__del__(self) magic method (Destructor)

A class contains two methods

1. __init__(self) constructor
2. __del__(self) destructor

Constructor method is executed on creation of object.
Destructor method is executed on deletion of object.
These methods are called automatically on creation of object and deletion of object.
Constructor is for allocating resources to object.
Destructor is for de-allocating resources of object.

Example:

```
class Employee:
   def __init__(self):
      print("Inside Constructor")
      print("Object Created...")
   def __del__(self):
      print("Inside Destructor")
      print("Object Deleted")


def fun1():
   emp2=Employee()


emp1=Employee()
emp1=None
fun1()
```

Output

```
Inside Constructor
Object Created...
Inside Destructor
Object Deleted
Inside Constructor
Object Created...
Inside Destructor
Object Deleted
```

GUI Programming (Graphical User Interface)/Windows based Applications (Tkinter)

Types of user interfaces

1. CUI (Character User Interface or Command User Interface)
2. GUI (Graphical User Interface)

Interacting with application by typing commands is called CUI application.
Interacting with application by using Graphical component is called GUI application or windows based application or desktop application.

For developing GUI applications python provides number of libraries.

1. Tkinter
2. Kivy
3. Wxpython
4. PyQT

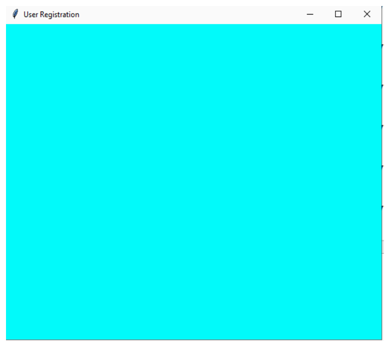
Tkinter stands for Tool Kit for Graphical User Interface. It is standard library which comes with python software.

Tkinter is library or package which provides set of widgets (Graphical Component). Every widget is one type or class.

Tk() class
This class or data type is used for creating window object.

Example

```
import tkinter

w=tkinter.Tk()
w.title("User Registration")
w['bg']='cyan'
w.geometry("600x500+200+200")
```

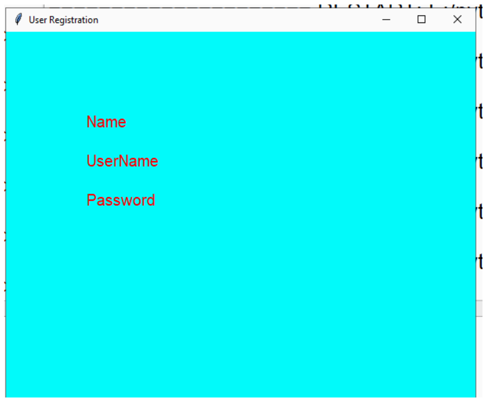


Label class or widget or data type
Label class is used for creating label object.
Label object represents display item. It is used to display data or information on window.

Syntax:
Label(window-object,properties)

```
import tkinter
```

```
w=tkinter.Tk()
w.title("User Registration")
w['bg']='cyan'
w.geometry("600x500+200+200")
l1=tkinter.Label(w,text="Name",font=("Arial",15),fg="red",bg="cyan")
l2=tkinter.Label(w,text="UserName",font=("Arial",15),fg="red",bg="cya
n")
l3=tkinter.Label(w,text="Password",font=("Arial",15),fg="red",bg="cyan"
)

l1.place(x=100,y=100)
l2.place(x=100,y=150)
l3.place(x=100,y=200)
```



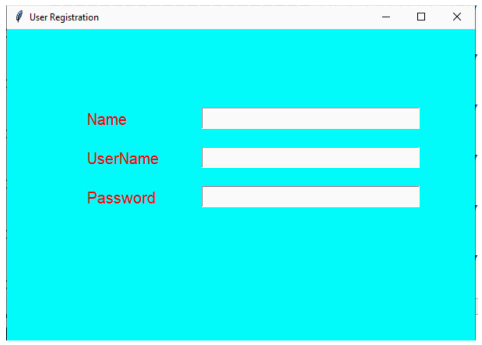
Entry class or Data type or widget

This class or data type is used to create entry field or input field. The data which is input inside entry field is type string.

Syntax: Entry(window-object, properties)

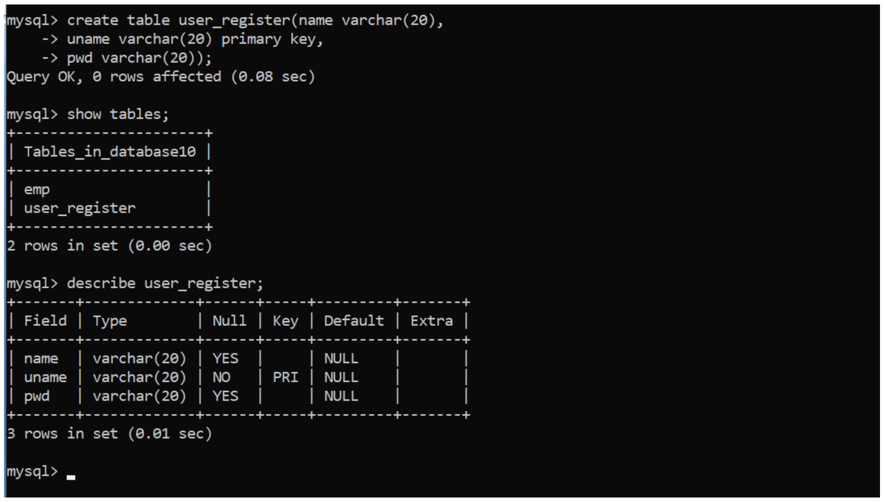
```
import tkinter

w=tkinter.Tk()
w.title("User Registration")
w['bg']='cyan'
w.geometry("600x500+200+200")
l1=tkinter.Label(w,text="Name",font=("Arial",15),fg="red",bg="cyan")
l2=tkinter.Label(w,text="UserName",font=("Arial",15),fg="red",bg="cya
n")
l3=tkinter.Label(w,text="Password",font=("Arial",15),fg="red",bg="cyan"
)
e1=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue")
e2=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue")
e3=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue",show='$')


l1.place(x=100,y=100)
l2.place(x=100,y=150)
l3.place(x=100,y=200)
e1.place(x=250,y=100)
e2.place(x=250,y=150)
e3.place(x=250,y=200)
```

Output



Button class or Widget or data type
This data type is used for creating button object. Button object is used to perform operations.

Syntax: Button(window-object,properties)



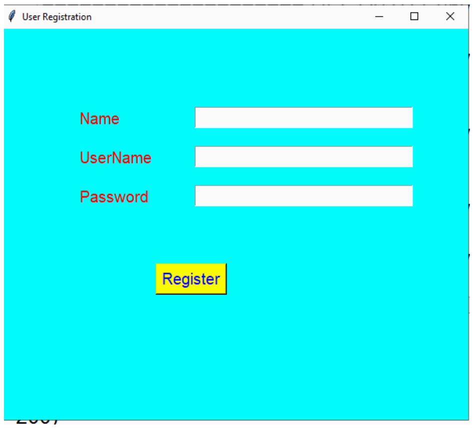
```
mysql> create table user_register(name varchar(20),
    -> uname varchar(20) primary key,
    -> pwd varchar(20));
Query OK, 0 rows affected (0.08 sec)

mysql> show tables;
+---------------------+
| Tables_in_database10 |
+---------------------+
| emp                 |
| user_register       |
+---------------------+
2 rows in set (0.00 sec)

mysql> describe user_register;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(20) | YES  |     | NULL    |       |
| uname | varchar(20) | NO   | PRI | NULL    |       |
| pwd   | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> _
```

Example
import tkinter
import mysql.connector

cn=mysql.connector.connect(database="database10",user="root",password="root")

w=tkinter.Tk()
w.title("User Registration")
w['bg']='cyan'
w.geometry("600x500+200+200")
l1=tkinter.Label(w,text="Name",font=("Arial",15),fg="red",bg="cyan")
l2=tkinter.Label(w,text="UserName",font=("Arial",15),fg="red",bg="cyan")

```
l3=tkinter.Label(w,text="Password",font=("Arial",15),fg="red",bg="cyan"
)
e1=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue")
e2=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue")
e3=tkinter.Entry(w,width=25,font=("Arial",15),fg="blue",show='$')

def reg():
    c=cn.cursor()
    name=e1.get()
    user=e2.get()
    pwd=e3.get()
    try:
        c.execute("insert into user_register
values(%s,%s,%s)",params=(name,user,pwd))
        print("user registered..")
        cn.commit()
    except:
        print("error in registering user")



b1=tkinter.Button(w,text="Register",font=("Arial",15),fg="blue",bg="yell
ow",command=reg)

l1.place(x=100,y=100)
l2.place(x=100,y=150)
l3.place(x=100,y=200)
e1.place(x=250,y=100)
e2.place(x=250,y=150)
e3.place(x=250,y=200)
b1.place(x=200,y=300)
```
Output


User Registration
Name
UserName
Password
Register