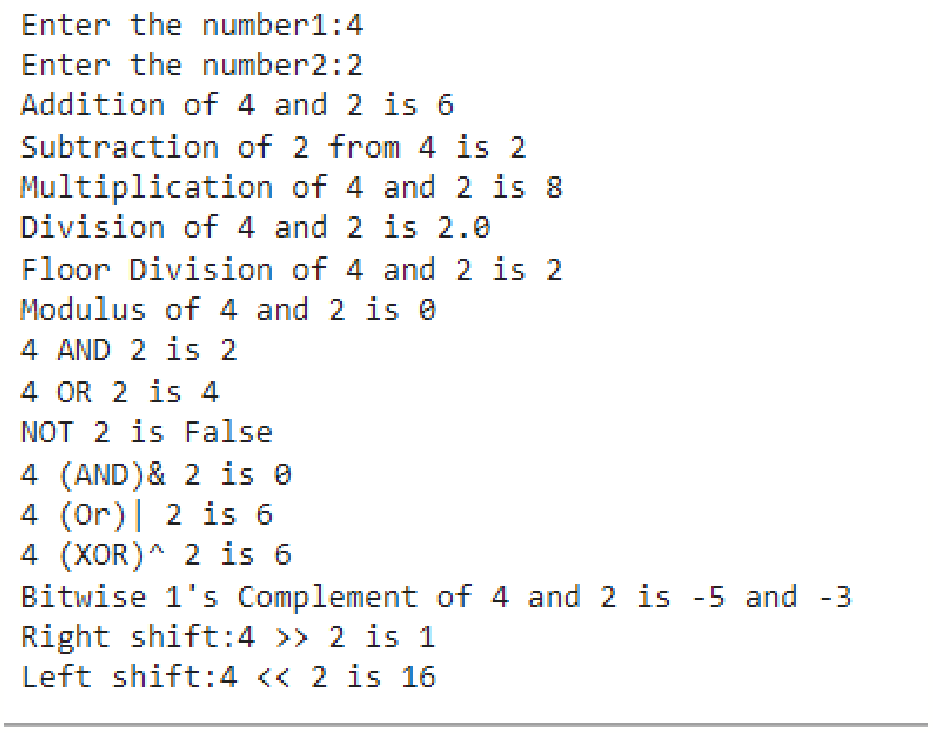
1.A. Arithmetic operators, Logical operator and Bitwise operator

```python
#getting input from end-user
a=int(input("Enter the number1:"))
b=int(input("Enter the number2:"))
#Arithmetic operator
#Addition
ad=a+b
print("Addition of {} and {} is {}".format(a,b,ad))
#Subtraction
su=a-b
print("Subtraction of {} from {} is {}".format(b,a,su))
#Multiplication
mu=a*b
print("Multiplication of {} and {} is {}".format(a,b,mu))
#Division
try:
  di=a/b
  print("Division of {} and {} is {}".format(a,b,di))
except ZeroDivisionError:
  print("Division:Anything divide by zero is undefined")
#Floor Division
try:
  fl=a//b
  print("Floor Division of {} and {} is {}".format(a,b,fl))
except ZeroDivisionError:
  print("Floor Division:Anything divide by zero is undefined")
#Modulus
try:
  mo=a%b
  print("Modulus of {} and {} is {}".format(a,b,mo))
except ZeroDivisionError:
  print("Modulus:Anything divide by zero is undefined")
#Logical operator
print("{} AND {} is {}".format(a,b,a and b))
print("{} OR {} is {}".format(a,b,a or b))
print("NOT {} is {}".format(b,not b))
#Bitwise operator
print("{} (AND)& {} is {}".format(a,b,a & b))
print("{} (Or)| {} is {}".format(a,b,a | b))
print("{} (XOR)^ {} is {}".format(a,b,a ^ b))
print("Bitwise 1's Complement of {} and {} is {} and {}".format(
a,b,~a,~b))
```

```
print("Right shift:{} >> {} is {}".format(a,b,a >> b))
print("Left shift:{} << {} is {}".format(a,b,a << b))
```



```
Enter the number1:4
Enter the number2:2
Addition of 4 and 2 is 6
Subtraction of 2 from 4 is 2
Multiplication of 4 and 2 is 8
Division of 4 and 2 is 2.0
Floor Division of 4 and 2 is 2
Modulus of 4 and 2 is 0
4 AND 2 is 2
4 OR 2 is 4
NOT 2 is False
4 (AND)& 2 is 0
4 (Or)| 2 is 6
4 (XOR)^ 2 is 6
Bitwise 1's Complement of 4 and 2 is -5 and -3
Right shift:4 >> 2 is 1
Left shift:4 << 2 is 16
```

1.B. MYSQL

Steps to create database:

1.Open **Mysql command line client**

2.check the Database already available or not by using this below command

mysql> show database; Then click enter

3.create new database by using this below command

mysql> create database database_name; Then click enter

4.To use the database,

mysql> use database_name; Then click enter

5.If you want to delete existing database,use this command

mysql> drop database database_name; Then click enter

```
import sqlite3 #This module is used in Google Colab
#import mysql.connector as mysql in jupyter notebook
import sys
#creating connector object in colab
sqlcon=sqlite3.connect("Student.db")
#connector_obj=mysql.connector.connect(host="localhost",user="ro
ot",passwd="password",database="database_name")
#creating cursor object
curobj=sqlcon.cursor()
#creating option
while (True):
  print("1. Create table")
  print("2. Insert rows")
  print("3. Delete rows")
  print("4. Update rows")
  print("5. Display rows")
  print("6. Describe Table")
  print("7. Exit")
  opt=int(input("enter your choice:"))
  if (opt==1):
    curobj.execute("CREATE TABLE IF NOT EXISTS stud (rollno inte
ger,name VARCHAR(30), m1 integer, m2 integer, total integer)")
    print("Table created \n")
  elif (opt==2):
    v1=int(input("Enter the student number:"))
    v2=input("Enter the student name:")
    v3=int(input("Enter the mark1:"))
    v4=int(input("Enter the mark2:"))
    v5=v3+v4
    sql = "INSERT INTO stud (rollno,name,m1,m2,total)

VALUES ('{}','{}','{}','{}','{}')".format(v1,v2,v3,v4,v5)
    #val = (v1,v2,v3,v4,v5)
    curobj.execute(sql)
    sqlcon.commit()
    print(curobj.rowcount, "record inserted \n.")
  elif (opt==3):
    v=int(input("Enter the roll number to be deleted:"))
    sql = "DELETE FROM stud WHERE rollno='{}';".format(v)
    #val=(v,)
    curobj.execute(sql)
    sqlcon.commit()
    print(curobj.rowcount, "record(s) deleted \n")
  elif (opt==4):
    v=int(input("Enter the student number:"))
    v1=int(input("Enter the new mark1:"))
    v2=int(input("Enter the new mark2:"))
    v3=v1+v2
    sql ="UPDATE stud SET m1='{}',m2='{}',total ='{}'
          WHERE rollno='{}'".format(v1,v2,v3,v)
    #val=[v1,v2,v3,v]
    curobj.execute(sql)
    sqlcon.commit()
    print(curobj.rowcount, "record(s) affected \n")
  elif (opt==5):
    curobj.execute("SELECT * FROM stud")
    print(curobj.fetchall())
  elif (opt==6):
    curobj.execute("DESCRIBE stud")
    myresult = curobj.fetchall()
    for x in myresult:
      print(x)
  elif (opt==7):
    sys.exit("Program terminated successfully")
```

2.A. Perfect number from 1-n using function with arguments and no return type

```
#creating function for perfect number
def perfect_no(n):
  i=1
  while (i<n+1):
    sum=0
    for j in range(1,i):
      if (i%j==0):
        sum+=j
    if (sum==i):
      print("{} is a perfect number".format(i))
    else:
      print("{} is not a perfect number".format(i))
    i+=1
#getting input from end-user to print perfect number from 1-n
n=int(input("Enter the number:"))
perfect_no(n)
```

Output:

```
Enter the number:9
1 is not a perfect number
```

```
2 is not a perfect number
3 is not a perfect number
4 is not a perfect number
5 is not a perfect number
6 is a perfect number
7 is not a perfect number
8 is not a perfect number
9 is not a perfect number
```

2.B Refer 1.B 1.B. MYSQL

3.A. To Print the nature of roots of a quadratic equation using function with arguments and return type

```
import math

#creating function
def quad_equation(a, b, c):
  d = b**2-4*a*c
  if d > 0:
    return "Determinant d =",d,"Roots are real and unequal "
    r1=(-b + math.sqrt(d))/(2 * a)
    r2=(-b - math.sqrt(d))/(2 * a)
    return "The Roots are {} and {}".format(r1,r2)
  elif d == 0:
    return "Roots are real and equal"
    return -b / (2*a)
  else: # d<0
    return 'Determinant d =",d,"Roots are imaginary'
a=int(input("Enter the Value of a:"))
b=int(input("Enter the Value of b:"))
c=int(input("Enter the Value of c:"))
quad_equation(a,b,c)
```

Output:

```
Enter the Value of a:5
Enter the Value of b:6
Enter the Value of c:4
'
Determinant d =",d,"Roots are imaginary
```

3.B.

	Age	Section	City	Gender	Favourite_Color
0	10	A	Gurgaon	M	Red
1	22	B	Delhi	F	NaN
2	13	C	Mumbai	F	Yellow
3	21	B	Delhi	M	NaN
4	12	B	Mumbai	M	Black
5	11	A	Delhi	M	Green
6	17	A	Mumbai	F	Red

a. Create the above data frame

```
import pandas as pd
import numpy as np
df=pd.DataFrame({'age':[10,22,13,21,12,11,17],'section':['A
','B','C','B','B','A','A'],'city':['Gurgaon','Delhi','Mumba
i','Delhi','Mumbai','Delhi','Mumbai'],'gender':['M','F','F'
,'M','M','M','F'],'favcolor':['RED',np.nan,'YELLOW',np.nan,
'BLACK','GREEN','RED']})
display(df)
```

Output:

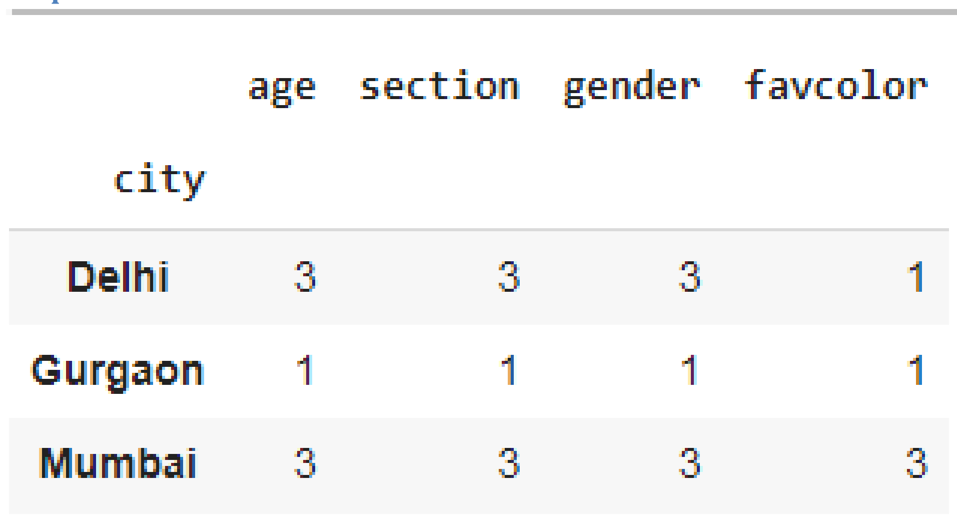
```
   age  section     city  gender  favcolor
0   10        A  Gurgaon       M       RED
1   22        B    Delhi       F       NaN
2   13        C   Mumbai       F    YELLOW
3   21        B    Delhi       M       NaN
4   12        B   Mumbai       M     BLACK
5   11        A    Delhi       M     GREEN
6   17        A   Mumbai       F       RED
```

b. Use Pandas groupby according to 'city'

```
    df.groupby(['city'])[['age','section','gender','favcolor']]
.count()
```

```
     #dataframe_object.groupby([`coloumn_name’][[column_name
list   separated by comma]].count()
```

Output:

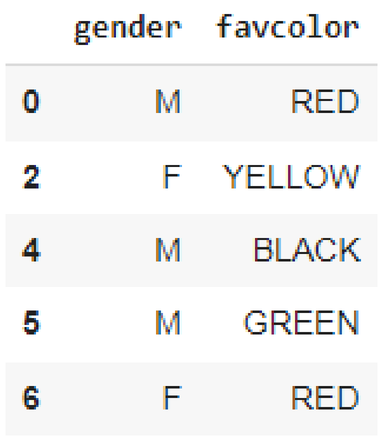


	age	section	gender	favcolor
city				
Delhi	3	3	3	1
Gurgaon	1	1	1	1
Mumbai	3	3	3	3

c.Display the gender and favorite color of the person whose age is less than 20.

```
df.loc[df.age<20, ['gender','favcolor']]
```

Output:



	gender	favcolor
0	M	RED
2	F	YELLOW
4	M	BLACK
5	M	GREEN
6	F	RED

d.List the city names ends with ‘I’

```
df[df.city.str.endswith('i')]
```

Output:

	age	section	city	gender	favcolor
1	22	B	Delhi	F	NaN
2	13	C	Mumbai	F	YELLOW
3	21	B	Delhi	M	NaN
4	12	B	Mumbai	M	BLACK
5	11	A	Delhi	M	GREEN
6	17	A	Mumbai	F	RED

e.Show the number of null values across each column

```
    df.isnull().sum()
```

Output:

```
age         0
section     0
city        0
gender      0
favcolor    2
dtype: int64
```

f.Fill the null values with orange

```
    df.fillna('orange')
```

note for 17.A.f: replace *'orange'* any other word given in exam

Output:

	age	section	city	gender	favcolor
0	10	A	Gurgaon	M	RED
1	22	B	Delhi	F	orange
2	13	C	Mumbai	F	YELLOW
3	21	B	Delhi	M	orange
4	12	B	Mumbai	M	BLACK
5	11	A	Delhi	M	GREEN
6	17	A	Mumbai	F	RED

4.A. To calculate the area of square,rectangle and triangle

```
def option():

  opt=input("Do u want to continue the menu selection(y/n): ")
  if opt.lower()=='y':
    area()
  else:
    print("Menu has exit")
def area():
    print("1. Area of Square")
    print("2. Area of Rectangle")
    print("3. Area of Triangle")
    ch=int(input("Enter your choice:1,2,3: "))
    if (ch==1):
      a=float(input("Enter the side of a square: "))
      area=a**2
      print("The area of square with side {} is {}".format(a,are
a))
      option()
    elif (ch==2):
      b=float(input("Enter the breadth of a rectangle: "))
      h=float(input("Enter the height of a rectangle: "))
      area=0.5*b*h
      print("The area of rectangle with breadth {} and
                     height {} is {}".format(b,h,area))
      option()
    elif (ch==3):
      l=float(input("Enter the length of a triangle:"))
      b=float(input("Enter the breadth of a triangle:"))
      area=l*b
      print("The area of triangle with breadth {} and
                    height {} is {}".format(l,b,area))
      option()
    else:
      print("Kindly enter values as 1,2 or 3")
      option()
area()
```

Output:

```
1. Area of Square
2. Area of Rectangle
3. Area of Triangle
Enter your choice:1,2,3: 1
Enter the side of a square: 2
```

```
The area of square with side 2.0 is 4.0
Do u want to continue the menu selection(y/n): y
1. Area of Square
2. Area of Rectangle
3. Area of Triangle
Enter your choice:1,2,3: 3
Enter the length of a triangle:2
Enter the breadth of a triangle:4
The area of triangle with breadth 2.0 and height 4.0 is 8.0
Do u want to continue the menu selection(y/n): n
Menu has exit
```

4.B. Hybrid Inheritance

```
#Hybrid Inheritance

class base1:
  def get1(self):
    self.snum=int(input("Enter the student number: "))
    self.sname=input("Enter the student name: ")
  def put1(self):
    print('Student Number: ',self.snum)
    print('Student Name: ',self.sname)
class base2(base1):
  def get2(self):
    self.mark1=int(input("Enter the Mark1: "))
    self.mark2=int(input("Enter the Mark2: "))
  def put2(self):
    print('Mark 1: ',self.mark1)
    print('Mark 2: ',self.mark2)
class base3:
  def get3(self):
    self.score=int(input("Enter the sports score for 10 : "))
  def put3(self):
    print('Score: ',self.score)
class child(base2,base3):
  def put(self):
    base1.get1(self)
    base1.put1(self)
    base2.get2(self)
    base2.put2(self)
    base3.get3(self)
    base3.put3(self)
    print("-"*50)
    print("Student number: ",self.snum)
    print("Student name: ",self.sname)
    print("Student Mark1: ",self.mark1)
    print("Student Mark2: ",self.mark2)
    print("Student sports score (10): ",self.score)
    self.total=self.mark1+self.mark2+self.score
    print("Student total marks (100): ",self.total)
    print("-"*50)
m=child()
m.put()
```

Output:

```
Enter the student number: 213152
Enter the student name: Paul
Student Number:  213152
Student Name:  Paul
Enter the Mark1: 50
Enter the Mark2: 60
Mark 1:  50
Mark 2:  60
Enter the sports score for 10 : 9
Score:  9
--------------------------------------------------
Student number:  213152
Student name:  Paul
Student Mark1:  50
Student Mark2:  60
Student sports score (10):  9
Student total marks (100):  119
```

5.A. To reverse the given number and find the sum of its digits using function without argument and return type

```
     def rev_and_sum_number():

        num = int(input("Enter the number: "))
        sum = 0
          print("The reverse number is:")
          while num > 0:
             rev = num % 10
             print(rev, end="")
             sum = sum + rev
             num = num // 10
          print()
          print("Sum of the digits:", sum)
       rev_and_sum_number()
```

Output:

```
Enter the number: 123
```

```
The reverse number is:
321
Sum of the digits: 6
```

5.B. File handling

```
#File Handling

f=open("Employee.dat","a")
for i in range(2):
 empnum=input("Enter Employee Number:")
 empname=input("Enter the Employee Name:")
 row=empnum+"-"+empname+"\n"
 f.write(row)
f.close()
#number of lines in a file
f=open("Employee.dat","r")
str=f.readlines()
print("File content Using readlines():\n",str)
f.close()
#readline() function
f=open("Employee.dat","r")
row=f.readline()
print("File contents using readline()")
while row:
 print(row)
 row=f.readline()
f.close()
#read() function
f=open("Employee.dat","r")
print("File contents using read()")
str1=f.read()
print(str1)
f.close()
```

Output:

```
Enter Employee Number:11477
Enter the Employee Name:Paul
Enter Employee Number:1148
Enter the Employee Name:Bala
File content Using readlines():
 ['11477-Paul\n', '1148-Bala\n']
File contents using readline()
11477-Paul

1148-Bala
```

```
File contents using read()
11477-Paul
1148-Bala
```

6.A. To print the number as prime or not from 1-n using function raise exception when n is not number

```python
try:
 n = int(input("Enter the nth number: "))
except ValueError:
 print("Please enter only number")
def prime_number(n):
 for num in range(1, n + 1):
    flag = True
    for i in range(2, num):
        if num % i == 0:
            print(num, " is not a prime number")
            flag = False
            break
    if flag:
       print(num, " is a prime number")
prime_number(n)
```

Output:

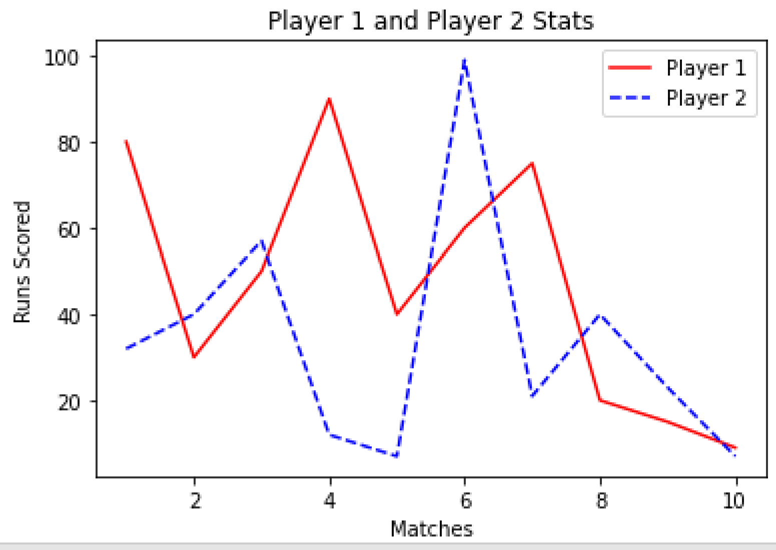
```
Enter the nth number: 11
1  is a prime number
2  is a prime number
3  is a prime number
4  is not a prime number
5  is a prime number
6  is not a prime number
7  is a prime number
8  is not a prime number
9  is not a prime number
10  is not a prime number
11  is a prime number
```

6.B. Line Graph

```python
import matplotlib.pyplot as plt
player1_runs=[80,30,50,90,40,60,75,20,15,9]
player2_runs=[32,40,57,12,7,99,21,40,23,7]
match=[1,2,3,4,5,6,7,8,9,10]
plt.xlabel('Matches')
plt.ylabel('Runs Scored')
plt.title('Player 1 and Player 2 Stats')
plt.plot(match,player1_runs,'r',label='Player 1', linestyle='sol
id')
plt.plot(match,player2_runs,'b',label='Player 2', linestyle='das
hed')
plt.legend()
```

Output:

```
<matplotlib.legend.Legend at 0x7fcd1e61cd90>
```



7.A. List Manipulation

```
#list1 - number divisible by 3
list1=[]
#list2 - number not divisible by 3
list2=[]
n=int(input('How many numbers going to enter?:'))
i=0
while(i<n):
 i=i+1
 a=int(input('Enter the Number: '))
 if(a%3==0):
    list1.append(a)
    list1.sort()
 else:
   list2.append(a)
   list2.sort()
#displaying the two list
print("Displaying the two sorted list")
print("-"*50)
print('List 1 - Numbers Divisbile By 3:',list1)
print('List 2 - Numbers Not Divisible By 3:',list2)
print("-"*50)
```

```python
#slicing the list
print("Slicing the sorted list")
print("-"*50)
print("list1[:3]:",list1[:3])
print("list2[4:]:",list2[4:])
print("list1[1:4]:",list1[1:4])
print("list2[:3]:",list2[1:4])
print("-"*50)
#accessing individual elements
print("Accessing individual element in list1")
for i in list1:
  print(i)
print("Accessing individual element in list2")
for i in list2:
  print(i)
#deleting the elements in list
list1.clear()
list2.clear()
print("After deleting the elements in list")
print("List1:",list1)
print("List2:",list2)
```

Output:

```
How many numbers going to enter?:10
Enter the Number: 1
Enter the Number: 2
Enter the Number: 3
Enter the Number: 4
Enter the Number: 5
Enter the Number: 6
Enter the Number: 7
Enter the Number: 8
Enter the Number: 9
Enter the Number: 10
Displaying the two sorted list
--------------------------------------------------
List 1 - Numbers Divisbile By 3: [3, 6, 9]
List 2 - Numbers Not Divisible By 3: [1, 2, 4, 5, 7, 8, 10]
--------------------------------------------------
Slicing the sorted list
--------------------------------------------------
list1[:3]: [3, 6, 9]
list2[4:]: [7, 8, 10]
list1[1:4]: [6, 9]
list2[:3]: [2, 4, 5]
--------------------------------------------------
Accessing individual element in list1
```

```
3
6
9
Accessing individual element in list2
1
2
4
5
7
8
10
After deleting the elements in list
List1: []
List2: []
```

7.B. Arithmetic operator overloading to subtract two object and logical operator overloading

```
Class point:

     def __init__(self,a,b):

         self.x=a

         self.y=b

      def __str__(self):

          t=(self.x,self.y)

          return str(t)

      def __sub__(self,other):

          a=self.x-other.x

          b=self.y-other.y

          return point(a,b)

P1=point(5,6)

P2=point(4,3)

print("The subtraction of two objects=",P1-P2)

class logical:

           def __init__(self,a):
               self.x=a
           def __str__(self):
               t=self.x
               return str(t)
            def __and__(self,other):
                b=self.x & other.x
                return b
            def __or__(self,other):
                b=self.x | other.x
                return b
L1=logical(0)
L2=logical(5)
print("AND Operation=",L1 & L2)
print("OR Operation=",L1 | L2)
```

Output:

```
The subtraction of two objects= (1, 3)
AND Operation= 0
OR Operation= 5
```

8.A.Tuple

```
#creating tuple for color
c=('Black','Blue','Red','White','Orange')
print(c)
print("-"*50)
#Tuple is an immutable data structure
try:
  c[0]='Grey'#Tuple does not support item assignment
except:
  print("Tuple does not support item assignment")
print("-"*50)
#Tuple Created using Slicing method
c1=c[:3]
print("New tuple using slicing method:",c1)
print("-"*50)
#accessing individual elements in tuple
print("Elements in color tuple:")
for i in c:
  print(i)
print("-"*50)
#converting tuple to list
c=list(c)
print("Type of Color tuple:",type(c))
#adding new elements
c.append('Yellow')
#displaying updated tuple
c=tuple(c)
print("Updated Tuple:", c)
print("-"*50)
```

Output:

```
('Black', 'Blue', 'Red', 'White', 'Orange')
--------------------------------------------------
Tuple does not support item assignment
--------------------------------------------------
New tuple using slicing method: ('Black', 'Blue', 'Red')
--------------------------------------------------
Elements in color tuple:
Black
Blue
Red
White
Orange
--------------------------------------------------
Type of Color tuple: <class 'list'>
Updated Tuple: ('Black', 'Blue', 'Red', 'White', 'Orange',
'Yellow')
```

8.B. Solving equation using scipy

```
import numpy as np
from scipy import linalg
#x+y+z=2
#6x-4y+5z=31
#5x+2y+2z=13
a = np.array([[1, 1, 1], [6, -4, 5], [5, 2, 2]])
b = np.array([[2], [31], [3]])
x = linalg.solve(a, b)
print(x)
A=np.array([[4,-3,0],[2,-1,-2],[1,5,7]])
x=linalg.det(A)
```

```
print("The determinant of \n {} is {}".format(A,x))
```

Output:

```
[[-0.33333333]
 [-2.37037037]
 [ 4.7037037 ]]
The determinant of
 [[ 4 -3  0]
 [ 2 -1 -2]
 [ 1  5  7]] is 60.0
```

9.A. Dictionary

```
players=['Rahul','Dowlath','Bharath','Danush','Bhagavathi']
scores=[42,105,90,88,78]
f=dict(zip(players,scores))
print("Player Dictionary:",f)
print("-"*50)
#get all keys
print("Values in dictionary:",f.keys())
print("-"*50)
#get all values
print("Values in dictionary:",f.values())
print("-"*50)
#get all key-value pair
print("Key-Value pair in dictionary:",f.items())
print("-"*50)
print('Key',':','Value')
#iterate
for key in f:
 print(key,':',f[key])
print("-"*50)
print('Key',':','Value')
#iterate
for key in f.items():
 print(key[0],key[1])
print("-"*50)
#length of the dictionary
print('Length of the Dictionary:',len(f))
print("-"*50)
#adding a key
f.update({'Dhina':150})
print("After updating new key Dhina:",f)
print("-"*50)
#update Values
f['Rahul']=55
print("After updating values in Rahul:",f)
print("-"*50)
#search
search='Rahul'
if search in f.keys():
 print('Runs Scored By',search,'is:',f[search])
 print("-"*50)
else:
 print('Key not Found')
 print("-"*50)
```

Output:

```
Player Dictionary: {'Rahul': 42, 'Dowlath': 105, 'Bharath': 90, 'Danush': 88, 'Bhagavathi': 78}
--------------------------------------------------
Values in dictionary: dict_keys(['Rahul', 'Dowlath', 'Bharath', 'Danush', 'Bhagavathi'])
--------------------------------------------------
Values in dictionary: dict_values([42, 105, 90, 88, 78])
--------------------------------------------------
Key-Value pair in dictionary: dict_items([('Rahul', 42), ('Dowlath', 105), ('Bharath', 90),
('Danush', 88), ('Bhagavathi', 78)])
--------------------------------------------------
Key : Value
Rahul : 42
Dowlath : 105
Bharath : 90
Danush : 88
Bhagavathi : 78
--------------------------------------------------
Key : Value
Rahul 42
Dowlath 105
Bharath 90
Danush 88
Bhagavathi 78
--------------------------------------------------
Length of the Dictionary: 5
--------------------------------------------------
After updating new key Dhina: {'Rahul': 42, 'Dowlath': 105, 'Bharath': 90, 'Danush': 88,
'Bhagavathi': 78, 'Dhina': 150}
--------------------------------------------------
After updating values in Rahul: {'Rahul': 55, 'Dowlath': 105, 'Bharath': 90, 'Danush': 88,
'Bhagavathi': 78, 'Dhina': 150}
--------------------------------------------------
Runs Scored By Rahul is: 55
--------------------------------------------------
```

9.B. Multiple Inheritance

```
class base1:
  def get1(self):
    self.num1=int(input('enter the first number:'))
  def put1(self):
    print('num1=',self.num1)
class base2:
  def get2(self):
    self.num2=int(input('enter the second number:'))
  def put2(self):
      print('num2=',self.num2)
class child(base1,base2):
  def put(self):
    base1.get1(self)
    base2.get2(self)
    self.add=self.num1+self.num2
    self.sub=self.num1-self.num2
    self.mul=self.num1*self.num2
    self.div=self.num1/self.num2
    self.biggest=(self.num1 > self.num2) or (self.num2 > self.nu
m1)
    print('num1=',self.num1)
    print('num2=',self.num2)
    print('Addition of {} and {} = {}'.format(self.num1,self.num
2,self.add))
    print("Subtraction of {} and {} = {}".format(self.num1,self.
num2,self.sub))
    print('multiplication of {} and {} = {}'.format(self.num1,se
lf.num2,self.mul))
    print('division of {} and {} = {}'.format(self.num1,self.num
2,self.div))
    if (self.num1>self.num2):
        print('The biggest number between {} and {} is {}'.forma
t(self.num1,self.num2,self.num1))
    else:
        print('The biggest between {} and {} is {}'.format(self.
num1,self.num2,self.num2))
c=child()
c.put()
c.put1()
c.put2()
```

Output:

```
enter the first number:20
enter the second number:12
num1= 20
num2= 12
```

```
Addition of 20 and 12 = 32
Subtraction of 20 and 12 = 8
multiplication of 20 and 12 = 240
division of 20 and 12 = 1.6666666666666667
The biggest number between 20 and 12 is 20
num1= 20
num2= 12
```

10.A. User defined module

1. Code this below #Module code in separate Notebook
2. Downoad it as ***.py***
3. Upload your module_name.py file in your Notebook
4. open the new Notebook
5. import module_name
6. code the #usage of Module

```
# Module
def my_func(n):
 l=[]
 i=0
 while i<n:
    a=int(input('Enter the Number:'))
    l.append(a)
    i=i+1
 MAXIMUM=max(l)
 MINIMUM=min(l)
 s = 0
 s = sum(l)
 r= 0
 r= len(l)
 AVERAGE=s/r
 print("Maximum:",MAXIMUM)
 print("Minimum:",MINIMUM)
 print("Average:",AVERAGE)
```

```
#Usage Of Module

import module_name

n=int(input("Enter the value of N:"))

 module.my_func(n)
```

print(dir(module))

10.B. Creating matrix using numpy

```
import numpy as np
x= np.arange(2, 11).reshape(3,3)
print("x=",x)
y = np.arange(1,10).reshape(3,3)
print("y=",y)
z=np.add(x,y)
a=np.dot(x,y)
print("-"*50)
print('Addition of matrix:\n',z)
print("-"*50)
print('Multiplication of matrix:\n',a)
print("-"*50)
print("Displaying the results in C-order")
print("Addition of x and y:\n")
for i in np.nditer(z,order='C'):
    print(i)
print("-"*50)
print("Multiplication of x and y:\n")
for i in np.nditer(a,order='C'):
    print(i)
print("-"*50)
print("Displaying the results in F-order")
print("Addition of x and y:\n")
for i in np.nditer(z,order='F'):
    print(i)
print("-"*50)
print("Multiplication of x and y:\n")
for i in np.nditer(a,order='F'):
    print(i)
print("-"*50)
```

Output:

```
x= [[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
y= [[1 2 3]
 [4 5 6]
 [7 8 9]]
--------------------------------------------------
Addition of matrix:
 [[ 3  5  7]
 [ 9 11 13]
 [15 17 19]]
```

```
---------------------------------------------------
Multiplication of matrix:
 [[ 42  51  60]
 [ 78  96 114]
 [114 141 168]]
---------------------------------------------------
Displaying the results in C-order
Addition of x and y:

3
5
7
9
11
13
15
17
19
---------------------------------------------------
Multiplication of x and y:

42
51
60
78
96
114
114
141
168
---------------------------------------------------
Displaying the results in F-order
Addition of x and y:

3
9
15
5
11
17
7
13
19
---------------------------------------------------
Multiplication of x and y:

42
```

```
78
114
51
96
141
60
114
168
--------------------------------------------------
```

11.A. Working of class

```
#Class and Objects
class Sample:
#Class variable classvar
#Object variables rollno,name,age
 classvar=0
 def __init__(self):
    Sample.classvar+=1
    self.rollno=Sample.classvar
    self.name=input("Enter the name:")
    #private variable
    self.__age=int(input("Enter the age:"))
 def put(self):
    print("Object Variables: Roll Number={} Name={} Age={}".form
at(self.rollno,self.name,self.__age))
 def __del__(self):
    del self.rollno
    del self.name
    del self.__age
    del Sample.classvar
    print()
obj1=Sample()
obj1.put()
obj2=Sample()
obj2.put()
#Change name and age. Only the name will be changed but not age
print("After changing, the values are")
obj2.name="Swathi"
obj2.age=25
obj2.put()
```

Output:

```
Enter the name:Paul
Enter the age:24
Object Variables: Roll Number=1 Name=Paul Age=24
Enter the name:Alan
Enter the age:25
Object Variables: Roll Number=2 Name=Alan Age=25
After changing, the values are
Object Variables: Roll Number=2 Name=Swathi Age=25
```

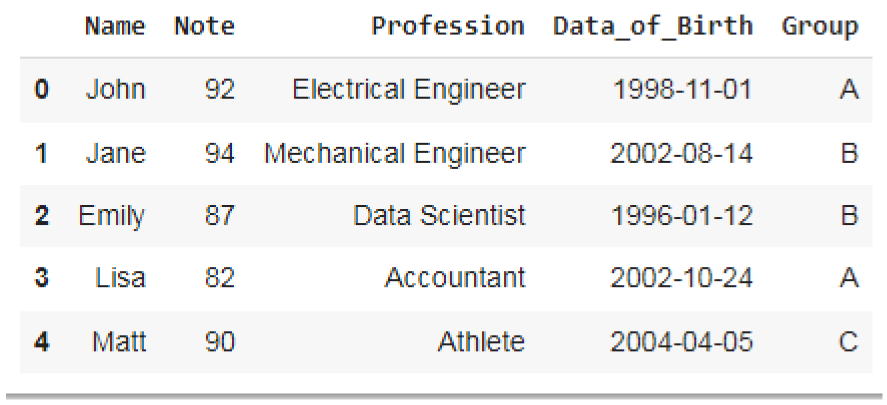
11.B.

	Name	Note	Profession	DOB	Group
0	John	92	Electrical Engineering	1998-11-01	A
1	Jane	94	Mechanical Engineering	2002-08-14	B
2	Emily	87	Data Scientist	1996-01-12	B
3	Lisa	82	Accontant	2002-10-24	A
4	Matt	90	Athelete	2004-04-05	C

a.Create the above DataFrame

```
import pandas as pd
data = pd.DataFrame(
                {'Name':['John','Jane','Emily','Lisa','Matt'],
                 'Note':[92,94,87,82,90],
                 'Profession':['Electrical Engineer','Mechanical
Engineer','Data Scientist','Accountant','Athlete'],
                 'Data_of_Birth':['1998-11-01','2002-08-
14','1996-01-12','2002-10-24','2004-04-05'],
                 'Group':['A','B','B','A','C']})
display(data)
```

Output:

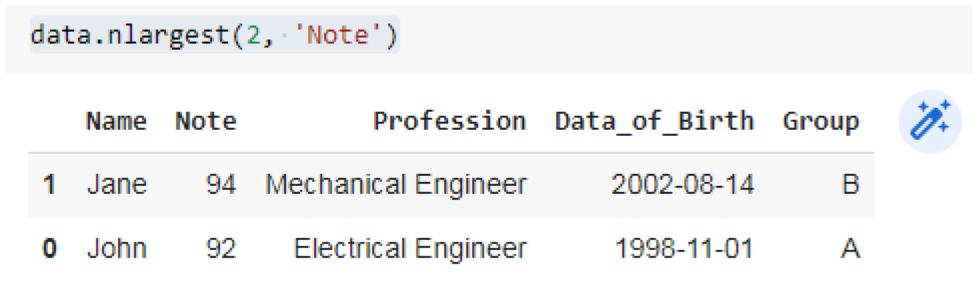


```
    Name  Note          Profession  Data_of_Birth  Group
0   John    92  Electrical Engineer    1998-11-01      A
1   Jane    94  Mechanical Engineer    2002-08-14      B
2  Emily    87       Data Scientist    1996-01-12      B
3   Lisa    82           Accountant    2002-10-24      A
4   Matt    90              Athlete    2004-04-05      C
```

b.List the nlargest and nsmallest based on ‘note’ column(let n=2)

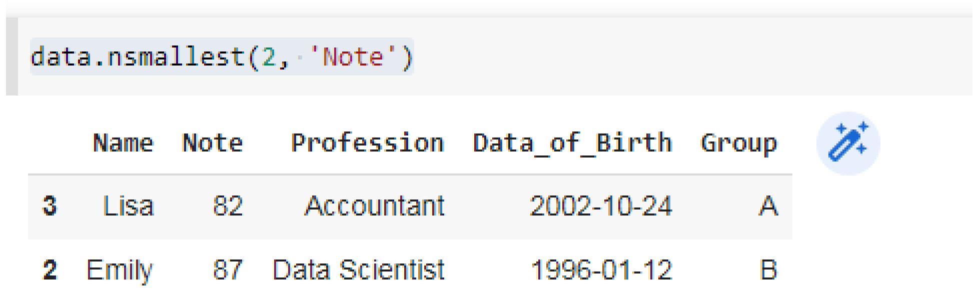
```
data.nlargest(2, 'Note')
data.nsmallest(2, 'Note')
```

Output:



```
data.nlargest(2, 'Note')
```

	Name	Note	Profession	Data_of_Birth	Group
1	Jane	94	Mechanical Engineer	2002-08-14	B
0	John	92	Electrical Engineer	1998-11-01	A



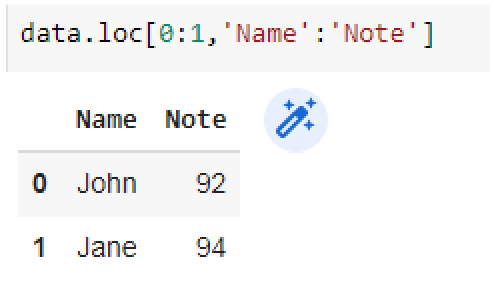
```
data.nsmallest(2, 'Note')
```

	Name	Note	Profession	Data_of_Birth	Group
3	Lisa	82	Accountant	2002-10-24	A
2	Emily	87	Data Scientist	1996-01-12	B

c.List the first 2 row with the name and note columns(use loc)

```
data.loc[0:1,'Name':'Note'] or
```
data.loc[1:2,:'Note']

Output:



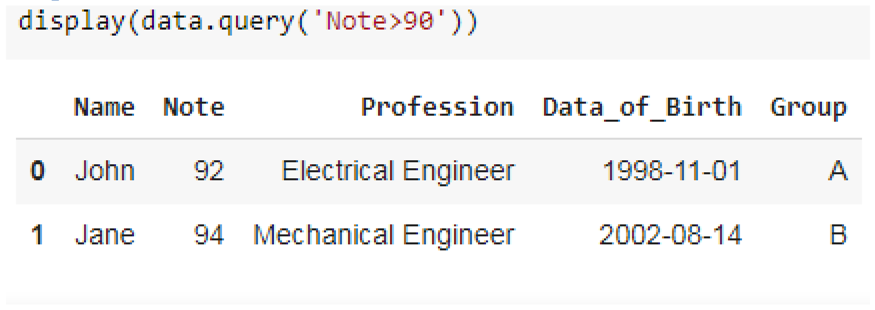
```
data.loc[0:1,'Name':'Note']
```

	Name	Note
0	John	92
1	Jane	94

d.List the rows whose note is greater than 90

```
display(data.query('Note>90'))
```

Output:



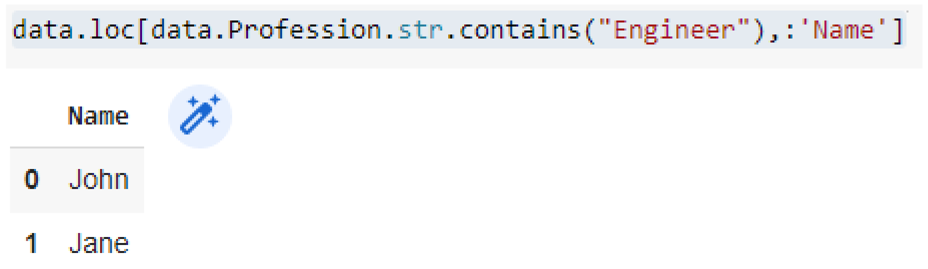
```
display(data.query('Note>90'))
```

	Name	Note	Profession	Data_of_Birth	Group
0	John	92	Electrical Engineer	1998-11-01	A
1	Jane	94	Mechanical Engineer	2002-08-14	B

e.List the name who are engineers

```
data.loc[data.Profession.str.contains("Engineer"),:'Name']
```

Output:



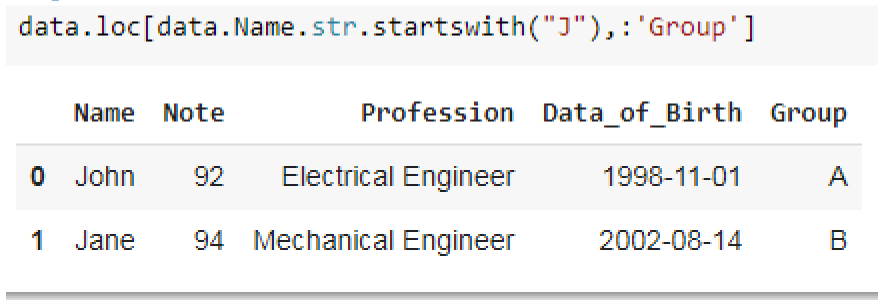
```
data.loc[data.Profession.str.contains("Engineer"),:'Name']
```

	Name
0	John
1	Jane

f.List the details of person whose name starts with 'J'

```
data.loc[data.Name.str.startswith("J"),:'Group']
```

Output:



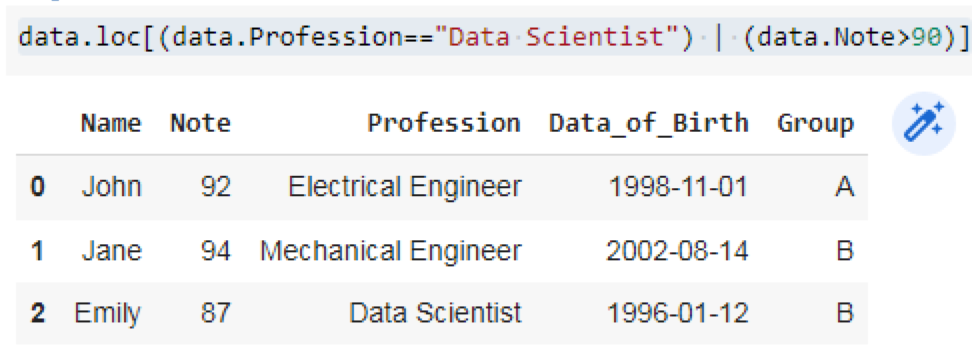
```
data.loc[data.Name.str.startswith("J"),:'Group']
```

	Name	Note	Profession	Data_of_Birth	Group
0	John	92	Electrical Engineer	1998-11-01	A
1	Jane	94	Mechanical Engineer	2002-08-14	B

13.b. List the people who are data scientist or have a note more than 90

```
data.loc[(data.Profession=="Data Scientist") | (data.Note>90)]
```

Output:



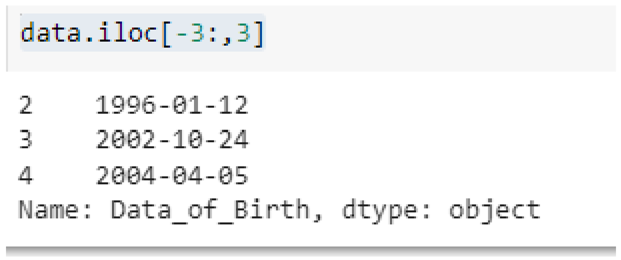
```
data.loc[(data.Profession=="Data Scientist") | (data.Note>90)]
```

	Name	Note	Profession	Data_of_Birth	Group
0	John	92	Electrical Engineer	1998-11-01	A
1	Jane	94	Mechanical Engineer	2002-08-14	B
2	Emily	87	Data Scientist	1996-01-12	B

13.c.List the last 3 row and the third column(use iloc)

```
data.iloc[-3:,3]
```

Output:



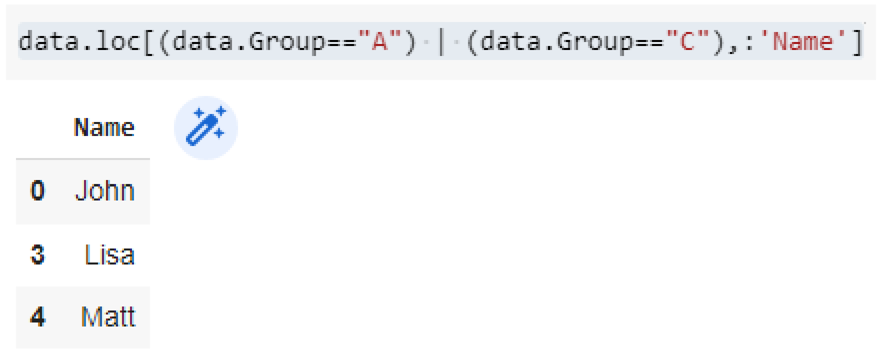
```
data.iloc[-3:,3]
```

```
2     1996-01-12
3     2002-10-24
4     2004-04-05
Name: Data_of_Birth, dtype: object
```

13.d.List the names who are either in group A or C(use isin operator)

```
data.loc[(data.Group=="A") | (data.Group=="C"),:'Name']
```

Output:



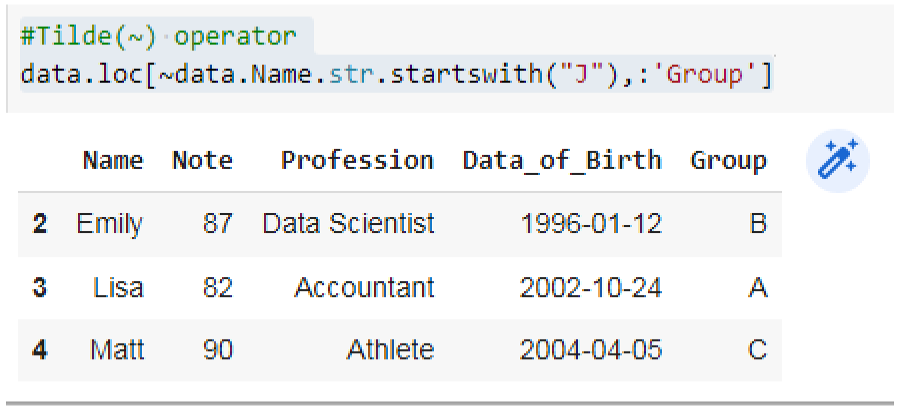
```
data.loc[(data.Group=="A") | (data.Group=="C"),:'Name']
```

	Name
0	John
3	Lisa
4	Matt

13.f. List the details of person whose names are not starts with ‘J’

```
#Tilde(~) operator
data.loc[~data.Name.str.startswith("J"),:'Group']
```

Output:



```
#Tilde(~) operator
data.loc[~data.Name.str.startswith("J"),:'Group']
```

	Name	Note	Profession	Data_of_Birth	Group
2	Emily	87	Data Scientist	1996-01-12	B
3	Lisa	82	Accountant	2002-10-24	A
4	Matt	90	Athlete	2004-04-05	C

12.A.Multilevel Inheritance

```
class empbase:
  def get1(self):
      self.num=int(input('enter the employee number:'))
      self.name=input('enter the employee name:')
```

```
        self.basic=int(input('enter the employee basic pay:'))
    def put1(self):
        print("Employee number: ",self.num)
        print("Employee name: ",self.name)
class empchild1(empbase):
    def get2(self):
        self.allow=int(input('enter the employee allowances:'))
        self.ded=int(input('enter the employee deductions:'))
    def put2(self):
        print("Employee allowances: ",self.allow)
        print("Employee deductions: ",self.ded)
class empchild2(empchild1):
    def get3(self):
        self.gross=self.basic+self.allow+self.ded
        self.net=self.gross-self.basic
    def put3(self):
        print("Employee gross salary: ",self.gross)
        print("Employee net salary: ",self.net)
obj=empchild2()
obj.get1()
obj.get2()
obj.get3()
obj.put1()
obj.put2()
obj.put3()
```

Output:

```
enter the employee number:2131
enter the employee name:Santa
enter the employee basic pay:5000
enter the employee allowances:2000
enter the employee deductions:750
Employee number:  2131
Employee name:  Santa
Employee allowances:  2000
Employee deductions:  750
Employee gross salary:  7750
Employee net salary:  2750
```

12.B.1-D Array

```
import numpy as np
a=np.arange(5)
print("One Dimension array is",a)
print("Size of the array:",len(a))
print("Dimension of array:",a.ndim)
print("POWER(a^2)",pow(a,2))
print("SUM OF ARRAY ELEMENTS OF A=",sum(a))
print("LARGEST OF ARRAY ELEMENTS OF A=",max(a))
print("SMALLEST OF ARRAY ELEMENTS OF A=",min(a))
print("MEAN OF ARRAY ELEMENTS A=",np.mean(a))
print("VARIANCE OF ARRAY ELEMENTS A=",np.var(a))
print("STANDARD DEVIATION OF ARRAY ELEMENTS A=",np.std(a))
print("SORTED ARRAY A=",np.sort(a))
print("UNIQUE ELEMENTS IN ARRAY A=",np.unique(a))
print("PRODUCT OF ARRAY ELEMENTS A=",np.prod(a))
print("LOGARITHMIC VALUE OF ARRAY ELEMENTS A=",np.log(a))
print("SQUARE ROOT OF ARRAY ELEMENTS A=",np.sqrt(a))
```

Output:

```
One Dimension array is [0 1 2 3 4]
Size of the array: 5
Dimension of array: 1
POWER(a^2) [ 0  1  4  9 16]
SUM OF ARRAY ELEMENTS OF A= 10
LARGEST OF ARRAY ELEMENTS OF A= 4
SMALLEST OF ARRAY ELEMENTS OF A= 0
MEAN OF ARRAY ELEMENTS A= 2.0
VARIANCE OF ARRAY ELEMENTS A= 2.0
STANDARD DEVIATION OF ARRAY ELEMENTS A= 1.4142135623730951
SORTED ARRAY A= [0 1 2 3 4]
UNIQUE ELEMENTS IN ARRAY A= [0 1 2 3 4]
PRODUCT OF ARRAY ELEMENTS A= 0
LOGARITHMIC VALUE OF ARRAY ELEMENTS A= [      -inf 0.
0.69314718 1.09861229 1.38629436]
SQUARE ROOT OF ARRAY ELEMENTS A= [0.         1.
1.41421356 1.73205081 2.        ]
```

13.A.Single Inheritance

```
class Empbase:
  def get1(self):
      self.enum=int(input("Enter the Employee Number: "))
      self.ename=input("Enter the Employee Name: ")
      self.basic=int(input("Enter Basic"))
  def put1(self):
      print("Employee Number: ",self.enum)
      print("Employee Name: ",self.ename)
      print("Basic:", self.basic)
class Empchild(Empbase):
  def get2(self):
      self.ded=int(input("Enter Deduction: "))
      self.allow=int(input("Enter Allowance: "))
      self.gross=self.basic+self.ded+self.allow
```

```
        self.net=self.gross-self.basic
    def put2(self):
        print("Deduction: ",self.ded)
        print("Allowance: ",self.allow)
        print("Gross: ",self.gross)
        print("Net: ",self.net)
obj=Empchild()
obj.get1()
obj.get2()
obj.put1()
obj.put2()
```

Output:

```
Enter the Employee Number: 2131
Enter the Employee Name: HODPA
Enter Basic5999
Enter Deduction: 999
Enter Allowance: 1999
Employee Number:  2131
Employee Name:  HODPA
Basic: 5999
Deduction:  999
Allowance:  1999
Gross:  8997
Net:  2998
```

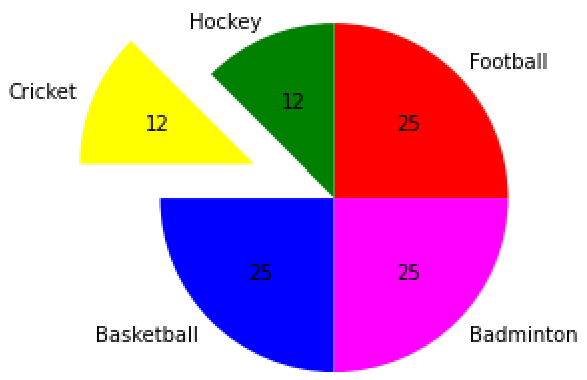
13.B.Refer 11.B 11.B.

14.A. Pie graph

Football	Hockey	Cricket	Basketball	Badminton
10	5	5	10	10

```
import matplotlib.pyplot as plt
score=[10,5,5,10,10]
game=['Football','Hockey','Cricket','Basketball','Badminton']
col=['Red','Green','Yellow','Blue','Magenta']
plt.pie(score,labels=game,colors=col,explode=[0,0,0.5,0,0])
plt.show()
```

Output:



14.B. Refer 1.B [1.B. MYSQL]

15.A.Hierarchical Inheritance

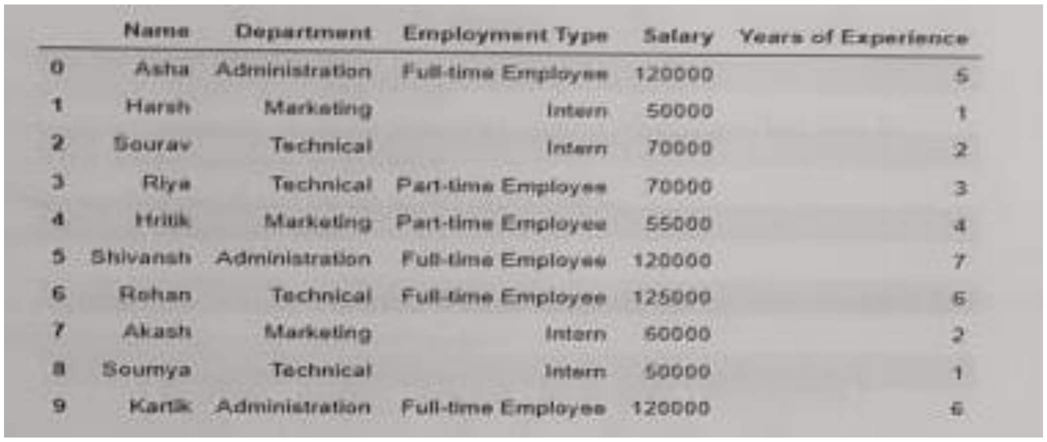
```
class base:
  def get(self):
      self.num1=int(input("Enter the num1:"))
      self.num2=int(input("Enter the num2:"))
  def put(self):
      print("Num1:",self.num1)
      print("Num2:",self.num2)
class child1(base):
  def put1(self):
      base.get(self)
      base.put(self)
      self.add=self.num1+self.num2
      self.sub=self.num1-self.num2
      self.mul=self.num1*self.num2
      self.div=self.num1/self.num2
      self.exp=self.num1**self.num2
      print('Additon:',self.add)
      print('Subtraction:',self.sub)
      print('Division:',self.div)
      print('Multiplication:',self.mul)
      print('Exponentitional:',self.exp)
class child2(base):
  def put2(self):
      base.get(self)
      base.put(self)
      self._and=self.num1 & self.num2
```

```
        self._or=self.num1 | self.num2
        self._not=~self.num1
        self._xor=self.num1 ^ self.num2
        print('AND:',self._and)
        print('OR:',self._or)
        print('NOT:',self._not)
        print('XOR:',self._xor)
c1=child1()
c2=child2()
c1.put1()
c2.put2()
```

Output:

```
Enter the num1:5
Enter the num2:6
Num1:  5
Num2:  6
Additon:  11
Subtraction:  -1
Division:  0.8333333333333334
Multiplication:  30
Exponentitional:  15625
Enter the num1:5
Enter the num2:4
Num1:  5
Num2:  4
AND:  4
OR:  5
NOT:  -6
XOR:  1
```

15.B.

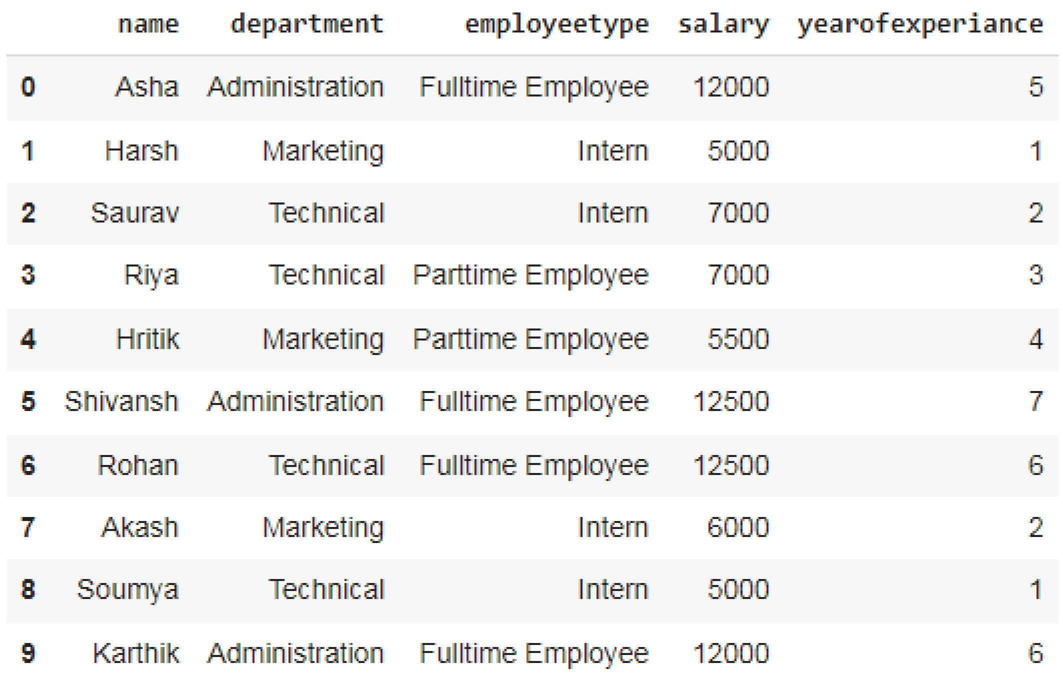


	Name	Department	Employment Type	Salary	Years of Experience
0	Asha	Administration	Full-time Employee	120000	5
1	Harsh	Marketing	Intern	50000	1
2	Sourav	Technical	Intern	70000	2
3	Riya	Technical	Part-time Employee	70000	3
4	Hritik	Marketing	Part-time Employee	55000	4
5	Shivansh	Administration	Full-time Employee	120000	7
6	Rohan	Technical	Full-time Employee	125000	6
7	Akash	Marketing	Intern	60000	2
8	Soumya	Technical	Intern	50000	1
9	Kartik	Administration	Full-time Employee	120000	6

a.create the data frame

```
import pandas as pd
d2=pd.DataFrame({'name':['Asha','Harsh','Saurav','Riya','Hritik'
,'Shivansh','Rohan','Akash','Soumya','Karthik'],
                 'department':['Administration','Marketing','Tec
hnical','Technical','Marketing','Administration','Technical','Ma
rketing','Technical','Administration'],
                 'employeetype':['Fulltime Employee','Intern','I
ntern','Parttime Employee','Parttime Employee','Fulltime Employe
e','Fulltime Employee','Intern','Intern','Fulltime Employee'],
                 'salary':[12000,5000,7000,7000,5500,12500,12500
,6000,5000,12000],
                 'yearofexperience':[5,1,2,3,4,7,6,2,1,6]})
display(d2)
```

Output:



	name	department	employeetype	salary	yearofexperiance
0	Asha	Administration	Fulltime Employee	12000	5
1	Harsh	Marketing	Intern	5000	1
2	Saurav	Technical	Intern	7000	2
3	Riya	Technical	Parttime Employee	7000	3
4	Hritik	Marketing	Parttime Employee	5500	4
5	Shivansh	Administration	Fulltime Employee	12500	7
6	Rohan	Technical	Fulltime Employee	12500	6
7	Akash	Marketing	Intern	6000	2
8	Soumya	Technical	Intern	5000	1
9	Karthik	Administration	Fulltime Employee	12000	6

b.Use pandas groupby to group rows department

```
d2.groupby(['department'])[['name','employeetype','salary','year
ofexperience']].count()
```

Output:

	name	employeetype	salary	yearofexperience
department				
Administration	3	3	3	3
Marketing	3	3	3	3
Technical	4	4	4	4

c.List the part-time employees of marketing department

```
d2.loc[(d2.department=='Marketing')&(d2.employeetype=="Parttime
Employee")]
```

Output:

	name	department	employeetype	salary	yearofexperience
4	Hritik	Marketing	Parttime Employee	5500	4

d.Find average salary of each group

```
d2.groupby('department').mean("salary")
```

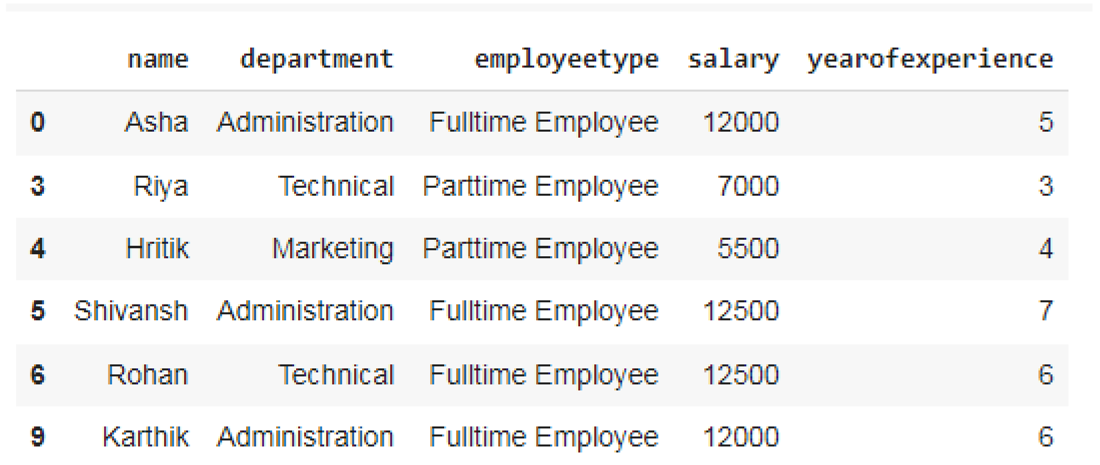
Output:

	salary	yearofexperience
department		
Administration	12166.666667	6.000000
Marketing	5500.000000	2.333333
Technical	7875.000000	3.000000

e.Display the employee details whose experience is greater than 2

```
d2.loc[d2.yearofexperience>2, ['name','department','employeetype
','salary','yearofexperience']]
```

Output:



```
      name    department       employeetype  salary  yearofexperience
0     Asha  Administration  Fulltime Employee   12000                 5
3     Riya       Technical  Parttime Employee    7000                 3
4   Hritik       Marketing  Parttime Employee    5500                 4
5 Shivansh  Administration  Fulltime Employee   12500                 7
6    Rohan       Technical  Fulltime Employee   12500                 6
9  Karthik  Administration  Fulltime Employee   12000                 6
```

f.Print the summary info of the dataset

```
d2.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column             Non-Null               Count  Dtype
---  ------             --------------          -----
 0   name               10 non-null            object
 1   department         10 non-null            object
 2   employeetype       10 non-null            object
 3   salary             10 non-null            int64
 4   yearofexperience   10 non-null            int64
dtypes: int64(2), object(3)
memory usage: 528.0+ bytes
```

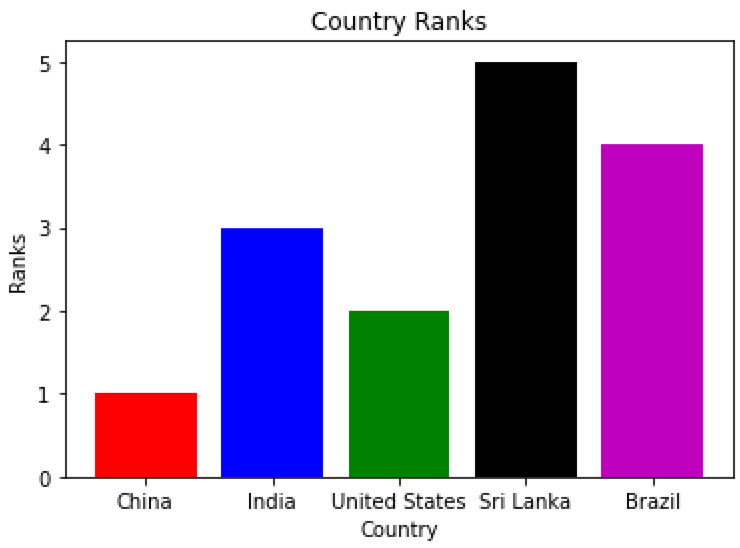
16.A.Bar graph

Country	Rank in Area
China	1
India	3
United States	2
Srilanka	5
Brazil	4

```python
#bar graph
import matplotlib.pyplot as plt
country=['China','India','United States','Sri Lanka','Brazil']
rank=[1,3,2,5,4]
plt.title('Country Ranks')
```

```
colors=('r','b','g','k','m')
plt.xlabel("Country")
plt.ylabel("Ranks")
plt.bar(country,rank,color=colors)
plt.show()
```

Output:



16.B. Refer 5.B **5.B. File handling**

17.A.Refer 3.B **3.B.**

17.B.Refer 14.A **14.A. Pie graph**

18.A.Refer 2.A **2.A. Perfect number from 1-n using function with arguments and no return type**

18.B.Refer 8.A **8.A.Tuple**

19.A.Refer 7.B **7.B. Arithmetic operator overloading to subtract two object and logical operator overloading**

19.B.Refer 9.A **9.A. Dictionary**