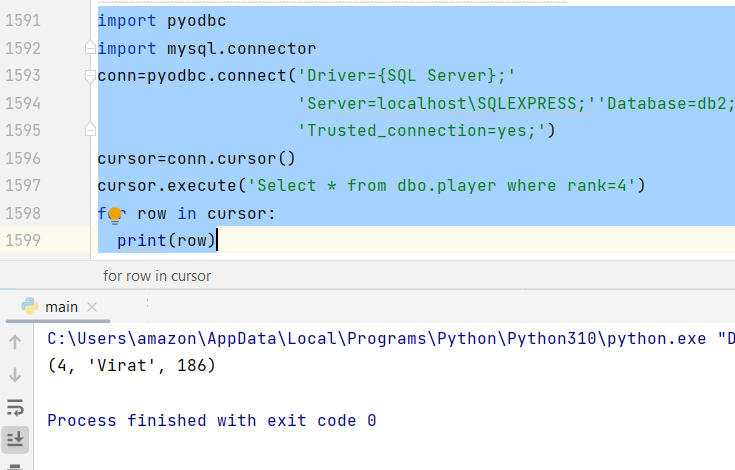
**Python Advanced 7**

import pyodbc  
import mysql.connector  
conn=pyodbc.connect('Driver={SQL Server};'  
 'Server=localhost\SQLEXPRESS;''Database=db2;'  
 'Trusted\_connection=yes;')  
cursor=conn.cursor()  
cursor.execute('Select \* from dbo.player where rank=4')  
for row in cursor:  
 print(row)



## Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

Import the *Tkinter* module.

Create the GUI application main window.

Add one or more of the above-mentioned widgets to the GUI application.

Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No. Operator & Description

1 Button

The Button widget is used to display buttons in your application.

2 Canvas

The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

3 Checkbutton

The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

4 Entry

The Entry widget is used to display a single-line text field for accepting values from a user.

5 Frame

The Frame widget is used as a container widget to organize other widgets.

6 Label

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7 Listbox

The Listbox widget is used to provide a list of options to a user.

8 Menubutton

The Menubutton widget is used to display menus in your application.

9 Menu

The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

10 Message

The Message widget is used to display multiline text fields for accepting values from a user.

11 Radiobutton

The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

12 Scale

The Scale widget is used to provide a slider widget.

13 Scrollbar

The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

14 Text

The Text widget is used to display text in multiple lines.

15 Toplevel

The Toplevel widget is used to provide a separate window container.

16 Spinbox

The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

17 PanedWindow

A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

18 LabelFrame

A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

19 tkMessageBox

This module is used to display message boxes in your applications.

Let us study these widgets in detail −

Standard attributes

Let us take a look at how some of their common attributes.such as sizes, colors and fonts are specified.

Dimensions

Colors

Fonts

Anchors

Relief styles

Bitmaps

Cursors

Let us study them briefly −

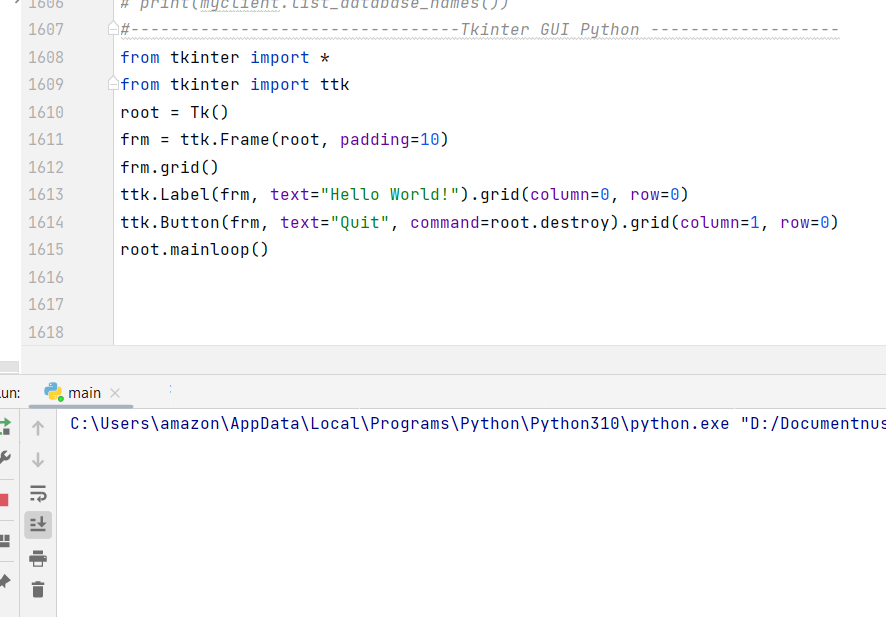
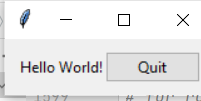
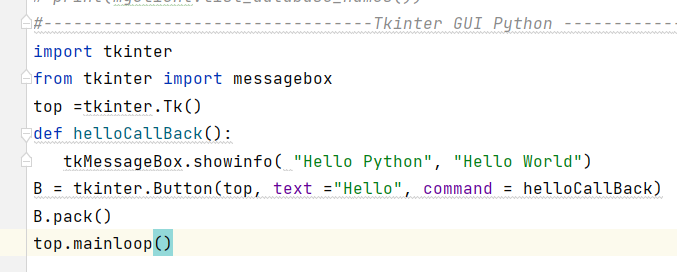
Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

The pack() Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.

The grid() Method − This geometry manager organizes widgets in a table-like structure in the parent widget.

The place() Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

from tkinter import \*  
from tkinter import ttk  
root = Tk()  
frm = ttk.Frame(root, padding=10)  
frm.grid()  
ttk.Label(frm, text="Hello World!").grid(column=0, row=0)  
ttk.Button(frm, text="Quit", command=root.destroy).grid(column=1, row=0)  
root.mainloop()

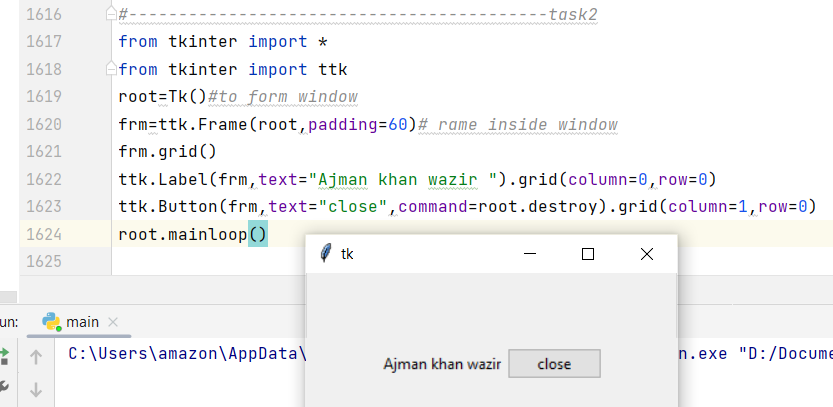
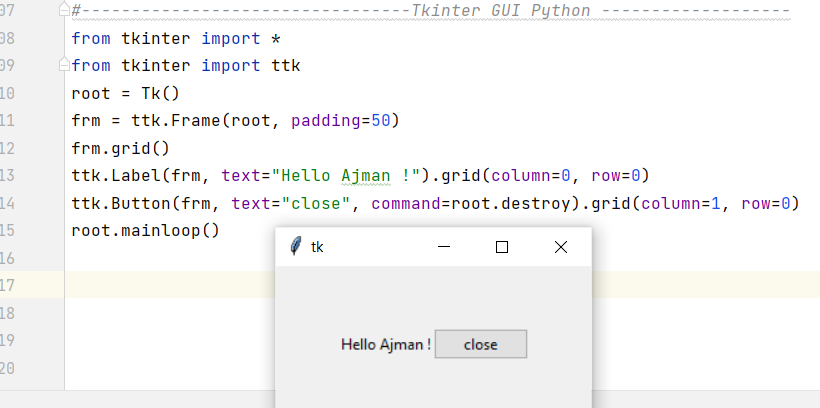
After the imports, the next line creates an instance of the [Tk](https://docs.python.org/3/library/tkinter.html" \l "tkinter.Tk" \o "tkinter.Tk) class, which initializes Tk and creates its associated Tcl interpreter. It also creates a toplevel window, known as the root window, which serves as the main window of the application.

The following line creates a frame widget, which in this case will contain a label and a button we’ll create next. The frame is fit inside the root window.

The next line creates a label widget holding a static text string. The grid() method is used to specify the relative layout (position) of the label within its containing frame widget, similar to how tables in HTML work.

A button widget is then created, and placed to the right of the label. When pressed, it will call the destroy() method of the root window.

Finally, the mainloop() method puts everything on the display, and responds to user input until the program terminates.



import tkinter as tk  
class App(tk.Frame):  
 def \_\_init\_\_(self, master):  
 super().\_\_init\_\_(master)  
 self.pack()  
 self.entrythingy = tk.Entry()  
 self.entrythingy.pack()  
 *# Create the application variable.* self.contents = tk.StringVar()  
 *# Set it to some value.* self.contents.set("this is a variable")  
 *# Tell the entry widget to watch this variable.* self.entrythingy["textvariable"] = self.contents  
 *# Define a callback for when the user hits return.  
 # It prints the current value of the variable.* self.entrythingy.bind('<Key-Return>',  
 self.print\_contents)  
 def print\_contents(self, event):  
 print("Hi. The current entry content is:",  
 self.contents.get())  
root = tk.Tk()  
myapp = App(root)  
myapp.mainloop()

