**Ex no 4 Implementation of Event Driven Programming Paradigm**

**Aim:**

To implement mouse click and keyboard click events in python.

**5a.Program to implement mouse click event**

**Algorithm:**

* Import the package tkinter
* Define the right click event
* Print the right click event detail.
* Define the left click event
* Print the left click event detail.
* Define the middle click event
* Print the middle click event detail.
* Binding the event with buttons
* Create object and run main loop.
* Stop

**Program:**

import tkinter as tk

from tkinter import \*

root=tk.Tk()

def rightclick(event):

print("rightclick")

def leftclick(event):

print("leftclick")

def middleclick(event):

print("middleclick")

frame=Frame(root,width=300,height=200)

frame.bind('<Button-1>',leftclick)

frame.bind('<Button-2>',middleclick)

frame.bind('<Button-3>',rightclick)

frame.pack()

root.mainloop()

**Output:**

**OUTPUT :**

leftclick

rightclick

leftclick

rightclick

leftclick

leftclick

**4b.Program to implement mouse events**

**ALGORITHM :**

* Import tkinter package
* Create class App
* Initialize the constructor
* Set the frame with green color
* Bind button, double button and button release events
* Bind motion ,enter and leave events
* Create object and run main loop.
* Stop

**PROGRAM :**

from tkinter import \*

import tkinter as tk

class App(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

frame = tk.Frame(self, bg="green", height=100, width=100)

frame.bind("<Button-1>", self.print\_event)

frame.bind("<Double-Button-1>", self.print\_event)

frame.bind("<ButtonRelease-1>", self.print\_event)

frame.bind("<B1-Motion>", self.print\_event)

frame.bind("<Enter>", self.print\_event)

frame.bind("<Leave>", self.print\_event)

frame.pack(padx=50, pady=50)

def print\_event(self, event):

position = "(x={}, y={})".format(event.x, event.y)

print(event.type, "event", position)

if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.mainloop()

**OUTPUT :**

Enter event (x=86, y=1)

Leave event (x=46, y=100)

Enter event (x=48, y=95)

Leave event (x=7, y=100)

Enter event (x=50, y=98)

Leave event (x=47, y=103)

Enter event (x=56, y=95)

Leave event (x=115, y=122)

**4c. Program to capture keyboard events**

**ALGORITHM :**

* Import tkinter package
* Define Key press and click events
* Bind the keypress and button events
* Create object and run main loop.
* Stop

**PROGRAM :**

from tkinter import \*

import tkinter as tk

import tkinter as event

root = Tk()

def key(event):

print("pressed", repr(event.char))

def callback(event):

frame.focus\_set()

print("clicked at", event.x, event.y)

frame = Frame(root, width=100, height=100)

frame.bind("<Key>", key)

frame.bind("<Button-1>", callback)

frame.pack()

root.mainloop()

**OUTPUT :**

clicked at 61 21

pressed 'w'

pressed 'w'

pressed 'b'

pressed 'b'

pressed 'b'

pressed 'b'

pressed 'b'

clicked at 47 54

clicked at 47 54

clicked at 47 54

**4d**. Program to implement the keypress event

**ALGORITHM :**

* Import tkinter package
* Define class App
* Define focus in and key events
* Bind the events
* Create object and run main loop.
* Stop

**PROGRAM :**

import tkinter as tk

class App(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

entry = tk.Entry(self)

entry.bind("<FocusIn>", self.print\_type)

entry.bind("<Key>", self.print\_key)

entry.pack(padx=20, pady=20)

def print\_type(self, event):

print(event.type)

def print\_key(self, event):

args = event.keysym, event.keycode, event.char

print("Symbol: {}, Code: {}, Char: {}".format(\*args))

if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.mainloop()

**OUTPUT :**

FocusIn

Symbol: s, Code: 83, Char: sty

Symbol: d, Code: 68, Char: d

Symbol: a, Code: 65, Char: a

Symbol: Caps\_Lock, Code: 20, Char:

Symbol: Caps\_Lock, Code: 20, Char:

Symbol: v, Code: 86, Char: v

Symbol: c, Code: 67, Char: c

**RESULT:**

Thus the Python program to implement various mouse click and keyboard events have been executed successfully

**Ex no: 5** **Implementation on Declarative Programming Paradigm**

**Aim:**

To implement Declarative programming in python

6a. **To demonstrate table creation and perform insert with SQL Using Python code (SQLite)**

**Algorithm:**

* Start the program
* Connect to the database using connect function in sqlite3.
* Create the cursor object
* Create the table
* Insert the records into the table
* Display the data from table
* Stop

**PROGRAM:**

#import the required packages

import sqlite3

#create a connection

con = sqlite3.connect('Students.db')

#create a cursor object

c=con.cursor()

#Create a table:

c.execute('''CREATE TABLE student(roll\_no INTEGER,name TEXT,age INTEGER);''')

#now to insert data:

c.execute('''INSERT INTO student VALUES(49,'Aman Bhai Patel',19)''')

#commit the changes to the database

con.commit()

#to see the data

for row in c.execute('''SELECT \* FROM student'''):

print(row)

Output



6b. Create two table using SQlite and have id as one of the column attribute and compare id with two table and if the same id is present in the two tables, print all the following information present with the id in two tables one by one.

**Algorithm:**

1. Start the program
2. Connect to the database using connect function in sqlite3.
3. Create two tables
4. Insert the records into the table
5. Display the data from table
6. Select needed field using where clause
7. Stop

PROGRAM:

# Import required packages

import sqlite3

# create a connection and cursor

con = sqlite3.connect("file3.db")

c = con.cursor()

# create the 2 tables and insert some random data

c.execute('''CREATE TABLE t1(id INTEGER,name TEXT);''')

c.execute('''CREATE TABLE t2(id INTEGER,job TEXT);''')

c.execute('''INSERT INTO t1 VALUES(1,'Aman'),(2,'Aviraj'),(1,'Nithish'),(2,'Venkat');''')

c.execute('''INSERT INTO t2 VALUES(1,'Job1'),(2,'Job2');''')

# commit changes

con.commit()

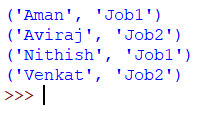
# send the select command to the sqlite3 backend:

task = '''SELECT t1.name,t2.job FROM t1,t2 WHERE t1.id = t2.id;'''

for row in c.execute(task):

print(row)

Output:



6c. Create a table and perform select and insert, delete, update, modify with SQL Using Python code (SQLite)

**Algorithm:**

* Start the program
* Connect to the database using connect function in sqlite3.
* Create the company table
* Insert the records into the table
* Display the data from table
* Display the all columns from the table
* Update the table
* Delete the records from the database
* Stop

**Program:**

# import required packages

import sqlite3

# Create a connection anf cursor

con = sqlite3.connect('file4.db')

c = con.cursor()

# create a function to print the whole table:

def printall():

global c

for row in c.execute("SELECT \* FROM datatable"):

print(row)

# Create a sample table and insert data into it:

c.execute("CREATE TABLE datatable(ID INTEGER,Name TEXT);")

# INSERT

namelist = [(1,'Aman'),(2,'Aviraj'),(3,'Venkat')]

c.executemany("INSERT INTO datatable VALUES(?,?)",namelist)

con.commit()

print("\nInitial table: ")

printall()

# DELETE

print("\nDeleting an enrty : (3,Venkat)")

c.execute("DELETE FROM datatable WHERE Name = 'Venkat';")

print("\nTable is now: ")

printall()

# UPDATE

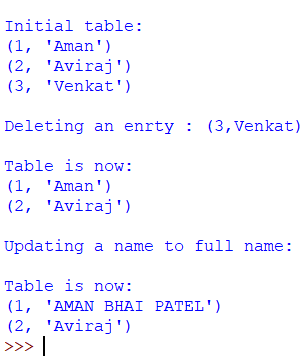
print("\nUpdating a name to full name:")

print("\nTable is now: ")

c.execute("UPDATE datatable SET Name='AMAN BHAI PATEL' WHERE ID=1;")

printall()

Output:



5d. Write a Python Program to create a company table that consists of name, age,address and salary attributes using sqlite3 and to insert five records into the table and to perform select, display, update and delete operations.

**ALGORITHM:**

* Start the program
* Connect to the database using connect function in sqlite3.
* Create the company table
* Insert the records into the table
* Display the data from table
* Display the all columns from the table
* Update the table
* Delete the records from the database
* Stop

**PROGRAM:**

import sqlite3

# connect to a database

conn = sqlite3.connect('test.db')

print ("Opened database successfully");

# To Create a table

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute('''CREATE TABLE COMPANY (ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL);''')

print("Table created successfully")

conn.close()

# To insert records into a table

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");

conn.commit()

print ("Records created successfully")

conn.close()

# To display the data from the table

import sqlite3

conn = sqlite3.connect('test.db')

print("Opened database successfully")

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

print ("ADDRESS = ", row[2])

print ("SALARY = ", row[3], "\n")

print("Operation done successfully")

conn.close()

# To display all columns from a database

import sqlite3

conn = sqlite3.connect('test.db')

print("Opened database successfully")

conn.execute('''CREATE TABLE COMPANY12345

(ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL);''')

print("Table created successfully")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY)\

VALUES (1, 'Paul', 32, 'California', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (2, 'Allen', 25, 'Texas', 15000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )")

conn.commit()

print("Records created successfully")

cursor = conn.execute("SELECT ID,NAME,AGE,ADDRESS,SALARY from COMPANY12345")

for row in cursor:

print("ID = ", row[0])

print("NAME = ", row[1])

print("AGE=", row[2])

print("ADDRESS = ", row[3])

print("SALARY = ", row[4])

print("Operation done successfully");

conn.close()

# To update the table

import sqlite3

conn = sqlite3.connect('test.db')

print("Opened database successfully")

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")

conn.commit()

print ("Total number of rows updated :", conn.total\_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

print ("ADDRESS = ", row[2])

print ("SALARY = ", row[3], "\n")

print("Operation done successfully")

conn.close()

# To perform delete operation

import sqlite3

conn = sqlite3.connect('test.db')

print( "Opened database successfully")

conn.execute("DELETE from COMPANY where ID = 2;")

conn.commit()

print ("Total number of rows deleted :", conn.total\_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

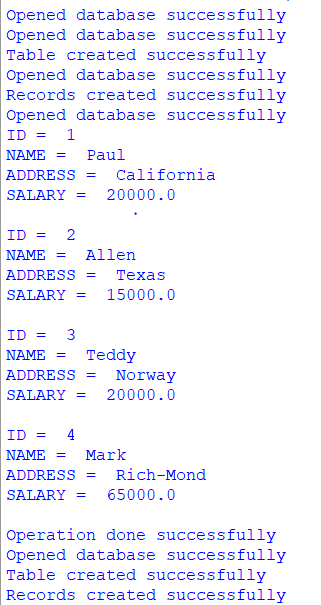
print ("ADDRESS = ", row[2])

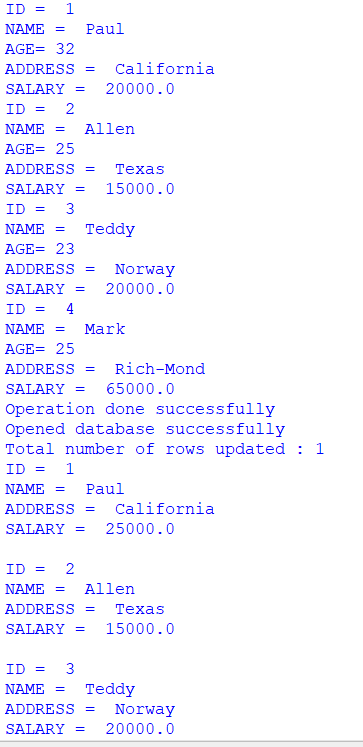
print ("SALARY = ", row[3], "\n")

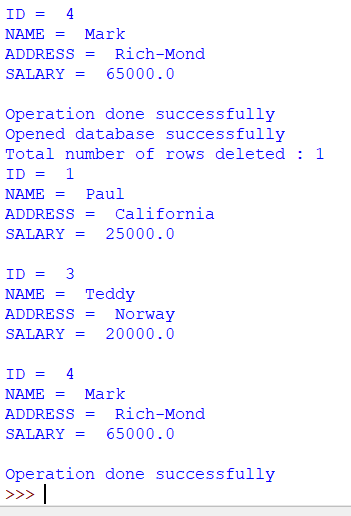
print ("Operation done successfully");

conn.close()

Output:







**Result:**

Thus the Python program to implement table creation and insertion with SQL, to implement creation of two tables and print the details depending upon particular field, to perform delete, update, modify are executed successfully.

**Ex.No:6**

**Date:**

**IMPLEMENTATION OF IMPERATIVE PROGRAMMING PROGRAM**

**Questions:**

To Write a Python Programs to implement sequential search

.

To Write a python program to implement bubble sort.