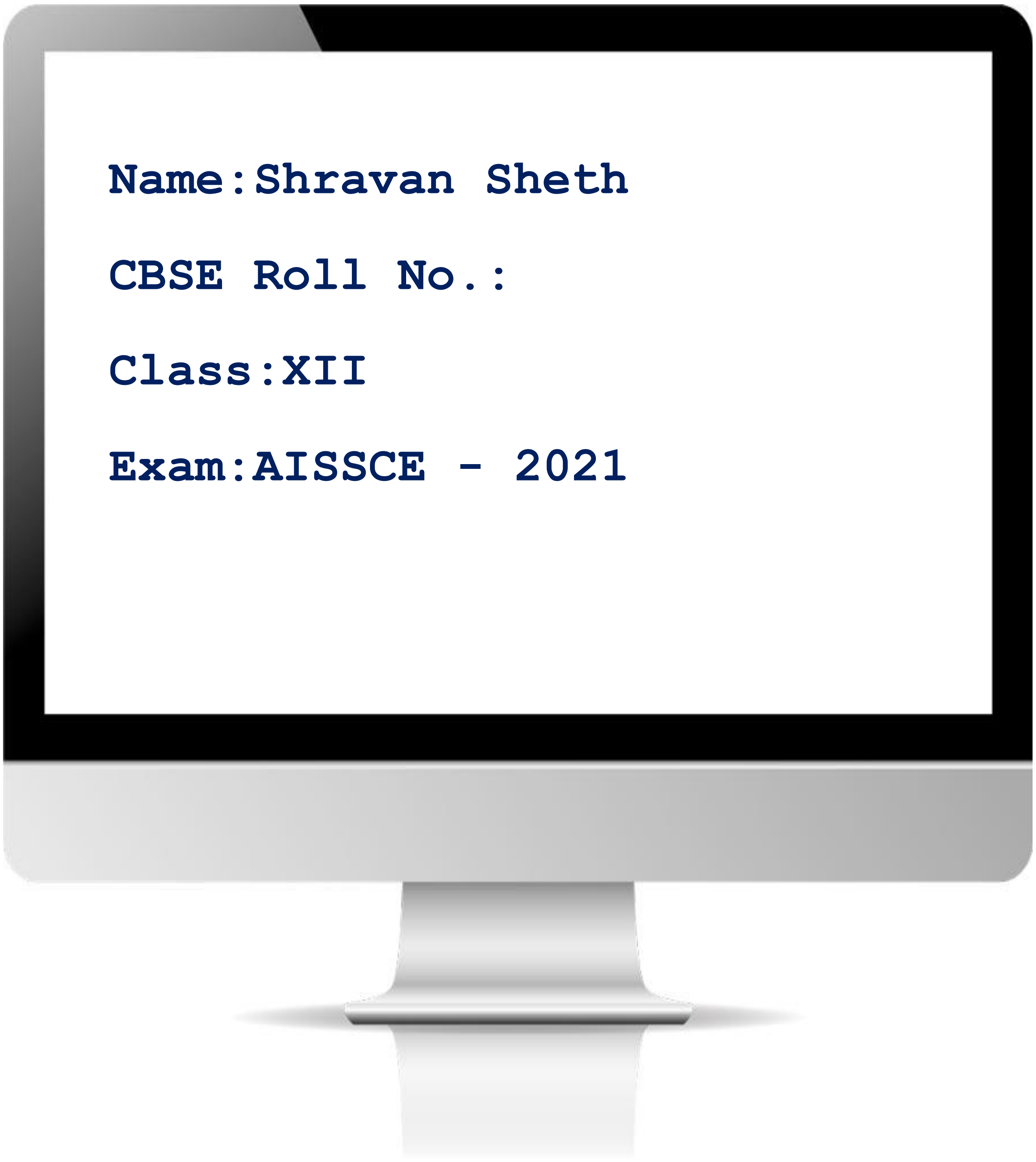
INDIRA NATIONAL SCHOOL

WAKAD,PUNE

ACADEMIC YEAR 2020-2021

COMPUTER SCIENCE PRACTICAL FILE



Name:Shravan Sheth

CBSE Roll No.:

Class:XII

Exam:AISSCE - 2021

INDEX

	isHAppy() : checks if the given number is a happy number by calling the function Sum_sq_digits() and displays an appropriate message.			
S.No	**Programs**	**Date of Program**	**Date of Submission**	**Signature**
10	Write a program to count 'He' or 'His' present in the text file 'story.txt'	25-Sep-2020	28-Sep-2020	
11	Write a program to count and display the number of lines starting with alphabet 'W' present in the text file "poem.txt"	25-Sep-2020	28-Sep-2020	
12	A file 'sports.dat' contains information in following format: Event-Participant Define a function Athletic_display() that would read contents from file sports.dat and creates a file named Athletic.dat copying only those records from sports.dat where the event name is 'Athletics' Write a program to invoke Athletic_display() .	29-Sep-2020	30-Sep-2020	
13	Write a database connectivity program to fetch all the records from student table.	30-Sep-2020	1-Oct-2020	
14	Write a database connectivity program to insert a new student record to student table.	30-Sep-2020	1-Oct-2020	
15	Write a database connectivity program to delete a student record from student table.	06-Oct-2020	07-Oct-2020	
16	Write a database connectivity program to modify student record of a student table.	08-Oct-2020	09-Oct-2020	
17	Write a program to Search for an element in a list using Binary search technique.	26-Oct-2020	27-Oct-2020	
18	Write a program to insert an element into an ordered list.	26-Oct-2020	27-Oct-2020	
19	Write a program to delete an element from an ordered list.	26-Oct-2020	27-Oct-2020	

20	Write a program to sort a list using Bubble sort technique.	26-Oct-2020	27-Oct-2020	
21	Write a program to sort a list using Insertion sort technique.	23-Nov-2020	24-Nov-2020	
22	Write a menu driven program to implement a stack.	25-Nov-2020	28-Nov-2020	
23	Write a menu driven program to implement a Queue.	25-Nov-2020	28-Nov-2020	

DATABASES AND SQL

S.No	Database	Date of Query	Date of Submission	Signature
1	SQL Queries on Employees and EmpSalary Tables.	20-Apr-2020	22-Apr-2020	
2	SQL Queries on Furniture and Arrivals Tables.	27-Apr-2020	29-Apr-2020	

INDIRA NATIONAL SCHOOL, PUNE

CERTIFICATE

This is to certify that Master Shravan Sheth

Studying in Indira National School,Pune of **Class XII** Science CBSE Examination Number:____________________

Has satisfactorily performed the practicals in Computer Science subject

as laid down by the CBSE Board during the academic year **2020 - 2021**.

________ ___________ _________

Principal’s External Internal
Signature Examiner Examiner

Date:
School Stamp:

PYTHON PROGRAMS

-Shravan Sheth

Program 1: Write a Program that reads a date as an integer in the format MMDDYYYY. The program will print the date in the format <Month Name> <day>, <year>

```
#Program to read date in MMDDYYYY format and print in Month
#Day, Year format

#Creating dictionary to hold months
months = {1 : 'January', 2 : 'February', 3 : 'March',
          4 : 'April', 5 : 'May', 6 : 'June', 7 : 'July',
          8 : 'August', 9 : 'September', 10 :'October',
          11 : 'November', 12 :'December'}

date = int(input("Enter date in MMDDYYYY format : "))

yr = date % 10000 #extracting year
date //= 10000

dy = date % 100  #extracting day
date //= 100

mn = date  #extracting month

print(f"Date is {months[mn]} {dy}, {yr}”)
```

Output:

```
Enter date in MMDDYYYY format : 01062003
Date is January 6, 2003



Enter date in MMDDYYYY format : 05142003
Date is May 14, 2003
```

Program 2: Create a dictionary whose keys are month names and whose values are the number of days in the corresponding months -
a) Ask the user to enter a month name and use the dictionary to tell them how many days are in the month.
b) Print out all of the keys in alphabetical order.
c) Print out all of the months with 31 days.
d) Print out the key – value pairs sorted by the number of days in each month.

```
#Program to store and extract data relating to months and the
#No. of days in them

#Dictionary to store no. of days with month names as key
mn_days = {'January' : 31, 'February' :28, 'March' : 31,
            'April' : 30, 'May' : 31, 'June' : 30, 'July' :31,
            'August' : 30, 'September' : 30, 'October' : 31,
            'November' : 30, 'December' : 31}

#Printing no. of days in user-given month
mn = input("Enter a month : ")
mn = mn.lower().capitalize()
if mn in mn_days :
    print(mn, "has", mn_days[mn], "days")

#Printing keys in alphabetical order
print("\nKeys in alphabetical order - ")
L1 = list(mn_days.keys())
L1.sort()
for ele in L1 :
    print(ele, end = ', ')

#Printing months with 31 days
print("\n\nMonths with 31 days - ")
for key in mn_days :
    if mn_days[key] == 31 :
        print(key, end = ', ')

#Printing months sorted by no. of days
print("\n\nMonths sorted by no. of days - ")
L2 =  []
for key, val in mn_days.items() :
    L2.append((val, key))  # L2 = [(no. of days, month),…]
L2.sort(reverse = True)
for val, key in L2 :  #Printing sorted L2 in desired form
    print(key, ':', val, end = ', ')
```

Output:

```
Enter a month : november
November has 30 days

Keys in alphabetical order -
April, August, December, February, January, July, June, March,
May, November, October, September,

Months with 31 days -
January, March, May, July, October, December,

Months sorted by no. of days -
October : 31
May : 31
March : 31
July : 31
January : 31
December : 31
September : 30
November : 30
June : 30
August : 30
April : 30
February : 28
```

Program 3: Write a function nthRoot() that receives two parameters x and n and returns nth root of x i.e. X ^(1/n). The default value of n is 2. Write a program to find nth root by invoking nthRoot( ). Program should run till user wants to continue.

```
#Program to find nth Root of a number using functions

#Function to return nth Root of argument
def nthRoot(x, n = 2) :
    val = x**(1/n)
    return val

#__main__

#To test default input
print("Testing default input for nthRoot() - ")
print("Returned value for nthRoot(9) :", nthRoot(9))

cont = True
while cont == True : #While user wants to continue…
    print()
    x = int(input("\nEnter x : ")) #Getting input
    n = int(input("Enter n : "))
    print("nthRoot of x =", nthRoot(x, n)) #Invoking nthRoot()
    print()
    s1 = input("Do you want to try again : ")
    if s1[0].lower() != 'y' :  #if first letter is not y or Y
        cont = False
```

Output:

```
Testing default input for nthRoot() -
Returned value for nthRoot(9): 3.0


Enter x : 16
Enter n : 2
nthRoot of x = 4.0

Do you want to try again : yes


Enter x : 27
Enter n : 3
nthRoot of x = 3.0

Do you want to try again : no
```

Program 4: Write a function named volumeBox( ) to calculate and return the volume of a box.
Function should have the following input parameters -
a)Length of box
b)Width of box
c)Height of box
Default values for Length, Width and Height parameters are 1,1,1. Write a program to find volume of a box by invoking volumeBox( ) . Program should run till user wants to continue.

```
#Program to find volume of a box by creating function for the
#same

#Function to return lbh (Volume of box)
def volumeBox(l = 1, b = 1, h = 1) :
    V = l * b * h
    return V

#__main__

#To test default input
print("Testing default input for volumeBox() - ")
print("Returned value for volumeBox() :", volumeBox())

cont = True
while cont == True :  #While user wants to continue
    print()
    l = int(input("Enter length : "))   #Getting input
    b = int(input("Enter breadth : "))
    h = int(input("Enter height : "))
    print("Volume of box =", volumeBox(l, b, h)) #Invoking
                                                 #volumeBox()

    print()
    s1 =  input("Do you want to try again : ")
    if s1[0].lower() != 'y' : # if first letter is y or Y
        cont = False
```

Output:

```
Testing default input for volumeBox() -
Returned value for volumeBox() : 1

Enter length : 2
Enter breadth : 3
Enter height : 4
Volume of box = 24

Do you want to try again : yes

Enter length : 1
Enter breadth : 5
Enter height : 6
Volume of box = 30


Do you want to try again : no
```

Program 5: Write a function calci() to calculate addition, subtraction, multiplication, division of two numbers and return the results to calling function. Write a program to call calci() and print the results. Program should run till user wants to continue.

```
#Program to return sum, product, ratio and difference of two
#numbers using a function

#Function to return sum, product, difference and ratio
def calci(a, b) :
    return a+b, a-b, a*b, a/b

#__main__
cont = True
while cont == True :  #While user wants to continue
    print()
    a = int(input("Enter a : ")) #Getting input
    b = int(input("Enter b : "))

    #Invoke calci() and print results
    add, sub, mult, div = calci(a, b)
    print(f"Sum = {add}, Difference = {sub}")
    print(f"Product = {mult}, Division = {div}")

    print()
    s1 = input("Do you want to try again : ")
    if s1[0].lower() != 'y' :  #If first letter is y or Y
        cont = False
```

Output:

```
Enter a : 12
Enter b : 4
Sum = 16, Difference = 8
Product = 48, Division = 3.0

Do you want to try again : yes

Enter a : 3
Enter b : 2
Sum = 5, Difference = 1
Product = 6, Division = 1.5

Do you want to try again : no
```

Program 6: Write Python functions for the following -
i) A function cubeN() that takes a number as argument and calculates and prints cube of it . If there is no value passed to the function then function should calculate cube of 2.
ii) A function checkS() that accepts two strings and compares them . If both are equal prints 'True' otherwise prints 'False'
Write a Menu driven program to invoke the above two functions.

```
#Menu driven program which can perform comparison of strings
#or cubing of a number as per user's choice

#Returns cube
def cubeN(n = 2) :
    val = n**3
    return val

#Returns True if s1 and s2 are identical, else False
def checkS(s1, s2) :
    return s1 == s2

#__main__
print('''MENU DRIVEN PROGRAM -
1 : Calculate cube of a number
2 : Compare two strings''')

ch = int(input("\nEnter your choice : "))
if ch == 1 :   #if user wants to compute cube
    n = int(input("\nEnter a number : "))
    print("Cube is", cubeN(n))
elif ch == 2 :  #if user wants to compare strings
    s1 = input("\nEnter first string : ")
    s2 = input("Enter second string : ")
    if checkS(s1, s2) : # if strings are identical
          print("The two strings are identical")
    else :
          print("The two strings are not identical")
```

Output:

```
MENU DRIVEN PROGRAM -
1 : Calculate cube of a number
2 : Compare two strings

Enter your choice : 1
Enter a number : 5
Cube is 125
```

```
MENU DRIVEN PROGRAM -
1 : Calculate cube of a number
2 : Compare two strings

Enter your choice : 2

Enter first string : sunday
Enter second string : Sunday
The two strings are not identical
```

```
MENU DRIVEN PROGRAM -
1 : Calculate cube of a number
2 : Compare two strings

Enter your choice : 2

Enter first string : Sunday
Enter second string : Sunday
The two strings are identical
```

Program 7: Write a recursive function to print Fibonacci series up to n.

```
#Program to print Fibonacci series using recursion

#Recursively evaluated function to calculate nth term of
Fibonacci series
def fib(n) :
    if n == 1 :  #First term is 0
        return 0
    elif n == 2 : #Second term is one
        return 1
    else :  #nth term is sum of previous two terms
        return fib(n-1) + fib(n-2)

#__main__
n = int(input("Enter n : "))

print("Fibonacci series till", n, "-")

#Loop to print Fibonacci series till n
i = 1
while fib(i) <= n :
    print(fib(i), end = ' ')
    i += 1
```

Output:

```
Enter n : 10
Fibonacci series till 10 -
0 1 1 2 3 5 8



Enter n : 1000
Fibonacci series till 1000 -
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Program 8: Write a recursive function to print sum of digits of a number.

```
#Program to evaluate sum of digits of a number using recursion

def sum_of_digits(n) :
    if n == 0 :  #if no digits left…
        return 0
    else : #sum = first digit + sum of remaining digits
        return sum_of_digits(n//10) + (n % 10)

#__main__
n = int(input("Enter a number : "))
print(f"Sum of digits = {sum_of_digits(n)}")
```

Output:

```
Enter a number : 1234
Sum of digits = 10



Enter a number : 3609
Sum of digits = 18
```

Program 9: A happy number is a number in which the eventual sum of the square of the digits of the number is equal to 1. Write a program that takes number and check if it is happy number by using following functions in it.
a) Sum_sq_digits() : returns the sum of the square of the digits of the number, using the recursive technique.
b) isHAppy() : checks if the given number is a happy number by calling the function Sum_sq_digits() and displays an appropriate message.

```
#Program to find out whether a given number is a happy number
#by using a function that calculates the sum of the squares if
#the digits of a number recursively

#Function to calculate sum of squares of digits recursively
def sum_sq_digits(n) :
    if n == 0 :  #if no digits left…
        return 0
    else :  #sum = (first digit)^2 + (sum of squares of
        d = n % 10                   #remaining digits)
        return d**2 + sum_sq_digits(n//10)

L = [] #L stores values of n at each stage of recursion
def is_happy(n) :
    global L

    nextn = sum_sq_digits(n)
    if nextn == 1 : #if sum of squares of digits is 1
        return True
    elif nextn in L : #if values of n will cycle endlessly
        return False
    else :
        L.append(nextn) # add nextn to L
        return is_happy(nextn) #check if nextn is happy no.

#__main__
n = int(input("Enter a number : "))
if is_happy(n) :
    print(n, "is a happy number !")
else :
    print(n, "is not a happy number !")
```

Output:

```
Enter a number : 28
28 is a happy number !


Enter a number : 12
12 is not a happy number !
```

Program 10: Write a program to count ‘He’ of ‘His’ present in the text file ‘story.txt’. (Text shown in program)

```
text = '''King Krishnadevaraya loved horses and had the best
collection of horse breeds in the
Kingdom. Well, one day, a trader came to the King and told him
that he had brought
with him a horse of the best breed in Arabia.
He invited the King to inspect the horse. King Krishnadevaraya
loved the horse; so the
trader said that the King could buy this one and that he had
two more like this one,
back in Arabia that he would go back to get. The King loved
the horse so much that he
had to have the other two as well. He paid the trader 5000
gold coins in advance.
The trader promised that he would return within two days with
the other horses.'''

F = open("story.txt", "w") #Writing text into file
F.write(text)
F.close()

F = open("story.txt", "r") #Seprating txt into words
words = F.read().split()
F.close()

count = 0              #Counting no. of words
for wrd in words :
    if wrd.lower() == "he" or wrd.lower() == "his" :
        count += 1

print("No of ocurrences of 'he' or 'his' in text :", count)
```

Output:

```
No of ocurrences of 'he' or 'his' in text : 7
```

Program 11: Write a program to count and display the no. of lines starting with alphabet ‘W’ present in text file ‘poem.txt’. (File content shown in program)

```
text = '''When the gloomy gray sky turns to clear azure blue,
And the snow disappears from the ground,
When the birds start to sing, and our moods start to lift,
Then we know Spring is coming around.
          When the first flower bulbs poke their heads toward
the sun,
          Golden daffodils, hyacinths, too;
          When the brown grass turns green, and the
wildflowers bloom,
          Then sweet Spring makes its showy debut.'''

F = open("poem.txt", "w")  #Writing text into file
F.write(text)
F.close()


F = open("poem.txt", "r")

count = 0
l = F.readline()
while l : #while lines still left
    i = 0
    while l[i] in [' ', '   ']: #Skip spaces
        i += 1

    if l[i] == "W" : #If first letter is 'W'
        count += 1

    l = F.readline()
F.close()

print("No of lines beginning with 'W' :", count)
```

Output:

```
No of lines beginning with 'W' : 4
```

Program 12: A file ‘sports.dat’ contains information in following format : Event – Participant. Define a function Athletic_display() that creates a file Athletic.dat, copying only those records from sports.dat where the event name is ‘Athletics’. Write a program to invoke Athletic.display()

```
import pickle


def Athletic_display() :
    with open("sports.dat", "rb") as F_sp : #get data
        L_sp = pickle.load(F_sp)

    L_ath = []
    for (ev, sp) in L_sp :
        if ev == "Athletics"  :#add needed records to L_ath
            L_ath.append((ev, sp))

    with open("Athletic.dat", "wb") as F_ath : #store L_ath in
                                                #athletic.dat
        pickle.dump(L_ath, F_ath)



#__main__

sports_data = [("Football", "Raj"), ('Athletics', "Harish"),
('Tennis', "John"), ("Athletics", "Sam"),
("Football", "Paras"), ("Athletics", "Kartik")]

with open("sports.dat", "wb") as F : #write data to sports.dat
    pickle.dump(sports_data, F)

Athletic_display()

print("Records stored in 'sports.dat' : ") #print data
with open("sports.dat", "rb") as F :
    print(pickle.load(F))

print()

print("Records stored in 'Athletics.dat' : ") # print data
with open("Athletic.dat", "rb") as F :
    print(pickle.load(F))
```

Output:

```
Records stored in 'sports.dat' :
[('Football', 'Raj'), ('Athletics', 'Harish'), ('Tennis',
'John'), ('Athletics', 'Sam'), ('Football', 'Paras'),
('Athletics', 'Kartik')]

Records stored in 'Athletics.dat' :
[('Athletics', 'Harish'), ('Athletics', 'Sam'), ('Athletics',
'Kartik')]
```

Program 13: Write a database connectivity program to fetch all records from student table

Rno	Name	Avg
100	Raj	76.6
101	Harish	89.6
102	Kiran	98.9
103	Vedanth	92.5

```
import mysql.connector as msc

def print_table(L) :   #prints records one after another
    for row in L :
        print(row)
    print()

con = msc.connect(user = ‘admin’, password = ‘sqlpassword‘,
host = 'localhost', database = 'school')
cur = con.cursor()

cur.execute("SELECT * FROM student;") #retrieve records
print("Records in student : ")
print_table(cur.fetchall())

cur.close()
con.close()
```

Output:

```
Records in student :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)
```

Program 14: Write a database connectivity program to add a record to student table

```
import mysql.connector as msc

def print_table(L) : #prints records one after another
    for row in L :
        print(row)
    print()

con = msc.connect(user = 'admin', password = 'sqlpassword',
host = 'localhost', database = 'school')
cur = con.cursor()

cur.execute("SELECT * FROM student;") #show initial records
print("Before record is inserted : ")
print_table(cur.fetchall())

record  = (104, 'Aryan', 65.5)   #insert new record
cur.execute(f"INSERT INTO student VALUES {record}")
con.commit()

cur.execute("SELECT * FROM student;") #show final records
print("After record is inserted : ")
print_table(cur.fetchall())

cur.close()
con.close()
```

Output:

```
Before record is inserted :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)

After record is inserted :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)
(104, 'Aryan', 65.5)
```

Program 15: Write a database connectivity program to delete a record from student table

```
import mysql.connector as msc

def print_table(L) : #prints records one after another
    for row in L :
        print(row)
    print()

con = msc.connect(user = ‘admin’, password = 'sqlpassword',
host = 'localhost', database = 'school')
cur = con.cursor()

cur.execute("SELECT * FROM student;") #show inital records
print("Before record is deleted : ")
print_table(cur.fetchall())

rno = 104     #delete record with rno 104
cur.execute('''DELETE
               FROM student
               WHERE rno = {}'''.format(rno))
con.commit()

cur.execute("SELECT * FROM student;") #show final records
print("After record is deleted : ")
print_table(cur.fetchall())

cur.close()
con.close()
```

Output:

```
Before record is deleted :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)
(104, 'Aryan', 65.5)

After record is deleted :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)
```

Program 16: Write a database connectivity program to update a record into the student table

```
import mysql.connector as msc

def print_table(L) : #prints records one after another
    for row in L :
        print(row)
    print()

con = msc.connect(user = ‘admin’, password = 'sqlpassword',
host = 'localhost', database = 'school')
cur = con.cursor()

cur.execute("SELECT * FROM student;")  #show initial records
print("Before record is updated : ")
print_table(cur.fetchall())

rno = 100
u_rec = (rno, 'Raj', 81.0)    #update record for given rno
cur.execute(f'''UPDATE student
                SET name = '{u_rec[1]}', avg = {u_rec[2]}
                WHERE rno = {rno};''')

cur.execute("SELECT * FROM student;")  #show final records
print("After record is  updated : ")
print_table(cur.fetchall())

cur.close()
con.close()
```

Output:

```
Before record is updated :
(100, 'Raj', 76.6)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)

After record is  updated :
(100, 'Raj', 81.0)
(101, 'Harish', 89.6)
(102, 'Kiran', 98.9)
(103, 'Vedanth', 92.5)
```

Program 17: Write a program to search for an element in a list using binary search technique

```
def find_ele(ele, L) : #finds index using binary search
    beg = 0
    end = len(L) - 1
    while beg <= end :
        mid = (beg + end)//2
        if L[mid] == ele :
            return mid
        elif ele < L[mid] :
            end = mid - 1 #search lower half
        else :
            beg = mid + 1 #search uppper half
    else :
        return None

#__main__
s1 = input("\nEnter list (type n to exit) : ")
while s1.lower() != 'n' : #exit if 'n'
    L = list(eval(s1))

    s2 = input("\nEnter the element(type n to exit) : ")
    while s2.lower() != 'n' :
        ele = int(s2)
        index = find_ele(ele, L)
        if index == None : #if element not found
            print(ele, "is not in the entered list")
        else :
            print(ele, "found at index", index)
        s2 = input("\nEnter the element(type n to exit) : ")

    s1 = input("\nEnter list (type n to exit) : ")
```

Output:

```
Enter list (type n to exit) : 1, 2, 7

Enter the element(type n to exit) : 1
1 found at index 0

Enter the element(type n to exit) : 2
2 found at index 1

Enter the element(type n to exit) : 3
```

```
3 is not in the entered list

Enter the element(type n to exit) : 7
7 found at index 2

Enter the element(type n to exit) : n

Enter list (type n to exit) : 8, 9

Enter the element(type n to exit) : 11
11 is not in the entered list

Enter the element(type n to exit) : n

Enter list (type n to exit) : n
```

Program 18: Write a program to insert an element into an ordered list

```
def find_index(ele, L) : #finds index for insertion
    for i in range(len(L)) :
        if ele <= L[i] :
            return i
    else :
        return len(L)

#__main__
s1 = input("\nEnter list (type n to exit) : ")
while s1.lower() != 'n' : #exit if 'n'
    L = list(eval(s1))

    s2 = input("\nEnter the element(type n to exit) : ")
    while s2.lower() != 'n' :
        ele = int(s2)
        L.insert(find_index(ele, L), ele)  #find index and
                                           #insert
        print("Element is inserted : ", L)

        s2 = input("\nEnter the element(type n to exit) : ")

    s1 = input("\nEnter list (type n to exit) : ")
```

Output:

```
Enter list (type n to exit) : 1, 4, 7

Enter the element(type n to exit) : 2
Element is inserted :  [1, 2, 4, 7]

Enter the element(type n to exit) : 3
Element is inserted :  [1, 2, 3, 4, 7]

Enter the element(type n to exit) : 4
Element is inserted :  [1, 2, 3, 4, 4, 7]

Enter the element(type n to exit) : 7
Element is inserted :  [1, 2, 3, 4, 4, 7, 7]

Enter the element(type n to exit) : 0
Element is inserted :  [0, 1, 2, 3, 4, 4, 7, 7]

Enter the element(type n to exit) : 9
```

```
Element is inserted :  [0, 1, 2, 3, 4, 4, 7, 7, 9]

Enter the element(type n to exit) : n

Enter list (type n to exit) : n
```

Program 19: Write a program to delete an element from an ordered list

---

```
def find_index(ele, L) : #finds index for deletion
    for i in range(len(L)) :
        if ele == L[i] :
            return i
    else :
        return None

#__main__
s1 = input("\nEnter list (type n to exit) : ")
while s1.lower() != 'n' : #exit if 'n'
    L = list(eval(s1))

    s2 = input("\nEnter the element(type n to exit) : ")
    while s2.lower() != 'n' :
        ele = int(s2)
        index = find_index(ele, L) #find index
        if index == None :  #if element not present
            print(ele, "not in list")
        else :
            del L[index]
            print("Element is deleted", L)
        s2 = input("\nEnter the element(type n to exit) : ")

    s1 = input("\nEnter list (type n to exit) : ")
```

Output:

```
Enter list (type n to exit) : 0, 1, 3, 6, 7, 11, 13

Enter the element(type n to exit) : 3
Element is deleted [0, 1, 6, 7, 11, 13]

Enter the element(type n to exit) : 0
Element is deleted [1, 6, 7, 11, 13]
```

```
Enter the element(type n to exit) : 13
Element is deleted [1, 6, 7, 11]

Enter the element(type n to exit) : 7
Element is deleted [1, 6, 11]

Enter the element(type n to exit) : 5
5 not in list

Enter the element(type n to exit) : n

Enter list (type n to exit) : n
```

Program 20: Write a program to sort a list using the bubble sort method

---

```
def bubble_sort(L) :
    for i in range(1, len(L) - 1) :  #i = no of elements to
                                     #leave at end of list
        for j in range(len(L) - i) :
            if L[j] > L[j + 1] :
                L[j], L[j+1] = L[j+1], L[j]

#__main__
s1 = input("\nEnter list (type n to exit) : ")
while s1.lower() != 'n' :
    L = list(eval(s1))  #input, sort and print
    bubble_sort(L)
    print("Sorted list : ", L)

    s1 = input("\nEnter list (type n to exit) : ")
```

Output:

```
Enter list (type n to exit) : 1, 9, 2, 7, 13, -3
Sorted list :  [-3, 1, 2, 7, 9, 13]

Enter list (type n to exit) : 1, 9, 3, 6, 2
Sorted list :  [1, 2, 3, 6, 9]

Enter list (type n to exit) : n
```

Program 21 : Write a program to sort a list using the insertion sort method

```
def insertion_sort(L) :
    for i in range(1, len(L)) :
        j = i
        while j > 0 and L[j] < L[j - 1] :
            L[j], L[j-1] = L[j-1], L[j]
            j = j - 1

#__main__
s1 = input("\nEnter list (type n to exit) : ")
while s1.lower() != 'n' :
    L = list(eval(s1)) #input, sort and print
    insertion_sort(L)
    print("Sorted list : ", L)

    s1 = input("\nEnter list (type n to exit) : ")
```

Output:

```
Enter list (type n to exit) : 3, 7, 1, 2, -4, 2.5
Sorted list :  [-4, 1, 2, 2.5, 3, 7]

Enter list (type n to exit) : 1, 4, 2, 6, 2, 0
Sorted list :  [0, 1, 2, 2, 4, 6]

Enter list (type n to exit) : n
```

Program 22 : Write a menu driven program to implement a stack

---

```
def pop() :
    global L, top

    val = L[top]  #del and return top value
    del L[top]
    top = top - 1
    return val

def push(ele) :
    global L, top

    L.append(ele)
    top = top + 1

def peek() :
    global L, top

    return L[top]

def traverse() :
    global L, top

    s = f"{peek()}    <-- top" #creates a string showing
                               #traversal of stack
    for i in range(top - 1, - 1, -1) :
        s += "\n{}".format(L[i])
    return s

def clear() :
    global L, top

    L.clear()
    top = -1



#__main__

L = []
top = -1

menu = '''
Type 1 to push
Type 2 to peek
Type 3 to pop
Type t to traverse
Type c to clear stack
```

```
(Type quit to close ) '''

print(menu)
s = input("\n>> ").lower()
while s != "quit" :
    if s == '1' :
        ele = int(input("Enter item to be pushed : "))
        push(ele)
        print(ele, "is pushed into stack")
    elif s == '2' :
        if top >= 0 :
            print(peek())
        else :
            print("Stack is empty")
    elif s == '3' :
        if top >= 0 :
            print(pop())
            print("An item has been popped")
        else :
            print("Underflow error")
    elif s == 't' :
        if top >= 0 :
            print(traverse())
        else :
            print("Stack is empty")
    elif s ==  'c' :
        clear()
        print("Stack is cleared")
    elif s.isspace() or s == '' :
        print(menu)
    else :
        print("Invalid")

    s = input("\n>> ").lower()
```

Output:

```
Type 1 to push
Type 2 to peek
Type 3 to pop
Type t to traverse
Type c to clear stack
(Type quit to close )

>> 1
Enter item to be pushed : 10
10 is pushed into stack
```

```
>> 1
Enter item to be pushed : 20
20 is pushed into stack

>> 1
Enter item to be pushed : 30
30 is pushed into stack

>> 2
30

>> 3
30
An item has been popped

>> 2
20

>> 1
Enter item to be pushed : 50
50 is pushed into stack

>> t
50    <-- top
20
10

>> c
Stack is cleared

>> 2
Stack is empty

>> quit
```

Output:

```
Type 1 to push
Type 2 to peek
Type 3 to pop
Type t to traverse
Type c to clear stack
(Type quit to close )
```

```
>> 4
Invalid

>>

Type 1 to push
Type 2 to peek
Type 3 to pop
Type t to traverse
Type c to clear stack
(Type quit to close )

>> 1
Enter item to be pushed : 10
10 is pushed into stack

>> 3
10
An item has been popped

>> 2
Stack is empty

>> 3
Underflow error

>> t
Stack is empty

>> quit
```

1.To show firstname,lastname,address and city of all employees living in Paris.

SELECT firstname, lastname, address, city
FROM employees
WHERE city = 'paris';

```
+-----------+----------+-----------------+-------+
| firstname | lastname | address         | city  |
+-----------+----------+-----------------+-------+
| GOERGE    | SMITH    | 83 FIRST STREET | PARIS |
| PETER     | THOMPSON | 11 RED ROAD     | PARIS |
+-----------+----------+-----------------+-------+
```

2.To display the content of EMPLOYEES table in descending order of firstname.

SELECT *
FROM employees
ORDER BY firstname DESC;

```
+-------+-----------+----------+-------------------+----------
--+
| empid | firstname | lastname | address           | city
|
+-------+-----------+----------+-------------------+----------
--+
|   215 | SARAH     | ACKERMAN | 440 US.110        | HOWARD
|
|   152 | SAM       | TONES    | 33 ELM STREET     | NEW DELHI
|
|   300 | ROBERT    | SAMUEL   | 9 FIFTH CROSS     |WASHINGTON
|
|   400 | RACHEL    | LEE      | 121 HARRISON ST.  | NEW YORK
|
|   441 | PETER     | THOMPSON | 11 RED ROAD       | PARIS
|
|   105 | MARY      | JONES    | 842 VINE AVE.     | NEW YORK
|
|   244 | MANILA    | SENGUPTA | 24 FRIENDS STREET | NEW DELHI
|
|   335 | HENRY     | WILLIAMS | 12 MOORE STREET   | BOSTON
|
|    10 | GOERGE    | SMITH    | 83 FIRST STREET   | PARIS
|
+-------+-----------+----------+-------------------+----------
--+
```

3.To display empid & First name of all employees who are clerk,Salesman or Director.

```
SELECT empid, firstname
FROM employees NATURAL JOIN empsalary
WHERE designation IN ('director', 'salesman', 'clerk');
```

```
+-------+-----------+
| empid | firstname |
+-------+-----------+
|   152 | SAM       |
|   244 | MANILA    |
|   300 | ROBERT    |
|   335 | HENRY     |
|   400 | RACHEL    |
|   441 | PETER     |
+-------+-----------+
```

4.To display empid ,firstname,lastname , salary,benefits and total salary that is calculated as salary+benefits

```
SELECT empid, firstname, lastname, salary, benefits, (salary + benefits) AS 'total salary'
FROM employees NATURAL JOIN empsalary;
```

```
+-------+-----------+----------+----------+----------+-------
------+
| empid | firstname | lastname | salary   | benefits | total
salary |
+-------+-----------+----------+----------+----------+-------
------+
|   105 | MARY      | JONES    | 65000.00 | 15000.00 |
80000.00 |
|   152 | SAM       | TONES    | 80000.00 | 25000.00 |
105000.00 |
|   215 | SARAH     | ACKERMAN | 75000.00 | 12500.00 |
87500.00 |
|   244 | MANILA    | SENGUPTA |  5000.00 |  1200.00 |
6200.00 |
|   300 | ROBERT    | SAMUEL   |  4500.00 |  1000.00 |
5500.00 |
|   335 | HENRY     | WILLIAMS |  4500.00 |  1000.00 |
5500.00 |
|   400 | RACHEL    | LEE      |  3200.00 |   750.00 |
3950.00 |
|   441 | PETER     | THOMPSON |  2800.00 |   750.00 |
3550.00 |
+-------+-----------+----------+----------+----------+-------
------+
```

5.To display sum and avg of salaries of all Employees.

```
SELECT SUM(salary), AVG(salary)
FROM empsalary;
```

```
+-------------+--------------+
| SUM(salary) | AVG(salary)  |
+-------------+--------------+
|   315000.00 | 35000.000000 |
+-------------+--------------+
```

6.To display maximum and minimum salaries of all Managers.

```
SELECT MAX(salary), MIN(salary)
FROM empsalary
WHERE designation = 'manager';
```

```
+-------------+-------------+
| MAX(salary) | MIN(salary) |
+-------------+-------------+
|    75000.00 |    65000.00 |
+-------------+-------------+
```

7.To display empid and salary of all the employees who are getting salary in the range of 32000 -65000

```
SELECT empid, salary
FROM empsalary
WHERE salary BETWEEN 32000 AND 65000;
```

```
+-------+----------+
| empid | salary   |
+-------+----------+
|   105 | 65000.00 |
+-------+----------+
```

8.To display all employees whose first name has letter ‘A’ .

```
SELECT *
```

FROM employees
WHERE firstname LIKE '%A%';

```
+-------+-----------+----------+------------------+---------
-+
| empid | firstname | lastname | address          | city
|
+-------+-----------+----------+------------------+---------
-+
|   105 | MARY      | JONES    | 842 VINE AVE.    | NEW YORK
|
|   152 | SAM       | TONES    | 33 ELM STREET    | NEW DELHI
|
|   215 | SARAH     | ACKERMAN | 440 US.110       | HOWARD
|
|   244 | MANILA    | SENGUPTA | 24 FRIENDS STREET | NEW DELHI
|
|   400 | RACHEL    | LEE      | 121 HARRISON ST.  | NEW YORK
|
+-------+-----------+----------+------------------+---------
-+
```

9.To display all the employees who stay in NEW DELHI, NEW YORK, PARIS

SELECT *
FROM employees
WHERE city IN ('new delhi', 'new york', 'paris');

```
+-------+-----------+----------+------------------+---------
-+
| empid | firstname | lastname | address          | city
|
+-------+-----------+----------+------------------+---------
-+
|    10 | GOERGE    | SMITH    | 83 FIRST STREET   | PARIS
|
|   105 | MARY      | JONES    | 842 VINE AVE.    | NEW YORK
|
|   152 | SAM       | TONES    | 33 ELM STREET    | NEW DELHI
|
|   244 | MANILA    | SENGUPTA | 24 FRIENDS STREET | NEW DELHI
|
|   400 | RACHEL    | LEE      | 121 HARRISON ST.  | NEW YORK
|
|   441 | PETER     | THOMPSON | 11 RED ROAD       | PARIS
|
+-------+-----------+----------+------------------+---------
-+
```

10. To display all the salesmen having salary more than 5000.

SELECT *
FROM empsalary
WHERE designation = 'salesman' AND salary > 5000 ;

```
Empty set (0.00 sec)
```

11.To add a new column Bonus of type float(6,2) in Table EMPSALARY.

```
ALTER TABLE empsalary
ADD bonus float(6, 2);

Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 1

 SELECT * FROM empsalary;

+-------+----------+----------+-------------+-------+
| empid | salary   | benefits | designation | bonus |
+-------+----------+----------+-------------+-------+
|   101 | 75000.00 | 15000.00 | Manager     |  NULL |
|   105 | 65000.00 | 15000.00 | Manager     |  NULL |
|   152 | 80000.00 | 25000.00 | Director    |  NULL |
|   215 | 75000.00 | 12500.00 | Manager     |  NULL |
|   244 |  5000.00 |  1200.00 | Clerk       |  NULL |
|   300 |  4500.00 |  1000.00 | Clerk       |  NULL |
|   335 |  4500.00 |  1000.00 | Clerk       |  NULL |
|   400 |  3200.00 |   750.00 | Salesman    |  NULL |
|   441 |  2800.00 |   750.00 | Salesman    |  NULL |
+-------+----------+----------+-------------+-------+
```

12 To remove the table from the database.

```
DROP TABLE empsalary;

Query OK, 0 rows affected (0.02 sec)
```

1.To show all information about Baby cots from FURNITURE table.

```
SELECT *
FROM furniture
WHERE type = 'baby cot';
```

```
+------+--------------+----------+-------------+---------+----------+
| fno  | itemname     | type     | dateofstock | price   | discount |
+------+--------------+----------+-------------+---------+----------+
|    2 | Pink feather | Baby cot | 2002-01-20  | 7000.00 |    20.00 |
|    3 | Dolphin      | Baby cot | 2002-02-19  | 9500.00 |    20.00 |
|    6 | Donald       | Baby cot | 2002-02-24  | 6500.00 |    15.00 |
+------+--------------+----------+-------------+---------+----------+
```

2.To list the item name which are priced more than 15000 from FURNITURE table.

```
SELECT itemname
FROM furniture
WHERE price > 15000 ;
```

```
+--------------+
| itemname     |
+--------------+
| White Lotus  |
| Decent       |
| Comfort zone |
| Royal Finish |
| Royal Tiger  |
+--------------+
```

3.To list item name and type of those items in which date of stock is before 22/01/02 from FURNITURE table in descending order of item name.

```
SELECT itemname, type
FROM furniture
WHERE dateofstock < '2002-01-22'
ORDER BY itemname DESC;
```

```
+---------------+--------------+
| itemname      | type         |
+---------------+--------------+
| Pink feather  | Baby cot     |
| Econo sitting | Sofa         |
| Decent        | Office Table |
| Comfort zone  | Double Bed   |
+---------------+--------------+
```

4.To count the number of items whose type is ‘Sofa’ from FURNITURE table.

```
SELECT COUNT(*)
FROM furniture
WHERE type = 'sofa' ;
```

```
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
```

5.To display count of different types of furniture newly arrived.

```
SELECT COUNT(DISTINCT type)
FROM arrivals;
```

```
+----------------------+
| COUNT(DISTINCT type) |
+----------------------+
|                    3 |
+----------------------+
```

6.To display average of discount from furniture where type =’Office Table’

```
SELECT AVG(discount)
FROM furniture
WHERE type = 'office table';
```

```
+---------------+
| AVG(discount) |
+---------------+
|      30.000000 |
+---------------+
```

7. To display all item name where price<10000 and discount<=20

```
SELECT itemname
FROM furniture
WHERE discount <= 20 AND price <= 10000
UNION
SELECT itemname
FROM arrivals
WHERE discount <= 20 AND price <= 10000 ;
+--------------+
| itemname     |
+--------------+
| Pink feather |
| Dolphin      |
| Donald       |
| Micky        |
+--------------+
```

8.To increase the Price of Sofa by Rs.1000 of both FURNITURE & ARRIVALS Table.

```
UPDATE furniture
SET price = price + 1000
WHERE type = 'sofa';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0
 ON emp1.deptno = dept1.deptno

SELECT * FROM furniture;
```

```
+-------+-----------------+--------------+-------------+----------+----------+
| fno  | itemname       | type        | dateofstock | price    | discount |
+------+-----------------+--------------+-------------+----------+----------+
|    1 | White Lotus     | Double Bed   | 2002-02-23  | 30000.00 |    25.00 |
|    2 | Pink feather    | Baby cot     | 2002-01-20  |  7000.00 |    20.00 |
|    3 | Dolphin         | Baby cot     | 2002-02-19  |  9500.00 |    20.00 |
|    4 | Decent          | Office Table | 2002-01-01  | 25000.00 |    30.00 |
|    5 | Comfort zone    | Double Bed   | 2002-01-12  | 25000.00 |    25.00 |
|    6 | Donald          | Baby cot     | 2002-02-24  |  6500.00 |    15.00 |
|    7 | Royal Finish    | Office Table | 2002-02-20  | 18000.00 |    30.00 |
|    8 | Royal Tiger     | Sofa         | 2002-02-22  | 32000.00 |    30.00 |
|    9 | Econo sitting   | Sofa         | 2001-02-13  | 10500.00 |    25.00 |
|   10 | Eating Paradise | Dining Table | 2002-02-19  | 11500.00 |    25.00 |
+------+-----------------+--------------+-------------+----------+----------+
```

```
UPDATE arrivals
SET price = price + 1000
WHERE type = 'sofa';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
SELECT * FROM arrivals;
------+---------------+------------+-------------+----------+----------+
| fno  | itemname      | type       | dateofstock | price    | discount |
+------+---------------+------------+-------------+----------+----------+
|   11 | World Comfort | Double Bed | 2003-03-23  | 25000.00 |    25.00 |
|   12 | Old Fox       | Sofa       | 2003-02-20  | 18000.00 |    20.00 |
|   13 | Micky         | Baby cot   | 2003-02-21  |  7500.00 |    15.00 |
+------+---------------+------------+-------------+----------+----------
```

10.To delete all the records from FURNITURE table.

```
DELETE
FROM furniture ;
Query OK, 10 rows affected (0.01 sec)
```

11. To display all sofas having discount in the range 25-30.

```
SELECT *
FROM arrivals
WHERE type = 'sofa' AND discount BETWEEN 25 AND 30 ;
Empty set (0.00 sec)
```

12. To insert the following new row in Furniture table.

15	**Study Time**	**Writing Table**	**21/01/02**	**10000**	**25**

```
INSERT INTO furniture
VALUES(15, 'Study time', 'Writing table', '2002-01-21', 10000, 25) ;
Query OK, 1 row affected (0.01 sec)
```

```
SELECT * FROM furniture;
+------+------------+---------------+-------------+----------+----------+
| fno  | itemname   | type          | dateofstock | price    | discount |
+------+------------+---------------+-------------+----------+----------+
|   15 | Study time | Writing table | 2002-01-21  | 10000.00 |    25.00 |
+------+------------+---------------+-------------+----------+----------+
```