SONA COLLEGE OF ARTS AND SCIENCE
Affiliated to Periyar University, Salem

DEPARTMENT OF COMPUTER APPLICATIONS

III BCA

PRACTICAL VI :

PYTHON PROGRAMMING

Name : _____

Register Number: _____

ACADEMIC YEAR: 2023 – 2024

SONA COLLEGE OF ARTS AND SCIENCE
Affiliated to Periyar University, Salem

*This is to certify that the Practical Record "***Practical – VI: PYTHON PROGRAMMING"** *is a bona-fide work done by* _____ *Reg. No.* _____ *submitted to the Department of Computer Applications, during the academic year* ***2023 - 2024.***

SUBJECT IN-CHARGE **HEAD OF THE DEPARMRNT**

Submitted for University Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Ex.No.:1	Simple Arithmetic Calculator
Date:	

Aim:

To create a simple calculator to do all the arithmetic operations

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application and type the coding

Step 3: Get two inputs from the user (choice corresponding to various operations like addition, subtraction, multiplication, division, and exponentiation)

Step 4: Perform the operations based on the user's choice.

Step 5: Print the output

Step 6: End the program

CODING:

```
def add(number_1, number_2):
        return number_1 + number_2
def subtract(number_1, number_2):
        return number_1 - number_2
def multiply(number_1, number_2):
        return number_1 * number_2
def divide(number_1, number_2):
        return number_1 / number_2
def expnumber_1, number_2):
        return number_1 ** number_2
print("Please select which of the following arithmetic operation you want me to
perform-\n" \
    "1. Add\n" \
    "2. Subtract\n" \
    "3. Multiply\n" \
    "4. Divide\n 5. Exponent\n")
f='y'
while(f=='Y' or f=='y'):
  operation = int(input(" 1, 2, 3,  4 or 5 :"))
  number_1 = int(input('Enter the first number: '))
  number_2 = int(input('Enter the second number: '))
  if operation == 1:
        print(number_1, "+", number_2, "=",add(number_1, number_2))
  elif operation == 2:
     print(number_1, "-", number_2, "=", subtract(number_1, number_2))
  elif operation == 3:
        print(number_1, "*", number_2, "=",multiply(number_1, number_2))
  elif operation == 4:
        print(number_1, "/", number_2, "=",divide(number_1, number_2))
```

```
    elif operation == 5:
            print(number_1, "**", number_2, "=",exp(number_1, number_2))
    else:
            print("Please enter Valid input")
    f=input("Do you want to continue (y/n)")
```

OUTPUT:

```
Please select which of the following arithmetic operation you want me to perform-
1. Add
2. Subtract
3. Multiply
4. Divide
 5. Exponent

 1, 2, 3,  4 or 5 :1
Enter the first number: 45
Enter the second number: 45
45 + 45 = 90
Do you want to continue (y/n)y
 1, 2, 3,  4 or 5 :2
Enter the first number: 789
Enter the second number: 654
789 - 654 = 135
Do you want to continue (y/n)y
 1, 2, 3,  4 or 5 :3
Enter the first number: 6
Enter the second number: 7
6 * 7 = 42
Do you want to continue (y/n)y
 1, 2, 3,  4 or 5 :4
Enter the first number: 7
Enter the second number: 8
7 / 8 = 0.875
Do you want to continue (y/n)y
1, 2, 3,  4 or 5 :5
Enter the first number: 3
Enter the second number: 4
3 ** 4 = 81
Do you want to continue (y/n)
```

Result:

Thus the program has executed successfully and the result verified with the inputs and outputs.

Ex.No.:2	Using Control flow tools
Date:	

Aim:

To find a given year is leap year or not using if…else

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application or Jupyter and type the coding

Step 3: Get the input year from the user

Step 4: Find the given year is leap or not using if..else

Step 5: Print the output

Step 6: End the program

CODING:

```
def check_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

year = int(input("Enter a year: "))

if check_year(year):
    print("{0} is a leap year".format(year))
else:
    print("{0} is not a leap year".format(year))
```

OUTPUT:

```
Enter a year: 2024
2024 is a leap year

Enter a year: 2022
2022 is not a leap year
```

Result:

Thus the program has been executed successfully and verified the result.

Ex.No.:3	Using for loop
Date:	

Aim:

To write a program which uses for loop for finding the prime numbers between a specified ranges.

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application and type the coding

Step 3: Get two inputs from the user for specifying the range

Step 4: Check whether a particular number is prime or not by using for and if

Step 5: Print the output

Step 6: End the program

CODING:

```
lower = int(input("enter the lower range number"))
upper = int(input("enter the upper range number"))
print("Prime numbers between", lower, "and", upper, "are:")
for num in range(lower, upper + 1):
    if num > 1:
     for i in range(2, num):
        if (num % i) == 0:
            break
     else:
        print(num)
```

OUTPUT:

```
enter the lower range number 5
enter the upper range number 50
Prime numbers between 5 and 50 are:
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Result:

Thus the program has executed successfully and the result verified with the inputs and outputs.

Ex.No.: 4 A	Use list as stack Data Structure
Date:	

Aim:

To implement list as Stack Data structure

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application or Jupyter note book and type the coding

Step 3: Write functions to implement stack operations like push, pop, display and isEmpty

Step 4: Get the choice from the user and depending on the choice perform the stack operations

Step 5: Print the output

Step 6: End the program

CODING:

```
def  isEmpty(stk):              # checks whether the stack is empty or not
  if stk==[ ]:
    return True
  else:
    return False

def  push(stk,item):            # Allow additions to the stack
  stk.append(item)
  top=len(stk)-1

def  pop(stk):
  if isEmpty(stk):              # verifies whether the stack is empty or not
    print("Underflow")
  else:     # Allow deletions from the stack
    item=stk.pop()
    if len(stk)==0:
      top=None
    else:
      top=len(stk)
      print("Popped item is "+str(item))

def display(stk):
  if isEmpty(stk):
    print("Stack is empty")
  else:
    top=len(stk)-1
    print("Elements in the stack are: ")
    for i in range(top,-1,-1):
      print (str(stk[i]))
```

```
print("Please select which of the following Stack operation you want me to perform-\n" \
      "1. Push\n" \
      "2. Pop\n" \
      "3. Display\n" \
      "4. Invalid choice\n")
stk=[ ]
top=None
f='y'
while f=='y' or f=='Y':
   choice=int(input("Enter choice:"))
   if choice==1:
      item=input("Enter item to push:")
      push(stk,item)
   elif choice==2:
      pop(stk)
   elif choice==3:
      display(stk)
   else:
      print("invalid input")
   f=input("you want to continue (y/n)")
```

OUTPUT:

```
1. Push
2. Pop
3. Display
4. Invalid choice
Enter choice: 1
Enter item to push: 56
you want to continue (y/n) y
```

Enter choice: 1
Enter item to push: india
you want to continue (y/n)y
Enter choice:3
Elements in the stack are:
india
56
you want to continue (y/n)y
Enter choice:2
Popped item is india
you want to continue (y/n)y
Enter choice:3
Elements in the stack are:
56
you want to continue (y/n)n

Result:

Thus the program has been executed successfully and implemented Stack Data structure.

Ex.No.:4 B	Use list as Queue Data Structure
Date:	

Aim:

To implement list as Queue Data structure

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application or Jupyter note book and type the coding

Step 3: Write functions to implement queue operations like Enqueue, Dequeue and Display

Step 4: Get the choice from the user and depending on the choice perform the Queue operations

Step 5: Print the output

Step 6: End the program

CODING:

```
def enqueue(data):  #Adding elements to queue at the rear end
  queue.insert(0,data)
def dequeue():          #Removing the front element from the queue
  if len(queue)>0:
    return queue.pop()
  return ("Queue Empty!")
def display():  #To display the elements of the queue
  if  len(queue)>0:
      print("Elements on queue are:");
      for i in range(len(queue)):
         print(queue[i])
   else:
      print("Queue empty")
queue=[ ]
f='y'
print("Please select which of the following Queue operation you want me to perform-\n" \
      "1. Enqueue\n" \
      "2. Dequeue\n" \
      "3. Display\n" \
      "4. Invalid choice\n")
while f=='y' or f=='Y':
   choice=int(input("Enter choice"))
   if choice==1:
      item=input("Enter item to add")
      enqueue(item)
   elif choice==2:
      print("Popped Element is: "+str(dequeue()))
   elif choice==3:
      display()
   else:
```

```
        print("invalid input")
    f=input("you want to continue (y/n)")
```

OUTPUT:

Please select which of the following Queue operation you want me to perform-

1. Enqueue

2. Dequeue

3. Display

4. Invalid choice

Enter choice 1

Enter item to add 45

you want to continue (y/n)y

Enter choice1

Enter item to add 34

you want to continue (y/n)y

Enter choice 3

Elements on queue are:

34

45

you want to continue (y/n)y

Enter choice 2

Popped Element is: 45

you want to continue (y/n)y

Enter choice 2

Popped Element is: 34

you want to continue (y/n)y

Enter choice 2

Popped Element is: Queue Empty!

you want to continue (y/n)y

Enter choice3

Queue empty

you want to continue (y/n)n

Result:

Thus the program has been successfully implemented Queue Data structure and verified the result.

Ex.No.:4 C	Implementation of Tuples and Sequences
Date:	

Aim:

To implement tuples and sequences

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application or Jupyter note book and type the coding

Step 3: Create a Tuple in different ways and do the possible operations with tuples

Step 4: Create strings for implementing sequences and do the various operations with strings

Step 5: Print the output

Step 6: End the program

CODING:

```
print("Tuples Implementation\n")
print("----------------------------\n")
T1 = ( )          #Empty Tuple
T2 = (10, 30, 20, 40, 60)              # Tuple with integer values
T3 = ("C", "Java", "Python","R")   #Tuple with string values
T4 = (501,"abc", 19.5)                 #Tuple with mixed type values
T5 = (90,)
print(T1)
print(T2)
print(T3)
print(T4)
print(T5)
print(T2+T3)
print(T2*2)
print(T3[2])
print(T3[1:4])
print('cpp' in T3)
print(6 not in T3)
list1 = [1, 2, 4, 5, 6]
print("\nTuple using List: ")
print(tuple(list1))
Tuple1 = tuple('Welcome')
print("\nTuple with the use of function: ")
print(Tuple1)
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))          # nested tuple
print("Nested Tuple")
print(my_tuple)
del T5          #Delete a tuple
```

```
print("Sum of Tuple values = "+str(sum(T2)))
print("Length of a Tuple ="+str(len(T3)))
print("Maximum of Tuple value="+str(max(T2)))
print("Minimum of Tuple value="+str(min(T2)))
print("Sorted of Tuple ="+str(sorted(Tuple1)))
print("Counting of a specific item in a Tuple="+str(T2.count(30)))
print("Index of a specific item in a Tuple="+str(T2.index(30)))
#String as Sequence
print("String as Sequence\n")
print("------------------------\n")
str1 = "JAVATPOINT"
print("Printing from starting index="+str1[0:])
print("Printing from start to specified index="+str1[:3])
print("Printing in a reverse order="+str1[::-1])
print("Printing from backwards specific index="+str1[-4:-1])
str2="Hello"
print("Concatenation of strings="+str2+str1)
print("Repetition of string="+str2*3)
```

OUTPUT:

```
Tuples Implementation
----------------------------
( )
(10, 30, 20, 40, 60, 30)
('C', 'Java', 'Python', 'R')
(501, 'abc', 19.5)
(90,)
```

(10, 30, 20, 40, 60, 30, 'C', 'Java', 'Python', 'R')

(10, 30, 20, 40, 60, 30, 10, 30, 20, 40, 60, 30)

Python

('Java', 'Python', 'R')

False

True

Tuple using List:

(1, 2, 4, 5, 6)

Tuple with the use of function:

('w', 'e', 'l', 'c', 'o', 'm', 'e')

('mouse', [8, 4, 6], (1, 2, 3))

Sum of Tuple values = 190

Length of a Tuple =4

Maximum of Tuple value=60

Minimum of Tuple value=10

Sorted of Tuple =['c', 'e', 'e', 'l', 'm', 'o', 'w']

Counting of a specific item in a Tuple=2

Index of a specific item in a Tuple=1

String as Sequence

Printing from starting index=JAVATPOINT

Printing from start to specified index=JAV

Printing in a reverse order=TNIOPTAVAJ

Printing from backwards specific index=OIN

Concatenation of strings=HelloJAVATPOINT

Repetition of string=HelloHelloHello

Ex.No.:5	Creating & Using a Module
Date:	

Aim:

To create a new module for mathematical operations and use it

Algorithm:

Step 1: Start the Program

Step 2: Open the Spyder Application and type the coding

Step 3: Create a new module(program) called mymodule that performs all possible mathematical operations

Step 4: Import mymodule in other programs and call the functions from mymodule

Step 5: Print the output

Step 6: End the program

CODING:

mymodule.py

```
import math
def circle_area(r):
   return math.pi * r**2
def ceil_no(p):
   return math.ceil(p)
def floor_no(p):
   return math.floor(p)
def fact(n):
   return math.factorial(n)
def gcd(a, b):
   return math.gcd(a, b)
def pnr(p, n, r):
   return p * math.pow(1 + n, r)
def sq_root(x):
   return math.sqrt(x)
def sine(x):
   return math.sin(x)
def cosine(x):
   return math.cos(x)
def logar(x, y):
   return math.log(x, y)
```

create new file and type this:

```
import mymodule as mx
r = int(input("Enter the radius: "))
n = mx.circle_area(r)
print("The area of the circle with radius {} = {}".format(r, n))
```

```
n = float(input("Enter a number: "))
res_ceil = mx.ceil_no(n)
res_floor = mx.floor_no(n)
print("Ceil of the number {} = {}".format(n, res_ceil))
print("Floor of the number {} = {}".format(n, res_floor))
n = int(input("Enter a number to find factorial: "))
res_fact = mx.fact(n)
print("The factorial of {}! = {}".format(n, res_fact))
n = int(input("Enter the first number to find GCD: "))
n1 = int(input("Enter the second number to find GCD: "))
res_gcd = mx.gcd(n, n1)
print("The GCD of {} and {} is = {}".format(n, n1, res_gcd))
n = int(input("Enter a number to find square root: "))
res_sqrt = mx.sq_root(n)
print("The square root of {} = {}".format(n, res_sqrt))
n = int(input("Enter the angle (degree) to find sine value: "))
res_sine = mx.sine(n)
print("The sine value of {} = {}".format(n, res_sine))
n = int(input("Enter the angle (degree) to find cosine value: "))
res_cosine = mx.cosine(n)
print("The cosine value of {} = {}".format(n, res_cosine))
n = int(input("Enter the value to find logarithm: "))
n1 = int(input("Enter the base value for logarithm: "))
res_logar = mx.logar(n, n1)
print("The log value of {} to the base {} = {}".format(n, n1, res_logar))
```

OUTPUT:

Enter the radius: 5

The area of the circle with radius 5 = 78.53981633974483

Enter a number: 5.67

Ceil of the number 5.67 = 6

Floor of the number 5.67 = 5

Enter a number to find factorial: 5

The factorial of 5! = 120

Enter the first number to find GCD: 10

Enter the second number to find GCD: 100

The GCD of 10 and 100 is = 10

Enter a number to find square root: 36

The square root of 36 = 6.0

Enter the angle (degree) to find sine value: 90

The sine value of 90 = 0.8939966636005579

Enter the angle (degree) to find cosine value: 90

The cosine value of 90 = -0.4480736161291701

Enter the value to find logarithm: 100

Enter the base value for logarithm: 10

The log value of 100 to the base 10 = 2.0

Result:

Thus the program has implemented read and write from / into files and also directory has created and deleted successfully.

Ex.No. 6	READ AND WRITE FILES, CREATE AND DELETE DIRECTORY
Date:	

AIM:

To write a program that performs to read and write files and to create and delete directories.

ALGORITHM:

STEP 1: Start the program

STEP 2: Open the spyder Application and type the code

STEP 3: Create a file in “write” mode and write the strings one by one.

STEP 4.: Open the same file in “Append” mode and append the string as a last.

STEP 5: Open the file in “read” mode and read the contents of the file.

STEP 6: close the file object.

STEP 7: Create a directory with a specific name using mkdir()

STEP 8: Delete a specified directory using rmdir()

STEP 9: Stop the execution

CODING:

```
file1 = open("myfile.txt", "w")          # write mode
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
file1.writelines(L)
file1.close()

file1 = open("myfile.txt", "a")          # append mode
file1.write("Today \n")
file1.close()

file1 = open("myfile.txt", "r")          # Read mode
print("Output of Readlines after appending")
print(file1.read())
file1.close()

file1 = open("myfile.txt", "w")          # Write-Overwrites
file1.write("Tomorrow \n")
file1.close()

file1 = open("myfile.txt", "r")
print("Output of Readlines after writing")
print(file1.read())
file1.close()

import os
directory = "pydir1"
parent_dir = "D:/python/"
path = os.path.join(parent_dir, directory)
dir_list = os.listdir(parent_dir)
print("List of directories and files before creation:")
```

```
print(dir_list)
print( )
print(“Creating a Directory\n”)

try:
   os.mkdir(path)
   print("Directory '% s' created" % directory)
except OSError as error:
   print(error)
dir_list = os.listdir(parent_dir)
print("List of directories and files after creation:")
print(dir_list)
print( )
print(“Deleting a Directory\n”)
try:
   os.rmdir(path)
   print("Directory '% s' has been removed successfully" % directory)
except OSError as error:
   print(error)
   print("Directory '% s' can not be removed" % directory)
dir_list = os.listdir(parent_dir)
print("List of directories and files after deletion:")
print(dir_list)
```

OUTPUT:

```
Output of Readlines after appending
This is Delhi
This is Paris
This is London
Today
```

Output of Readlines after writing

Tomorrow

List of directories and files before creation:

[ ]

Creating a Directory

Directory 'pydir1' created

List of directories and files after creation:

['pydir1']

Deleting a Directory

Directory 'pydir1' has been removed successfully

List of directories and files after deletion:

[ ]

Result:

Thus the program has implemented read and write from / into files and also directory has created and deleted successfully.

Ex.No. 7	EXCEPTION HANDLING
Date:	

AIM:

To write a program to handle the exceptions in python

ALGORITHM:

STEP 1: Start the program

STEP 2: Open the spyder Application and type the code

STEP 3: Create and catch ZeroDivisionError (Divide by Zero)

STEP 4: Create and catch NameError (Access undefined variable / function)

STEP 5: Create and catch TypeError (using mismatch data types)

STEP 6: Create and catch IndexError (Index out of bounds)

STEP 7: : Create and catch AttributeError (Access unavailable attribute)

STEP 8: Stop the execution

CODING:

```
def fun(a,b):
    c = ((a+b) / (a-b))
    print (c)

try:
    fun(2,3)
    fun(3,3)
except ZeroDivisionError:
    print ("a/b result in 0")
    print("ZeroDivisionError Occurred and Handled")

try:
   print("The sum=",s)
except NameError:
    print("NameError Occurred and Handled")

a=5
b='0'
try:
    print(a+b)
except TypeError:
    print('TypeError Occurred')

list1 = [2,4,6,8]
try:
    print(list1[5])
except IndexError:
    print("Index Out of Bound.")
```

```
try:
    r=list1.length
except AttributeError:
   print("List has no attribute length")
```

OUTPUT:

```
-5.0
a/b result in 0
ZeroDivisionError Occurred and Handled
NameError Occurred and Handled
TypeError Occurred
Index Out of Bound.
List has no attribute length
```

Result:

Thus the program has caught different exceptions and handled successfully.

Ex.No. 8	USING CLASSES
Date:	

AIM:

To write a program that uses classes.

ALGORITHM:

STEP 1: Start the program

STEP 2: Open the spyder Application and type the code

STEP 3: Create a parent class called Person with data members name and idnumber

STEP 4.: initialize values of data members using constructor

STEP 5: Create a class called Employee which derived from class Person

STEP 6: Create object for the derived class Employee and pass the in initial values for data members

STEP 7: call the function display() using object

STEP 8: Stop the execution

CODING;

```
class Person(object):

  def __init__(self, name, idnumber):      # __init__ is known as the constructor
      self.name = name
      self.idnumber = idnumber

    def display(self):
      print('Name = ',self.name)
      print('ID Number =',self.idnumber)

# child class
class Employee(Person):
    def __init__(self, name, idnumber, salary, post):
      self.salary = salary
      self.post = post

      # invoking the __init__ of the parent class
      Person.__init__(self, name, idnumber)

# creation of an object variable or an instance
a = Employee('Rahul', 886012, 200000, "Intern")

# calling a function of the class Person using its instance
a.display()
```

OUTPUT:

```
Name =  Rahul
ID Number = 886012
```

RESULT:

Thus the program implemented Single Inheritance using the concept of classes

Ex.No. 9	CONNECT WITH MYSQL AND CREATE ADDRESS BOOK
Date:	

AIM:

To write a program to connect with MySQL and create address book.

ALGORITHM:

Step 1: Start the program

Step 2: Import the MySQL module

Step 3: Define a function to create a table 'contacts' if it does not exist

Step 4: Define a function insert_contact to insert a new contact into the contacts table

Step 5: Define a function delete_contact to delete a contact from the contacts table based on the ID

Step 6: Define a function update_contact to update a contact in the contacts table based on the ID

Step 7: Define a function display_contacts to display all contacts in the contacts table

Step 8: Prompt the user to enter their choice

Step 9: Handle choices with if…elif…else statement

Step 11: Stop the program

CODING:

```
import MySQLdb
def create_table():
   conn = MySQLdb.connect(host="localhost", user="root", passwd="sHj@6378#jw",
db="address_book")
   cursor = conn.cursor()
   cursor.execute('''CREATE TABLE IF NOT EXISTS contacts (
                ssn INTEGER PRIMARY KEY,
                name TEXT,
                address TEXT,
                city TEXT,
                state TEXT,
                postcode INTEGER,
                country TEXT)''')
   conn.commit()
   conn.close()

def insert_contact(ssn, name, address, city, state, postcode, country):
   conn = MySQLdb.connect(host="localhost", user="root", passwd="sHj@6378#jw",
db="address_book")
   cursor = conn.cursor()
   cursor.execute("INSERT INTO contacts (ssn, name, address, city, state, postcode, country)
VALUES (%s, %s, %s, %s, %s, %s, %s)", (ssn, name, address, city, state, postcode, country))
   conn.commit()
   conn.close()

def delete_contact(ssn):
   conn = MySQLdb.connect(host="localhost", user="root", passwd="sHj@6378#jw",
db="address_book")
   cursor = conn.cursor()
   cursor.execute("DELETE FROM contacts WHERE ssn=%s", (ssn,))
```

```
    conn.commit()
    conn.close()

def update_contact(ssn, name, address, city, state, postcode, country):
    conn = MySQLdb.connect(host="localhost", user="root", passwd="sHj@6378#jw",
db="address_book")
    cursor = conn.cursor()
    cursor.execute("UPDATE contacts SET name=%s, address=%s, city=%s, state=%s,
postcode=%s, country=%s WHERE ssn=%s", (name, address, city, state, postcode, country,
ssn))
    conn.commit()
    conn.close()

def display_contacts():
    conn = MySQLdb.connect(host="localhost", user="root", passwd="sHj@6378#jw",
db="address_book")
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM contacts")
    rows = cursor.fetchall()
    for row in rows:
        print(row)
    conn.close()

if __name__ == "__main__":
    create_table()
    while True:
        print("\nADDRESS Book:")
        print("1. Add Record")
        print("2. Delete Record")
        print("3. Update Record")
        print("4. Display Record")
```

```
print("5. Exit")
choice = input("Enter your choice: ")
if choice == '1':
    ssn = input("Enter SSN NO: ")
    name = input("Enter name: ")
    address = input("Enter address: ")
    city = input("Enter city: ")
    state = input("Enter state: ")
    postcode = input("Enter postcode: ")
    country = input("Enter country: ")
    insert_contact(ssn, name, address, city, state, postcode, country)
    print("Record added successfully!")
elif choice == '2':
    ssn = input("Enter SSN of Record to delete: ")
    delete_contact(ssn)
    print("Record deleted successfully!")
elif choice == '3':
    ssn = input("Enter SSN of Record to update: ")
    name = input("Enter new name: ")
    address = input("Enter new address: ")
    city = input("Enter new city: ")
    state = input("Enter new state: ")
    postcode = input("Enter new postcode: ")
    country = input("Enter new country: ")
    update_contact(ssn, name, address, city, state, postcode, country)
    print("Record updated successfully!")
elif choice == '4':
    print("\nList of Record:")
    display_contacts()
elif choice == '5':
    print("Exiting program...")
```

```
            break
        else:
            print("Invalid choice! Please enter a valid option.")
```

OUTPUT:

```
ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 1

Enter SSN NO: 101

Enter name: KALA

Enter address: 12,Nehru street

Enter city: Trichy

Enter state: Tamil Nadu

Enter postcode: 624005

Enter country: India
Record added successfully!

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 1

Enter SSN NO: 102

Enter name: Rajesh

Enter address: 345, Junction Main Road
```

Enter city: Salem

Enter state: Tamil Nadu

Enter postcode: 636005

Enter country: India
Record added successfully!

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 4

List of Record:
(101, 'KALA', '12,Nehru street', 'Trichy', 'Tamil Nadu', 624005, 'India')
(102, 'Rajesh', '345, Junction Main Road', 'Salem', 'Tamil Nadu', 636005, 'India')

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 3

Enter SSN of Record to update: 102

Enter new name: Kishore

Enter new address: 56, first cross street

Enter new city: Salem

Enter new state: Tamil nadu

Enter new postcode: 636001

Enter new country: India
Record updated successfully!

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 4

List of Record:
(101, 'KALA', '12,Nehru street', 'Trichy', 'Tamil Nadu', 624005, 'India')
(102, 'Kishore', '56, first cross street', 'Salem', 'Tamil nadu', 636001, 'India')

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 2

Enter SSN of Record to delete: 101
Record deleted successfully!

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit

Enter your choice: 4

List of Record:
(102, 'Kishore', '56, first cross street', 'Salem', '636001', 636001, 'India')

ADDRESS Book:
1. Add Record
2. Delete Record
3. Update Record
4. Display Record
5. Exit
Enter your choice: 5

Result:

Thus, the address book is created and manipulated with MySQL and Python.

Ex.No.:10	STRING HANDLING AND REGULAR EXPRESSIONS
Date:	

AIM:

To write a program using String handling and Regular Expressions

ALGORITHM:

STEP 1: Start the program

STEP 2: Open the spyder Application and type the code

STEP 3: Import the package re for handling Regular Expressions

STEP 4: Implement all functions in 're' package like search, split, sub, compile, span, findall, etc. with required inputs.

STEP 5: Verify the result and Stop the Execution

CODING:

```
import re
pattern = 'a...s$'
test_string = 'abyss'
result = re.match(pattern, test_string)
if result:
        print("Search successful.")
else:
        print("Search unsuccessful.")
txt = "The rain in Spain"
x = re.findall("in", txt)
print(x)
x = re.search("\s", txt)
print("The first white-space character is located in position:", x.start())
x = re.split("\s", txt)
print(x)
x = re.sub("\s", "9", txt)
print(x)
x = re.sub("\s", "9", txt, 2)
print(x)
x = re.search("ai", txt)
print(x)
x = re.search(r"\bS\w+", txt)
print(x.span())
txt="guido@python.org or the older address guido@google.com."
email = re.compile('\w+@\w+\.[a-z]{3}')
email.findall(txt)
email.sub('--@--.--', txt)
email.findall('barack.obama@whitehouse.gov')
regex = re.compile('[aeiou]')
regex.split('consequential')
```

```
s = 'A computer science portal for geeks'
match = re.search('portal', s)
re.match(pattern, s)
print('Start Index:', match.start())
print('End Index:', match.end())
s = 'Readability'
pattern = r'[aeiou]'
matches = re.finditer(pattern, s)
for match in matches:
   print(match)
```

OUTPUT:

```
 Search successful.
 ['in', 'in', 'in']
 The first white-space character is located in position: 3
 ['The', 'rain', 'in', 'Spain']
 The9rain9in9Spain
 The9rain9in Spain
 <re.Match object; span=(5, 7), match='ai'>
 (12, 17)
 Start Index: 19
 End Index: 25
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='a'>
<re.Match object; span=(4, 5), match='a'>
<re.Match object; span=(6, 7), match='i'>
 <re.Match object; span=(8, 9), match='i'>
```

Result:

Thus the program implemented all functions in ‘re’ package.