



18CSC207J-Advanced Programming Practices

Record Work

Register Number: RA2011003010207

Name of the Student: SAURABH PANDEY

Semester / Year: 4th Semester / 2nd Year

Department : CSE



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUR - 603 203

BONAFIDE CERTIFICATE

Register No.: RA2011003010207

Certified to be the bonafide record of work done by Saurabh Pandey of Computer Science Engineering (Core) – C1 section, B. Tech Degree course in the Practical 18CSC207J-Advanced Programming Practices in SRM Institute of Science and Technology, Kattankulathur during the academic year 2021-2022.

Date:

Lab Incharge:

Submitted for University Examination held in SRM Institute of Science and Technology, Kattankulathur.

Examiner-1

Examiner-2

INDEX SHEET

Exp. No.	Date of Experiment	Name of the Experiment	Page No.	Marks (10)	Staff Signature
1		Structured Programming Paradigm	1-15		
2		Procedural Programming Paradigm	16-25		
3		Object Oriented Programming	26-35		
4		Event Driven Programming Paradigm	36-59		
5		Declarative Programming	60-66		
6		Parallel and Concurrent Programming	67-74		
7		Network Programming/Socket Programming	75-81		
8		Functional Programming	82-89		
9		Logic Programming	90-94		
10		Symbolic Programming	95-99		

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Structured Programming Paradigm

Problem:

Write a Python program to find those numbers which are divisible by 8 and multiple of 5, between 1000 and 2000 (both included)

Algorithm:

Step 1: Iterate in range from 1000 to 2000

Step 2: For each iteration check if the number is divisible by 5 and 8

Step 3: Print the numbers if the above condition is true.

Program:

```
n=[]                                #declaring variable
for x_882 in range(1000,2000):      #for Loop of x_882 in the range 1000 and 2000
(inclusive)
    if (x_882%8==0) and (x_882%5==0): #if x_882 is divisible by 8 and 5
        n.append(str(x_882))          #add x_882 into n
print(','.join(n))                  #after comma join the upcoming numbers
```

Input/Output

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q1.py"
1000,1040,1080,1120,1160,1200,1240,1280,1320,1360,1400,1440,1480,1520,1560,
1600,1640,1680,1720,1760,1800,1840,1880,1920,1960
```

Problem:

Write a Python program to guess a number between 1 to 9. Note : User is prompted to enter a guess. If the user guesses wrong then the prompt appears again until the guess is correct, on successful guess, user will get a "Well guessed!" message, and the program will exit.

Algorithm:

Step 1: Import random as header file

Step 2: Input a number

Step 3: Guess a number between 1 and 10 randomly using inbuilt function

Step 4: Check if input number and random is matched

Step 5: Print well guessed if condition is true

Step 6: Repeat the above process of 2.

Program:

```
import random          #importing function to generate random numbers
n_882 , guess_882 = random.randint(1,10),0      # declaring variable and generating
random int
while n_882!=guess_882:                         # while Loop till n not equal to guess
    guess_882=int(input("guess a number between 1 to 10 until u get it right:")) #convert
the input into an integer data type

print("guessed well!!")                          #prints this when the guess is right
```

Problem:

Write a Python program to construct the following pattern, using a nested for loop.

```
*
```

```
* *
```

```
* * *
```

```
* * * *
```

```
* * * * *
```

```
* * * *
```

```
* * *
```

```
* *
```

```
*
```

Algorithm:

Step 1: Break the pattern into 2 parts.

Step 2: Iterate for the above part for 5 times using i

Step 3: Iterate for in range 0 to I and print *

Step 4: Iterate for second half for 4 times.

Step 5: Iterate in reverse manner from I to 0

Step 6: print *

Program:

```
n_882=5
for i in range(n_882):                      #for Loop, i from 0 to n
    for j in range(i):                        #j from 0 to i
        print('* ',end="")
    print()

for i in range(n_882,0,-1):                  #i from 5 to 0, decreement
```

```
for j in range(i):          #j from 0 to i
    print('* ',end="")
print("")
```

Input/Output

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\tempCodeRunnerFile.py"
```

```
*
```

```
* *
```

```
* * *
```

```
* * * *
```

```
* * * * *
```

```
* * * *
```

```
* * *
```

```
*
```

Problem:

Write a Python program that accepts a word from the user and reverse it. (should not use any functions)

Algorithm:

Step 1: Input a word

Step 2: Reverse the word iterating each character and use the syntax in for loop.

Step 3: Print the word

Program:

```
word_882 = input("Input a word to reverse: ")
for char in range(len(word_882) - 1, -1, -1):  #char from Last letter till -1, decreement
    print(word_882[char], end="")                #print word the letter one after the other
print("\n")                                     #change line
```

Input/Output

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\tempCodeRunnerFile.py"
Input a word to reverse: nikhil
lihkin
```

Problem:

Write a Python program which takes two digits m (row) and n (column) as input and generates a two-dimensional array. The element value in the i-th row and j-th column of the array should be $i*j$. Note :

$i = 0, 1, \dots, m-1$

$j = 0, 1, \dots, n-1$.

Test Data : Rows = 3, Columns = 4

Expected Result : [[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]

Algorithm:

Step 1: Input rows(m) and column (n)

Step 2: Initialize a result array

Step 3: Iterate in rows(m) and for each row iterate in column (n)

Step 4: Calculating $i*j$

Step 5: print the resultant array

Program:

```
row_num = int(input("Input number of rows: ")) #number of rows and columns
col_num = int(input("Input number of columns: "))
array = [[0 for col in range(col_num)] for row in range(row_num)] #creates array
for row in range(row_num):
    for col in range(col_num):
        array[row][col]= row*col

print(array)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q5.py"
Input number of rows: 3
Input number of columns: 4
[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

Problem:

Write a Python program that accepts a string and calculate the number of digits and letters.

Algorithm:

Step 1: Input a string

Step 2: Initialize d and l as 0

Step 3: Traverse as character in string

Step 4: Check if the current character in digit

Step 5: If it in digit increment d else l

Step 6: Print d and l

Program:

```
s = input("Input a string")      #takes an input
d=l=0
for c in s:                      #for a letter/character in string
    if c.isdigit():              #using the function to check is c is a digit
        d=d+1                     #adding the value
    elif c.isalpha():             #using the function to check is c is a alpha
        l=l+1
    else:
        pass
print("Letters", l) print("Digits", d)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q6.py"
Input a stringsrmist2022
Letters 6
Digits 4
```

Problem:

Write a Python program to check the validity of password input by users.
Validation :

- At least 1 letter between [a-z] and 1 letter between [A-Z].
- At least 1 number between [0-9].
- At least 1 character from [\${#@].
- Minimum length 6 characters.
- Maximum length 16 characters.

Algorithm:

Step 1: Input a password as string

Step 2: Iterate in range 0 to string length

Step 3: Check for validation and conditions one by one

Step 4: If all are true print “Valid Password” else “Not Valid Password”

Program:

```
l, u, p, d = 0, 0, 0, 0
s = input("Input a string: ")
if (len(s) >= 8) and (len(s) <= 16):
    for i in s:
```

```

if (i.islower()):
    l+=1

if (i.isupper()):
    u+=1

if (i.isdigit()):
    d+=1

if(i=='@'or i=='$' or i=='_ ' or i=='!'):
    p+=1

if (l>=1 and u>=1 and p>=1 and d>=1 and l+p+u+d==len(s)):
    print("Valid Password")
else:
    print("Invalid Password")

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q7.py"
Input a string: srmist@2017
Invalid Password
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q7.py"
Input a string: Srmist@2022
Valid Password

```

Problem:

Write a Python program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.

Algorithm:

Step 1: Iterate from 100 to 400

Step 2: Extract each digit and check for even

Step 3: If yes, the print the digits of numbers \

Step 4: Continue with step 1 again till upon time.

Program:

```

items = []
for i in range(100, 401):
    s = str(i)
    if (int(s[0])%2==0) and (int(s[1])%2==0) and (int(s[2])%2==0):      #convert into int
and check if each digit is divisible by 2
        items.append(s)
print( ",".join(items))

```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\tempCodeRunnerFile.py"
200,202,204,206,208,220,222,224,226,228,240,242,244,246,248,260,262,264,266
,268,280,282,284,286,288,400
```

Problem:

Write a Python program to convert month name to a number of days.

Algorithm:

Step 1: Input the name of month

Step 2: Check if the month is feb, print 28/29 days

Step 3: Check if the month is Jan, March, May, August, print (31 days)

Step 4: Check if month in April ,June ,September ,November print 30 days.

Program:

```
month_name = input("Input the name of Month: ")

if month_name == "February":
    print("No. of days: 28/29 days")
elif month_name in ("April", "June", "September", "November"):
    print("No. of days: 30 days")
elif month_name in ("January", "March", "May", "July", "August", "October", "December"):
    print("No. of days: 31 day")
else:
    print("Wrong month name")
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\tempCodeRunnerFile.py"
Input the name of Month: January
No. of days: 31 day
```

Problem:

Write a Python program to sum of two given integers. However, if the sum is between 105 to 200 it will return 200.

Algorithm:

Step 1: Take 2 Integer as input

Step 2: Sum them up in sum 1
Step 3: If sum lies in range of 105 and 200
Step 4: else print sum1

Program:

```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
sum = a + b
if sum in range(105, 200):
    print(200)
else:
    print(sum)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q10.py"
Enter the first number: 106
Enter the second number: 20
200
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q10.py"
Enter the first number: 20
Enter the second number: 80
100
```

Problem:

Write a Python program to construct the following pattern, using a nested loop number.

Expected Input/Output:

```
999999999
88888888
7777777
666666
55555
4444
333
22
1
```

Algorithm:

Step 1: Take for loop and iterate for number of lives
Step 2: Iterating for 10 to i-1 in reverse direction
Step 3: Print values of 9-i

Program:

```
for i in range(9,0,-1):
```

```
print(str(i) * i)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q11.py"
99999999
88888888
7777777
666666
55555
4444
333
22
1
```

Problem:

Write a Python program to create a histogram from a given list of integers

Algorithm:

Step 1: Define a function histogram and iterate with n on items

Step 2: Take variable time and iterate for times and print *

Step 3: Reduce times by 1

Step 4: Print Output

Step 5: Call the functions with histogram [2,3,6,5]

Program:

```
def histogram( items ):
    for n in items:
        Input/Output = ''
        times = n
        while( times > 0 ):
            Input/Output += '*'
            times = times - 1
        print(Input/Output)

histogram([2, 3, 6, 5])
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q12.py"
**
***
*****
****
```

Problem:

Write a Python program that will return true if the two given integer values are equal or their sum or difference is 5.

Algorithm:

Step 1: Create a function taking two inputs

Step 2: Check if condition if $x==y$ or if absolute difference is 5 or if their sum is 5.

Step 3: If the condition is fulfilled, print true

Step 4: else print false

Program:

```
x = int(input("Enter the first number: "))
y = int(input("Enter the second number: "))
if x == y or abs(x-y) == 5 or (x+y) == 5:
    print("True")
else:
    print("False")
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q13.py"
Enter the first number: 10
Enter the second number: 25
False
```

Problem:

Write a Python program to compute the distance between the points (x_1, y_1) and (x_2, y_2) .

Algorithm:

Step 1: Input 4 integer in pairs of 2 separated by ,

Step 2: Using square root function, sum of (x_1-x_2) square and sum of (y_1-y_2) square.

Step 3: Print the distance

Program:

```
import math
p1 = [4, 0]
p2 = [6, 6]
distance = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
print(distance)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\tempCodeRunnerFile.py"
6.324555320336759
```

Problem:

Function that takes a sequence of numbers and determines whether all are different from each other

Algorithm:

Step 1: Create function test_distinct

Step 2: Create a set of the data provide.

Step 3: Set data Structure doesn't take any duplicate numbers

Step 4: If the length of set and data in same print true.

Program:

```
data=input("set of intergers from 0-9: ")
if len(data) == len(set(data)):
    print("True")
else:
    print("False")
```

Input/Output:

```
PS C:\User> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q15.py"
set of intergers from 0-9: 5
True
```

Problem:

Write a Python program to count the number of each character of a given text

Algorithm:

Step 1: Enter a string

Step 2: Iterate with n in string

Step 3: Using keys, calculate dict[n] not increase it if the key is preexisting

Step 4: Else create a new key with value 1

Step 5: Dictionary library of python is used to keep track of character.

Program:

```
test_str = input("Enter a string :")
```

```

all_freq = {}

for i in test_str:
    if i in all_freq:
        all_freq[i] += 1
    else:
        all_freq[i] = 1

print ("Count of all characters  is :\n "
       + str(all_freq))

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Py
thon-1\A1\q16.py"
Enter a string :abababba23221
Count of all characters  is :
{'a': 4, 'b': 4, '2': 3, '3': 1, '1': 1}

```

Problem:

Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive.

Algorithm:

- Step 1: Change the number to string
- Step 2: Iterate while number in positive
- Step 3: Decrement n by its sum of digits
- Step 4: Update the string number by list of string number

Program:

```

n=int(input("the number: "))
n_str = str(n)
while (n > 0):
    n -= sum([int(i) for i in list(n_str)])
    n_str = list(str(n))
    print(n)

```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q17.py"
the number: 25
25
18
9
0
```

Problem:

Write a Python program to find the digits which are absent in a given mobile number.

Algorithm:

- Step 1: Input numbers in list**
- Step 2: Sort the set in range from start to end -1**
- Step 3: Here negative indexes are used**
- Step 4: Start in 1st[0], end in 1st[-1]**

Program:

```
n=input("enter the phone number: ")
all_nums = set([0,1,2,3,4,5,6,7,8,9])
n = set([int(i) for i in n])
n = n.symmetric_difference(all_nums)
n = sorted(n)
print(n)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q18.py"
enter the phone number: 6202073290
[1, 4, 5, 8]
```

Problem:

Write a Python program to reverse the digits of a given number and add it to the original, If the sum is not a palindrome repeat this procedure

Algorithm:

- Step 1: Enter the number**
- Step 2: Take a infinite loop of while**
- Step 3: Take k and assign seeing value of n in it**
- Step 4: Check if it's a palindrome , if yes break**
- Step 5: Else reverse the number and add it to a number and update the sum.**
- Step 6: Return the value of n**

Program:

```
n=int(input("enter the number: "))
s = 0 #declare the variable sum
while True:
    k = str(n) # converting the integer to string
    if k == k[::-1]: #reversing the number
        break
    else: #if not palindrome then again reverse the number
        m = int(k[::-1]) # reverse statement
        n += m

print(n) #after Loop ends then
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q19.py"
enter the number: 25
77
```

Problem:

Write a Python program to print the length of the series and the series from the given 3rd term, 3rd last term and the sum of a series.

Algorithm:

Step 1: Input third term,third last term and sum of series.

Step 2: Calculate the integral part of the formula

Step 3: Print total number of terms

Step 4: In order to calculate the common difference, use the formula used in arithmetic progression.

Step 5: Calculating the first term

Step 6: Iterating in range of n-1, print integral part of a

Step 7: Increment a by d after every iteration

Program:

```
tn = int(input("Input third term of the series: "))
tltn = int(input("Input 3rd last term: "))
s_sum = int(input("Sum of the series: "))
n = int(2*s_sum/(tn+tltn))
print("Length of the series: ",n)

if n-5==0:
    d = (s_sum-3*tn)//6
```

```
else:  
    d = (tltn-tn)/(n-5)  
  
a = tn-2*d  
j = 0  
print("Series:")  
for j in range(n-1):  
    print(int(a),end=" ")  
    a+=d  
print(int(a),end=" ")
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A1\q20.py"  
Input third term of the series: 51  
Input 3rd last term: 20  
Sum of the series: 452  
Length of the series: 12  
Series:  
59 55 51 46 42 37 33 28 24 19 15 11
```

Result: Hence the Structured Programming has been successfully executed and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Procedural Programming Paradigm

Problem:

Given a string, find its mirroring image

Algorithm:

Step 1: Input string from user

Step 2: Lookup dictionary for all valid mirrorable English character

Step 3: find string 2 in string 1

Step 4: If string 2 in rotation of string 1 return true

Step 5: else return false

Step 6: Exit

Program:

```
#1. Given a string, find its mirroring image
def mirror(a_882):
    rev_882=a_882[::-1]
    print(rev_882)
wd_882=input("Enter A Word:")
mirror(wd_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"
Enter A Word:Nikhil
lihkiN
```

Problem:

Check if two strings are Rotationally Equivalent

Sample Input/Output

string 1 is : srmist

string 2 is : tsrmis

Are two strings Rotationally equal ? : True

Algorithm:

Step 1:Input 2 strings

Step 2: Concentrate

Step 3: If the string 2 is present is concentrate string, string 2 is rotation of string 1.

Step 4: return true

Step 5: else return false

Step 6: Exit

Program:

```
wd_882=input("Enter first word: ")
wd1_882=input("Enter second word:")
res = False
for i in range(len(wd_882)):
    if wd_882[i: ] + wd_882[ :i] == wd1_882:
        res = True
        break
print("Are two strings Rotationally equal ? : " + str(res))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"
Enter first word: sam
Enter second word:nick
Are two strings Rotationally equal ? : False
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"
Enter first word: mom
Enter second word:mom
Are two strings Rotationally equal ? : True
```

Problem:

Given a number n, the task is to generate a random binary string of length n.

Algorithm:

Step 1: Input n

Step 2: Input string and key

Step 3: Randint function used to create random “0” or “1”

Step 4: Append the randomly generated “0” and “1” to string, key.

Step 5: return string,key

Step 6: Exit

Program:

```
import random
n_882= int(input("Enter the length of binary string:"))
k_882= random.randint(1,5)
for i in range(1,n_882+1):
    if((k_882+i)%2 == 0):
        print("0",end=' ')
    else:
        print("1",end=' ')
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q3.py"
Enter the length of binary string:6
010101
```

Problem:

Given a string, remove punctuation and any special characters

Algorithm:

Step 1: Input string from user

Step 2: Look up dictionary created for all valid punctuation

Step 3: Remove any invalid parameters as you move along the string

Step 4: Print the obtained string

Step 5: Exit

Program:

```
st_882=input("Enter a string: ")
p_882='''!()[]-{};:'">\,<./?@#$%^&*_~'''
for ele in st_882:
    if ele in p_882:
        st_882=st_882.replace(ele,"")
print("String after filtering:",st_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q4.py"
Enter a string: nick@3442
String after filtering: nick3442
```

Problem:

Write a Python program to compute element-wise sum of given tuples.

Input

(11, 2, 3, 14)
(13, 5, 22, 10)

(12, 2, 3, 10)

Input/Output
(36, 9, 28, 34)

Algorithm:

Step 1: Input the tuple from user
Step 2: Store the sum in results
Step 3: result =tuple(mag(sum,zip(x,y,z)))
Step 4: print the result
Step 5: Exit

Program:

```
a_882= (1,2,3)
b_882 = (4,5,6)
c_882= (7,8,9)
print("Original list:")
print(a_882)
print(b_882)
print(c_882)
print("The sum of tuples", "")
print(tuple(map(sum, zip(a_882,b_882,c_882))))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q5.py"
Original list:
(1, 2, 3)
(4, 5, 6)
(7, 8, 9)
The sum of tuples
(12, 15, 18)
```

Problem:

Write a Python program to remove an empty tuple(s) from a list of tuples.

Algorithm:

Step 1: Input the tuple from user
Step 2: filter method is used to remove empty one.
Step 3: tuples=filter(move,tuples)
Step 4: print tuples
Step 5: Exit

Program:

```
L_882= [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]
L_882 = [i for i in L_882 if i]
print(L_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q6.py"
[('',), ('a', 'b'), ('a', 'b', 'c'), 'd']
```

Problem:

Write a Python program to count the elements in a list until an element is a tuple

Algorithm:

Step 1: Input the list from user

Step 2: Use instance() to verify whether we are encountering a tuple

Step 3: If instance (num,tuple) in true then break

Step 4: else counter ++

Step 5: print counter

Step 6: Exit

Program:

```
num_882 = [10,20,30,(10,20),40]
c_882 = 0
for i in num_882:
    if isinstance(i, tuple):
        break
    c_882+= 1
print("Number of element before tuple: ",c_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"
Number of element before tuple: 3
```

Problem:

Write a Python program to Convert Tuple Matrix to Tuple List

Sample Input : [[(9, 51), (7, 9)], [(11, 1), (22, 19)]]

Input/Output : [(9, 7, 11, 22), (51, 9, 1, 19)]

Algorithm:

Step 1: Input a list from user

Step 2: We perform list comprehension and zip() is used to perform column pairing to render as tuple pairs.

Step 3: res=list(zip(*tup))

Step 4: print the list

Step 5: Exit

Program:

```
list_882= [(9, 51), (7, 9)], [(11, 1), (22, 19)]  
  
print("The original list is : " + str(list_882))  
  
temp_882= [ele for sub in list_882 for ele in sub]  
  
res = list(zip(*temp_882))  
  
print("The converted tuple list : " + str(res))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"  
The original list is : [(9, 51), (7, 9)], [(11, 1), (22, 19)]  
The converted tuple list : [(9, 7, 11, 22), (51, 9, 1, 19)]
```

Problem:

Write a Python program to count unique values in the list.

Algorithm:

Step 1: Input list from user

Step 2: Using counter function we will create a dictionary

Step 3: The keys of dictionary will be unique items and values will be the numbers of that key present in list

Step 4: Print unique number

Step 5: Exit

Program:

```
list_882=[1,3,3,2,4,4]  
l_882=[]  
count_882=0  
for i in list_882:
```

```
if i not in l_882:  
    count_882+=1  
    l_882.append(i)  
print("No of unique items are: ",count_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q9.py"  
No of unique items are: 4
```

Problem:

Python Program to print all Possible Combinations from the three Digits

Algorithm:

Step 1: Input 3 digits

Step 2: Using internal permutation U print all the combination

Step 3: This method takes a list as an input and returning an object list of tuples that contain all permutation in a list from

Step 4: print all the combination

Step 5: Exit

Program:

```
def comb_882(L):  
  
    for i in range(3):  
        for j in range(3):  
            for k in range(3):  
                if (i!=j and j!=k and i!=k):  
                    print(L[i], L[j], L[k])  
  
print("Possible combination :")  
comb_882([1, 2, 3])
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q10.py"  
Possible combination :  
1 2 3  
1 3 2  
2 1 3  
2 3 1  
3 1 2  
3 2 1
```

Problem:

Write a Python program (using function) to print the even numbers from a given list.

Algorithm:

- Step 1: Input list from user**
- Step 2: Iterate along the array**
- Step 3: Check divisible by 2**
- Step 4: Print the number**
- Step 5 : Exit**

Program:

```
def even(a_882):  
    for i in a_882:  
        if(i % 2 == 0):  
            print(i)  
print("Even Numbers from list: ")  
l_882= [1,2,3,4,5,6]  
even(l_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q11.py"  
Even Numbers from list:  
2  
4  
6
```

Problem:

Write a Python function (using function) that checks whether a passed string is palindrome or not.

Algorithm:

- Step 1: Take string from user**
- Step 2: Iterate string as argument**
- Step 3: If last letter is equal to first character recursively call the function with argument as the sliced list with the first character and last character removed.**
- Step 4: Exit**

Program:

```
def pal(wd_882):
    k_882= wd_882[::-1]
    if(wd_882== k_882):
        print("Palindrome Number")
    else:
        print("Not a Palindrome")
wd_882= input("Enter a word :")
pal(wd_882)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q12.py"
Enter a word :madam
Palindrome Number
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\q12.py"
Enter a word :memory
Not a Palindrome
```

Problem:

Write a Python function (using function) that checks whether a given number is prime or not

Algorithm:

Step 1: Input the number from user

Step 2: Iterate from 2 to sqrt of (number)

Step 3: If we found any fact then we can print as not a prime number

Step 4: Initialize a flag that maintain states whether it is prime or not.

Program:

```
number_882= int(input("Enter any number: "))

if number_882 > 1:
    for i in range(2, number_882):
        if (number_882 % i) == 0:
            print(number_882, "is not a prime number")
            break
    else:
        print(number_882, "is a prime number")

else:
    print(number_882, "is not a prime number")
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\Python-1> python -u "c:\Users\nikhi\Downloads\Python-1\A2\tempCodeRunnerFile.py"
Enter any number: 25
25 is not a prime number
```

Result: Hence the Procedural Programming has been successfully implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Object Oriented Programming Paradigm

Problem:

Write a Python class named SRMIST with six attributes school, dept1, dept2, dept2 and dept3. Add a new attribute specialization and display the entire attribute and their values of the said class. Now remove the dept1 and dept2 attribute and display the entire attribute with values.

Algorithm:

Step 1: Create class ‘SRMIST’

Step 2: school='SRMIST' ,dep1='Computer Science', dept2="Artificial Intelligence",dep3="Mech Engineering",dep4="Biotech"

Step 3: SRMIST. Specialization= ‘Blockchain’

Step 4: Print attributes with value

Step 5: Exit

Program:

```
class SRMIST:  
    school = 'SRMIST'  
    dept1 = 'Computer Science'  
    dept2 = 'Artificial Intelligence'  
    dept3 = 'Mechanical Engineering'  
    dept4 = 'Biotech'  
    print("Original attributes and their values of the Student class:")  
    for attr, value in SRMIST.__dict__.items():  
        if not attr.startswith('_'):  
            print(f'{attr} -> {value}')  
    print("\nAfter adding the specialization, attributes and their values with the said  
class:")  
    SRMIST.specialization = 'Blockchain'  
    for attr, value in SRMIST.__dict__.items():  
        if not attr.startswith('_'):  
            print(f'{attr} -> {value}')  
    print("\nAfter removing the dept1,dept2 attributes and their values from the said  
class:")  
    del SRMIST.dept1  
    del SRMIST.dept2  
    #delattr(Student, 'student_name')  
    for attr, value in SRMIST.__dict__.items():  
        if not attr.startswith('_'):  
            print(f'{attr} -> {value}')
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab3\tempCodeRunnerFile.py"
Original attributes and their values of the Student class:
school -> SRMIST
dept1 -> Computer Science
dept2 -> Artificial Intelligence
dept3 -> Mechanical Engineering
dept4 -> Biotech

After adding the specialization, attributes and their values with the said class:
school -> SRMIST
dept1 -> Computer Science
dept2 -> Artificial Intelligence
dept3 -> Mechanical Engineering
dept4 -> Biotech
specialization -> Blockchain

After removing the dept1,dept2 attributes and their values from the said class:
school -> SRMIST
dept3 -> Mechanical Engineering
dept4 -> Biotech
specialization -> Blockchain
```

Problem:

Write a Python program to create four empty classes, CTECH, CINTEL, NWC and DSBS. Now create some instances and check whether they are instances of the said classes or not. Also, check whether the said classes are subclasses of the built-in object class or not.

Algorithm:

```
Step 1: class Ctech ;pass
        class Cintel; pass
        class NWC; pass
        class DSBS;pass
Step 2: student1=Ctech()
        Marks1=Cintel()
        Str=NWC()
        Ma=DSBS()
Step 3: print(isinstance(marks1,CINTEL))
Step 4: print(isinstance(Ctech,object))
Step 5: Exit
```

Program:

```
class CTECH:  
    pass  
class CINTEL:  
    pass  
class NWC:  
    pass  
  
class DSBS:  
    pass  
  
student1 = CTECH()  
marks1 = CINTEL()  
stu=NWC()  
ma=DSBS()  
  
print(isinstance(student1, CTECH))  
print(isinstance(marks1, CTECH))  
print(isinstance(stu, CTECH))  
print(isinstance(ma, CTECH))  
  
print(isinstance(marks1, CINTEL))  
print(isinstance(student1, CINTEL))  
print(isinstance(stu, CINTEL))  
  
print(isinstance(ma, CINTEL))  
  
print("\nCheck whether the said classes are subclasses of the built-in object class or  
not.")  
print(issubclass(CTECH, object))  
print(issubclass(CINTEL, object))  
print(issubclass(NWC, object))  
print(issubclass(DSBS, object))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab  
3\tempCodeRunnerFile.py"  
True  
False  
False  
False  
True  
False  
False  
False  
  
Check whether the said classes are subclasses of the built-in object class or not.  
True  
True  
True  
True
```

Problem:

Write a program to print the names of the departments students by creating a Dept class. If no name is passed while creating an object of the Dept class, then the name should be "SCO", otherwise the name should be equal to the String value passed while creating the object of the Dept class.

Algorithm:

Step 1: def __init__(self,*args)
Step 2: if(len args)=1
Step 3: self.dept=args[0]
Step 4: elif(len(args))==0
Step 5: self.dept='SLO'
Step 6: print self.dept
Step 7: Exit

Program:

```
class Dept:  
    def __init__(self, *args):  
        if len(args) == 1:  
            self.dept=args[0]  
  
        elif len(args) == 0:  
            self.dept="SCO"  
  
    def deptname(self):  
        print(self.dept)  
  
d1=Dept()  
d1.deptname()  
  
d2=Dept("CSE")  
d2.deptname()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab  
3\q3.py"  
SCO  
CSE
```

Problem:

Create a class named 'Rectangle' with two data members- length and breadth and a function to calculate the area which is 'length*breadth'. The class has three constructors which are :
having no parameter - values of both length and breadth are assigned zero.
having two numbers as parameters - the two numbers are assigned as length and breadth respectively.
having one number as parameter - both length and breadth are assigned that number.
Now, create objects of the 'Rectangle' class having none, one and two parameters and print their areas.

Algorithm:

Step 1: class Rectangle
Step 2: l=0,b=0
Step 3: def __init__(self,*args)
Step 4: if(len(args)==2
 Selfl=args[0]
 Selfb=args[1]
Step 5: else
Step 6: self.l=0
 Self.b=0
Step 7: area=self.l*self.b

Program:

```
class rectangle:  
    length=0  
    breadth=0  
  
    def __init__(self, *args):  
        if len(args) == 2:  
            self.length=args[0];  
            self.breadth=args[1]  
  
        elif len(args) == 1:  
            self.length=args[0]
```

```

        self.breadth=args[0]

    else:

        self.length=0

        self.breadth=0

    def area(self):

        return self.length*self.breadth;

r1=rectangle(5,10)
print(r1.area())
r2=rectangle(10)
print(r2.area())
r3=rectangle()
print(r3.area())

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab
3\q4.py"
50
100
0

```

Problem:

Create a class named 'PrintDT' to print various numbers of different datatypes by creating different functions with the same name 'python_data' having a parameter for each datatype. (example : tuple, list, string)

Algorithm:

- Step 1:** def pyd(self,list)
- Step 2:** print self.list
- Step 3:** def pyd(self,tuple)
- Step 4:** print(tuple)
- Step 5:** def pyd(self,str)
- Step 6:** print(str)
- Step 7:** p=PrintDT()
- Step 8:** Exit

Program:

```
class PrintDT:  
    def py_data(self, list):  
        self.list=[]  
        print(self.list)  
  
    def py_data(self, tuple):  
        self.tuple=()  
        print(tuple)  
  
    def py_data(self, str):  
        self.str=''  
        print(str)  
  
p=PrintDT()  
p.py_data([1,2,3])  
p.py_data(('a',[8,4,6], "mouse"))  
p.py_data('amit')
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab  
3\tempCodeRunnerFile.py"  
[1, 2, 3]  
('a', [8, 4, 6], 'mouse')  
amit
```

Problem:

A student from SRMIST has his/her money deposited Rs.15000, Rs.30000 and Rs. 40,000 in banks-CUB, HDFC and Indian Bank respectively. We have to print the money deposited by him/her in a particular bank.

Create a class named 'Banks_SRMIST' with a function 'getBalance' which returns 0. Make its three subclasses named 'CUB', 'HDFC' and 'Indian_Bank' with a function with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the function 'getBalance' by the object of each of the three banks.

Algorithm:

```
Step 1: class Bank_Srmist  
Step 2: def getBalance()  
Step 3: class WB(Banks_Srmist)  
Step 4: def getBalance (balance)  
Step 5: return balance  
Step 6: class HDFC(Bank_Srmist)  
Step 7: def getBalance(balance)
```

Return balance

Step 8: print

Step 9: Exit

Program:

```
class Banks_SRMIST:  
    def getBalance():  
        return 0  
class CUB(Banks_SRMIST):  
  
    def getBalance(balance):  
        return balance  
class HDFC(Banks_SRMIST):  
  
    def getBalance(balance):  
        return balance  
class Indian_Bank(Banks_SRMIST):  
  
    def getBalance(balance):  
        return balance  
Banks_SRMIST()  
print(CUB.getBalance(15000))  
print(HDFC.getBalance(30000))  
print(Indian_Bank.getBalance(40000))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab  
3\tempCodeRunnerFile.py"  
15000  
30000  
40000
```

Problem:

Create a Time class and initialize it with hours and minutes.

Make a method addTime which should take two time object and add them. E.g.- (2 hour and 50 min)+(1 hr and 20 min) is (4 hr and 10 min)

Make a method displayTime which should print the time.

Make a method DisplayMinute which should display the total minutes in the Time. E.g.- (1 hr 2 min) should display 62 minute.

Algorithm:

Step 1: create class Time
Step 2: def __init__(self,hours,min)
Step 3: self.hours=hours
Step 4: self.mins=mins
Step 5: def additive(t1,t2)
Step 6: t3.hours=t1.hours+t2.hours
Step 7: print
Step 8: Exit

Program:

```
class Time():

    def __init__(self, hours, mins):
        self.hours = hours
        self.mins = mins

    def addTime(t1, t2):
        t3 = Time(0,0)
        if t1.mins+t2.mins > 60:
            t3.hours = (t1.mins+t2.mins)//60
        t3.hours = t3.hours+t1.hours+t2.hours
        t3.mins = (t1.mins + t2.mins) % 60
        return t3

    def displayTime(self):
        print ("Time is",self.hours,"hours and",self.mins,"minutes.")

    def displayMinute(self):
        print ((self.hours*60)+self.mins)

a = Time(2,40)
b = Time(1,30)
c = Time.addTime(a,b)
c.displayTime()
c.displayMinute()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab
3\tempCodeRunnerFile.py"
Time is 4 hours and 10 minutes.
250
```

Problem:

Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter.

Algorithm:

Step 1: Create class Traingle
Step 2: def findPerimeter(self,s1,s2,s3)
Step 3: findArea(self,s1,s2,s3)
Step 4: p=s1+s2+s3
Step 5: s=p/2
Step 6: area=(0.5**(s*(s-s1)*(s-s2)*(s-s3)))
Step 7: print
Step 8: Exit

Program:

```
class Triangle:  
    def findPerimeter(self, s1, s2, s3):  
        return (s1 + s2 + s3)  
  
    def findArea(self, s1, s2, s3):  
        p = (s1 + s2 + s3)  
        s = p/2  
        return (s * (s-s1) * (s-s2)*(s-s3))**0.5  
  
s1 = float(input("Enter the first side of the triangle : "))  
s2 = float(input("Enter the second side of the triangle : "))  
s3 = float(input("Enter the third side of the triangle : "))  
  
u = Triangle()  
  
print("The perimeter of the triangle is : {0:.2f}".format(  
    u.findPerimeter(s1, s2, s3)))  
print("The area of the triangle is : {0:.2f}".format(u.findArea(s1, s2, s3)))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab  
3\tempCodeRunnerFile.py"  
Enter the first side of the triangle : 20  
Enter the second side of the triangle : 14  
Enter the third side of the triangle : 12  
The perimeter of the triangle is : 46.00  
The area of the triangle is : 82.65
```

Result: Hence the Object Oriented Programming has been implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Event Driven Programming Paradigm

Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

Step 5: Call the mainloop

Step 6: Exit

Program:

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox

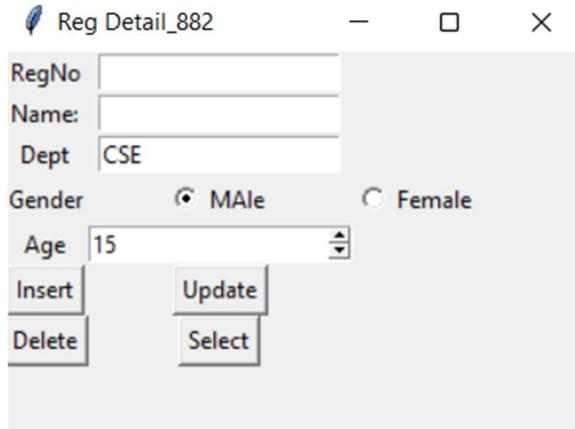
def hello1():
    msg = messagebox.show_882info( "Confirmation","Inserted Sucessfully")
def hello2():
    msg = messagebox.show_882info( "Confirmation","Updated Sucessfully")
def hello3():
    msg = messagebox.show_882info( "Confirmation","Deleted Sucessfully")
def hello4():
    msg = messagebox.show_882info( "Confirmation","Select Button")
root_882=tk.Tk()
root_882.title('Reg Detail_882')
root_882.geometry('300x200')
l1_882=tk.Label(root_882,text="RegNo")
l1_882.grid(row=0)
t1_882=tk.Entry(root_882)
t1_882.grid(row=0,column=1)
l2_882=tk.Label(root_882,text="Name:")
l2_882.grid(row=1)
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1)
v = StringVar(root_882, value='CSE')
l3_882=tk.Label(root_882,text="Dept")
l3_882.grid(row=2)
t3_882=tk.Entry(root_882,textvariable=v)
t3_882.grid(row=2,column=1)
l4_882=tk.Label(root_882,text="Gender")
l4_882.grid(row=3)
radio1_882=IntVar()
```

```

radio2_882=IntVar()
rb1_882=Radiobutton(root_882,text='MAle',variable=radio1_882,value=0)
rb1_882.grid(row=3,column=1)
rb2_882=Radiobutton(root_882,text='Female',variable=radio1_882,value=1)
rb2_882.grid(row=3,column=2)
l5_882=tk.Label(root_882,text="Age")
l5_882.grid(row=4)
spin_882 = Spinbox(root_882,from_=15,to=20)
spin_882.grid(row=4,column=1)
b1_882=tk.Button(root_882, text='Insert',command=hello1)
b1_882.grid(row=5,column=0)
b2_882=tk.Button(root_882, text='Update',command=hello2)
b2_882.grid(row=5,column=1)
b3_882=tk.Button(root_882, text='Delete',command=hello3)
b3_882.grid(row=6,column=0)
b4_882=tk.Button(root_882, text='Select',command=hello4)
b4_882.grid(row=6,column=1)
root_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```

import tkinter as tk
from tkinter import *

```

```

from tkinter import messagebox

def hello1():
    msg = messagebox.show_882info( "Confirmation","Inserted Sucessfully")
def hello2():
    msg = messagebox.show_882info( "Confirmation","Updated Sucessfully")
def hello3():
    msg = messagebox.show_882info( "Confirmation","Deleted Sucessfully")
def hello4():
    msg = messagebox.show_882info( "Confirmation","Select Button")
root_882=tk.Tk()
root_882.title('tk')
root_882.geometry('300x200')
l1_882=tk.Label(root_882,text="Custid")
l1_882.grid(row=0)
t1_882=tk.Entry(root_882)
t1_882.grid(row=0,column=1)
l2_882=tk.Label(root_882,text="Customer Name:")
l2_882.grid(row=1)
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1)

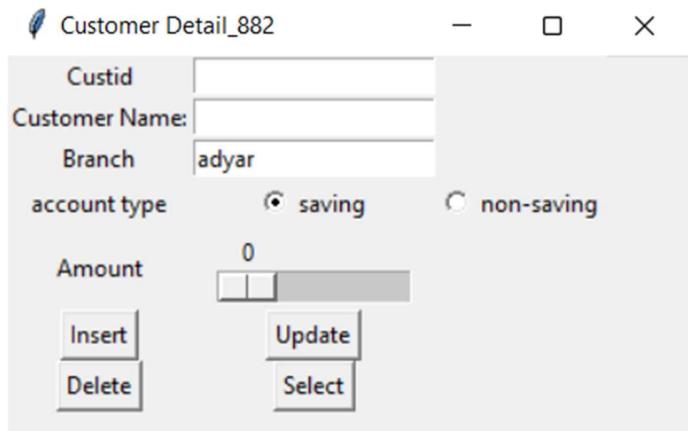
```

```

v = StringVar(root_882, value='adyar')
l3_882=tk.Label(root_882,text="Branch")
l3_882.grid(row=2)
t3_882=tk.Entry(root_882,textvariable=v)
t3_882.grid(row=2,column=1)
l4_882=tk.Label(root_882,text="account type")
l4_882.grid(row=3)
radio1_882=IntVar()
radio2_882=IntVar()
rb1_882=Radiobutton(root_882,text='saving',variable=radio1_882,value=0)
rb1_882.grid(row=3,column=1)
rb2_882=Radiobutton(root_882,text='non-saving',variable=radio1_882,value=1)
rb2_882.grid(row=3,column=2)
l5_882=tk.Label(root_882,text="Amount")
l5_882.grid(row=4)
w_882 = Scale(root_882, from_=0, to=200, orient=HORIZONTAL)
w_882.grid(row=4,column=1)
b1_882=tk.Button(root_882, text='Insert',command=hello1)
b1_882.grid(row=5,column=0)
b2_882=tk.Button(root_882, text='Update',command=hello2)
b2_882.grid(row=5,column=1)
b3_882=tk.Button(root_882, text='Delete',command=hello3)
b3_882.grid(row=6,column=0)
b4_882=tk.Button(root_882, text='Select',command=hello4)
b4_882.grid(row=6,column=1)
root_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```
import tkinter as tk
from tkinter import *

root_882=tk.Tk()
root_882.title('Employee Detail_882')
l1_882=tk.Label(root_882,text="Empid")
l1_882.grid(row=0)
t1=tk.Entry(root_882)
t1.grid(row=0,column=1)
l2=tk.Label(root_882,text="Employee Name:")
l2.grid(row=1)
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1)
l3_882=tk.Label(root_882,text="Job")
l3_882.grid(row=2)
t3_882=tk.Entry(root_882)
t3_882.grid(row=2,column=1)
empty=IntVar()
def emtype():
    print(empty.get())
g=tk.Label(root_882,text="Employee type")
g.grid(row=3,column=0)
```

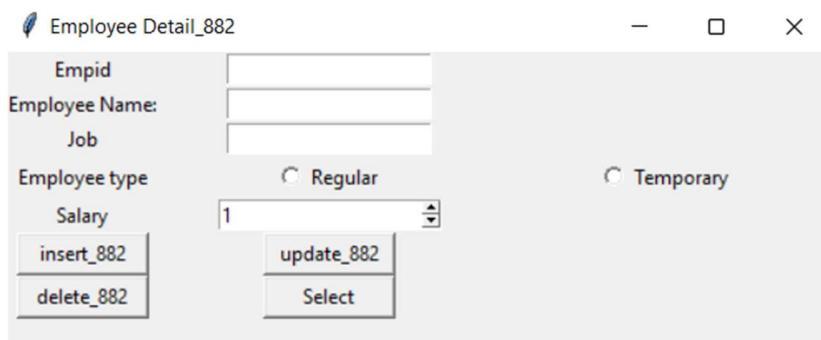
```

g1=RadioButton(root_882,text="Regular",variable=emty,width=25,value=1,command=emtype)
g1.grid(row=3,column=1)
g2=RadioButton(root_882,text="Temporary",variable=emty,width=25,value=2,command=emtype)
g2.grid(row=3,column=2)
age=tk.Label(root_882,text="Salary")
age.grid(row=4,column=0)
age=Spinbox(root_882,from_=1, to = 30)
age.grid(row=4,column=1)
insert_882=tk.Button(root_882,text="insert_882",width=10,command=root_882.destroy)
insert_882.grid(row=5,column=0)
update_882=tk.Button(root_882,text="update_882",width=10,command=root_882.destroy)
update_882.grid(row=5,column=1)
delete_882=tk.Button(root_882,text="delete_882",width=10,command=root_882.destroy)
delete_882.grid(row=6,column=0)
select=tk.Button(root_882,text="Select",width=10,command=root_882.destroy)
select.grid(row=6,column=1)

root_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```

import tkinter as tk
from tkinter import *

root_882=tk.Tk()
root_882.title('Booking_882')
l1_882=tk.Label(root_882,text="bookingid")

```

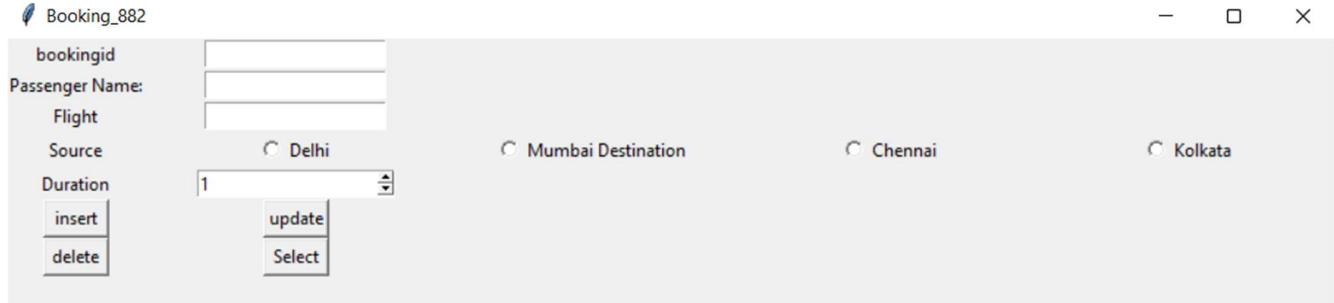
```

l1_882.grid(row=0)
t1_882=tk.Entry(root_882)
t1_882.grid(row=0,column=1)
l2_882=tk.Label(root_882,text="Passenger Name:")
l2_882.grid(row=1)
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1)
l3_882=tk.Label(root_882,text="Flight")
l3_882.grid(row=2)
t3_882=tk.Entry(root_882)
t3_882.grid(row=2,column=1)
fli=IntVar()
def flidis():
    print(fli.get())
g=tk.Label(root_882,text="Source")
g.grid(row=3,column=0)
g1=Radiobutton(root_882,text="Delhi",variable=fli,width=25,value=1,command=flidis)
g1.grid(row=3,column=1)
g2=Radiobutton(root_882,text="Mumbai
Destination",variable=fli,width=25,value=2,command=flidis)
g2.grid(row=3,column=2)
g2=Radiobutton(root_882,text="Chennai",variable=fli,width=25,value=3,command=flidis)
g2.grid(row=3,column=3)
g2=Radiobutton(root_882,text="Kolkata",variable=fli,width=25,value=4,command=flidis)
g2.grid(row=3,column=4)
age=tk.Label(root_882,text="Duration")
age.grid(row=4,column=0)
age=Spinbox(root_882,from_=1, to = 30)
age.grid(row=4,column=1)
insert_882=tk.Button(root_882,text="insert",width=5,command=root_882.destroy)
insert_882.grid(row=5,column=0)
update_882=tk.Button(root_882,text="update",width=5,command=root_882.destroy)
update_882.grid(row=5,column=1)
delete_882=tk.Button(root_882,text="delete",width=5,command=root_882.destroy)
delete_882.grid(row=6,column=0)
select=tk.Button(root_882,text="Select",width=5,command=root_882.destroy)
select.grid(row=6,column=1)

root_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```
import tkinter as tk
from tkinter import *

root_882=tk.Tk()
root_882.geometry("750x250")
root_882.title('Movie Booking_882')
l1_882=tk.Label(root_882,text="Movie booking id")
l1_882.grid(row=0)
t1_882=tk.Entry(root_882)
t1_882.grid(row=0,column=1)
l2_882=tk.Label(root_882,text="Person Name:")
l2_882.grid(row=1)
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1)
l3_882=tk.Label(root_882,text="Movie Name")
l3_882.grid(row=2)
t3_882=tk.Entry(root_882)
t3_882.grid(row=2,column=1)
boo=IntVar()
def tick():
    print(boo.get())
g=tk.Label(root_882,text="Class")
g.grid(row=3,column=0)
g1=Radiobutton(root_882,text="A",variable=boo,width=25,value=1,command=tick)
g1.grid(row=3,column=1)
g2=Radiobutton(root_882,text="B",variable=boo,width=25,value=2,command=tick)
```

```

g2.grid(row=3, column=2)
g=tk.Label(root_882, text="Time of Show")
g.grid(row=3, column=3)
g1=Checkbutton(root_882, text='7:15 pm')
g1.grid(row=3, column=4)
g2=Checkbutton(root_882, text='9 am')
g2.grid(row=3, column=5)
age=tk.Label(root_882, text="No. of Tickets")
age.grid(row=4, column=0)
age=Scale(root_882, from_=1, to= 20, orient=HORIZONTAL)
age.grid(row=4, column=1)
insert_882=tk.Button(root_882, text="insert", width=5, command=root_882.destroy)
insert_882.grid(row=5, column=0)
update_882=tk.Button(root_882, text="update", width=5, command=root_882.destroy)
update_882.grid(row=5, column=1)
delete_882=tk.Button(root_882, text="delete", width=5, command=root_882.destroy)
delete_882.grid(row=6, column=0)
select=tk.Button(root_882, text="Select", width=5, command=root_882.destroy)
select.grid(row=6, column=1)

root_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package**
- Step 2: define functions that works on button actions**
- Step 3: Create multiple labels and Text fields according to the requirement**
- Step 4: Call the defined functions in the command methods of button.**
- Step 5: Call the mainloop**
- Step 6: Exit**

Program:

```
import tkinter as tk
from tkinter import *

root_882=tk.Tk()
root_882.geometry("400x200")
root_882.title('Annual Rate_882')
l1_882=tk.Label(root_882,text="Annual Rate:")
l1_882.grid(row=0,sticky='w')
t1_882=tk.Entry(root_882)
t1_882.grid(row=0,column=1,sticky='w')
l2_882=tk.Label(root_882,text="Number of Payments:")
l2_882.grid(row=1,sticky='w')
t2_882=tk.Entry(root_882)
t2_882.grid(row=1,column=1,sticky='w')
l3_882=tk.Label(root_882,text="Loan Principle:")
l3_882.grid(row=2,sticky='w')
```

```
t3_882=tk.Entry(root_882)
t3_882.grid(row=2,column=1,sticky='w')
l2_882=tk.Label(root_882,text="Monthly Payments:")
l2_882.grid(row=3,sticky='w')
t2_882=tk.Entry(root_882)
t2_882.grid(row=3,column=1,sticky='w')
l2_882=tk.Label(root_882,text="Remaining Loan:")
l2_882.grid(row=4,sticky='w')
t2_882=tk.Entry(root_882)
t2_882.grid(row=4,column=1,sticky='w')
final=tk.Button(root_882,text="Final balance",width=10,command=root_882.destroy)
final.grid(row=5,column=0,sticky='w')
monthly=tk.Button(root_882,text="Monthly Payment",width=15,command=root_882.destroy)
monthly.grid(row=5,column=1)
qui=tk.Button(root_882,text="Quit",width=5,command=root_882.destroy)
qui.grid(row=5,column=2,sticky=tk.N)
root_882.mainloop()
```

Input/Output:



Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

Step 5: Call the mainloop

Step 6: Exit

Program:

```
from tkinter import *

master = Tk()
myText=StringVar()
master.title('Form_882')
Label(master, text="Form",bg='light green').grid(row=0,column=1)
Label(master, text="Name",bg='light green').grid(row=1)
Label(master, text="Course",bg='light green').grid(row=2)
Label(master, text="Semester,",bg='light green').grid(row=3)
Label(master, text="Form No.",bg='light green').grid(row=4)
Label(master, text="Contact No.",bg='light green').grid(row=5)
Label(master, text="Email id.",bg='light green').grid(row=6)
Label(master, text="Address",bg='light green').grid(row=7)
master.configure(bg='light green')

e1_882 = Entry(master)
e2_882 = Entry(master)
e3_882 = Entry(master)
e4_882 = Entry(master)
e5_882 = Entry(master)
e6_882 = Entry(master)
e7_882 = Entry(master)

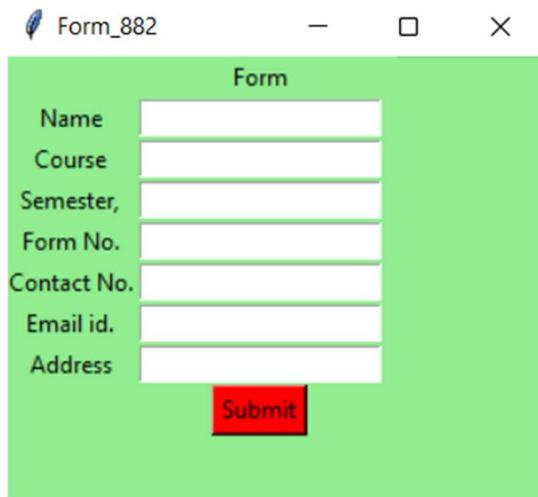
e1_882.grid(row=1, column=1)
e2_882.grid(row=2, column=1)
```

```

e3_882.grid(row=3, column=1)
e4_882.grid(row=4, column=1)
e5_882.grid(row=5, column=1)
e6_882.grid(row=6, column=1)
e7_882.grid(row=7, column=1)
b = Button(master, text="Submit",bg='RED')
b.grid(row=8,column=1)
mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```

from tkinter import *

expression = ""

def press(num):
    global expression
    expression = expression + str(num)
    equation.set(expression)

```

```

def equalpress():
    try:
        global expression
        total = str(eval(expression))
        equation.set(total)
        expression = ""
    except:
        equation.set(" error ")
        expression = ""

def clear():
    global expression
    expression = ""
    equation.set("0")

if __name__ == "__main__":
    gui = Tk()
    gui.title("Calculator_882")
    gui.geometry("270x150")
    equation = StringVar()
    expression_field = Entry(gui, textvariable=equation)
    expression_field.grid(columnspan=4, ipadx=70)
    clrscr = Button(gui, text=' C ', fg='black', bg='grey',
                    command=clear, height=1, width=7)
    clrscr.grid(row=2, column=0)

    squareRoot = Button(gui, text=' √ ', fg='black', bg='grey',
                         command=lambda: press('√'), height=1, width=7)
    squareRoot.grid(row=2, column=1)

    exponent = Button(gui, text=' x^y ', fg='black', bg='grey',
                      command=lambda: press('^'), height=1, width=7)
    exponent.grid(row=2, column=2)

    percent = Button(gui, text=' % ', fg='black', bg='grey',
                     command=lambda: press('%'), height=1, width=7)

    percent.grid(row=2, column=3)

    num1 = Button(gui, text=' 1 ', fg='black',
                  command=lambda: press(1), height=1, width=7)
    num1.grid(row=3, column=0)

    num2 = Button(gui, text=' 2 ', fg='black',
                  command=lambda: press(2), height=1, width=7)
    num2.grid(row=3, column=1)

```

```

num3 = Button(gui, text=' 3 ', fg='black',
             command=lambda: press(3), height=1, width=7)
num3.grid(row=3, column=2)

plus = Button(gui, text=' + ', fg='black', bg='grey',
              command=lambda: press('+'), height=1, width=7)
plus.grid(row=3, column=3)

num4 = Button(gui, text=' 4 ', fg='black',
              command=lambda: press(4), height=1, width=7)
num4.grid(row=4, column=0)

num5 = Button(gui, text=' 5 ', fg='black',
              command=lambda: press(5), height=1, width=7)
num5.grid(row=4, column=1)

num6 = Button(gui, text=' 6 ', fg='black',
              command=lambda: press(6), height=1, width=7)
num6.grid(row=4, column=2)

minus = Button(gui, text=' - ', fg='black', bg='grey',
               command=lambda: press("-"), height=1, width=7)
minus.grid(row=4, column=3)

num7 = Button(gui, text=' 7 ', fg='black',
              command=lambda: press(7), height=1, width=7)
num7.grid(row=5, column=0)

num8 = Button(gui, text=' 8 ', fg='black',
              command=lambda: press(8), height=1, width=7)
num8.grid(row=5, column=1)

num9 = Button(gui, text=' 9 ', fg='black',
              command=lambda: press(9), height=1, width=7)
num9.grid(row=5, column=2)

multiply = Button(gui, text=' * ', fg='black', bg='grey',
                  command=lambda: press('*'), height=1, width=7)
multiply.grid(row=5, column=3)

num0 = Button(gui, text=' 0 ', fg='black',
              command=lambda: press(0), height=1, width=7)
num0.grid(row=6, column=0)

decimal = Button(gui, text=' . ', fg='black',
                  command=lambda: press('.'), height=1, width=7)
decimal.grid(row=6, column=1)

```

```

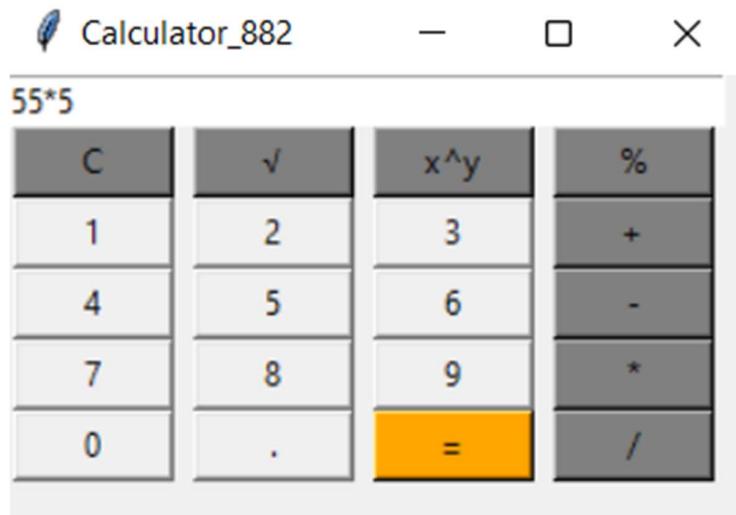
equals = Button(gui, text=' = ', fg='black', bg='orange',
               command=equalpress, height=1, width=7)
equals.grid(row=6, column=2)

divide = Button(gui, text=' / ', fg='black', bg='grey',
                command=lambda: press('/'), height=1, width=7)
divide.grid(row=6, column=3)

# start the GUI
gui.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```

from tkinter import *
from tkinter import ttk
root = Tk()
root.title("Phone list_882")
root.geometry("350x320")
root.resizable(0, 0)
root.configure(background="#7F7F7F")

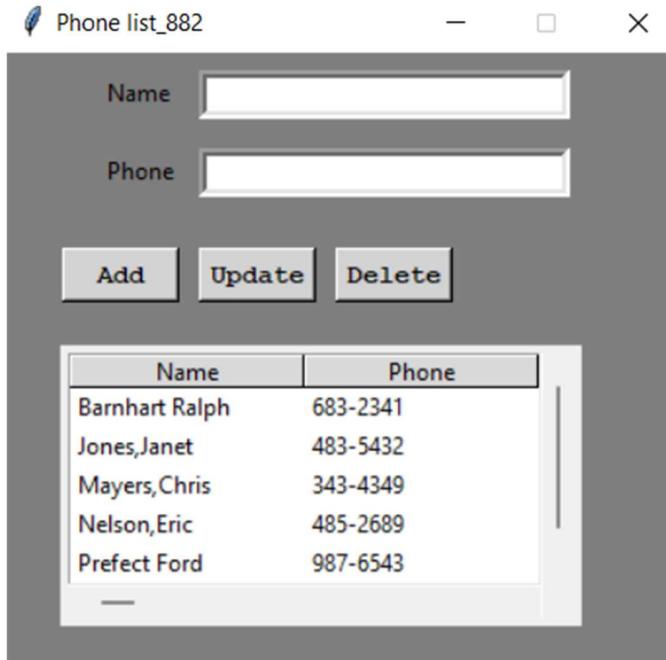
```

```

name = Label(root, text="Name",background="#7F7F7F").place(x=51, y=10)
nameEntry = Entry(root, width=30,bd=4).place(x=101, y=9)
phone = Label(root, text="Phone", background="#7F7F7F").place(x=51, y=50)
phoneEntry = Entry(root, width=30,bd=4).place(x=101, y=49)
add = Button(root, text="Add", bg="light grey",padx=13,font=("Courier New",10,'bold')).place(x=31, y=100)
update = Button(root, text="Update",bg="light grey",font=("Courier New",10,'bold')).place(x=101, y=100)
delete = Button(root, text="Delete", bg="light grey",font=("Courier New",10,'bold')).place(x=171, y=100)
f1=Frame(root,bd=4)
f1.pack()
sb=Scrollbar(f1)
sb.pack(side=RIGHT,fill=Y)
sb2= Scrollbar(f1, orient="horizontal")
sb2.pack(side=BOTTOM, fill=X)
f1.place(x=30, y=150)
tree = ttk.Treeview(f1,height=5,columns=(1,2),show='headings')
sb.configure(command=tree.yview)
tree.configure(yscrollcommand=sb.set)
tree.column(1,width=120)
tree.column(2,width=120)
tree.heading(1, text="Name")
tree.heading(2, text="Phone")
tree.insert(parent="", index=0, values=("Barnhart Ralph", "683-2341"))
tree.insert(parent="", index=1, values=("Jones,Janet", "483-5432"))
tree.insert(parent="", index=2, values=("Mayers,Chris", "343-4349"))
tree.insert(parent="", index=3, values=("Nelson,Eric", "485-2689"))
tree.insert(parent="", index=4, values=("Prefect Ford", "987-6543"))
tree.insert(parent="", index=5, values=("Smith,Bob", "689-1234"))
tree.insert(parent="", index=6, values=("Smith,Robert", "689-1234"))
tree.pack(side=LEFT)
s = ttk.Style()
s.theme_use("default")
s.map("Treeview")
root.mainloop()

```

Input/Output:



Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

Step 5: Call the mainloop

Step 6: Exit

Program:

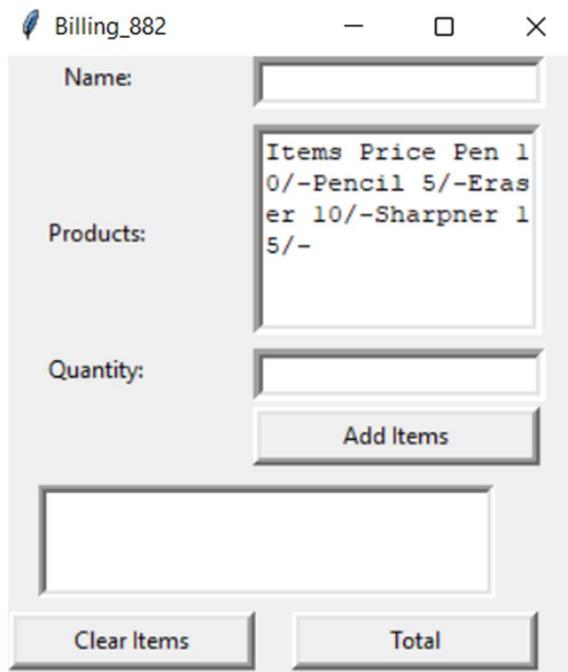
```
from tkinter import *
r=Tk()
r.title("Billing_882")
r.geometry("285x325")
label1=Label(r,text="Name:")
label1.place(x=30,y=0)
entry1=Entry(r,bd=5,width=23)
entry1.place(x=130,y=0)
text1=Text(r,height=6,width=17,bd=5)
text1.place(x=130,y=35)
s1 = """Items Price"""
s2 = """
s3 = """Pen 10/-"""
s4 = """Pencil 5/-"""
s5 = """Eraser 10/-"""
s6 = """Sharpner 15/-"""
```

```

text1.insert(END,s1)
text1.insert(END,s2)
text1.insert(END,s3)
text1.insert(END,s4)
text1.insert(END,s5)
text1.insert(END,s6)
entry2=Entry(r,bd=5,width=23)
entry2.place(x=130,y=150)
label2=Label(r,text="Products:")
label2.place(x=22,y=80)
label3=Label(r,text="Quantity:")
label3.place(x=22,y=150)
button1=Button(r,text="Add Items",width=19,bd=4)
button1.place(x=130,y=180)
text2=Text(r,height=3,width=28,bd=4)
text2.place(x=20,y=220)
button2=Button(r,text="Clear Items",bd=4,width=16)
button2.place(x=5,y=285)
button3=Button(r,text="Total",bd=4,width=16)
button3.place(x=150,y=285)
r.mainloop()

```

Input/Output:



Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

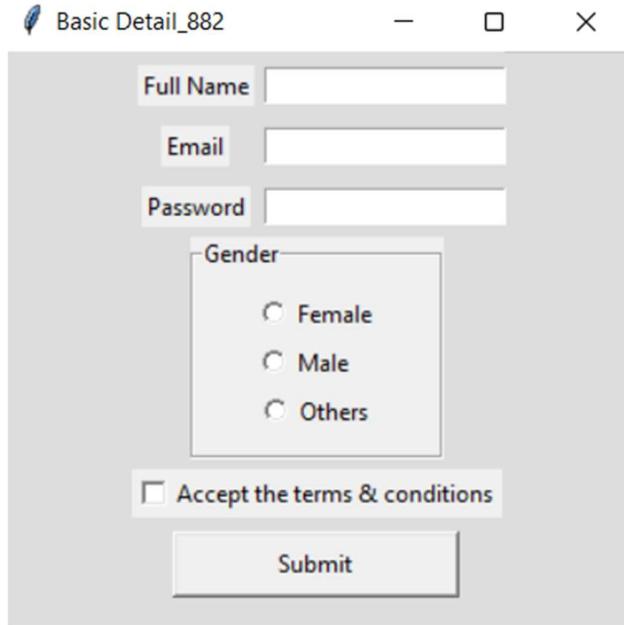
Step 5: Call the mainloop

Step 6: Exit

Program:

```
from tkinter import *
ws_882 =Tk()
ws_882.title('Basic Detail_882')
ws_882.geometry('250x300')
ws_882.configure(bg='#dddddd')
frame1_882 = Label(ws_882, bg='#dddddd')
frame1_882.pack()
frame2_882 = LabelFrame(frame1_882, text='Gender', padx=30, pady=10)
var =IntVar()
cb = IntVar()
Label(frame1_882, text='Full Name').grid(row=0, column=0, padx=5, pady=5)
Label(frame1_882, text='Email').grid(row=1, column=0, padx=5, pady=5)
Label(frame1_882, text='Password').grid(row=2, column=0, padx=5, pady=5)
Radiobutton(frame2_882, text='Female', variable=var, value=1).pack()
Radiobutton(frame2_882, text='Male', variable=var, value=2).pack(anchor=W)
Radiobutton(frame2_882, text='Others', variable=var, value=3).pack()
name_Tf = Entry(frame1_882)
name_Tf.grid(row=0, column=2)
Entry(frame1_882).grid(row=1, column=2)
Entry(frame1_882, show="*").grid(row=2, column=2)
frame2_882.grid(row=3, columnspan=3,padx=30)
Checkbutton(frame1_882, text='Accept the terms & conditions', variable=cb, onvalue=1, offvalue=0).grid(row=4, columnspan=4, pady=5)
submit_btn = Button(frame1_882, text="Submit", padx=50, pady=5)
submit_btn.grid(row=5, columnspan=4, pady=2)
ws_882.mainloop()
```

Input/Output:



Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

Step 5: Call the mainloop

Step 6: Exit

Program:

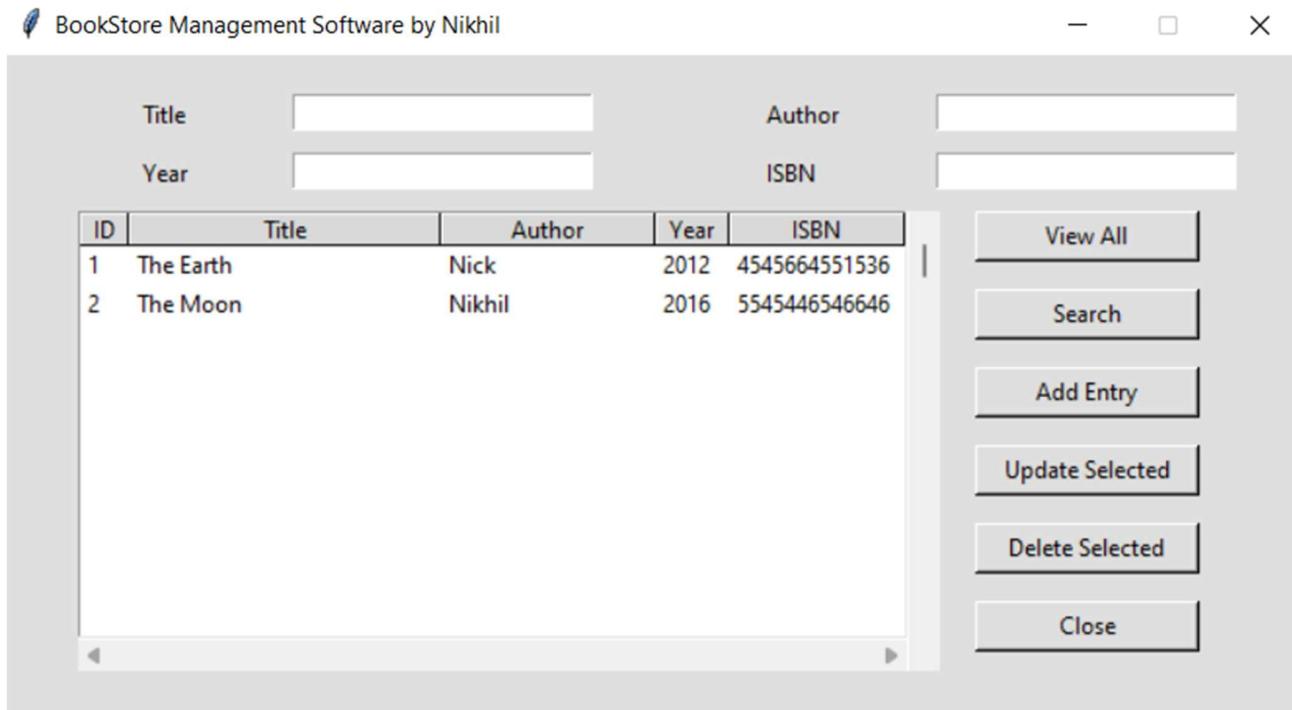
```
from tkinter import *
from tkinter import ttk
from tkinter import font
root = Tk()
root.geometry("670x340")
root.title("BookStore Management Software by Nikhil")
root.resizable(0, 0)
root.configure(bg="#DFDFDE")
f = "Helvetica 13 bold"
title = Label(root, text="Title",bg="#DFDFDE").place(x=70, y=20)
titleEntry = Entry(root,width=25).place(x=150, y=20)
year = Label(root, text="Year",bg="#DFDFDE").place(x=70, y=50)
yearEntry = Entry(root,width=25).place(x=150, y=50)
author = Label(root, text="Author",bg="#DFDFDE").place(x=390, y=20)
authorEntry = Entry(root,width=25).place(x=480, y=20)
isbn = Label(root, text="ISBN",bg="#DFDFDE").place(x=390, y=50)
isbnEntry = Entry(root,width=25).place(x=480, y=50)
viewAll = Button(root, text="View All", width=15,bg="#DFDFDE").place(x=500, y=80)
```

```

search = Button(root, text="Search", width=15,bg="#DFDFDE").place(x=500, y=120)
addEntry = Button(root, text="Add Entry", width=15,bg="#DFDFDE").place(x=500, y=160)
updateSelected = Button(root, text="Update Selected", width=15,bg="#DFDFDE").place(x=500,
y=200)
deleteSelected = Button(root, text="Delete Selected", width=15,bg="#DFDFDE").place(x=500,
y=240)
close = Button(root, text="Close", width=15,bg="#DFDFDE").place(x=500, y=280)
data_frame = Frame(root)
data_frame.pack()
# scrollbar
data_scroll = Scrollbar(data_frame)
data_scroll.pack(side=RIGHT, fill=Y)
data_scroll = Scrollbar(data_frame, orient="horizontal")
data_scroll.pack(side=BOTTOM, fill=X)
frame = ttk.Treeview(
data_frame, yscrollcommand=data_scroll.set, xscrollcommand=data_scroll.set
)
style = ttk.Style()
style.configure("Treeview.Heading", font="Ariel 10 bold")
frame.pack()
data_frame.place(x=40, y=80)
data_scroll.config(command=frame.yview)
data_scroll.config(command=frame.xview)
# define our column
frame["columns"] = ("ID", "Title", "Author", "Year", "ISBN")
frame.column("#0", width=0, stretch=NO)
frame.column("ID", width=25)
frame.column("Title", width=160)
frame.column("Author", width=110)
frame.column("Year", width=38)
frame.column("ISBN", width=90)
frame.heading("#0", text="")
frame.heading("ID", text="ID")
frame.heading("Title", text="Title")
frame.heading("Author", text="Author")
frame.heading("Year", text="Year")
frame.heading("ISBN", text="ISBN")
frame.insert(
parent="", index=0, values=("1", "The Earth", "Nick", "2012", "4545664551536")
)
frame.insert(
parent="", index=1, values=("2", "The Moon", "Nikhil", "2016", "5545446546646")
)
s = ttk.Style()
s.theme_use("default")
s.map("Treeview")
root.mainloop()

```

Input/Output:



Algorithm:

Step 1: import tkinter package

Step 2: define functions that works on button actions

Step 3: Create multiple labels and Text fields according to the requirement

Step 4: Call the defined functions in the command methods of button.

Step 5: Call the mainloop

Step 6: Exit

Program:

```
import tkinter as tk
from tkinter import ttk
r_882= tk.Tk()
r_882.title('Time Converter_882')
r_882.geometry("380x210")
label1_882=tk.Label(r_882,fg="#fff",text="Welcome to Real Time Currency
Converter",bg="#187498",font='Helvetica 12 bold',justify='center',width=38)
label1_882.place(x=0,y=0)
label2_882=tk.Label(r_882,text="1 USD = 78.93 Indian Rupee",font='Helvetica 12
bold',justify='center')
label2_882.place(x=81,y=40)
label3_882=tk.Label(r_882,text="Date : 2022-06-15",font='Helvetica 12
bold',justify='center')
label3_882.place(x=120,y=65)
n_882= tk.StringVar()
```

```

c1_882= ttk.Combobox(r_882, width = 13,justify='center', textvariable = n_882)
c1_882['values'] = (' USD ')
c1_882.place(x = 40, y = 120)
c1_882.current(0)
n2_882= tk.StringVar()
c2_882= ttk.Combobox(r_882, width = 13,justify='center', textvariable = n2_882)
c2_882['values'] = (' INR ')
c2_882.place(x = 230, y = 120)
c2_882.current(0)
e1_882=tk.Entry(r_882,justify='center',width=13)
e1_882.place(x=50,y=150)
e2_882=tk.Entry(r_882,justify='center',width=13)
e2_882.place(x=240,y=150)
b1_882=tk.Button(r_882,text="Convert",bg='blue',fg="#ffff")
b1_882.place(x=160,y=165)
r_882.mainloop()

```

Input/Output:



Algorithm:

- Step 1: import tkinter package
- Step 2: define functions that works on button actions
- Step 3: Create multiple labels and Text fields according to the requirement
- Step 4: Call the defined functions in the command methods of button.
- Step 5: Call the mainloop
- Step 6: Exit

Program:

```

from tkinter import *
from tkinter import ttk
def donothing():
    filewin = Toplevel(root)

```

```

button = Button(filewin, text="Do nothing button")
button.pack()
f = "Helvetica 13 bold"
root = Tk()
root.geometry("500x300")

```

```

root.title("Tab_882")
root.resizable(0,0)
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)
tabControl = ttk.Notebook(root)
tab1 = ttk.Frame(tabControl)
tab2 = ttk.Frame(tabControl)
tabControl.add(tab1, text="Tab 1")
tabControl.add(tab2, text="Tab 2")
tabControl.pack(expand=1, fill="both")
label_frame = LabelFrame(tab2, text="The Snake")
label_frame.pack(expand=YES, fill=BOTH)
disabledCheckBox = Checkbutton(label_frame, text="Disabled", font="Ariel 10 bold").place(x=10, y=10)
unCheckedCheckBox = Checkbutton(label_frame, text="Uncheked", font="Ariel 10 bold").place(x=190, y=10)
enabledCheckBox = Checkbutton(label_frame, text="Enabled", font="Ariel 10 bold").place(x=350, y=10)
blueCheckBox = Checkbutton(label_frame, text="Blue", font="Ariel 10 bold").place(x=10, y=40)
goldCheckBox = Checkbutton(label_frame, text="Gold", font="Ariel 10 bold").place(x=190, y=40)
redCheckBox = Checkbutton(label_frame, text="Red", font="Ariel 10 bold").place(x=350, y=40)
progressBarLabelFrame = LabelFrame(label_frame, text="ProgressBar", font="Ariel 10 bold")
progressBarLabelFrame.place(x=10, y=70)
runProgressButton = Button(progressBarLabelFrame, text="Run Progress", font="Ariel 10 bold").grid(
row=0, column=0

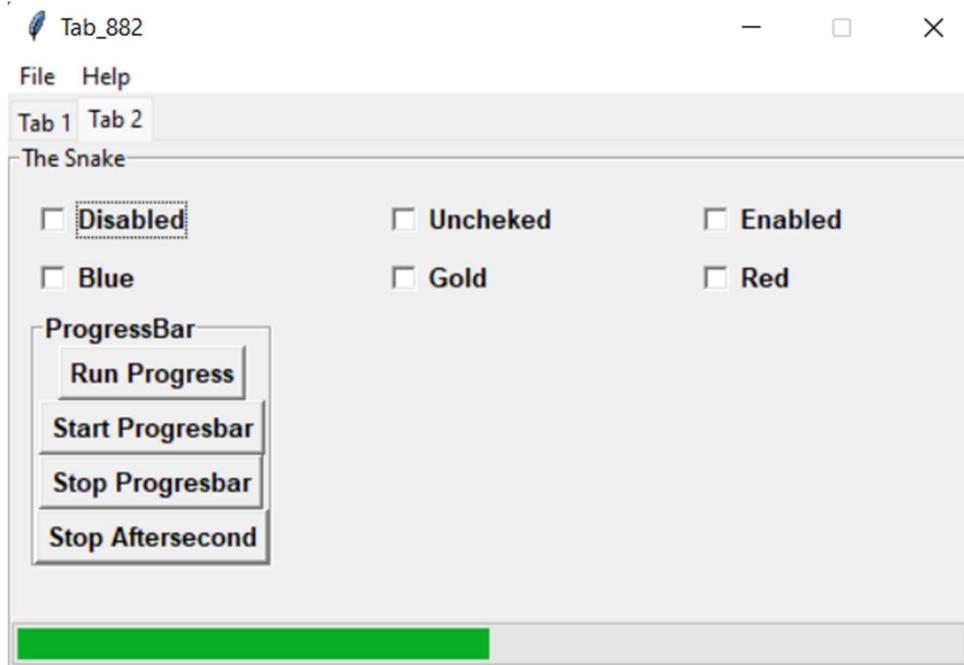
```

```

)
startProgressBar = Button(progressBarLabelFrame, text="Start Progresbar", font="Ariel 10 bold").grid(
row=1, column=0
)
stopProgressBar = Button(progressBarLabelFrame, text="Stop Progresbar", font="Ariel 10 bold").grid(
row=2, column=0
)
stopAfterSecond = Button(progressBarLabelFrame, text="Stop Aftersecond", font="Ariel 10 bold").grid(row=3, column=0)
progress = ttk.Progressbar(
label_frame, orient=HORIZONTAL, style="green.Horizontal.TProgressbar", Length=490,
mode="determinate", maximum=4, value=2
)
progress.pack(ipady=20)
progress.place(x=0, y=230)
root.config(menu=menuBar)
root.mainloop()

```

Input/Output:



Result: Hence the Event Driven Programming has been implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Declarative Programming Paradigm

Problem:

Create the below table and execute the insert, update and the below select statements.

recipes	recipes
id : int(11)	
name : varchar(400)	
description : text	
category_id : int(11)	
chef_id : int(255)	
created : datetime	

- i) Write a query to display the total number of recipes available with the description “Chinese”
- ii) Write a query to display the id, name of the recipes with chef_id 'BL000002'.
- iii) Write a query to display the description of the recipes whose name begins with 'P'.

Algorithm:

Step 1: import sqlite3 package

Step 2: connect the database to store, retrieve table & records

Step 3: Create a table using CREATE TABLE command

Step 4: Insert records in the table using INSERT command

Step 5: commit/save the record into the Table

Step 6: Execute the sql query one by one and print the result accordingly

Program:

```
import sqlite3
conn_882=sqlite3.connect('week4.db')
print("Opened database successfully")
conn_882.execute("CREATE TABLE recipes(id INT PRIMARY KEY NOT NULL,name
VARCHAR,description TEXT,category_id INT,chef_id text,created DATETIME);")
print("Table created successfully")
conn_882.execute("INSERT INTO recipes VALUES(101,'Chicken
Mugali','Indian',01,'BL00001','2022-02-14');")
conn_882.execute("INSERT INTO recipes VALUES(102,'Hakka
noodels','Chinese',02,'BL00002','2022-02-14');")
conn_882.execute("INSERT INTO recipes VALUES(103,'Octopus
Soup','Chinese',03,'BL00003','2022-02-14');")
conn_882.execute("INSERT INTO recipes VALUES(104,'Pasta','Italian',04,'BL00004','2022-02-
14');")
conn_882.commit()
print("Records inserted successfully");
print("\nDisplaying the table \n")
for rows in conn_882.execute("SELECT * from recipes;"):
```

```

    print(rows)
print("\nPART A\n")
for rows in conn_882.execute("SELECT * from recipes WHERE description='Chinese';"):
    print(rows)
print("\nPART B\n")
for rows in conn_882.execute("SELECT id,name from recipes WHERE chef_id='BL00002';"):
    print(rows)
print("\nPART C\n")
for rows in conn_882.execute("SELECT description from recipes WHERE name LIKE 'P%';"):
    print(rows)
conn_882.close()

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab5\1.py"
Opened database successfully
Table created successfully
Records inserted successfully

Displaying the table

(101, 'Chicken Mugali', 'Indian', 1, 'BL00001', '2022-02-14')
(102, 'Hakka noodels', 'Chinese', 2, 'BL00002', '2022-02-14')
(103, 'Octopus Soup', 'Chinese', 3, 'BL00003', '2022-02-14')
(104, 'Pasta', 'Italian', 4, 'BL00004', '2022-02-14')

PART A

(102, 'Hakka noodels', 'Chinese', 2, 'BL00002', '2022-02-14')
(103, 'Octopus Soup', 'Chinese', 3, 'BL00003', '2022-02-14')

PART B

(102, 'Hakka noodels')

PART C

('Italian',)

```

Problem:

Create a table movie of the below structure and assume data types.Movie_ID, Movie_Name, Genre, Language, Rating ,Do the following queries

- Update the movies rating by 10% and display it
- Delete the movies with movie_id 102
- Select movies whose rating is more than 3.

Algorithm:

Step 1: import sqlite3 package

Step 2: connect the database to store,retrieve table & records

Step 3: Create a table using CREATE TABLE command

Step 4: Insert records in the table using INSERT command

Step 5: commit/save the record into the Table

Step 6: Execute the sql query one by one and print the result accordingly

Program:

```
import sqlite3
conn_882=sqlite3.connect('week4.db')
print("Opened database successfully")
conn_882.execute("CREATE TABLE movie(Movie_ID INT PRIMARY KEY NOT NULL,Movie_Name
VARCHAR,Genre TEXT,Language text,Rating FLOAT;")"
print("Table created successfully")
conn_882.execute("INSERT INTO movie VALUES(101,'Interstellar','Sci-fi','English',8.3);")
conn_882.commit()
conn_882.execute("INSERT INTO movie VALUES(102,'The Batman','Superhero','English',8.2);")
conn_882.commit()
conn_882.execute("INSERT INTO movie VALUES(103,'Top Gun','Action','English',8.0);")
conn_882.commit()
print("Records inserted successfully");
print("\nDisplaying the table \n")
for rows in conn_882.execute("SELECT * from movie;"):
    print(rows)
print("\nPART A\n")
conn_882.execute("UPDATE movie SET rating=rating+0.1*rating;")
print("Rating updated successfully")
for rows in conn_882.execute("SELECT * from movie;"):
    print(rows)
print("\nPART B\n")
conn_882.execute("DELETE from movie WHERE Movie_Id=102;")
print("\nRecord Deleted Successfully")
print("\nPART C\n")
for rows in conn_882.execute("SELECT * from movie WHERE rating>3;"):
    print(rows)
conn_882.close()
```

Input/Output:

```
PS C:\> python -u "c:\Users\nikhi\Downloads\network\python\lab5\2.py"
Opened database successfully
Table created successfully
Records inserted successfully

Displaying the table

(101, 'Interstellar', 'Sci-fi', 'English', 8.3)
(102, 'The Batman', 'Superhero', 'English', 8.2)
(103, 'Top Gun', 'Action', 'English', 8.0)

PART A

Rating updated successfully
(101, 'Interstellar', 'Sci-fi', 'English', 9.13)
(102, 'The Batman', 'Superhero', 'English', 9.02)
(103, 'Top Gun', 'Action', 'English', 8.8)

PART B

Record Deleted Successfully

PART C

(101, 'Interstellar', 'Sci-fi', 'English', 9.13)
(103, 'Top Gun', 'Action', 'English', 8.8)
```

Problem:

Create a course database with the following fields Product(ID, Prod_name, Supplier_id, Unit_price, Package, OrderID), OrderItem(ID, Order_id, Product_id, Unit_price, Quantity) using Foreign key

- d. Display the total quantity of every product in the stock
- e. Sort the Unit_price based on the supplier_id
- f. Display the Product_name along with order_id and supplier_id

Algorithm:

Step 1: import sqlite3 package

Step 2: connect the database to store, retrieve table & records

Step 3: Create a table using CREATE TABLE command

Step 4: Insert records in the table using INSERT command

Step 5: commit/save the record into the Table

Step 6: Execute the sql query one by one and print the result accordingly

Program:

```
import sqlite3
conn_882=sqlite3.connect('course.db')
print("Opened database successfully")
conn_882.execute("CREATE TABLE Product(ID INT PRIMARY KEY NOT NULL,Prod_name
VARCHAR,Supplier_id text,Unit_price FLOAT,Package TEXT,order_ID text);")
conn_882.execute("CREATE TABLE OrderItem(ID INT PRIMARY KEY NOT NULL,Unit_price
FLOAT,order_ID text,Product_ID TEXT,quantity INT, FOREIGN KEY(order_ID) REFERENCES
Product(order_ID));")
print("Tables created successfully")
conn_882.execute("INSERT INTO Product VALUES(1,'Car','s01',3450.7,5,'o01');")
```

```

conn_882.execute("INSERT INTO Product VALUES(2,'Bike','s02',2450.3,4,'o02');")
conn_882.execute("INSERT INTO OrderItem VALUES(1,3450.7,'o01','p01',8);")
conn_882.execute("INSERT INTO OrderItem VALUES(2,2450.3,'o02','p02',10);")
conn_882.commit()
print("Records Inserted Successfully")
print("Showing Table Product\n")
for rows in conn_882.execute("SELECT * from Product"):
    print(rows)
print("\nShowing Table OrderItem\n")
for rows in conn_882.execute("SELECT * from OrderItem"):
    print(rows)
print("\nPart A\n")
for rows in conn_882.execute("SELECT Product.Prod_name,OrderItem.quantity FROM
Product,OrderItem where Product.order_ID=OrderItem.order_ID;"):
    print(rows)
print("\nPart B\n")
for rows in conn_882.execute("SELECT Unit_Price FROM Product ORDER BY Supplier_id"):
    print(rows)
print("\nPart C\n")
for rows in conn_882.execute("SELECT Prod_name,order_ID,Supplier_id FROM PRODUCT"):
    print(rows)
conn_882.close()

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Download
s\network\python\lab5\3.py"
Opened database successfully
Tables created successfully
Records Inserted Successfully
Showing Table Product
(1, 'Car', 's01', 3450.7, '5', 'o01')
(2, 'Bike', 's02', 2450.3, '4', 'o02')

Showing Table OrderItem
(1, 3450.7, 'o01', 'p01', 8)
(2, 2450.3, 'o02', 'p02', 10)

Part A
('Car', 8)
('Bike', 10)

Part B
(3450.7,)
(2450.3,)

Part C
('Car', 'o01', 's01')
('Bike', 'o02', 's02')

```

Problem:

Write a SQL lite3 statement to create a table named as job including columns job_id,job_title,Min-salary,Max_salary.job_id column does not contain any duplicate value at the time of insertion

Algorithm:

Step 1: import sqlite3 package

- Step 2: connect the database to store, retrieve table & records**
- Step 3: Create a table using CREATE TABLE command**
- Step 4: Insert records in the table using INSERT command**
- Step 5: commit/save the record into the Table**
- Step 6: Execute the sql query one by one and print the result accordingly**

Program:

```
import sqlite3
conn_882=sqlite3.connect('week4.db')
print("Opened database successfully")
conn_882.execute("CREATE TABLE job(job_id INT PRIMARY KEY NOT NULL,job_title
text,Min_salary INT,Max_salary INT);")
print("Table created successfully")
conn_882.execute("INSERT INTO job VALUES(01,'Front End Developer',60000,200000);")
conn_882.execute("INSERT INTO job VALUES(02,'Back End Developer',50000,150000);")
conn_882.commit()
print("Records inserted successfully");
print("\nDisplaying the table\n")
for rows in conn_882.execute("SELECT * from job;"):
    print(rows)
conn_882.close()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab5\4.py"
Opened database successfully
Table created successfully
Records inserted successfully

Displaying the table
(1, 'Front End Developer', 60000, 200000)
(2, 'Back End Developer', 50000, 150000)
```

Problem:

Write a SQL lite3 statement to create a table names as job_history including columns employee_id, start_date, end_date, job_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion and the foreign key column job_id contain only those values which are exists in the jobs table.

Algorithm:

- Step 1: import sqlite3 package**
- Step 2: connect the database to store, retrieve table & records**
- Step 3: Create a table using CREATE TABLE command**
- Step 4: Insert records in the table using INSERT command**
- Step 5: commit/save the record into the Table**
- Step 6: Execute the sql query one by one and print the result accordingly**

Program:

```
import sqlite3
conn_882=sqlite3.connect('week4.db')
print("Opened database successfully")
conn_882.execute("CREATE TABLE job(job_id INT PRIMARY KEY NOT NULL,job_title
text,Min_salary INT,Max_salary INT);")
conn_882.execute("CREATE TABLE job_history(employee_id INT PRIMARY KEY NOT
NULL,start_date text,end_date text,job_id INT,depatement_id INT, FOREIGN KEY(job_id)
REFERENCES job(job_id));")
print("Tables created successfully")
conn_882.execute("INSERT INTO job VALUES(01,'Front End Developer',60000,200000);")
conn_882.execute("INSERT INTO job VALUES(02,'Back End Developer',50000,150000);")
conn_882.execute("INSERT INTO job_history
VALUES(101,'1/Jan/2022','10/May/2022',01,454);")
conn_882.execute("INSERT INTO job_history
VALUES(102,'1/Jan/2021','31/Dec/2021',02,654);")
conn_882.commit()
print("Records inserted successfully");
print("\nDisplaying the table job\n")
for rows in conn_882.execute("SELECT * from job;"):
    print(rows)
print("\nDisplaying the table job_history\n")
for rows in conn_882.execute("SELECT * from job_history;"):
    print(rows)
conn_882.close()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Download
s\network\python\lab5\5.py"
Opened database successfully
Tables created successfully
Records inserted successfully

Displaying the table job

(1, 'Front End Developer', 60000, 200000)
(2, 'Back End Developer', 50000, 150000)

Displaying the table job_history

(101, '1/Jan/2022', '10/May/2022', 1, 454)
(102, '1/Jan/2021', '31/Dec/2021', 2, 654)
```

Result: Hence the Declarative Programming has been implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim : To implement Parallel Programming

Problem:

The Dining Philosopher Problem – The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher.

Program:

```
import threading
import time

# inheriting threading class in Thread module
class Philosopher(threading.Thread):
    running = True # used to check if everyone is finished eating

    # Since the subclass overrides the constructor, it must make sure to invoke the base
    # class constructor (
    # Thread.__init__() before doing anything else to the thread.
    def __init__(self, index, forkOnLeft, forkOnRight):
        threading.Thread.__init__(self)
        self.index = index
        self.forkOnLeft = forkOnLeft
        self.forkOnRight = forkOnRight

    def run(self):
        while (self.running):
            # Philosopher is thinking (but really is sleeping).
            time.sleep(30)
            print('Philosopher %s is hungry.' % self.index)
            self.dine()

    def dine(self):
        # if both the semaphores(forks) are free, then philosopher will eat
        fork1, fork2 = self.forkOnLeft, self.forkOnRight
        while self.running:
            fork1.acquire() # wait operation on left fork
            locked = fork2.acquire(False)
            if locked: break # if right fork is not available leave left fork
            fork1.release()
            print('Philosopher %s swaps forks.' % self.index)
            fork1, fork2 = fork2, fork1
        else:
            return
```

```

self.dining()
# release both the fork after dining
fork2.release()
fork1.release()

def dining(self):
    print('Philosopher %s starts eating. ' % self.index)
    time.sleep(30)
    print('Philosopher %s finishes eating and leaves to think.' % self.index)

def main():
    forks = [threading.Semaphore() for n in range(5)] # initialising array of semaphore
    i.e forks

    # here (i+1)%5 is used to get right and left forks circularly between 1-5
    philosophers = [Philosopher(i, forks[i % 5], forks[(i + 1) % 5])
                    for i in range(5)]

    Philosopher.running = True
    for p in philosophers: p.start()
    time.sleep(100)
    Philosopher.running = False
    print("Now we're finishing.")

if __name__ == "__main__":
    main()

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\py
thon\lab6\6_dining_philosopher.py"
Philosopher 3 is hungry.Philosopher 0 is hungry.
Philosopher 0 starts eating.

Philosopher 3 starts eating. Philosopher 4 is hungry.
Philosopher 2 is hungry.
Philosopher 1 is hungry.
Philosopher 2 swaps forks.

Philosopher 0 finishes eating and leaves to think.
Philosopher 1 starts eating.
Philosopher 3 finishes eating and leaves to think.
Philosopher 4 starts eating.
Philosopher 2 swaps forks.
Philosopher 1 finishes eating and leaves to think.Philosopher 0 is hungry.

Philosopher 2 starts eating.
Philosopher 4 finishes eating and leaves to think.
Philosopher 3 is hungry.Philosopher 0 starts eating.

Now we're finishing.

```

Problem:

Consider a situation where we have a file shared between many people.

- If one of the people tries editing the file, no other person should be reading or writing at the same time, otherwise changes will not be visible to him/her.
- However if some person is reading the file, then others may read it at the same time.

Program:

```
import threading as thread
import random

global x # Shared Data
x = 0
lock = thread.Lock() # Lock for synchronising access

def Reader():
    global x
    print('Reader is Reading!')
    lock.acquire() # Acquire the Lock before Reading (mutex approach)
    print('Shared Data:', x)
    lock.release() # Release the Lock after Reading
    print()

def Writer():
    global x
    print('Writer is Writing!')
    lock.acquire() # Acquire the Lock before Writing
    x += 1 # Write on the shared memory
    print('Writer is Releasing the lock!')
    lock.release() # Release the Lock after Writing
    print()

if __name__ == '__main__':
    for i in range(0, 10):
        randomNumber = random.randint(0, 100) # Generate a Random number between 0 and 100
        if randomNumber > 50:
            Thread1 = thread.Thread(target=Reader)
            Thread1.start()
        else:
            Thread2 = thread.Thread(target=Writer)
            Thread2.start()

    Thread1.join()
```

```
Thread2.join()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\lab6\7_reader_writer.py"
Writer is Writing!
Writer is Releasing the lock!Reader is Reading!

Shared Data:Reader is Reading!
Reader is Reading!1

Reader is Reading!
Shared Data:
Writer is Writing!
1Reader is Reading!

Reader is Reading!Shared Data:

Reader is Reading!1

Writer is Writing!

Shared Data: 1

Writer is Releasing the lock!

Shared Data: 2

Shared Data: 2

Shared Data: 2

Writer is Releasing the lock!
```

Result: Hence the Parallel and Concurrent Programming has been implemented and compiled successfully

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim : To implement concurrent programming

Problem:

Modify the provided ReverseHelloMultithreaded file so that it creates a thread (let's call it Thread 1). Thread 1 creates another thread (Thread 2); Thread 2 creates Thread 3; and so on, up to Thread 50. Each thread should print "Hello from Thread <num>!"; but you should structure your program such that the threads print their greetings in reverse order. When complete, ReverseHelloTest should run successfully.

Program:

```
import time # import time module
import threading
from threading import *

ThreadCount = 0
def req_thread(num): # define a square calculating function
    if num<=0:
        return 0
    req_thread(num-1)
    print(f'Hello from Thread {num+1}! I am Thread {num} :)')
t = time.time()
th1 = threading.Thread(target=req_thread, args=(50, ))
th1.start()
th1.join()
print(" Total time taking by threads is :", time.time() - t) # print the total time
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\tempCodeRunnerFile.py"
Hello from Thread 2! I am Thread 1 :)
Hello from Thread 3! I am Thread 2 :)
Hello from Thread 4! I am Thread 3 :)
Hello from Thread 5! I am Thread 4 :)
Hello from Thread 6! I am Thread 5 :)
Hello from Thread 7! I am Thread 6 :)
Hello from Thread 8! I am Thread 7 :)
Hello from Thread 9! I am Thread 8 :)
Hello from Thread 32! I am Thread 31 :)
Hello from Thread 33! I am Thread 32 :)
Hello from Thread 34! I am Thread 33 :)
Hello from Thread 35! I am Thread 34 :)
Hello from Thread 36! I am Thread 35 :)
Hello from Thread 37! I am Thread 36 :)
Hello from Thread 38! I am Thread 37 :)
Hello from Thread 39! I am Thread 38 :)
Hello from Thread 40! I am Thread 39 :)
Hello from Thread 41! I am Thread 40 :)
Hello from Thread 42! I am Thread 41 :)
Hello from Thread 43! I am Thread 42 :)
Hello from Thread 44! I am Thread 43 :)
Hello from Thread 45! I am Thread 44 :)
Hello from Thread 46! I am Thread 45 :)
Hello from Thread 47! I am Thread 46 :)
Hello from Thread 48! I am Thread 47 :)
Hello from Thread 49! I am Thread 48 :)
Hello from Thread 50! I am Thread 49 :)
Hello from Thread 51! I am Thread 50 :)
Total time taking by threads is : 0.01717209815979004
```

Problem:

To make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

Program:

```
import threading
import time

# Shared Memory variables
CAPACITY = 10
buffer = [-1 for i in range(CAPACITY)]
in_index = 0
out_index = 0

# Declaring Semaphores
mutex = threading.Semaphore()
empty = threading.Semaphore(CAPACITY)
full = threading.Semaphore(0)

# Producer Thread Class
class Producer(threading.Thread):
    def run(self):

        global CAPACITY, buffer, in_index, out_index
        global mutex, empty, full

        items_produced = 0
        counter = 0

        while items_produced < 20:
            empty.acquire()
            mutex.acquire()

            counter += 1
            buffer[in_index] = counter
            in_index = (in_index + 1)%CAPACITY
            print("Producer produced : ", counter)

            mutex.release()
            full.release()

            time.sleep(1)
            items_produced += 1

# Consumer Thread Class
class Consumer(threading.Thread):
    def run(self):
        global CAPACITY, buffer, in_index, out_index, counter
```

```
global mutex, empty, full

items_consumed = 0

while items_consumed < 20:
    full.acquire()
    mutex.acquire()

    item = buffer[out_index]
    out_index = (out_index + 1)%CAPACITY
    print("Consumer consumed item : ", item)

    mutex.release()
    empty.release()

    time.sleep(2.5)
    items_consumed += 1

# Creating Threads
producer = Producer()
consumer = Consumer()

# Starting Threads
consumer.start()
producer.start()

# Waiting for threads to complete
producer.join()
consumer.join()
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Do
wnloads\network\python\GUIpython\tempCodeRunnerFile.py"
Producer produced : 1
Consumer consumed item : 1
Producer produced : 2
Producer produced : 3
Consumer consumed item : 2
Producer produced : 4
Producer produced : 5
Consumer consumed item : 3
Producer produced : 6
Producer produced : 7
Producer produced : 8
Consumer consumed item : 4
Producer produced : 9
Producer produced : 10
Consumer consumed item : 5
Producer produced : 11
Producer produced : 12
Producer produced : 13
Consumer consumed item : 6
Producer produced : 14
Producer produced : 15
Consumer consumed item : 7
Producer produced : 16
Producer produced : 17
Consumer consumed item : 8
Producer produced : 18
Consumer consumed item : 9
Producer produced : 19
Consumer consumed item : 10
Producer produced : 20
Consumer consumed item : 11
Consumer consumed item : 12
Consumer consumed item : 13
Consumer consumed item : 14
Consumer consumed item : 15
Consumer consumed item : 16
Consumer consumed item : 17
Consumer consumed item : 18
Consumer consumed item : 19
Consumer consumed item : 20
```

Result: Hence the Concurrent Programming has been implemented and compiled successfully

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Network Programming

Problem:

Create Simple Client Server Application using TCP Socket where server issue a command which will be executed at the client side as a process of remote command execution.

Algorithm:

Client

- Step 1: import the socket module**
- Step 2: Create a socket object**
- Step 3: Choose a local machine name**
- Step 4: Reserve a part for your service**
- Step 5: Connect the host and port together**
- Step 6: Close the socket when done**

Server

- Step 1: import the socket module**
- Step 2: Create a socket object**
- Step 3: Choose a local machine name**
- Step 4: Reserve a part for your service**
- Step 5: Bind the port and host**
- Step 6: Now wait for client connection**
- Step 7: Establish the connection with the client**

Program:

Client.py

```
import socket # Import socket module
import os
s = socket.socket() # Create a socket object
host = socket.gethostname() # Get Local machine name
port = 12345 # Reserve a port for your service.
s.connect((host, port))
cmd1= s.recv(1024)
cmd1=str(cmd1,'utf-8')
print(cmd1)
os.system(cmd1)
s.send(b'Command Executed')
s.close() # Close the socket when done
```

Server.py

```
import socket # Import socket module
s = socket.socket() # Create a socket object
host = socket.gethostname() # Get Local machine name
port = 12345 # Reserve a port for your service.
s.bind((host, port)) # Bind to the port
```

```

s.listen(5) # Now wait for client connection.
while True:
    c,addr = s.accept() # Establish connection with client.
    print ('Got connection from', addr)
    c.send(b'id')
    print (c.recv(1024))
    c.close() # Close the connection

```

```

C:\Users\nikhi\Downloads\network\python\GUIpython>python q1.py
Got connection from ('10.6.193.45', 49775)

```

Problem:

Write a Socket-based Python server program that responds to client messages as follows: When it receives a message from a client, it simply converts the message into all uppercase letters and sends back the same to the client. Write both client and server programs demonstrating this.

Algorithm:

Client

- Step 1: import the socket module
- Step 2: Create a socket object
- Step 3: Choose a local machine name
- Step 4: Reserve a part for your service
- Step 5: Connect the host and port together
- Step 6: Close the socket when done

Server

- Step 1: import the socket module
- Step 2: Create a socket object
- Step 3: Choose a local machine name
- Step 4: Reserve a part for your service
- Step 5: Bind the port and host
- Step 6: Now wait for client connection
- Step 7: Establish the connection with the client

Program:

Client.py

```

import socket
# Import socket module
import os
s = socket.socket()
# Create a socket object
host = socket.gethostname()
# Get Local machine name
port = 12345
# Reserve a port for your service.
s.connect((host, port))

```

```

s.send(b"Sending request to UpperCase this line")
res1 = s.recv(1024)
print(res1)
s.close()
# Close the socket when done

```

Server.py

```

import socket
# Import socket module
s = socket.socket()
# Create a socket object
host = socket.gethostname()
# Get Local machine name
port = 12345
# Reserve a port for your service.
s.bind((host, port))
# Bind to the port
print(f'Server started at {host}:{port}')
s.listen(5)
# Now wait for client connection.
while True:
    c, addr = s.accept()
    # Establish connection with client.
    print('Got connection from', addr)
    req1 = c.recv(1024)
    req1 = str(req1, 'utf-8')
    str1 = req1.upper()
    # print(str1)
    str1 = bytes(str1, 'UTF-8')
    c.send(str1)
    c.close()
# Close the connection

```

Input/Output:

```

[1] Command Prompt - python server.py
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nikhi>cd Downloads
C:\Users\nikhi\Downloads>cd network
C:\Users\nikhi\Downloads\network>cd python
C:\Users\nikhi\Downloads\network\python>cd GUIpython
C:\Users\nikhi\Downloads\network\python\GUIpython>python server.py
Server started at Nick:12345
Get connection from ('10.6.193.45', 49859)

[2] Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nikhi>cd Downloads
C:\Users\nikhi\Downloads>cd network
C:\Users\nikhi\Downloads\network>cd python
C:\Users\nikhi\Downloads\network\python>cd GUIpython
C:\Users\nikhi\Downloads\network\python\GUIpython>python client.py
b'SENDING REQUEST TO UPPERCASE THIS LINE'

C:\Users\nikhi\Downloads\network\python\GUIpython>

```

Problem:

Write a ping-pong client and server application. When a client sends a ping message to the server, the server will respond with a pong message. Other messages sent by the client can be safely dropped by the server.

Algorithm:

Client

- Step 1: import the socket module
- Step 2: Create a socket object
- Step 3: Choose a local machine name
- Step 4: Reserve a part for your service
- Step 5: Connect the host and port together
- Step 6: Close the socket when done

Server

- Step 1: import the socket module
- Step 2: Create a socket object
- Step 3: Choose a local machine name
- Step 4: Reserve a part for your service
- Step 5: Bind the port and host
- Step 6: Now wait for client connection
- Step 7: Establish the connection with the client

Program:

Client.py

```
# echo-client.py
import socket
# Import socket module
import os
s = socket.socket()
# Create a socket object
host = socket.gethostname()
# Get Local machine name
port = 12345
# Reserve a port for your service.
s.connect((host, port))
s.send(b"ping")
res1 = s.recv(1024)
print(res1)
s.close()
# Close the socket when done
```

Server.py

```
# echo-server.py
# If port is in use try kill -9 $(lsof -t -i tcp:12345)
import socket
# Import socket module
s = socket.socket()
# Create a socket object
host = socket.gethostname()
# Get Local machine name
port = 12345
# Reserve a port for your service.
```

```

s.bind((host, port))
# Bind to the port
print(f'Server started at{host}:{port}')
s.listen(5)
# Now wait for client connection.
while True:
    c, addr = s.accept()
    # Establish connection with client.
    print ('Got connection from', addr)
    req1 = c.recv(1024)
    req1 =str(req1,'utf-8')
    if(req1=="ping"):
        c.send(b'pong')
    # print(str1)
    c.close()
# Close the connection

```

Input/Output:

 Command Prompt - python server1.py Microsoft Windows [Version 10.0.22000.675] (c) Microsoft Corporation. All rights reserved. C:\Users\nikhi>cd Downloads C:\Users\nikhi\Downloads>cd network C:\Users\nikhi\Downloads\network>cd python C:\Users\nikhi\Downloads\network\python>cd GUIpython C:\Users\nikhi\Downloads\network\python\GUIpython>python server1.py Server started atNick:12345 Got connection from ('10.6.193.45', 49910)	 Command Prompt Microsoft Windows [Version 10.0.22000.675] (c) Microsoft Corporation. All rights reserved. C:\Users\nikhi>cd Downloads C:\Users\nikhi\Downloads>cd network C:\Users\nikhi\Downloads\network>cd python C:\Users\nikhi\Downloads\network\python>cd GUIpython C:\Users\nikhi\Downloads\network\python\GUIpython>python client1.py b'pong' C:\Users\nikhi\Downloads\network\python\GUIpython>
--	--

Problem:

Write a Socket based program server-client you simulate a simple chat application where the server is multithreaded which can serve for multiple clients at the same time.

Algorithm:

Client

Step 1: import the socket module

Step 2: Create a socket object

Step 3: Choose a local machine name

Step 4: Reserve a part for your service

Step 5: Connect the host and port together

Step 6: Close the socket when done

Server

Step 1: import the socket module

Step 2: Create a socket object

Step 3: Choose a local machine name

Step 4: Reserve a part for your service

Step 5: Bind the port and host

Step 6: Now wait for client connection

Step 7: Establish the connection with the client

Program:

Client.py

```
import socket
ClientMultiSocket = socket.socket()
host ='127.0.0.1'
port =2004
print('Waiting for connection response')
try:
    ClientMultiSocket.connect((host, port))
except socket.error as e:
    print(str(e))
res = ClientMultiSocket.recv(1024)
print(str(res,'utf-8'))
while True :
    Input =input('Hey there: ')
    ClientMultiSocket.send(str.encode(Input))
    res = ClientMultiSocket.recv(1024)
    print(res.decode('utf-8'))
ClientMultiSocket.close()
```

Server.py

```
import socket
import os
from _thread import *
ServerSideSocket = socket.socket()
host ='127.0.0.1'
port =2004
ThreadCount =0
try:
    ServerSideSocket.bind((host, port))
except socket.error as e:
    print(str(e))
print('Socket is listening..')
ServerSideSocket.listen(5)
def multi_threaded_client(connection):
    connection.send(str.encode('Server is working :)))
    while True:
        data = connection.recv(2048)
        response ='Server message: '+ data.decode('utf-8')
        if not data:
            break
        connection.sendall(str.encode(response))
```

```

        connection.close()
while True:
    Client, address = ServerSideSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (Client, ))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))
ServerSideSocket.close()

```

Input/Output:

The image shows two separate Command Prompt windows. The left window, titled 'Command Prompt - python server2.py', displays the server's configuration and listening status. The right window, titled 'Command Prompt - python client2.py', shows the client connecting to the server and sending/receiving messages.

```

[cmd] Command Prompt - python server2.py
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nikhi>cd Downloads
C:\Users\nikhi\Downloads>cd network
C:\Users\nikhi\Downloads\network>cd python
C:\Users\nikhi\Downloads\network\python>cd GUIpython
C:\Users\nikhi\Downloads\network\python\GUIpython>python server2.py
Socket is listening..
Connected to: 127.0.0.1:50385
Thread Number: 1

[cmd] Command Prompt - python client2.py
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nikhi>cd Downloads
C:\Users\nikhi\Downloads>cd network
C:\Users\nikhi\Downloads\network>cd python
C:\Users\nikhi\Downloads\network\python>cd GUIpython
C:\Users\nikhi\Downloads\network\python\GUIpython>python client2.py
Waiting for connection response
Server is working :)
Hey there: Hello
Server message: Hello
Hey there: I am Nikhil Kumar
Server message: I am Nikhil Kumar

```

Result: Hence the Network Programming has been implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Functional Programming Paradigm

Problem:

Write a Python program to create Fibonacci series upto n using Lambda.

Algorithm:

- Step 1:Create a list of 0,1**
- Step 2: Using map ,giving function as append in list**
- Step 3: Range in given as 2**
- Step 4: print fibonacci**

Program:

```
from functools import reduce

fib = lambda n: reduce(lambda x, _: x+[x[-1]+x[-2]], range(n-2), [0, 1])

print(fib(5))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f1.py"
[0, 1, 1, 2, 3]
```

Problem:

Write a Python program to find the numbers of a given string and store them in a list, display the numbers which are bigger than the length of the list in sorted form. Use lambda function to solve the problem.

Input :

Original string: SOC 23 CTech 5 DSBS8 NWC 56 CINtel 20 5

Input/Output

Numbers in sorted form:

20 23 56

Algorithm:

- Step 1: Input the string**
- Step 2: Length of string calculated**
- Step 3: Sort the list**
- Step 4: if x>length ,print the numbers**

Program:

```
str1 = "sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5"
print("Original string: ",str1)
str_num=[i for i in str1.split(' ')]
length=len(str_num)
numbers=sorted([int(x) for x in str_num if x.isdigit()])
print('Numbers in sorted form:')
for i in ((filter(lambda x:x>length,numbers))):
    print(i,end=' ')
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f2.py"
Original string: sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5
Numbers in sorted form:
20 23 56
```

Problem:

Write a Python program to sort a given list of strings(numbers) numerically using lambda.

Algorithm:

Step 1:Enter the list

Step 2: Create a sort function

Step 3: Pass the sort function and list in map function

Step 4: print the list

Program:

```
def sort_numeric_strings(nums_str):
    result = sorted(nums_str, key=lambda el: int(el))
    return result
nums_str = ['4','12','45','7','0','100','200','-12','-500']
print("Original list:")
print(nums_str)
print("\nSort the said list of strings(numbers) numerically:")
print(sort_numeric_strings(nums_str))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\py  
thon\GUlPython\f3.py"  
Original list:  
['4', '12', '45', '7', '0', '100', '200', '-12', '-500']  
  
Sort the said list of strings(numbers) numerically:  
['-500', '-12', '0', '4', '7', '12', '45', '100', '200']
```

Problem:

Write a Python program to calculate the average value of the numbers in a given tuple of tuples using lambda.

Algorithm:

- Step 1: Enter the numbers**
- Step 2: Use reduce function**
- Step 3: create function of $(x+y)/2$**
- Step 4: Press the function and list in reduce**

Program:

```
def average_tuple(nums):  
    result = tuple(map(lambda x: sum(x) / float(len(x)), zip(*nums)))  
    return result  
  
nums = ((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))  
print ("Original Tuple: ")  
print(nums)  
print("\nAverage value of the numbers of the said tuple of  
tuples:\n",average_tuple(nums))  
nums = ((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))  
print ("\nOriginal Tuple: ")  
print(nums)  
print("\nAverage value of the numbers of the said tuple of  
tuples:\n",average_tuple(nums))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f4.py"
Original Tuple:
((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))

Average value of the numbers of the said tuple of tuples:
(30.5, 34.25, 27.0)

Original Tuple:
((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))

Average value of the numbers of the said tuple of tuples:
(25.5, -18.0, 3.75)
```

Problem:

Write a Python program to find the nested lists elements, which are present in another list using lambda.

Algorithm:

Step 1: Enter both element in list

Step 2: Enter function expression as x=y

Step 3: Pass the list1 and list2 as parameters

Step 4: Print the values

Program:

```
def intersection_nested_lists(l1, l2):
    result = [list(filter(lambda x: x in l1, sublist)) for sublist in l2]
    return result

nums1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
nums2 = [[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]
print("\nOriginal lists:")
print(nums1)
print(nums2)
print("\nIntersection of said nested lists:")
print(intersection_nested_lists(nums1, nums2))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUipython\f5.py"

Original lists:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]

Intersection of said nested lists:
[[12], [7, 11], [1, 5, 8]]
```

Problem:

Write a Python program to convert a given list of strings into list of lists using map function.

Algorithm:

Step 1: Create a function taking map function passing (list,string)

Step 2: Sort the result in separate list

Step 3: The list has each letter of every word

Program:

```
def strings_to_listOflists(str):
    result = map(list, str)
    return list(result)

colors = ["Red", "Green", "Black", "Orange"]
print('Original list of strings:')
print(colors)
print("\nConvert the said list of strings into list of lists:")

print(strings_to_listOflists(colors))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUipython\f6.py"
Original list of strings:
['Red', 'Green', 'Black', 'Orange']

Convert the said list of strings into list of lists:
[['R', 'e', 'd'], ['G', 'r', 'e', 'e', 'n'], ['B', 'l', 'a', 'c', 'k'], ['o', 'r', 'a', 'n', 'g', 'e']]
```

Problem:

Write a Python program to convert all the characters in uppercase and lowercase and eliminate duplicate letters from a given sequence. Use map() function.

Algorithm:

- Step 1: Enter the characters in list**
- Step 2: Enter function as $x!=y$**
- Step 3: Pass the function and list in map**
- Step 4: Print it as a list**

Program:

```
def change_cases(s):  
    return str(s).upper(), str(s).lower()  
  
chrars = {'a', 'b', 'E', 'f', 'a', 'i', 'o', 'U', 'a'}  
print("Original Characters:\n", chrars)  
  
result = map(change_cases, chrars)  
print("\nAfter converting above characters in upper and lower cases\nand eliminating  
duplicate letters:")  
print(set(result))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\py  
thon\GUIpython\tempCodeRunnerFile.py"  
Original Characters:  
{'f', 'U', 'o', 'b', 'E', 'a', 'i'}  
  
After converting above characters in upper and lower cases  
and eliminating duplicate letters:  
{('o', 'o'), ('A', 'a'), ('E', 'e'), ('I', 'i'), ('F', 'f'), ('B', 'b'), ('U', 'u')}
```

Problem:

Write a Python program to add two given lists and find the difference between lists. Use map() function.

Algorithm:

- Step 1: Enter two list**
- Step 2: Using map define lambda function**
- Step 3: Give parameters as 2 list 1**
- Step 4: print the list**

Program:

```
nums1 = [1, 2, 3]
nums2 = [4, 5, 6]
print("Original list:")
print(nums1)
print(nums2)
result = map(lambda x, y: x + y, nums1, nums2)
print("\nResult: after adding two list")
print(list(result))
result1 = map(lambda x, y: x - y, nums1, nums2)
print("\nResult: after subtracting two list")
print(list(result1))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f9.py"
Original list:
[1, 2, 3]
[4, 5, 6]

Result: after adding two list
[5, 7, 9]

Result: after subtracting two list
[-3, -3, -3]
```

Problem:

Write a Python program to filter only vowels from given sequence.

Algorithm:

Step 1: Pass the list as parameter

Step 2: Specify the function as $x='a' \parallel x='e' \parallel x='o' \parallel x='i' \parallel x='e'$

Step 3: If the elements match the above conditions print them in list

Program:

```
letters_882 = ['a', 'b', 'd', 'e', 'i', 'j', 'o']

def filter_vowels(letter):

    vowels = ['a', 'e', 'i', 'o', 'u']
    if(letter in vowels):
        return True
    else:
        return False
```

```
filtered_vowels = filter(filter_vowels, letters_882)
print('The filtered vowels are:')
for vowel in filtered_vowels:
    print(vowel)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f11.py"
The filtered vowels are:
a
e
i
o
```

Problem:

Write a python program to calculate factorial of given number (use reduce function)

Algorithm:

Step 1: Define a function manually which calculates factorial ,multiplying each value in range 1 to n
Step 2: Pass the number and the manual function in reduce function
Step 3: Print the result of reduce

Program:

```
import functools
def mult(x,y):
    print("x=",x," y=",y)
    return x*y

fact=functools.reduce(mult, range(1, 4))
print ('Factorial of 3: ', fact)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\network\python> python -u "c:\Users\nikhi\Downloads\network\python\GUIpython\f12.py"
x= 1  y= 2
x= 2  y= 3
Factorial of 3:  6
```

Result: Hence the Functional Programming has been implemented and compiled successfully.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Logic Programming

Problem:

Implement using pyDatalog:

Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2).

- Define a predicate brother(X,Y) which holds iff X and Y are brothers.
- Define a predicate cousin(X,Y) which holds iff X and Y are cousins.
- Define a predicate grandson(X,Y) which holds iff X is a grandson of Y.
- Define a predicate descendent(X,Y) which holds iff X is a descendent of Y.
- Consider the following genealogical tree:

```
a
/\ 
b c
/\ |
d e f
```

What are the answers generated by your definitions for the queries:

```
brother(X,Y)
cousin(X,Y)
grandson(X,Y)
descendent(X,Y)
```

Program:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('a,b,c,d,e,f,brother,cousin,grandson,descendent,X,Y')
+brother('b','c')
+brother('d','e')
+cousin('d','f')
+cousin('e','f')
+grandson('d','a')
+grandson('e','a')
+grandson('f','a')
+descendent('b','a')
+descendent('c','a')
+descendent('d','b')
+descendent('f','c')
print(pyDatalog.ask('brother(X,Y)'))
print(pyDatalog.ask('cousin(X,Y)'))
print(pyDatalog.ask('grandson(X,Y)'))
print(pyDatalog.ask('descendent(X,Y)'))
```

Input/Output:

{('d', 'e'), ('b', 'c')}

{('d', 'f'), ('e', 'f')}

{('e', 'a'), ('d', 'a'), ('f', 'a')}

{('f', 'c'), ('b', 'a'), ('c', 'a'), ('d', 'b')}

Result: Hence the program has been executed.**Problem:**

Encode the following facts and rules in pyDatalog:

- Bear is big
- Elephant is big
- Cat is small
- Bear is brown
- Cat is black
- Elephant is gray
- An animal is dark if it is black
- An animal is dark if it is brown

Write a query to find which animal is dark and big.

Program:

```
pyDatalog.create_terms('X,Y,Z,bear,elephant,cat,small,big,brown,black,gray,dark')
+big('elephant')
+big('bear')
+small('cat')
+black('cat')
+brown('bear')
+gray('elephant')
dark(X)<=black(X) or brown(X)
print(big(X),dark(X))
```

Input/Output:

X

bear

elephant X

cat

Result: Hence the program has been executed.

Problem:

The following are the marks scored by 5

students. Student Name Mark

Ram 90

Raju 45

Priya 85

Carol 70

Shyam 80

Enter the above data using pyDatalog.

Write queries for the following:

- a. Print Student name and mark of all students.
- b. Who has scored 80 marks?
- c. What mark has been scored by Priya?
- a. Write a rule ‘passm’ denoting that pass mark is greater than 50. Use the rule to print all students who failed.
- b. Write rules for finding grade letters for a marks and use the rule to find the gradeletter of a given mark.

Program:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,student,marks,passm,grades')
+student('ram')
+student('raju')
+student('priya')
+student('carol')
+student('shyam')
+marks('90','ram')
+marks('45','raju')
+marks('85','priya')
+marks('70','carol')
+marks('80','shyam')
+grades('ram','O')
+grades('priya','A')
+grades('shyam','A')
+grades('carol','B')
+grades('raju','E') print(marks(X,Y)) print(marks('80',X))
print(marks(X,'priya'))
passm(X)<=grades(X,'E') print(passm(X))
```

Input/Output:

X | Y

....|-.....

80 | shyam

70 | carol

85 | priya

45 | raju

90 | ram

X

....|-

shyam

X

-85

X

raju

Result: Hence the program has been executed.

Problem:

Write a recursive program to find factorial of a number using pyDatalog.

Program:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('factorial, N')
num=int(input('Enter any number:'))
factorial[N] = N*factorial[N-1]
factorial[1] = 1
print(factorial[num]==N)
```

Input/Output:

Enter any number:3

N

-6

Result: Hence the program has been executed.

18CSC207J-Advanced Programming Practice
SRM Institute of Science & Engineering- Kattankulathur Campus
School of Computing

Aim: To implement Symbolic Programming

Problem: Calculate $\sqrt{2}$ with 100 decimals.

Program:

```
from sympy import *
print(N(sqrt(2),100))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\A4-1\a1.py"
1.41421356237309504880168872420969807856967187537694807317667973799
0732478462107038850387534327641573
```

Result: Hence the program has been executed.

Problem: Calculate $(1/2+1/3)$ in rational arithmetic.

Program:

```
import sympy as sym
a = (sym.Rational(1, 2)+sym.Rational(1,3))
print(sym.simplify(a))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\A4-1\a1.py"
5/6
```

Result: Hence the program has been executed.

Problem: Calculate the expanded form of $(x+y)^6$.

Program:

```
import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
a = sym.expand((x+y)**6)
print(a)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\a4-1\
a1.py"
x**6 + 6*x**5*y + 15*x**4*y**2 + 20*x**3*y**3 + 15*x**2*y**4 + 6*x*y**5 + y**6
```

Result: Hence the program has been executed.

Problem: Simplify the trigonometric expression $\sin(x) / \cos(x)$

Program:

```
from sympy import *
x = symbols('x')
a = (sin(x)/(cos(x)))
s=trigsimp(a)
print(format(s))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\a4-1\
a1.py"
tan(x)
```

Result: Hence the program has been executed.

Problem: Calculate $\sin x - x^{3n}$

Program:

```
from sympy import *
x,n=symbols('x n')
exp= sin(x)-x*x**3*n
s=trigsimp(exp)
print(s)
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\a4-1\
a1.py"
-n*x**4 + sin(x)
```

Result: Hence the program has been executed.

Problem: Develop a python code for to carry out the operations on the given algebraic manipulation for the given expression $a^2 - ab + ab - b^2 = a^2 - b^2$ by using the symbolic programming paradigms principles.

Program:

```
import sympy as sym
a = sym.Symbol('a')
b = sym.Symbol('b')
```

```

lhs=sym.simplify(a**2-a*b+a*b-b**2)
rhs=sym.simplify(a**2-b**2)
print('lhs is ',lhs)
print('rhs is ',rhs)

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\a4-1\
a1.py"
lhs is  a**2 - b**2
rhs is  a**2 - b**2

```

Result: Hence the program has been executed.

Problem:

Give the Symbolic program for the expression given below:

a. $\int \int a^2 da$

Program:

```

from sympy import *
import functools
import operator
import random
a=symbols('a')
g=a**2
j=integrate(g,a)
print('before integration a**2')
print('after integration '+str(j))

```

Input/Output:

```

PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\a4-1\
a1.py"
before integration a**2
after integration a**3/3

```

Result: Hence the program has been executed.

Problem:

Give the Symbolic program for the expression given below:

b. $2x+y^2$

Program:

```

import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
a = sym.simplify(2*x+y**2)
print(a)

```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\A4-1\tempCodeRunnerFile.py"
2*x + y**2
```

Result: Hence the program has been executed.

Problem:

Give the Symbolic program for the expression given below:

- c. $\frac{1}{10} + \frac{1}{5}$

Program:

```
import sympy as sym
a = (sym.Rational(1, 10)+sym.Rational(1,5))
print(sym.simplify(a))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\A4-1\tempCodeRunnerFile.py"
3/10
```

Result: Hence the program has been executed.

Problem:

Give the Symbolic program for the expression given below:

- d. $\frac{d}{dx}(\sin(x))$

Program:

```
from sympy import *
import functools
import operator
import random
a=symbols('a')
x=symbols('x')
g=sin(x)
j=diff(g,a)

print('before differentiation sin(x)')
print('after differentiation '+str(j))
```

Input/Output:

```
PS C:\Users\nikhi\Downloads\GUI> python -u "c:\Users\nikhi\Downloads\GUI\A4\A4-1\tempCodeRunnerFile.py"
before differentiation sin(x)
after differentiation 0
```

Result: Hence the program has been executed.