KENDRIYA VIDYALAYA JHUNJHUNU

STUDENT NAME

Student Record Management

ABSTRACT

The Project is for Record Keeping Package for student. It includes Admission, Examination, Fees, Library and CCA modules.

Student name

CBSE AISSCE- 2019-20 , Python- MySQL Connectivity

Python Introduction

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Python Getting Started

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python –version
```

Python Comments
Comments can be used to explain Python code.
Comments can be used to make the code more readable.
Comments can be used to prevent execution when testing code.

Creating a Comment

```
#This is a comment
print("Hello, World!")

print("Hello, World!") #This is a comment
```

Multi Line Comments

Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line:

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Variable Names
A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

Python Data Types

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool

Python Strings

```
a = "Hello"
print(a)
```

Multiline Strings

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
```

```
ut labore et dolore magna aliqua."""
print(a)
```

String Format

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))

quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

String Methods

Method	Description
capitalize()	Converts the first character to upper case
Count()	Returns the number of times a specified value occurs in a string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet

isdigit()	Returns True if all characters in the string are digits
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isupper()	Returns True if all characters in the string are upper case
lower()	Converts a string into lower case
replace()	Returns a string where a specified value is replaced with a specified value
split()	Splits the string at the specified separator, and returns a list
strip()	Returns a trimmed version of the string
upper()	Converts a string into upper case

Python Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position

remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Tuples

A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Method	**Description**
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Dictionaries

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Method	**Description**

clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
get()	Returns the value of the specified key
items()	Returns a list containing the a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

What is PIP?

PIP is a package manager for Python packages.

Use the uninstall command to remove a package:

Python Try Except

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

```
try:
  print(x)
except:
  print("An exception occurred")

finally:
  print("The 'try except' is finished")
```

File Handling

The key function for working with files in Python is the open() function.

The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

```
f = open("demofile.txt")
```

Open a File on the Server

```
f = open("demofile.txt", "r")
print(f.read())
```

Read Lines

```
f = open("demofile.txt", "r")
print(f.readline())
```

Close Files

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Write to an Existing File

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file "w" - Write - will overwrite any existing content

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Delete a File

```
import os
os.remove("demofile.txt")

import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Delete Folder

```
import os
os.rmdir("myfolder")
```

MySQL Database

&

Python Connectivity

Install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

In this tutorial we will use the driver "MySQL Connector".

We recommend that you use PIP to install "MySQL Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector
```

Create Connection

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword"
)

print(mydb)
```

Creating a Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address
VARCHAR(255))")
```

Insert Into Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

Select From a Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

Update Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley
345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

CODING

```
# Databse , Table Ceration by using Python
import mysql.connector as cnt
import os
mydb=cnt.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="kvjjn"
    )
mycursor=mydb.cursor()
```

```
#if mydb.is_connected():

#    print("Connection Successfully Established")

#    print("Database= ",mydb)

#else:

#    print("No Connection Found")

# mycursor.execute("create database kvjjn")

#mycursor.execute("show databases")

#for i in mycursor:

#    print(i)

#mycursor.execute("use kvjjn")



#mycursor.execute(" create table stud ( sid int(2), name char(20))")

#mycursor.execute("CREATE TABLE teach (id int(2) prIMARY KEY, name
VARCHAR(20), address VARCHAR(20))")



#mycursor.execute("show tables")

#for i in mycursor:

#    print(i)

#mycursor.execute("desc stud")

#for i in mycursor:

#    print(i)

#sql = "INSERT INTO stud VALUES (102, 'Mohan')"

#mycursor.execute(sql)

#mydb.commit()
```

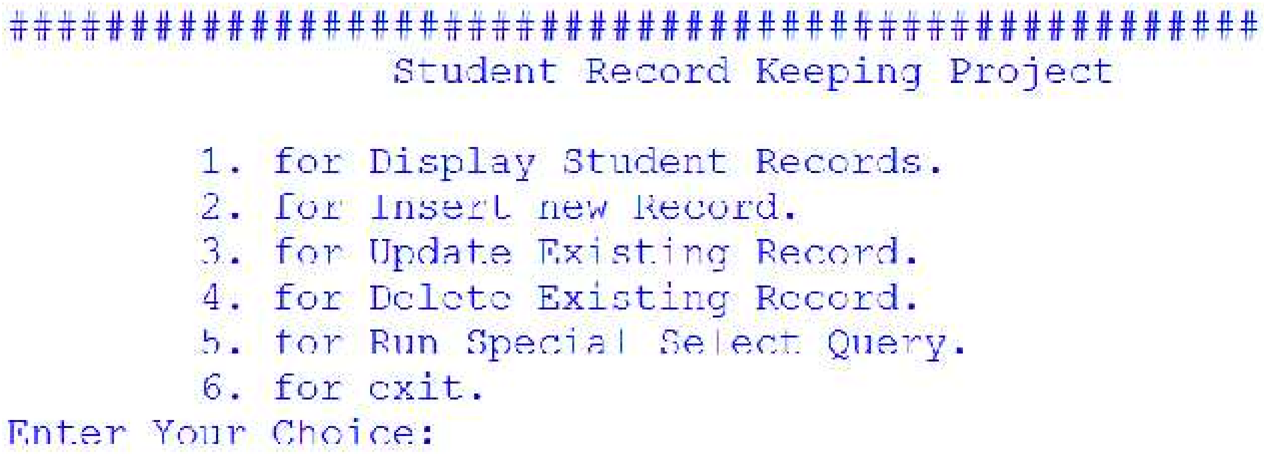
```
#print(mycursor.rowcount, "record inserted.")

def display():

    print("Table Name: stud")

    print("\nSID\tName")

    print("---------------")

    mycursor.execute("select * from stud")

    for i in mycursor:

        print(i)

    print(mycursor.rowcount," Rows Selected")


def run_query():

    print("\nStructure of Table: stud")

    print("---------------")

    mycursor.execute("desc stud")

    for i in mycursor:

        print(i)

    sql=input("Kindly type your own query :")

    print(sql)

    a=input()

    mycursor.execute(sql)

    for i in mycursor:

        print(i)

    mydb.commit()
    print(mycursor.rowcount," Records Affected")



def insert():
    print("Table Name: stud")
    sn=int(input("Enter Student ID(numeric): "))
    nm=input("Enter Student Name(char): ")
    sql = "INSERT INTO stud VALUES(%d,'%s')"%(sn,nm)
    mycursor.execute(sql)
    mydb.commit()
    print(mycursor.rowcount, "record inserted.")
    choice=input("One More Record? ")
    if choice.upper()=="Y":
        insert()
def delete():
    print("Table Name: stud")
    id=int(input("Enter Student ID (Numeric) to delete: "))
    sql="delete from stud where sid=%d"%(id)
    mycursor.execute(sql)
    if mycursor.rowcount==0:
        print("Record Not found!")
    else:
        print(mycursor.rowcount, end="")
        choice=input(" Records found. Want to Delete?(Y/N): ")
        if choice.upper()=="Y":
            mydb.commit()
            print(mycursor.rowcount," Record(s) Deleted")
        else:
            mydb.rollback()
            print(mycursor.rowcount," Aborted by user, No Record(s) Deleted")


def update():
    print("Table Name: stud")
    id=int(input("Enter ID (numeric)to Update: "))
    nm=input("Enter new name (char)of student to update: ")
    sql="update stud set name='%s' where sid=%d"%(nm,id)
    mycursor.execute(sql)
    if mycursor.rowcount==0:
        print("Record Not found!")
    else:
        print(mycursor.rowcount, end="")
        choice=input(" Records found. Want to update?(Y/N): ")
        if choice.upper()=="Y":
            mydb.commit()
            print(mycursor.rowcount," Record(s) Update Successfully")
        else:
            mydb.rollback()

            print(mycursor.rowcount," Abortedby user, No Record(s) Updated")


def menu():

    try:

        print("####################################################")

        print("\t\tStudent Record Keeping Project")

        print("Table Name: stud")

        str="""

        1. for Display Student Records.

        2. for Insert new Record.

        3. for Update Existing Record.

        4. for Delete Existing Record.

        5. for Run Special Select Query.

        6. for exit."""

        print(str)

        choices=int(input("Enter Your Choice: "))

        if choices==1:

            display()

            a=input("Press any key to continue...")

            os.system('cls')

            menu()

        elif choices==2:
```

```
    insert()

    a=input("Press any key to continue...")

    os.system('cls')

    menu()

elif choices==3:

    print("Before Update")

    display()

    update()

    print("After Update")

    display()

    a=input("Press any key to continue...")

    os.system('cls')

    menu()

elif choices==4:

    delete()

    a=input("Press any key to continue...")

    os.system('cls')

    menu()

elif choices==5:

    run_query()

    a=input("Press any key to continue...")

    os.system('cls')

    menu()
```

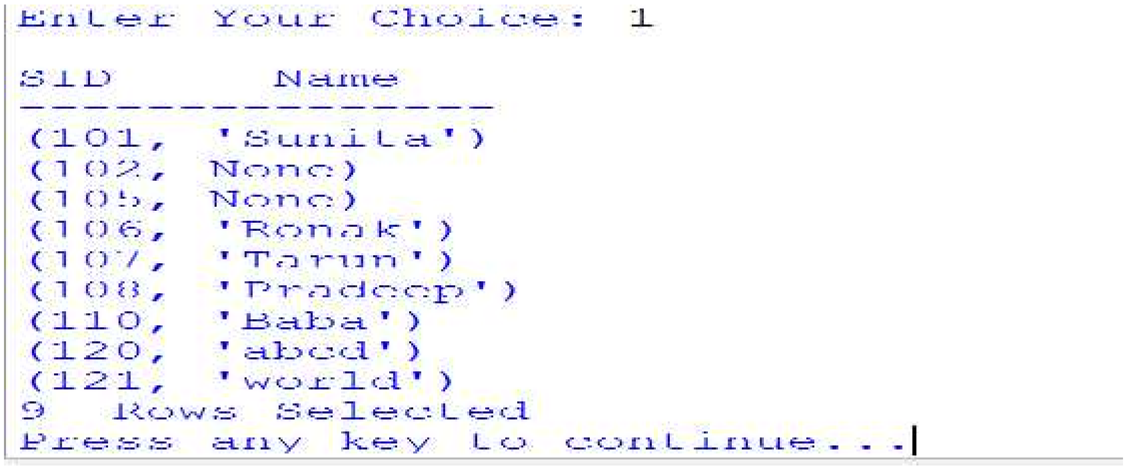
```
        elif choices==6:
            exit()
        else:
            print("Enter Correct Option:")
            a=input("Press any key to continue...")
            os.system('cls')
            menu()
    except:
        print("something went wrong: Crush Ending")
        exit()

menu()
```

OUTPUT

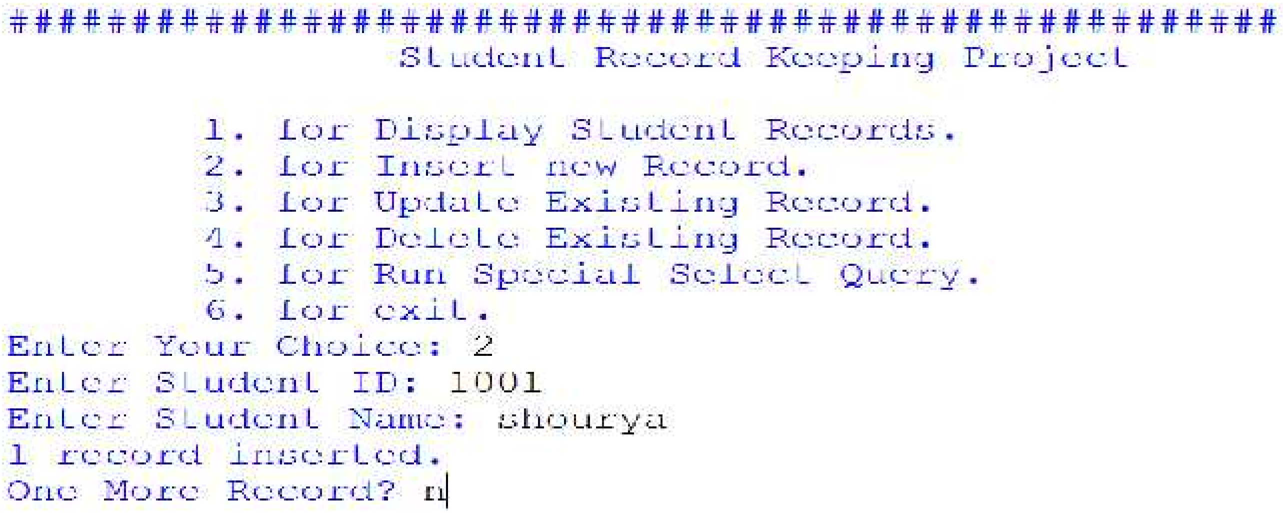
SCREENSHOTS



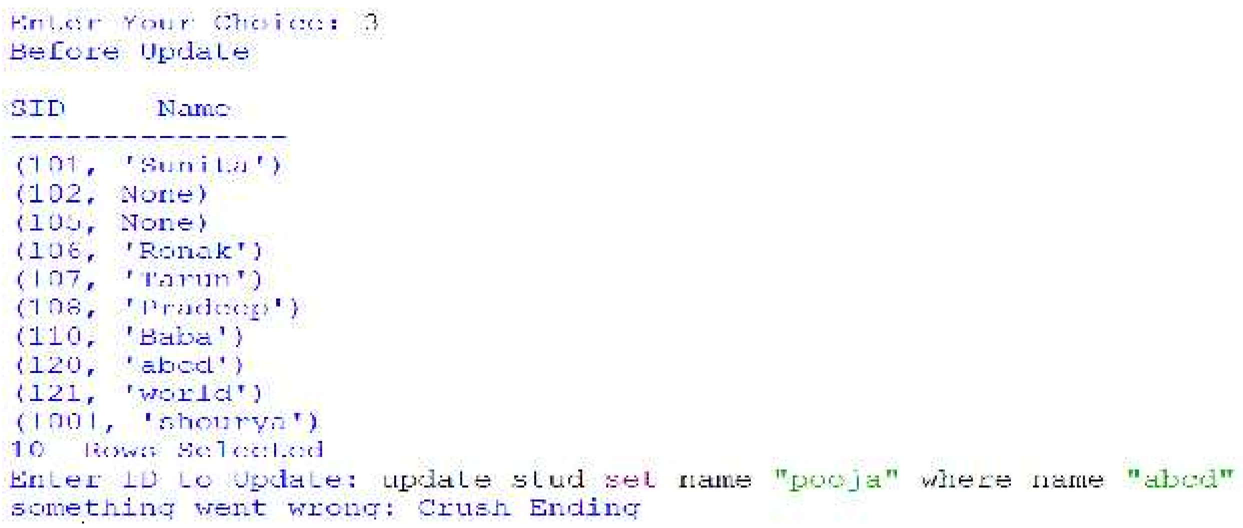
```
####################################################
                Student Record Keeping Project

        1. for Display Student Records.
        2. for Insert new Record.
        3. for Update Existing Record.
        4. for Delete Existing Record.
        5. for Run Special Select Query.
        6. for exit.
Enter Your Choice:
```



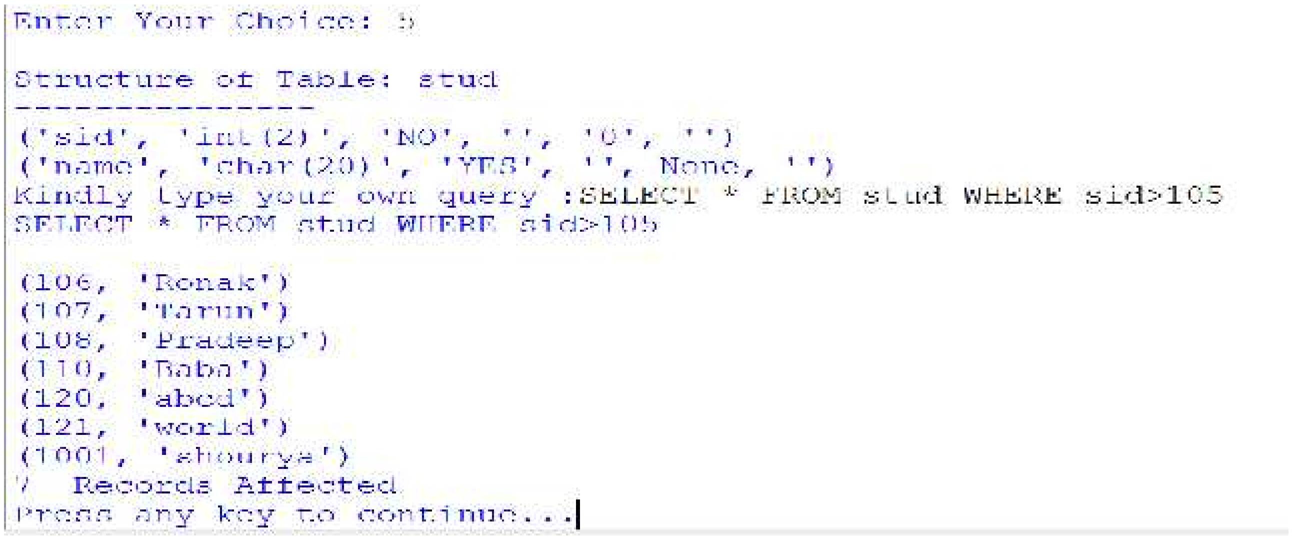
```
Enter Your Choice: 1

SID       Name
----------------
(101, 'Sunita')
(102, None)
(105, None)
(106, 'Ronak')
(107, 'Tarun')
(108, 'Pradeep')
(110, 'Baba')
(120, 'abcd')
(121, 'world')
9  Rows Selected
Press any key to continue...
```



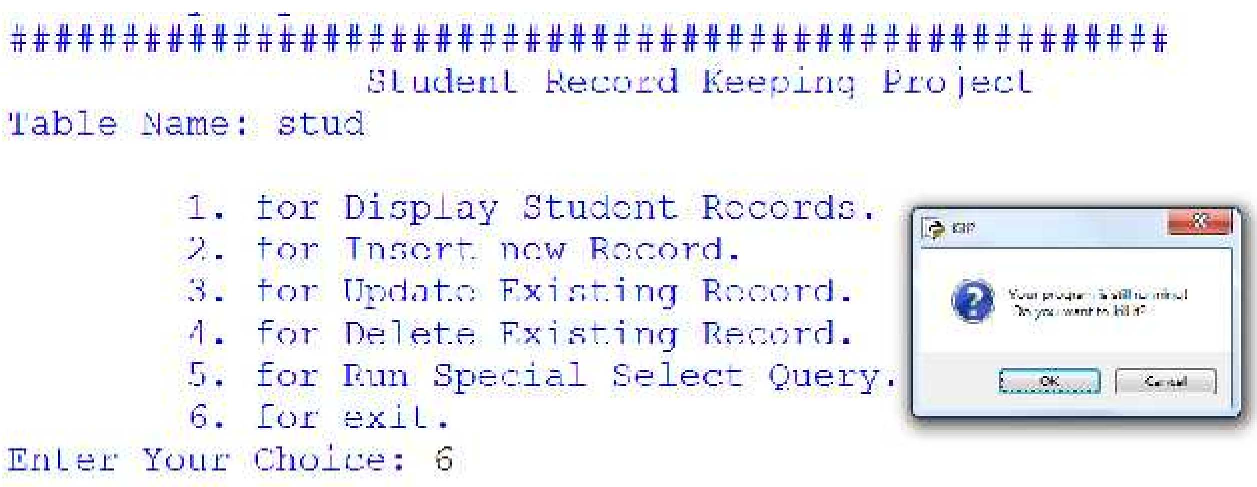
```
####################################################
                Student Record Keeping Project

        1. for Display Student Records.
        2. for Insert new Record.
        3. for Update Existing Record.
        4. for Delete Existing Record.
        5. for Run Special Select Query.
        6. for exit.
Enter Your Choice: 2
Enter Student ID: 1001
Enter Student Name: shourya
1 record inserted.
One More Record? n
```



```
Enter Your Choice: 3
Before Update

SID      Name
---------------
(101, 'Sunita')
(102, None)
(105, None)
(106, 'Ronak')
(107, 'Tarun')
(108, 'Pradeep')
(110, 'Baba')
(120, 'abcd')
(121, 'world')
(1001, 'shourya')
10  Rows Selected
Enter ID to Update: update stud set name "pooja" where name "abcd"
something went wrong: Crush Ending
```



```
Enter Your Choice: 5

Structure of Table: stud
----------------
('sid', 'int(2)', 'NO', '', '0', '')
('name', 'char(20)', 'YES', '', None, '')
Kindly type your own query :SELECT * FROM stud WHERE sid>105
SELECT * FROM stud WHERE sid>105

(106, 'Ronak')
(107, 'Tarun')
(108, 'Pradeep')
(110, 'Baba')
(120, 'abcd')
(121, 'world')
(1001, 'shourya')
7  Records Affected
Press any key to continue...
```



```
####################################################
                Student Record Keeping Project
Table Name: stud

        1. for Display Student Records.
        2. for Insert new Record.
        3. for Update Existing Record.
        4. for Delete Existing Record.
        5. for Run Special Select Query.
        6. for exit.
Enter Your Choice: 6
```

Your program is still running!
Do you want to kill it?

References.

- Sumita Arora Book
- Preeti Arora Book
- www.w3schools.com
- Internet

Credits.

- KV Jhunjhunu- Computer Lab.
- KV Jhunjhunu Staff
- Subject Teacher (…………………)
- Friends (…………………)
- Parents
- At last but not least, grace of God

(Signature of student)

Name of Student

CBSE Roll No.