AMITY SCHOOL OF ENGINERRING & TECHNOLOGY

AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

LAB REPORT & ASSIGNMENTS

(ACADEMIC YEAR 2021-22)

COURSE NAME: PYTHON PROGRAMMING LAB

COURSE CODE: CSE2410

DEPARTMENT: ASET

FACULTY NAME: DR. SONI SWETA

SUBMITTED BY

STUDENT NAME: GAURAV BHANOT

ENROLLMENT NUMBER: A70405220085

CLASS: BTECH CSE DIV B

SEMESTER: IV

DATE OF SUBMISSION: 2nd June 2022

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

CERTIFICATE OF SUBMISSION

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: - A70405220085

This is certified to be the bonafide work of student in Python

Laboratory during the academic year 2021-22.

Faculty In-charge
{Department of_____}
ASET, AUM

Department Coordinate
{Department of_____}
ASET, AUM

HoI
ASET & AIT, AUM

Date:

Stamp

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

INDEX

Faculty In-charge
{Department of_____}
ASET, AUM

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 1
Programs based on List, Tuple, Dictionary and String data structures and operations performed on it.

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Programs based on List, Tuple, Dictionary and String data structures and operations performed on it.

1.Write a Python script to check whether a given key already exists in a dictionary.

In [1]:
```
d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

def is_key_present(x):
    '''Using if and in'''
    if x in d:
        print('Key is present in the dictionary')
    else:
        print('Key is not present in the dictionary')
is_key_present(5)
is_key_present(9)
```

```
Key is present in the dictionary
Key is not present in the dictionary
```

In [2]:
```
def checkKey(dict, key):
    '''Using Inbuilt method keys()'''
    if key in dict.keys():
        print("Present, ", end =" ")
        print("value =", dict[key])
    else:
        print("Not present")

dict = {'a': 100, 'b':200, 'c':300}
key = 'b'
checkKey(dict, key)
key = 'w'
checkKey(dict, key)
```

```
Present,  value = 200
Not present
```

2.Write a Python program to get the key, value, and item in a dictionary.

In [3]:
```
dict_num = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
# dict.keys() is use for accessing keys in dictionary
print(dict_num.keys())
# dict.values() is use for accessing values in dictionary
print(dict_num.values())
# dict.items() is use for accessing key-values in dictionary
print(dict_num.items())
```

```
dict_keys([1, 2, 3, 4, 5, 6])
dict_values([10, 20, 30, 40, 50, 60])
dict_items([(1, 10), (2, 20), (3, 30), (4, 40), (5, 50), (6, 60)])
```

3.Write a Python program to combine two lists into a dictionary, where the elements of the first one serves as the keys and the elements of the second one serves as the values. The values of the first list need to be unique.

In [4]:
```
'''Using naive method '''
test_keys = [1, 2, 3]
test_values = ["Roy", "John", "Elon"]
```

```
print ("Original key list is : " + str(test_keys))
print ("Original value list is : " + str(test_values))

res = {}
for key in test_keys:
    for value in test_values:
        res[key] = value
        test_values.remove(value)
        break
print ("Resultant dictionary is : " +  str(res))
```

```
Original key list is : [1, 2, 3]
Original value list is : ['Roy', 'John', 'Elon']
Resultant dictionary is : {1: 'Roy', 2: 'John', 3: 'Elon'}
```

In [5]:
```
'''Using dictionary comprehension '''
test_keys = [1, 2, 3]
test_values = ["Roy", "John", "Elon"]
print ("Original key list is : " + str(test_keys))
print ("Original value list is : " + str(test_values))
res = {test_keys[i]: test_values[i] for i in range(len(test_keys))}
print ("Resultant dictionary is : " +  str(res))
```

```
Original key list is : [1, 2, 3]
Original value list is : ['Roy', 'John', 'Elon']
Resultant dictionary is : {1: 'Roy', 2: 'John', 3: 'Elon'}
```

4.Create a list and tuple and apply all the operations on them.

1.List Operation

In [6]:
```
myList  =  ['Mango','Apple','Orange','Grapes']
myList
```

Out[6]:
```
['Mango', 'Apple', 'Orange', 'Grapes']
```

a) append()

In [7]:
```
myList.append('Banana')
myList
```

Out[7]:
```
['Mango', 'Apple', 'Orange', 'Grapes', 'Banana']
```

b) extend()

In [8]:
```
myList.extend(['Guava','Pineapple'])
myList
```

Out[8]:
```
['Mango', 'Apple', 'Orange', 'Grapes', 'Banana', 'Guava', 'Pineapple']
```

c) remove()

In [9]:
```
myList.remove("Guava")
myList
```

Out[9]:
```
['Mango', 'Apple', 'Orange', 'Grapes', 'Banana', 'Pineapple']
```

d) pop()

In [10]:
```
myList.pop()
myList
```

Out[10]:
```
['Mango', 'Apple', 'Orange', 'Grapes', 'Banana']
```

e) insert()

In [11]:
```
myList.insert(2,'Guava')
myList
```

Out[11]:
```
['Mango', 'Apple', 'Guava', 'Orange', 'Grapes', 'Banana']
```

f) slice()

In [12]:
```
print(myList[2:4])
```

```
['Guava', 'Orange']
```

g) sort()

In [13]:
```
myList.sort()
myList
```

Out[13]:
```
['Apple', 'Banana', 'Grapes', 'Guava', 'Mango', 'Orange']
```

h) clear()

In [14]:
```
myList.clear()
myList
```

Out[14]:
```
[]
```

2.Tuple Operation

In [15]:
```
myTuple = (53,55,34,22,19,71)
myTuple
```

Out[15]:
```
(53, 55, 34, 22, 19, 71)
```

a) Accessing Items in a Tuple

In [16]:
```
print(myTuple[3])
```

```
22
```

b) Slicing Operation on Tuples

In [17]:
```
print(myTuple[2:])
```

```
(34, 22, 19, 71)
```

c) Finding length of Tuples

In [18]:
```
print(len(myTuple))
```

```
6
```

d) Membership Test on Tuples

In [19]:
```
print(53 in myTuple)
print(69 in myTuple)
```

```
True
False
```

e) max() and min()

In [20]:
```
print(max(myTuple))
```

```
71
```

In [21]:
```
print(min(myTuple))
```

```
19
```

f) sum()

In [22]:
```
print(sum(myTuple))
```

```
254
```

g) sorted()

In [23]:
```
print(sorted(myTuple))
```

```
[19, 22, 34, 53, 55, 71]
```

h) del()

In [24]:
```
del(myTuple)
myTuple
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-6bf531af9771> in <module>
      1 del(myTuple)
----> 2 myTuple

NameError: name 'myTuple' is not defined
```

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 2
Programs based on functions (positional argument, default argument and variable length arguments).

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Programs based on functions (positional argument, default argument and variable length arguments).

5.Create a function with variable length of arguments

In [1]:
```
def my_max(*args):
    '''Using Many Arguments with *args'''
    result = args[0]
    for num in args:
        if num > result:
            result = num
    return result

my_max(7, 15, 10, 7, 3)
```

Out[1]:
```
15
```

6.Create a function which return multiple values from a function

In [5]:
```
def multiple_values():
    '''This function returns a tuple'''
    str1 = "Amity"
    str2 = "University"
    return str1, str2;

str1, str2 = multiple_values()
print(str1)
print(str2)
```

```
Amity
University
```

In [6]:
```
def multiple_values():
    '''This function returns a list'''
    str1 = "India"
    str2 = "Delhi"
    return [str1, str2];

list = multiple_values()
print(list)
```

```
['India', 'Delhi']
```

7.Create a function with positional and default argument

In [7]:
```
#Following function has 2 default arguments
def student(firstname, lastname ='Sharma', standard ='10th'):
     print(firstname, lastname, 'studies in', standard, 'Standard')
# 1 positional argument
student('Rohit')

# 2 positional arguments
student('Rohit',  'Singh')
student('Rohit', '7th')

# 3 positional arguments
student('Rohit', 'Singh', '7th')
```

```
Rohit Sharma studies in 10th Standard
Rohit Singh studies in 10th Standard
```

Rohit 7th studies in 10th Standard
Rohit Singh studies in 7th Standard

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 3
Program based on exceptions handling

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Program based on exceptions handling

8.Write a python program to handle multiple error with one except statement.

In [4]:
```
try:
    print(3/0)
    name = 'Amity University'
    name += 5
except (ZeroDivisionError, NameError, TypeError) as error:
    print(error)
```

```
division by zero
```

In [5]:
```
try:
#   print(3/0)
    name = 'Amity University'
    name += 5
except (ZeroDivisionError, NameError, TypeError) as error:
    print(error)
```

```
can only concatenate str (not "int") to str
```

9.Write a python program to depict else clause with try-except

In [1]:
```
def divide(x, y):
    '''This code is to run else clause'''
    try:
        result = x // y
    except  ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")
    else:
        print("Yeah ! Your answer is :", result)

# Look at parameters and note the working of Program
divide(3, 2)
divide(3, 0)
```

```
Yeah ! Your answer is : 1
Sorry ! You are dividing by zero
```

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 4
Program based on file handling.

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Program based on file handling.

10. Write a Python program to read last n lines of a file.

In [6]:
```
with open('states.txt') as f:
    lines = [line.rstrip() for line in f]
    print(lines)
```

```
['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Chhattisgarh', 'Goa', 'Gu
jarat', 'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir', 'Jharkhand', 'Karnatak
a', 'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram', 'N
agaland', 'Odisha', 'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telangana', 'Tri
pura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal']
```

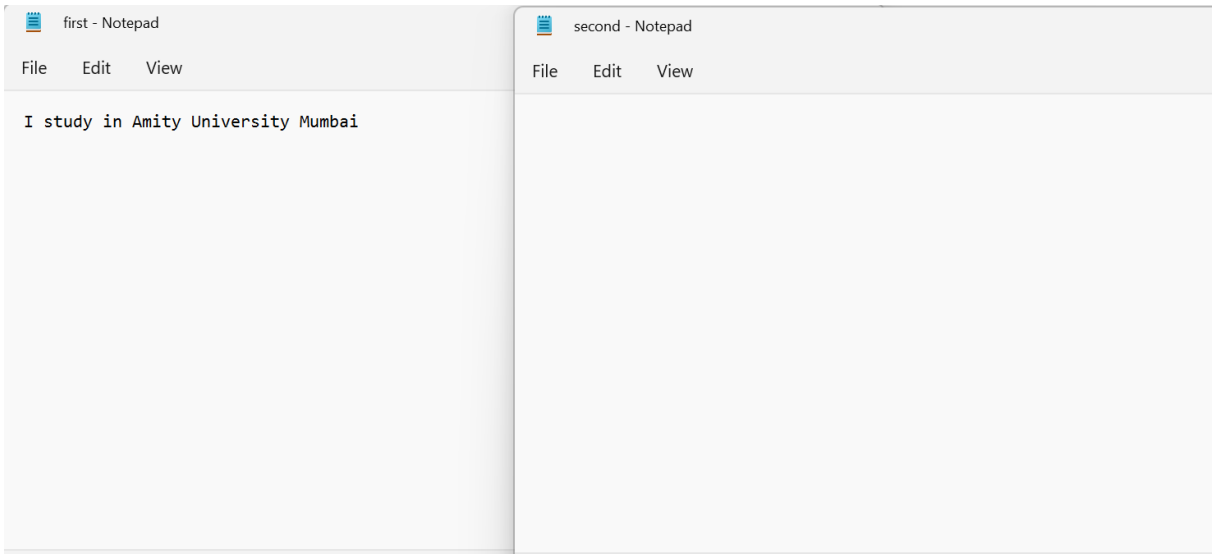
In [3]:
```
def  read_lastnlines(fname,n):
    '''Reading last n lines of file'''
    with open('states.txt') as f:
        for line in (f.readlines() [-n:]):
            print(line)

read_lastnlines('states.txt',3)
```

```
Uttar Pradesh

Uttarakhand

West  Bengal
```

11. Write a Python program to copy the contents of a file to another file.

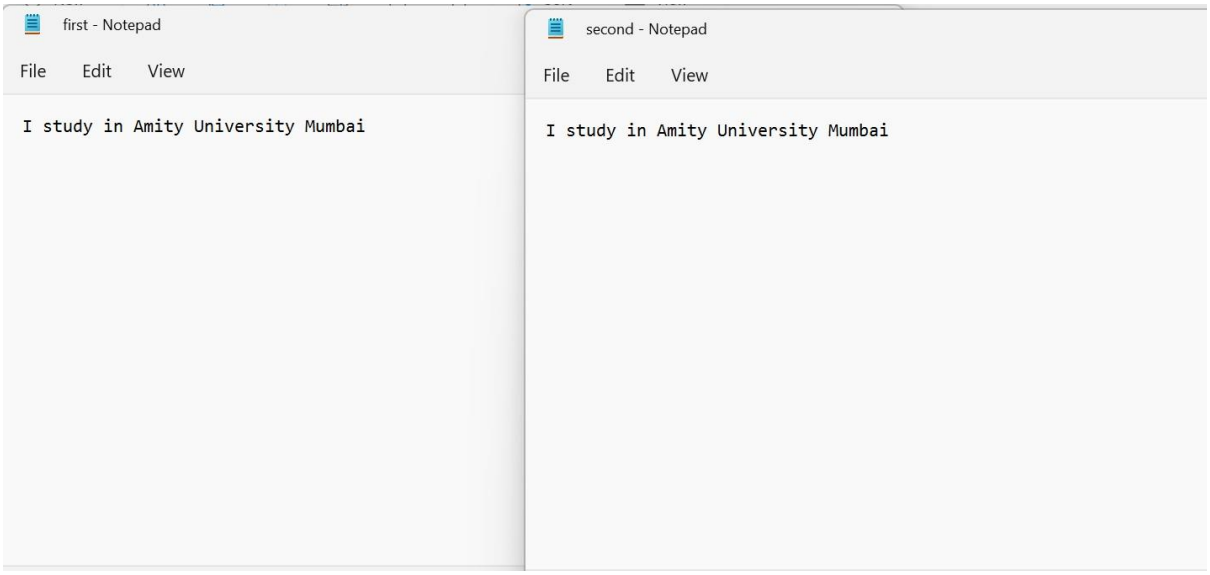
Method1 : Using File handling to read and append

Before Copying File



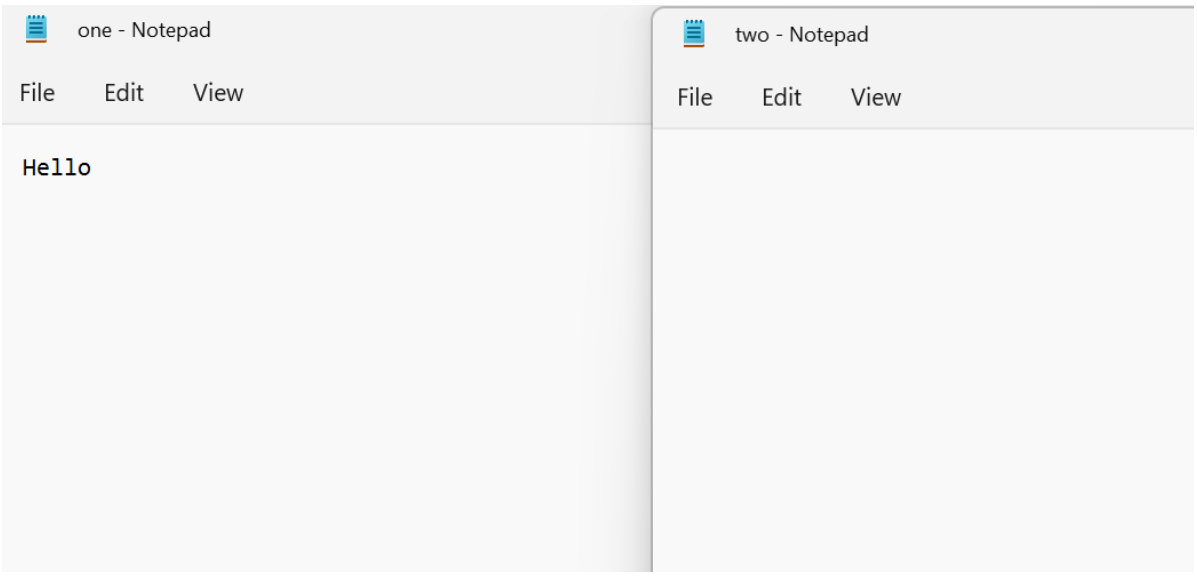
In [7]:
```
with open('first.txt','r') as firstfile, open('second.txt','a') as secondfile:
          for line in firstfile:
             secondfile.write(line)
```

After Copying File



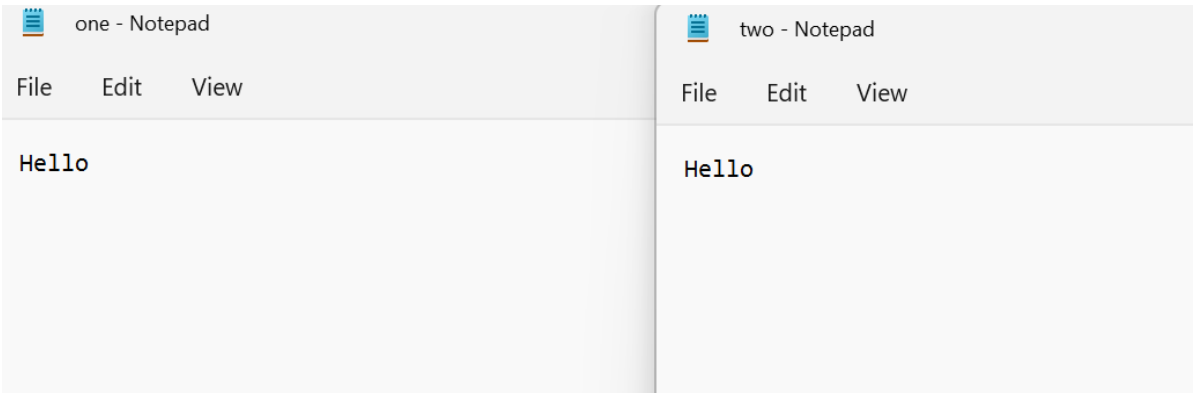
Method2 : Using File handling to read and write

Before Copying File



In [9]:
```
with open('one.txt','r') as firstfile, open('two.txt','w') as secondfile:
    for line in firstfile:
            secondfile.write(line)
```

After Copying File



AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 5
Programs based on N dimensional array using Numpy array

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Programs based on N dimensional array using Numpy array

12. Creation of array using Numpy module and perform many operations on it.

In [1]:
```
import numpy as np
a = np.array([5,6,9])
a
```

Out[1]:
```
array([5, 6, 9])
```

In [2]:
```
print(a)
```

```
[5 6 9]
```

In [3]:
```
a.ndim
```

Out[3]:
```
1
```

In [4]:
```
b = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(b)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [5]:
```
b.ndim
```

Out[5]:
```
2
```

In [6]:
```
b.shape
```

Out[6]:
```
(3, 3)
```

In [7]:
```
b.itemsize
```

Out[7]:
```
4
```

In [8]:
```
b.dtype
```

Out[8]:
```
dtype('int32')
```

In [9]:
```
a.size
```

Out[9]:
```
3
```

In [10]:
```
b.size
```

Out[10]:
```
9
```

In [11]:
```
c = np.zeros((3,4))
```

In [12]:
```
c
```

Out[12]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
```

```
       [0., 0., 0., 0.]])
```

In [13]:
```
d = np.ones((2,3))
d
```

Out[13]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [14]:
```
e = np.arange(1,7)
e
```

Out[14]:
```
array([1, 2, 3, 4, 5, 6])
```

In [15]:
```
f = np.arange(1,9,2)
f
```

Out[15]:
```
array([1, 3, 5, 7])
```

In [16]:
```
g = np.linspace(1,5,10)
g
```

Out[16]:
```
array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,
       3.22222222, 3.66666667, 4.11111111, 4.55555556, 5.        ])
```

In [17]:
```
g2 = np.linspace(1,5,20)
g2
```

Out[17]:
```
array([1.        , 1.21052632, 1.42105263, 1.63157895, 1.84210526,
       2.05263158, 2.26315789, 2.47368421, 2.68421053, 2.89473684,
       3.10526316, 3.31578947, 3.52631579, 3.73684211, 3.94736842,
       4.15789474, 4.36842105, 4.57894737, 4.78947368, 5.        ])
```

In [18]:
```
h = np.array([[1,2],[3,4]])
h
h.min()
```

Out[18]:
```
1
```

In [19]:
```
h.max()
```

Out[19]:
```
4
```

In [20]:
```
h.sum()
```

Out[20]:
```
10
```

In [21]:
```
h
```

Out[21]:
```
array([[1, 2],
       [3, 4]])
```

In [22]:
```
#axis = 0 is row
# axis = 1 is column
h.sum(axis=0)
```

Out[22]:
```
array([4, 6])
```

In [23]:
```
h.sum(axis=1)
```

Out[23]:
```
array([3, 7])
```

In [24]:
```
np.sqrt(h)
```

Out[24]:
```
array([[1.        , 1.41421356],
       [1.73205081, 2.        ]])
```

In [25]:
```
np.std(h)
```

Out[25]:
```
1.1180339887498951
```

In [26]:
```
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
x + y
```

Out[26]:
```
array([[ 6,  8],
       [10, 12]])
```

In [27]:
```
x[0:2]
```

Out[27]:
```
array([[1, 2],
       [3, 4]])
```

In [28]:
```
a[0:2]
```

Out[28]:
```
array([5, 6])
```

In [29]:
```
x[0,1]
```

Out[29]:
```
2
```

In [30]:
```
b[0:2,2]
```

Out[30]:
```
array([3, 6])
```

In [31]:
```
b[-1]
```

Out[31]:
```
array([7, 8, 9])
```

In [32]:
```
b[2]
```

Out[32]:
```
array([7, 8, 9])
```

In [33]:
```
b[0]
```

Out[33]:
```
array([1, 2, 3])
```

In [34]:
```
b[2,0:2]
```

Out[34]:
```
array([7, 8])
```

In [35]:
```
for row in b:
  print(row)
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

In [36]:
```
for cell in b.flat:
  print(cell)
```

```
1
2
```

```
3
4
5
6
7
8
9
```

In [37]:
```
i = np.arange(6).reshape(3,2)
j =  np.arange(6,12).reshape(3,2)
i
```

Out[37]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [38]:
```
j
```

Out[38]:
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [39]:
```
np.vstack((i,j))
```

Out[39]:
```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [40]:
```
np.hstack((i,j))
```

Out[40]:
```
array([[ 0,  1,  6,  7],
       [ 2,  3,  8,  9],
       [ 4,  5, 10, 11]])
```

In [41]:
```
k = np.arange(30).reshape(2,15)
```

In [42]:
```
k
```

Out[42]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])
```

In [43]:
```
np.hsplit(k,3)
```

Out[43]:
```
[array([[ 0,  1,  2,  3,  4],
        [15, 16, 17, 18, 19]]),
 array([[ 5,  6,  7,  8,  9],
        [20, 21, 22, 23, 24]]),
 array([[10, 11, 12, 13, 14],
        [25, 26, 27, 28, 29]])]
```

In [44]:
```
np.vsplit(k,2)
```

Out[44]:
```
[array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]]),
 array([[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]])]
```

In [45]:
```
i = j>2
i
```

Out[45]:
```
array([[ True,  True],
       [ True,  True],
       [ True,  True]])
```

In [46]:
```
i = i>2
```

In [47]:
```
i
```

Out[47]:
```
array([[False, False],
       [False, False],
       [False, False]])
```

In [48]:
```
j = i > 4
```

In [49]:
```
j
```

Out[49]:
```
array([[False, False],
       [False, False],
       [False, False]])
```

In [50]:
```
v = j < 2
v
```

Out[50]:
```
array([[ True,  True],
       [ True,  True],
       [ True,  True]])
```

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 6
Programs based on Data Manipulation using Pandas

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

Programs based on Data Manipulation using Pandas

13. Creation of series using Pandas module and perform many operations on it.

In [1]:
```
import pandas as pd
```

In [2]:
```
s1 = pd.Series([1,2,3,4,5])
s1
```

Out[2]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [3]:
```
type(s1)
```

Out[3]:
```
pandas.core.series.Series
```

In [4]:
```
s2 = pd.Series({'a': 10, 'b': 20, 'c': 30})
s2
```

Out[4]:
```
a    10
b    20
c    30
dtype: int64
```

In [5]:
```
idex = ['x','y','z']
s2.index = idex
s2
```

Out[5]:
```
x    10
y    20
z    30
dtype: int64
```

In [6]:
```
s2['y']
```

Out[6]:
```
20
```

In [7]:
```
df  =  pd.DataFrame({"Name":  ['John','Roy','Ram','Shyam'],"Marks":[50,40,60,90]})
df
```

Out[7]:

	Name	Marks
0	John	50
1	Roy	40
2	Ram	60
3	Shyam	90

In [8]:
```
type(df)
```

Out[8]:
```
pandas.core.frame.DataFrame
```

In [9]:
```
df.head(2)
```

Out[9]:

	Name	Marks
0	John	50
1	Roy	40

In [10]:
```
df.tail(1)
```

Out[10]:

	Name	Marks
3	Shyam	90

In [11]:
```
df.shape
```

Out[11]:
```
(4, 2)
```

In [12]:
```
df.describe()
```

Out[12]:

	Marks
count	4.000000
mean	60.000000
std	21.602469
min	40.000000
25%	47.500000
50%	55.000000
75%	67.500000
max	90.000000

In [13]:
```
df = pd.read_csv("churn_prediction.csv")
```

In [14]:
```
df.head(5)
```

Out[14]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category	b
0	1	3135	66	Male	0.0	self_employed	187.0	2	
1	2	310	35	Male	0.0	self_employed	NaN	2	
2	4	2356	31	Male	0.0	salaried	146.0	2	
3	5	478	90	NaN	NaN	self_employed	1020.0	2	
4	6	2531	42	Male	2.0	self_employed	1494.0	3	

5 rows × 21 columns

In [30]:
```
df.tail(5)
```

Out[30]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_categor
28377	30297	1845	10	Female	0.0	student	1020.0	

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_categor
28378	30298	4919	34	Female	0.0	self_employed	1046.0	
28379	30299	297	47	Male	0.0	salaried	1096.0	
28380	30300	2585	50	Male	3.0	self_employed	1219.0	
28381	30301	2349	18	Male	0.0	student	1232.0	

5 rows × 21 columns

In [15]:
```
df.describe()
```

Out[15]:

	customer_id	vintage	age	dependents	city	customer_nw_category
count	28382.000000	28382.000000	28382.000000	25919.000000	27579.000000	28382.000000
mean	15143.508667	2364.336446	48.208336	0.347236	796.109576	2.225530
std	8746.454456	1610.124506	17.807163	0.997661	432.872102	0.660443
min	1.000000	180.000000	1.000000	0.000000	0.000000	1.000000
25%	7557.250000	1121.000000	36.000000	0.000000	409.000000	2.000000
50%	15150.500000	2018.000000	46.000000	0.000000	834.000000	2.000000
75%	22706.750000	3176.000000	60.000000	0.000000	1096.000000	3.000000
max	30301.000000	12899.000000	90.000000	52.000000	1649.000000	3.000000

In [16]:
```
df.min()
```

```
<ipython-input-16-c3612c624a3f>:1: FutureWarning: Dropping of nuisance columns in Da
taFrame reductions (with 'numeric_only=None') is deprecated; in a future version thi
s will raise TypeError.  Select only valid columns before calling the reduction.
  df.min()
```

Out[16]:
```
customer_id                        1.00
vintage                          180.00
age                                1.00
dependents                         0.00
city                               0.00
customer_nw_category               1.00
branch_code                        1.00
days_since_last_transaction        0.00
current_balance                -5503.96
previous_month_end_balance     -3149.57
average_monthly_balance_prevQ   1428.69
average_monthly_balance_prevQ2 -16506.10
current_month_credit               0.01
previous_month_credit              0.01
current_month_debit                0.01
previous_month_debit               0.01
current_month_balance          -3374.18
previous_month_balance         -5171.92
churn                              0.00
dtype: float64
```

In [17]:
```
df[20:30]
```

Out[17]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category
20	23	5724	45	Male	0.0	self_employed	1020.0	2
21	24	2083	29	Female	0.0	self_employed	1020.0	2
22	25	3101	41	Female	0.0	self_employed	905.0	2
23	26	1897	34	Male	0.0	self_employed	931.0	2
24	27	754	48	Male	2.0	salaried	218.0	2
25	28	606	76	Male	NaN	self_employed	1533.0	3
26	29	2620	36	Male	2.0	self_employed	1563.0	1
27	30	1391	56	Male	0.0	self_employed	836.0	3
28	31	4175	55	Male	0.0	self_employed	118.0	1
29	32	2204	33	Female	0.0	salaried	834.0	2

10 rows × 21 columns

Data Extraction

In [18]:
```
df.iloc[0:3,0:2]
```

Out[18]:

	customer_id	vintage
0	1	3135
1	2	310
2	4	2356

In [19]:
```
df.iloc[30:40,3:]
```

Out[19]:

	gender	dependents	occupation	city	customer_nw_category	branch_code	days_since_last_
30	Male	0.0	retired	1096.0	1	4029	
31	Female	0.0	retired	1020.0	1	60	
32	Male	0.0	self_employed	1366.0	2	797	
33	NaN	0.0	self_employed	834.0	3	8	
34	Male	0.0	student	623.0	2	512	
35	Male	7.0	self_employed	549.0	2	264	
36	Male	0.0	self_employed	630.0	2	36	
37	Male	3.0	self_employed	1271.0	2	101	
38	Male	0.0	self_employed	834.0	2	1130	
39	Male	0.0	salaried	485.0	3	823	

In [21]:
```
df["city"]
```

Out[21]:
```
0        187.0
```

```
1           NaN
2         146.0
3        1020.0
4        1494.0
          ...
28377    1020.0
28378    1046.0
28379    1096.0
28380    1219.0
28381    1232.0
Name: city, Length: 28382, dtype: float64
```

In [22]:
```
df.loc[0:3,('branch_code','current_balance')]
```

Out[22]:

	branch_code	current_balance
0	755	1458.71
1	3214	5390.37
2	41	3913.16
3	582	2291.91

In [23]:
```
df1 = df.drop(["gender"],axis = 1)
df1
```

Out[23]:

	customer_id	vintage	age	dependents	occupation	city	customer_nw_category	branc
0	1	3135	66	0.0	self_employed	187.0	2	
1	2	310	35	0.0	self_employed	NaN	2	
2	4	2356	31	0.0	salaried	146.0	2	
3	5	478	90	NaN	self_employed	1020.0	2	
4	6	2531	42	2.0	self_employed	1494.0	3	
...	...	...	...	...	...	...	...	
28377	30297	1845	10	0.0	student	1020.0	2	
28378	30298	4919	34	0.0	self_employed	1046.0	2	
28379	30299	297	47	0.0	salaried	1096.0	2	
28380	30300	2585	50	3.0	self_employed	1219.0	3	
28381	30301	2349	18	0.0	student	1232.0	2	

28382 rows × 20 columns

In [25]:
```
df.mean()
```

```
<ipython-input-25-c61f0c8f89b5>:1: FutureWarning: Dropping of nuisance columns in Da
taFrame reductions (with 'numeric_only=None') is deprecated; in a future version thi
s will raise TypeError.  Select only valid columns before calling the reduction.
  df.mean()
```

Out[25]:
```
customer_id                15143.508667
vintage                     2364.336446
age                           48.208336
dependents                     0.347236
city                         796.109576
customer_nw_category           2.225530
```

```
branch_code                          925.975019
days_since_last_transaction           69.997814
current_balance                     7380.551804
previous_month_end_balance          7495.770548
average_monthly_balance_prevQ       7496.779856
average_monthly_balance_prevQ2      7124.209162
current_month_credit                3433.252240
previous_month_credit               3261.694458
current_month_debit                 3658.744549
previous_month_debit                3339.761353
current_month_balance               7451.132765
previous_month_balance              7495.177129
churn                                  0.185329
dtype: float64
```

14. Creation of ND array using Pandas module and perform many data manipulation operations on it using IRIS dataset.

In [26]:
```
import seaborn as sns
iris = sns.load_dataset('iris')
iris
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [27]:
```
iris.head(5)
```

Out[27]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [28]:
```
iris.tail(5)
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width	species

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [29]:
```
iris.describe()
```

Out[29]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [31]:
```
iris[20:30]
```

Out[31]:

	sepal_length	sepal_width	petal_length	petal_width	species
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
22	4.6	3.6	1.0	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
26	5.0	3.4	1.6	0.4	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
29	4.7	3.2	1.6	0.2	setosa

Data Extraction

In [34]:
```
iris.iloc[0:3,0:2]
```

Out[34]:

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2

In [35]:
```
iris.iloc[30:40,3:]
```

Out[35]:

	petal_width	species
30	0.2	setosa
31	0.4	setosa
32	0.1	setosa
33	0.2	setosa
34	0.2	setosa
35	0.2	setosa
36	0.2	setosa
37	0.1	setosa
38	0.2	setosa
39	0.2	setosa

In [37]:
```
iris["species"]
```

Out[37]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

In [39]:
```
iris.loc[0:3,('petal_width','species')]
```

Out[39]:

	petal_width	species
0	0.2	setosa
1	0.2	setosa
2	0.2	setosa
3	0.2	setosa

In [40]:
```
iris1 = iris.drop(["species"],axis = 1)
iris1
```

Out[40]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	sepal_length	sepal_width	petal_length	petal_width
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [41]:
```
iris.mean()
```

```
<ipython-input-41-7eed97565d6e>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  iris.mean()
```

Out[41]:
```
sepal_length    5.843333
sepal_width     3.057333
petal_length    3.758000
petal_width     1.199333
dtype: float64
```

AMITY SCHOOL OF ENGINERRING & TECHNOLOGY
AMITY INSTITUTE OF TECHNOLOGY

AMITY UNIVERSITY

MAHARASHTRA

(Academic Year 2021-22)

LAB 7
Create any two GUI applications using Tkinter

Student Name: Gaurav Bhanot

Class: BTECH CSE DIV B Semester: IV

Enrolment Number: A70405220085

Faculty In-charge
{Department of_____}
ASET, AUM

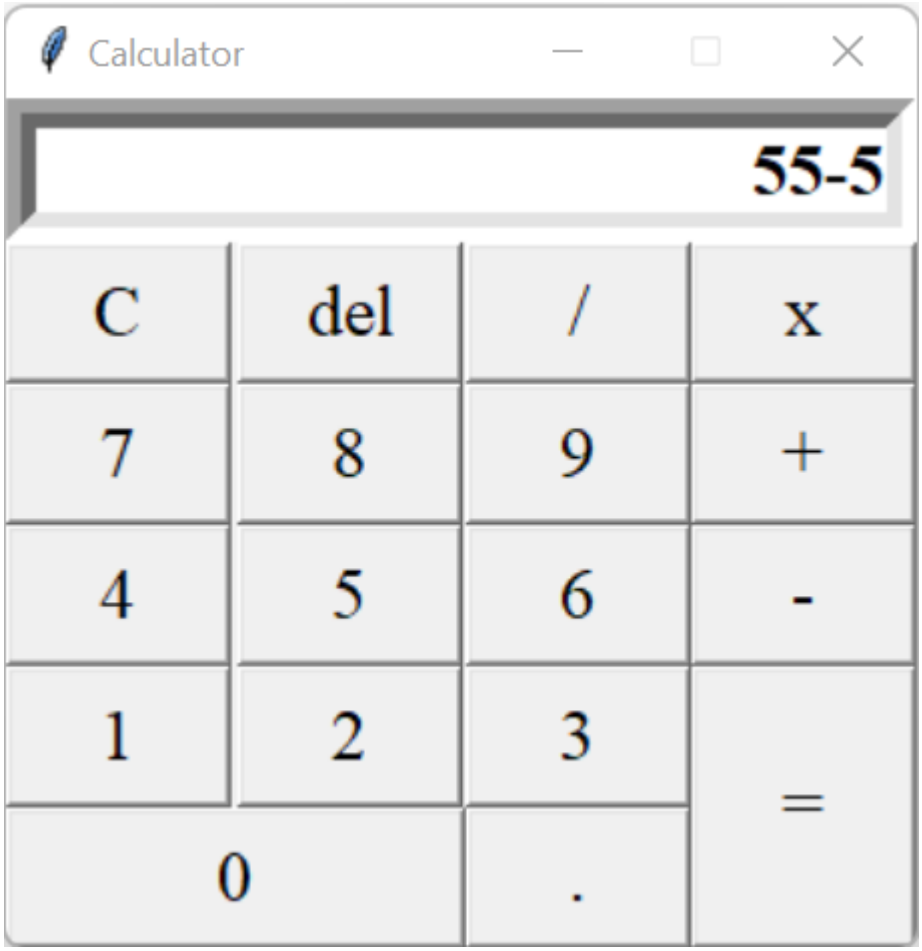
15. Create any two GUI applications using Tkinter

a) Create a simple calculator

In [1]:

```
from tkinter import *

win = Tk()
win.title("Calculator")
num = StringVar()
win.resizable(False,False)
# StringVar is a class that provides helper functions for directly creating and acce
# interpreter.
n1 = ""
entry = Entry(win, textvariable=num, font=("Times", 18, "bold"), width=5, bd=10, jus
# bd : This option used to represent the size of the border around the indicator and

entry.grid(sticky="WE",  column=1,  columnspan=4)


# The Grid geometry manager puts the widgets in a 2-dimensional table. The master wi
# rows and columns, and each “cell” in the resulting table can hold a widget

# The eval() method parses the expression passed to this method and runs python expr

def add(n):
    n1 = num.get()
    if n == "Error":
        num.set(n)
    else:
        num.set(n1 + n)


def cal():
    try:
        num.set(eval(num.get()))
    except:
        num.set("Error")


def delete():
    num.set(num.get()[:-1])


def reset():
    num.set("")


clear = Button(win, text="C", command=reset, font="Times 18", width=5).grid(row=1, c
delete = Button(win, text="del", command=delete, font="Times 18", width=5).grid(row=
div = Button(win, text="/", command=lambda: add("/"), font="Times 18", width=5).grid
mul = Button(win, text="x", command=lambda: add("*"), font="Times 18", width=5).grid

seven = Button(win, text="7", command=lambda: add("7"), font="Times 18", width=5).gr
eight = Button(win, text="8", command=lambda: add("8"), font="Times 18", width=5).gr
nine = Button(win, text="9", command=lambda: add("9"), font="Times 18", width=5).gri
addition = Button(win, text="+", command=lambda: add("+"), font="Times 18", width=5)

four = Button(win, text="4", command=lambda: add("4"), font="Times 18", width=5).gri
five = Button(win, text="5", command=lambda: add("5"), font="Times 18", width=5).gri
six = Button(win, text="6", command=lambda: add("6"), font="Times 18", width=5).grid
```

```
sub = Button(win, text="-", command=lambda: add("-"), font="Times 18", width=5).grid
one = Button(win, text="1", command=lambda: add("1"), font="Times 18", width=5).grid
two = Button(win, text="2", command=lambda: add("2"), font="Times 18", width=5).grid
three = Button(win, text="3", command=lambda: add("3"), font="Times 18", width=5).gr

equal = Button(win, text="=", command=cal, font="Times 18", width=5).grid(sticky="NS
zero = Button(win, text="0", command=lambda: add("0"), font="Times 18", width=11).gr
dot = Button(win, text=".", command=lambda: add("."), font="Times 18", width=5).grid

win.mainloop()
```



b) Create a registration and login form

In [ ]:

```
from tkinter import *
from tkinter import messagebox
import sqlite3

f = ('Times', 14)

con = sqlite3.connect('userdata.db')
cur = con.cursor()
cur.execute('''CREATE TABLE IF NOT EXISTS record(
                    name text,
                    email text,
                    contact number,
                    gender text,
                    country text,
                    password text
                )
            ''')
con.commit()

ws = Tk()
```

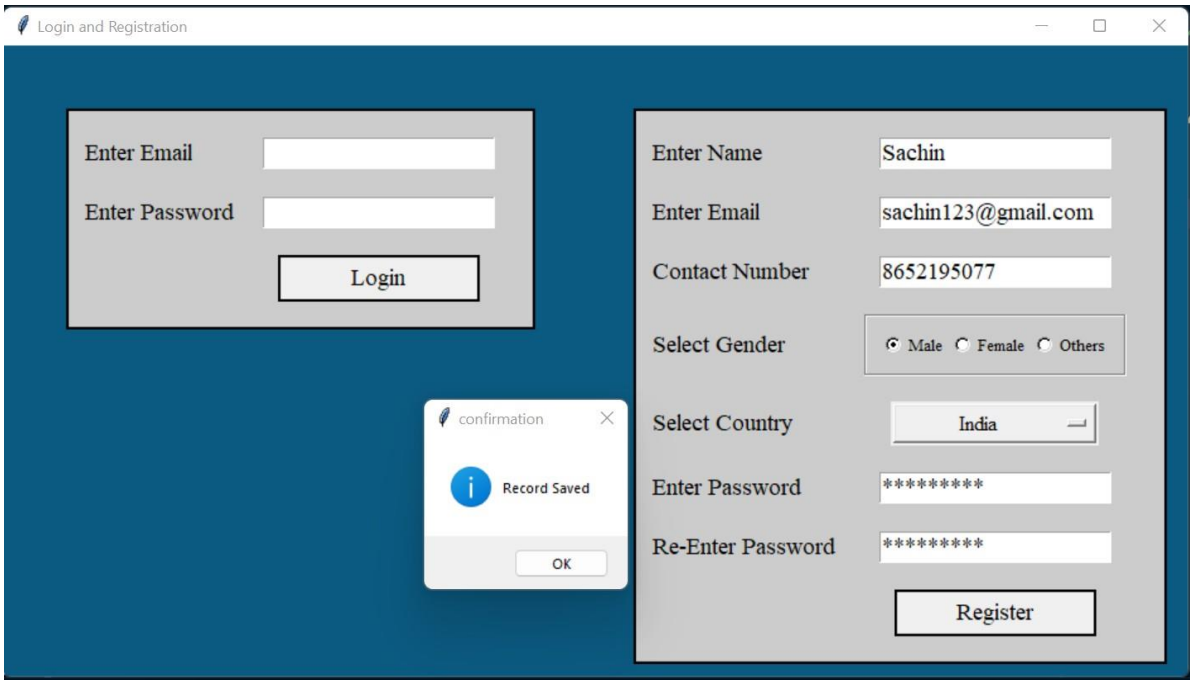
```
ws.title('Login  and  Registration')
ws.geometry('940x500')
ws.config(bg='#0B5A81')


def insert_record():
    check_counter = 0
    warn = ""
    if register_name.get() == "":
        warn = "Name can't be empty"
    else:
        check_counter += 1

    if register_email.get() == "":
        warn = "Email can't be empty"
    else:
        check_counter += 1

    if register_mobile.get() == "":
        warn = "Contact can't be empty"
    else:
        check_counter += 1

    if var.get() == "":
        warn = "Select Gender"
    else:
        check_counter += 1

    if variable.get() == "":
        warn = "Select Country"
    else:
        check_counter += 1

    if register_pwd.get() == "":
        warn = "Password can't be empty"
    else:
        check_counter += 1

    if pwd_again.get() == "":
        warn = "Re-enter password can't be empty"
    else:
        check_counter += 1

    if register_pwd.get() != pwd_again.get():
        warn = "Passwords didn't match!"
    else:
        check_counter += 1

    if check_counter == 8:
        try:
            con  =  sqlite3.connect('userdata.db')
            cur = con.cursor()
            cur.execute("INSERT INTO record VALUES (:name, :email, :contact, :gender
                'name': register_name.get(),
                'email':  register_email.get(),
                'contact':  register_mobile.get(),
                'gender': var.get(),
                'country':  variable.get(),
                'password':  register_pwd.get()

            })
            con.commit()
            messagebox.showinfo('confirmation',  'Record  Saved')
```

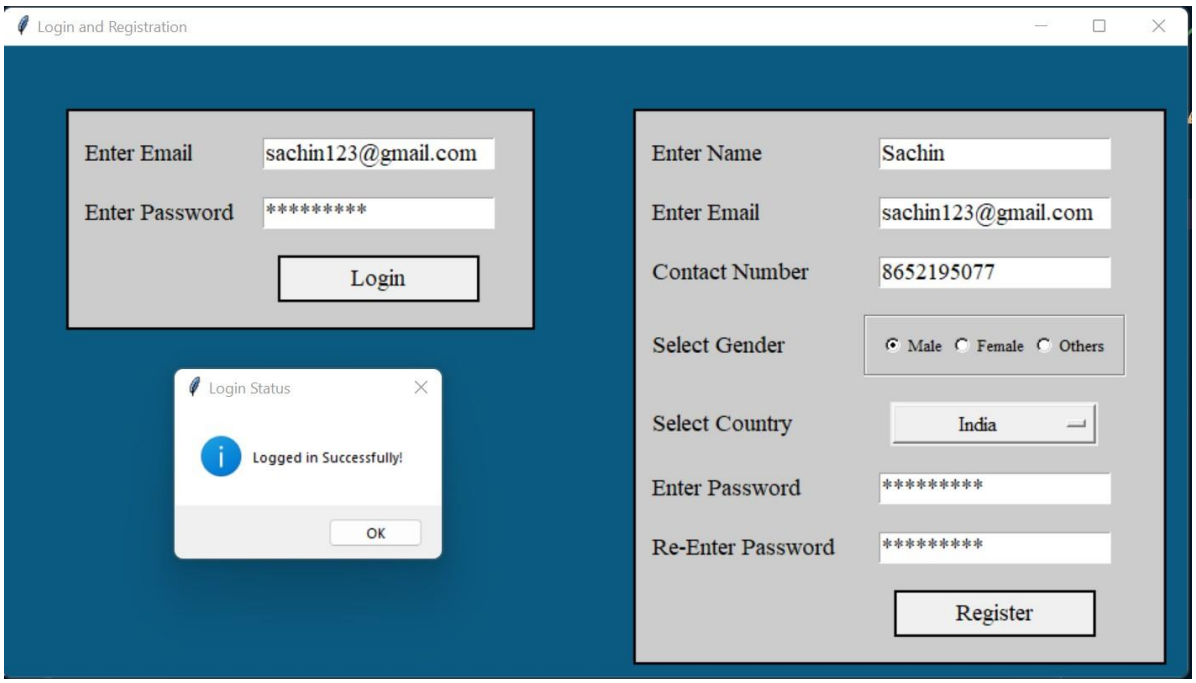
```
        except Exception as ep:
            messagebox.showerror('', ep)
    else:
        messagebox.showerror('Error', warn)


def login_response():
    try:
        con = sqlite3.connect('userdata.db')
        c = con.cursor()
        for row in c.execute("Select * from record"):
            username = row[1]
            pwd = row[5]

    except Exception as ep:
        messagebox.showerror('', ep)

    uname = email_tf.get()
    upwd = pwd_tf.get()
    check_counter = 0
    if uname == "":
        warn = "Username can't be empty"
    else:
        check_counter += 1
    if upwd == "":
        warn = "Password can't be empty"
    else:
        check_counter += 1
    if check_counter == 2:
        if (uname == username and upwd == pwd):
            messagebox.showinfo('Login Status', 'Logged in Successfully!')

        else:
            messagebox.showerror('Login Status', 'invalid username or password')
    else:
        messagebox.showerror('', warn)


var = StringVar()
var.set('male')

countries = []
variable = StringVar()
world = open('countries.txt', 'r')
for country in world:
    country = country.rstrip('\n')
    countries.append(country)
variable.set(countries[22])

# widgets
left_frame = Frame(
    ws,
    bd=2,
    bg='#CCCCCC',
    relief=SOLID,
    padx=10,
    pady=10
)

Label(
    left_frame,
    text="Enter Email",
    bg='#CCCCCC',
    font=f).grid(row=0, column=0, sticky=W, pady=10)
```

```
Label(
    left_frame,
    text="Enter Password",
    bg='#CCCCCC',
    font=f
).grid(row=1, column=0, pady=10)

email_tf = Entry(
    left_frame,
    font=f
)
pwd_tf = Entry(
    left_frame,
    font=f,
    show='*'
)
login_btn = Button(
    left_frame,
    width=15,
    text='Login',
    font=f,
    relief=SOLID,
    cursor='hand2',
    command=login_response
)

right_frame = Frame(
    ws,
    bd=2,
    bg='#CCCCCC',
    relief=SOLID,
    padx=10,
    pady=10
)

Label(
    right_frame,
    text="Enter Name",
    bg='#CCCCCC',
    font=f
).grid(row=0, column=0, sticky=W, pady=10)

Label(
    right_frame,
    text="Enter Email",
    bg='#CCCCCC',
    font=f
).grid(row=1, column=0, sticky=W, pady=10)

Label(
    right_frame,
    text="Contact Number",
    bg='#CCCCCC',
    font=f
).grid(row=2, column=0, sticky=W, pady=10)

Label(
    right_frame,
    text="Select Gender",
    bg='#CCCCCC',
    font=f
).grid(row=3, column=0, sticky=W, pady=10)
```

```
Label(
    right_frame,
    text="Select  Country",
    bg='#CCCCCC',
    font=f
).grid(row=4, column=0, sticky=W, pady=10)

Label(
    right_frame,
    text="Enter  Password",
    bg='#CCCCCC',
    font=f
).grid(row=5, column=0, sticky=W, pady=10)

Label(
    right_frame,
    text="Re-Enter  Password",
    bg='#CCCCCC',
    font=f
).grid(row=6, column=0, sticky=W, pady=10)

gender_frame = LabelFrame(
    right_frame,
    bg='#CCCCCC',
    padx=10,
    pady=10,
)

register_name = Entry(
    right_frame,
    font=f
)

register_email = Entry(
    right_frame,
    font=f
)

register_mobile = Entry(
    right_frame,
    font=f
)

male_rb = Radiobutton(
    gender_frame,
    text='Male',
    bg='#CCCCCC',
    variable=var,
    value='male',
    font=('Times', 10),

)

female_rb = Radiobutton(
    gender_frame,
    text='Female',
    bg='#CCCCCC',
    variable=var,
    value='female',
    font=('Times', 10),

)

others_rb = Radiobutton(
```

```
    gender_frame,
    text='Others',
    bg='#CCCCCC',
    variable=var,
    value='others',
    font=('Times', 10)

)

register_country = OptionMenu(
    right_frame,
    variable,
    *countries)

register_country.config(
    width=15,
    font=('Times', 12)
)
register_pwd = Entry(
    right_frame,
    font=f,
    show='*'
)
pwd_again = Entry(
    right_frame,
    font=f,
    show='*'
)

register_btn = Button(
    right_frame,
    width=15,
    text='Register',
    font=f,
    relief=SOLID,
    cursor='hand2',
    command=insert_record
)


email_tf.grid(row=0, column=1, pady=10, padx=20)
pwd_tf.grid(row=1, column=1, pady=10, padx=20)
login_btn.grid(row=2, column=1, pady=10, padx=20)
left_frame.place(x=50, y=50)

register_name.grid(row=0, column=1, pady=10, padx=20)
register_email.grid(row=1, column=1, pady=10, padx=20)
register_mobile.grid(row=2, column=1, pady=10, padx=20)
register_country.grid(row=4, column=1, pady=10, padx=20)
register_pwd.grid(row=5, column=1, pady=10, padx=20)
pwd_again.grid(row=6, column=1, pady=10, padx=20)
register_btn.grid(row=7, column=1, pady=10, padx=20)
right_frame.place(x=500, y=50)

gender_frame.grid(row=3, column=1, pady=10, padx=20)
male_rb.pack(expand=True, side=LEFT)
female_rb.pack(expand=True, side=LEFT)
others_rb.pack(expand=True,  side=LEFT)

ws.mainloop()
```





	name	email	contact	gender	country	password
1	Sachin	sachin123@gmail.com	8652195077	male	India	sachin123