```
import tkinter as t
import mysql.connector
import tkinter.messagebox

cn=mysql.connector.connect(database="database10",user="root",p
assword="root")
window=t.Tk()
window.geometry("300x300+200+200")
window.title("Student System")

# Place Buttons
def add():
    awindow=t.Tk()
    awindow.geometry("200x200")
    awindow.title("Adding Student")

    l1=t.Label(awindow,text="Rollno",font=("Arial",15))
    l2=t.Label(awindow,text="Name",font=("Arial",15))
    l3=t.Label(awindow,text="Course",font=("Arial",15))
    l4=t.Label(awindow,text="Fee",font=("Arial",15))

    e1=t.Entry(awindow,width=10,font=("Arial",15))
    e2=t.Entry(awindow,width=10,font=("Arial",15))
    e3=t.Entry(awindow,width=10,font=("Arial",15))
    e4=t.Entry(awindow,width=10,font=("Arial",15))

    def add_student():
        c=cn.cursor()
        try:
            c.execute("insert into student
values(%s,%s,%s,%s)",params=(e1.get(),e2.get(),e3.get(),e4.get()))
            tkinter.messagebox.showinfo(message="Student Added")
```

```
            e1.delete(0,t.END)
            e2.delete(0,t.END)
            e3.delete(0,t.END)
            e4.delete(0,t.END)
            cn.commit()
            awindow.destroy()
        except:
            tkinter.messagebox.showerror(message="Unable to Add
Student")
    def close_window():
        awindow.destroy()


b1=t.Button(awindow,text="Add",font=("Arial",15),command=add_st
udent)

b2=t.Button(awindow,text="Exit",font=("Arial",15),command=close_wi
ndow)

    l1.grid(row=1,column=1)
    l2.grid(row=2,column=1)
    l3.grid(row=3,column=1)
    l4.grid(row=4,column=1)

    e1.grid(row=1,column=2)
    e2.grid(row=2,column=2)
    e3.grid(row=3,column=2)
    e4.grid(row=4,column=2)

    b1.grid(row=5,column=1)
    b2.grid(row=5,column=2)
```

```
def update():
    uwindow=t.Tk()
    uwindow.geometry("200x200")
    uwindow.title("Update Student")
    l1=t.Label(uwindow,text="Rollno",font=("Arial",15))
    l2=t.Label(uwindow,text="Course",font=("Arial",15))
    l3=t.Label(uwindow,text="Fee",font=("Arial",15))

    e1=t.Entry(uwindow,width=10,font=("Arial",15))
    e2=t.Entry(uwindow,width=10,font=("Arial",15))
    e3=t.Entry(uwindow,width=10,font=("Arial",15))
    def update_data():
        c=cn.cursor()
        c.execute("update student set course=%s,fee=%s where
rollno=%s",params=(e2.get(),e3.get(),e1.get()))
        k=c.rowcount
        if k>=1:
            tkinter.messagebox.showinfo(message="Student Details
Updated")
            cn.commit()
        else:
            tkinter.messagebox.showerror(message="Invalid Rollno")


b1=t.Button(uwindow,text="Update",font=("Arial",15),command=upd
ate_data)

b2=t.Button(uwindow,text="Exit",font=("Arial",15),command=lambda:
uwindow.destroy())

    l1.grid(row=1,column=1)
    l2.grid(row=2,column=1)
```

```
    l3.grid(row=3,column=1)
    e1.grid(row=1,column=2)
    e2.grid(row=2,column=2)
    e3.grid(row=3,column=2)
    b1.grid(row=4,column=1)
    b2.grid(row=4,column=2)
def remove():
    rwindow=t.Tk()
    rwindow.geometry("200x200")
    rwindow.title("Remove Student")
    l1=t.Label(rwindow,text="Rollno",font=("Arial",15))
    e1=t.Entry(rwindow,width=10,font=("Arial",15))
    def delete():
        c=cn.cursor()
        c.execute("delete from student where
rollno=%s",params=(e1.get(),))
        k=c.rowcount
        if k>=1:
            tkinter.messagebox.showinfo(message="Student Removed")
            cn.commit()
        else:
            tkinter.messagebox.showerror(message="Invalid Rollno")


b1=t.Button(rwindow,text="Delete",font=("Arial",15),command=delet
e)

b2=t.Button(rwindow,text="Exit",font=("Arial",15),command=lambda:r
window.destroy())

    l1.grid(row=1,column=1)
    e1.grid(row=1,column=2)
```

```
    b1.grid(row=2,column=1)
    b2.grid(row=2,column=2)

def display():
   dwindow=t.Tk()
   dwindow.geometry("200x200")
   dwindow.title("Display Details")
   l1=t.Label(dwindow,text="Rollno",font=("Arial",15))
   e1=t.Entry(dwindow,width=10,font=("Arial",15))
   def display_data():
      c=cn.cursor()
      c.execute("select course,fee from student where
rollno=%s",params=(e1.get(),))
      row=c.fetchone()
      if row!=None:
         tkinter.messagebox.showinfo(message=f'{row[0]}-->{row[1]}')
      else:
         tkinter.messagebox.showerror(message="Invalid Rollno")

b1=t.Button(dwindow,text="Display",font=("Arial",15),command=displ
ay_data)

b2=t.Button(dwindow,text="Exit",font=("Arial",15),command=lambda:
dwindow.destroy())

   l1.grid(row=1,column=1)
   e1.grid(row=1,column=2)
   b1.grid(row=2,column=1)
   b2.grid(row=2,column=2)
b1=t.Button(window,text="Add
Student",width=20,font=("Arial",15),command=add)
```

```
b2=t.Button(window,text="Update
Student",width=20,font=("Arial",15),command=update)
b3=t.Button(window,text="Remove
Student",width=20,font=("Arial",15),command=remove)
b4=t.Button(window,text="Display
Student",width=20,font=("Arial",15),command=display)

b1.pack()
b2.pack()
b3.pack()
b4.pack()
```

Python Logging using logging module

In application development we find different types of messages.

1. Information messages
2. Debugging messages
3. Error Messages
4. Warning Messages

5. Critical Messages

Disadvantage of displaying messages using print()

1. Does not have control on print statement
2. It is very complex maintain messages
3. Cannot find difference between one message with another message
4. It cannot send data on network or file or database

The above limitations can be overcome using logging.

Logging module provides various functions for implementation of logging for application or project.

Level	Numeric value	What it means / When to use it
`logging.NOTSET`	0	When set on a logger, indicates that ancestor loggers are to be consulted to determine the effective level. If that still resolves to `NOTSET`, then all events are logged. When set on a handler, all events are handled.
`logging.DEBUG`	10	Detailed information, typically only of interest to a developer trying to diagnose a problem.
`logging.INFO`	20	Confirmation that things are working as expected.
`logging.WARNING`	30	An indication that something unexpected happened, or that a problem might occur in the near future (e.g. 'disk space low'). The software is still working as expected.
`logging.ERROR`	40	Due to a more serious problem, the software has not been able to perform some function.
`logging.CRITICAL`	50	A serious error, indicating that the program itself may be unable to continue running.

1. logging.debug(message)
2. logging.info(message)
3. logging.warning(message)
4. logging.error(message)
5. logging.critical(message)

Example:

```
import logging

logging.debug("This is debug message")
logging.error("This is error message")
logging.warning("This is warning message")
logging.info("This is info message")
logging.critical("This is critical message")
```

Output

```
ERROR:root:This is error message
WARNING:root:This is warning message
CRITICAL:root:This is critical message
```

Example:

```
import logging


logging.basicConfig(filename="log1.txt",level=logging.DEBUG)
logging.debug("This is debug message")
logging.error("This is error message")
logging.warning("This is warning message")
logging.info("This is info message")
logging.critical("This is critical message")
```

Output

Log messages are stored in log1.txt

Example:

```
import logging
```

```
try:
    n1=int(input("Enter first number "))
    n2=int(input("Enter Second number"))
    n3=n1/n2
    print(n1,n2,n3)
except ZeroDivisionError:
    logging.error("cannot divide number with zero")
except ValueError:
    logging.error("input value must be integer type")
```

Output

```
Enter first number 5
Enter Second number0
ERROR:root:cannot divide number with zero
```

FAQ
Projects-2
Resume Formats
MongoDB--Python

4Months

Mail-id: pythonbygupta@gmail.com