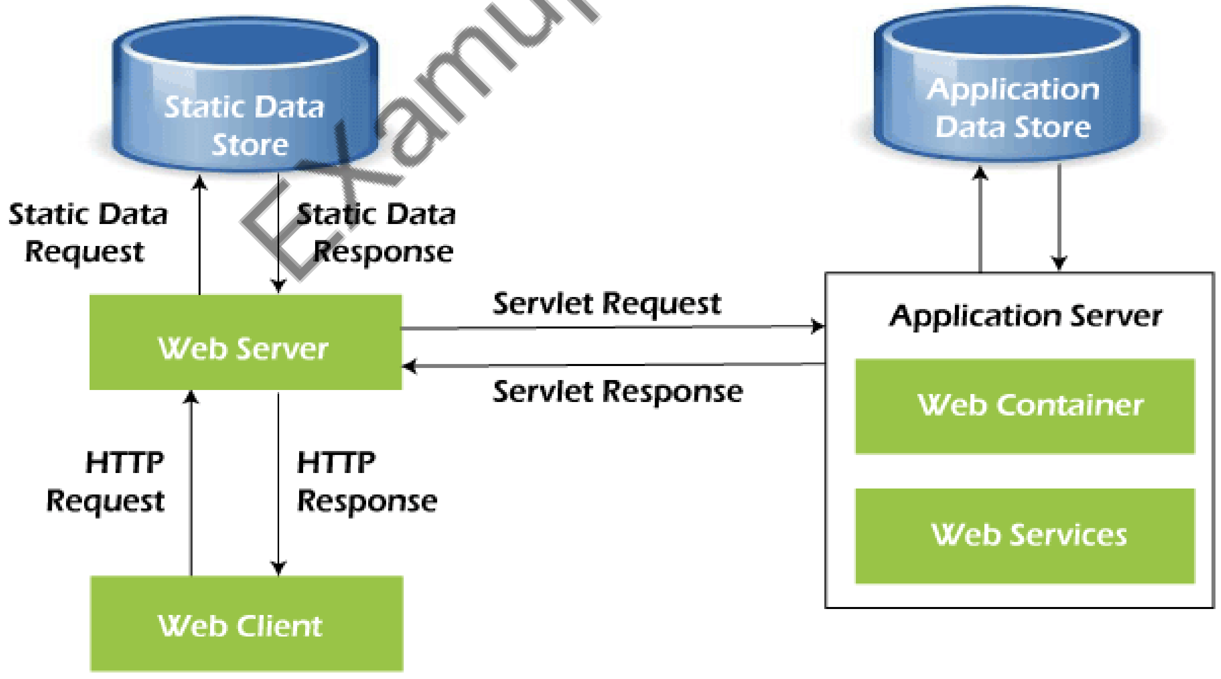
The hardware of the web servers are connected to the Internet that manages thedata exchange facility within different connected devices. In contrast, the software

of web server software is responsible for controlling how a user accesses delivered files. Typically, web server management is an ideal example of the client/server model. Therefore, **it is compulsory for all computers that host websites (whether with state or dynamic web page content) to have web server software.**

How do web servers work?

The term web server can denote server hardware or server software, or in most cases, both hardware and software might be working together.

Working of web servers



On the hardware side, a web server is defined as a computer that stores software and another website raw data, such as HTML files, images, text documents, and JavaScript files. The hardware of the web servers are connected to the web and supports the data exchange with different devices connected to the Internet.

On the software side, a web server includes server software accessed through website domain names. It controls how web users access the web files and ensures the supply of website content to the end-user. The web server contains several components, including an HTTP server.

Examples of web server uses

Web servers are mostly used for:

- sending and receiving mails on Internet by using SMTP (Simple Mail transfer Protocol);
- fetching requests for File Transfer Protocol (FTP) files; and
- designing, developing, and publishing websites.

Many Web servers, even the basic one, also support the server-side scripting technique. Server-side scripting is a web development method used to employ scripts on a web server that produces a customized response for each user. This technique operates on the server machine and consists of an extensive feature set, including database access. The server-side scripting process will have various scripting languages such ASP

, PHP
, Java
, JavaScript
, Python
, ruby
and many more. This technique also enables the HTML files to be created dynamically.

Creating Web Servers in Python

Its role is to perform the necessary HTTP communication between client and server. The base server class is named **HTTPServer** and is found in the **BaseHTTPServer module**.

It the client request and returns the appropriate file, whether static or dynamically generated by CGI.

The complexity of the handler determines the complexity of your Web server. The Python standard library provides three different handlers.

The following are three modules and their classes

Module	**Description**
BaseHTTPServer	Provides the base Web server and base handler classes, HTTPServer and **BaseHTTPRequestHandler**, respectively
SimpleHTTPServer	Contains the **SimpleHTTPRequestHandler** class to perform GET and HEAD requests
CGIHTTPServer	Contains the **CGIHTTPRequestHandler** class to process POST requests and perform CGI execution

Q: explain different modules CGI (Common Gateway Interface)

Related Modules

These are a list of modules which you may use for Web development.

Module/Package	**Description**
Web Applications	
cgi	*Gets Common Gateway Interface (CGI) form data*
cgitb	Handles CGI tracebacks
htmllib	*Older HTML parser for simple HTML files;* HTML-Parser *class extends from* sgmllib.SGMLParser
HTMLparser	Newer non-SGML-based parser for HTML and XHTML
htmlentitydefs	*HTML general entity definitions*
Cookie	*Server-side cookies for HTTP state management*
cookielib	Cookie-handling classes for HTTP clients
webbrowser	Controller: launches Web documents in a browser
sgmllib	*Parses simple SGML files*
robotparser	Parses robots.txt files for URL "fetchability" analysis
httplib	Used to create HTTP clients
XML Processing	
xmllib	*(Outdated/deprecated) original simple XML parser*
xml	XML package featuring various parsers (some below)
xml.sax	Simple API for XML (SAX) SAX2-compliant XML parser
xml.dom	Document Object Model [DOM] XML parser
xml.etree	Tree-oriented XML parser based on the Element flexible container object

xml.parsers.expat	Interface to the non-validating Expat XML parser
xmlrpclib	Client support for XML Remote Procedure Call (RPC) via HTTP
XML Processing	
SimpleXMLRPCServer	Basic framework for Python XML-RPC servers
DocXMLRPCServer	Framework for self-documenting XML-RPC servers
Web Servers	
BaseHTTPServer	*Abstract class with which to develop Web servers*
SimpleHTTPServer	*Serve the simplest HTTP requests (HEAD and GET)*
CGIHTTPServer	*In addition to serving Web files like* SimpleHTTPServers, *can also process CGI (HTTP POST) requests*
wsgiref	Standard interface between Web servers and Python Web application
Mail Client Protocols	
poplib	Use to create POP3 clients
imaplib	Use to create IMAP4 clients
Mail and MIME Processing and Data Encoding Formats	
email	Package for managing e-mail messages, including MIME and other RFC2822-based message
mailbox	Classes for mailboxes of e-mail messages
mailcap	Parses mailcap files to obtain MIME application delegations
Internet Protocols	
httplib	Used to create HTTP clients
ftplib	*Used to create FTP (File Transfer Protocol) clients*
gopherlib	*Used to create Gopher clients*
telnetlib	*Used to create Telnet clients*
nntplib	*Used to create NNTP (Network News Transfer Protocol [Usenet]) clients*