Assignment No. 9

26/03/2023

Problem Statement: Python program to demonstrate MYSQL database connectivity with python. Create a GUI based application using widgets Entry, Label, Text, Button, RadioButton, CheckButton, ListBox, Menu, Spinbox (any five). Save the details in a database and read back from file on python prompt.

Theory:

MySQL DB: MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for MySQL however, is for the purpose of a web database. We will use mysql.connector library to establish a connection between Python project and MySQL workbench. Db is the object created using mysql.connector.connect class which stores all the information about databases such database name, password, and table name.

Python provides various libraries and modules for database connectivity. Here are some of the most used ones:

1. **SQLite3**: SQLite3 is a lightweight database system that comes with Python's standard library. It provides a simple way to connect to and manipulate a SQLite database.

Here's an example of how to create a connection to a SQLite database using Python:

```
import sqlite3 conn = sqlite3.connect('example.db')
```

2. **MySQL Connector**: MySQL Connector is a module that provides connectivity to MySQL databases. It can be installed using pip.

Here's an example of how to create a connection to a MySQL database using Python:

```
import mysql.connector
conn = mysql.connector.connect(
host="localhost",
user="yourusername",
password="yourpassword",
database="yourdatabase"
)
```

3. **psycopg2**: psycopg2 is a PostgreSQL database adapter for Python. It can be installed using pip.

Here's an example of how to create a connection to a PostgreSQL database using Python:

```
import psycopg2
conn = psycopg2.connect(
host="localhost",
user="yourusername",
password="yourpassword",
database="yourdatabase"
 )
```

Once you have created a connection, you can execute SQL statements using the **cursor** object.

```
cursor = conn.cursor()
cursor.execute("SELECT * FROM yourtable")
result = cursor.fetchall()
```

After you have executed the SQL statement, you can fetch the results using **fetchall()** method.

Note that you should always close the connection after you are done with it to release the resources.

```
conn.close()
```

PROGRAM CODE:

```
from getpass import getpass
from mysql.connector import connect, Error
global_id = 1
try:
        with connect(
                host="localhost",
                user=input("Enter username: "),
                password=getpass("Enter password: "),
                ) as connection:
                        print(connection)
                        except Error as e:
                                print(e)
import mysql.connector as m
# creating connection
conn = m.connect(user='python',password = 'python',host='localhost', database = "filedatabase")
# create cursor object
cursor = conn.cursor()
def create_database():
        # create new database
        create_db_query = "CREATE DATABASE filedatabase"
        # execute query
        cursor.execute(create_db_query)
        conn.commit()
def create_database_table():
        create_table_query = """ CREATE TABLE details (id INT PRIMARY KEY, text VARCHAR(100))"""
        cursor.execute(create_table_query)
        conn.commit()
def print_tables_in_database():
        show_tables_query = "SHOW TABLES"
        cursor.execute(show_tables_query)
        print("\n\nSHOW TABLES")
        print("-" * 50)
        for row in cursor.fetchall():
                print(row)
```

```
        conn.commit()

def insert_text_details(text2save, title2save):
        global global_id
        global_id += 1
        print(text2save, title2save)
        create_insert_query= " INSERT INTO details VALUES (%s, %s, %s) "
        records = [(global_id, text2save, title2save)]
        cursor.executemany(create_insert_query, records)
        # cursor.execute(create_insert_query)
        print("Values INSERTED in TABLE details")
        conn.commit()
def alter_table():
        create_alter_query = "ALTER TABLE details ADD COLUMN title VARCHAR (30)" #
        Adding a column in the table
        cursor.execute(create_alter_query)
        print("ALTER TABLE details: Added column Title")
        conn.commit()
def delete_records_with_id():
        create_update_query="""DELETE FROM details WHERE id >= 1 """
        cursor.execute(create_update_query)
        conn.commit()
        return messagebox.showinfo("DELETED", "Records Deleted Successfully")
def describe_table_on_terminal():
        create_desc_query=""" DESCRIBE details"""
        cursor.execute(create_desc_query)
        print("\n\nDESCRIBE TABLE details")
        print("-" * 50)
        for row in cursor.fetchall():
                print(row)
                conn.commit()
def print_records_on_terminal():
        describe_table_on_terminal()
        create_select_query=""" SELECT * FROM details """
        cursor.execute(create_select_query)
        print("\n\nPrinting TABLE details")
        print("-" * 100)
for row in cursor.fetchall():
        print(row)
        conn.commit()
        # create_database()
        # create_database_table()
        # print_tables_in_database()
        # describe_table_on_terminal()
        # alter_table()
        # delete_records_with_id(1)
        # print_records_on_terminal()
        conn.commit()
```

```
        from tkinter import *
        from tkinter import messagebox, filedialog
        # Functions
def save_file():
        '''saves the file in a .txt format'''
        text2save = str(text.get(1.0, END))
        title = name_var.get()
        print(text2save, title)
        insert_text_details(text2save, title)
        # Shows info box if file saved;
        return messagebox.showinfo("Saved", "File Saved to Database Successfully")
def clear_file():
        '''clears the input field '''
        text.delete(1.0, END)
def print_in_cmd():
        '''prints the file in terminal'''
        print_records_on_terminal()
        # Shows info box if file printed in the terminal;
        return messagebox.showinfo("Printed", "Records Printed Successfully")
def donothing():
        pass
        # create root window
root = Tk()
# root window title and dimension
root.title("Welcome to File Editor")
# Set geometry (widthxheight)
root.geometry('1200x550')
name_var=StringVar()
# creating a label for TITLE using widget Label
Label(root, text = 'TITLE', font=('calibre',14, 'bold')).place(x=30, y=70)
Entry(root, textvariable = name_var, font=('calibre',13,'normal')).place(x=30, y=110)
# Text Input
text=Text(root, x=30, y=40, padx = 10, pady = 10, wrap=WORD)
text.pack()
# adding a label to the root window
lbl = Label(root, text = "File Editor")
# Text to display where to enter text
Label(root, text="Enter Text: ", font=('calibre', 15, 'bold')).place(x=180, y=220, anchor="center")
# button widget with blue color text inside
btn = Button(root, text = "Save", fg = "blue", command=save_file)
btn.place(x = 350, y = 450)
# button widget with blue color text inside
btn = Button(root, text = "Print", fg = "blue", command=print_in_cmd)
btn.place(x = 475, y = 450)
# button widget with blue color text inside
btn = Button(root, text = "Alter", fg = "blue", command=alter_table)
btn.place(x = 600, y = 450)
# button widget with blue color text inside
```

```
btn = Button(root, text = "Delete Records", fg = "red",
command=delete_records_with_id)
btn.place(x = 725, y = 450)
# menu bar at the top
menubar=Menu(root)
filemenu=Menu(menubar,tearoff=0)
filemenu.add_command(label="New", command=clear_file)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=save_file)
filemenu.add_command(label="Save as...", command=save_file)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
# Adding cascade
menubar.add_cascade(label="File", menu=filemenu)
editmenu=Menu(menubar,tearoff=0)
editmenu.add_command(label="Undo", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
# Adding cascade
menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu=Menu(menubar,tearoff=0)
helpmenu.add_command(label="Help",command=donothing)
# Adding cascade
menubar.add_cascade(label="Help",menu=helpmenu)
# configuring the root menu
root.config(menu=menubar)
# Execute Tkinter
root.mainloop()
```

Output :

```
Values INSERTED in TABLE details
Values INSERTED in TABLE details
Values INSERTED in TABLE details

SHOW TABLES
```

```
1 firstdata
2 seconddata
3 thirddata
```