Noida Institute Of
Engineering & Technology
AN AUTONOMOUS INSTITUTE

Subject Name : Problem Solving Using
Advance Python Lab
Subject Code : **ACSE0252**

Submitted By:

Name: Gaurav Singh
Branch: CSE(AI)
Section: AI-II-B
Year/Sem: 1stYear/IInd Sem
Session: 2020-21
Roll No: 2001331160049
ERP : 0201CSAI117

Submitted To:

Mr. Sarvachan Verma

In [ ]:

```
'''
Problem Solving Using Advance Python Lab
Name: Gaurav Singh
Section: AI-II-B
Roll Number: 2001331160049
ERP: 0201CSAI117
Email: 0201csai117@niet.co.in
'''
```

In [1]:

```
'''1.Write a program illustrating class definition and accessing class members.
The Synatx for Accessing class member is : object_name.class_member_name'''

class Definition():

    '''Illustrating class definition and accessing class member'''

    classVar="Gaurav"       #ClassVaribale

    def __init__(self,name=None):
        self.name=name       #InstanceVariable
        print("Object Created Sucessfully thats why i am getting printed")

    @staticmethod
    def staticmethod():
        print("I am static method called by class name")

    @classmethod
    def classmethod(cls):
        print("I am class method called by class name or object name")


#ObjectCreation
obj=Definition()       #Object of Definition class is created

#Calling the Static Method
Definition.staticmethod()

#Calling Class Method
obj.classmethod()

#Accessing the class Variable
print("I am class variable called by class name: ",Definition.classVar)

#Creating 2nd Object of class
obj2=Definition("Gaurav")  #Passing a value to instance variable

#Calling the instance variable by name of object
print("I am instance variable called by object name: ",obj2.name)

#Accessing the docstring of class
```

```
print("I am Docstring: ",Definition.__doc__)
```

```
Object Created Sucessfully thats why i am getting printed
I am static method called by class name
I am class method called by class name or object name
I am class variable called by class name:  Gaurav
Object Created Sucessfully thats why i am getting printed
I am instance variable called by object name:  Gaurav
I am Docstring:  Illustrating class definition and accessing class member
```

In [1]:

```
#2. Write a program to implement default constructor(Non-Parameterised),
#parameterized constructor, and destructor

class ConDis():
    '''Implementing constructor - (Default and Parameterised) and Distrucutor'''
    class_var=0

    def __init__(self):
        '''It is Default constructor as it has no parameter.
           python always search for the last constructor.
           As It support constructor overriding'''

        print("I am default Constructor")

    def __init__(self,var):
        '''It is parameterised Construcutor'''

        ConDis.class_var+=1
        self.var=var
        print("Value of Instance Variable is: ",self.var)
        print("value of Class variable is :",ConDis.class_var)

    #Inplementing Distructor
    def __del__(self):
        '''__del__() is analogous to distructor in C++ and JAVA'''
        ConDis.class_var-=1    #Decreasing the count
        print("Object with value %d is going out of scope"%self.var)

#Creating First Object of Class
obj1=ConDis(10)

#Creating Second Object of Class
obj2=ConDis(20)

#Invoking The Distructor
del obj1
del obj2

#Checking if obj1 and obj2 is distructed or not
# print(obj1)
```

```
Value of Instance Variable is:  10
value of Class variable is : 1
Value of Instance Variable is:  20
value of Class variable is : 2
Object with value 10 is going out of scope
Object with value 20 is going out of scope
```

In [2]:

```
#3. Create a Python class named Rectangle constructed by a length and width.
#a. Create a method called area which will compute the area of a rectangle

class Rectangle():

    def __init__(self,length=None,width=None):

        self.length=int(input("Enter the Length of Rectangle in CM = "))
        self.width=int(input("Enter the Breadth of Rectangle in CM = "))

        print("Length is =",self.length,"cm")
        print("Width is =",self.width,"cm")


    def Area(self):
        '''Area of Rectangle=length*width'''
        print("The Area of Rectange is =",self.length*self.width,"sq unit")

R=Rectangle()
R.Area()
```

```
Enter the Length of Rectangle in CM = 10
Enter the Breadth of Rectangle in CM = 15
Length is = 10 cm
Width is = 15 cm
The Area of Rectange is = 150 sq unit
```

In [1]:

```
'''4.Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a
(these should all be numbers).
a.Write an instance  method called add which returns the sum of the attributes x and y.
b.Write a class method called multiply, which takes a single number parameter a and returns the
c.Write a static method called subtract, which takes two number parameters, b and c, and retur
d.Write a method called value which returns a tuple containing the values of x and y.'''

class Numbers():
  MULTIPLIER=int(input("Enter value of MULTIPLIER: "))

  def __init__(self):
    self.x=int(input("Enter the value of x: "))
    self.y=int(input("Enter the value of y: "))

  #Instance Method
  def add(self):
    return self.x+self.y

  @classmethod
  def multiply(cls,a):
    cls.a=a
    return cls.a*Numbers.MULTIPLIER

  @staticmethod
  def subtract(b,c):
    return b-c

  def value(self):
    return (self.x,self.y)

number=Numbers()
print("Sum=",number.add())
print("Product=",number.multiply(10))
print("Subtract=",Numbers.subtract(89,79))
print("Value of x and y in TUPLE : ",number.value())

print("-*"*10,"THANK FOR USING","-*"*10)
```

```
Enter value of MULTIPLIER: 10
Enter the value of x: 10
Enter the value of y: 5
Sum= 15
Product= 100
Subtract= 10
Value of x and y in TUPLE :  (10, 5)
-*-*-*-*-*-*-*-*-*-* THANK FOR USING -*-*-*-*-*-*-*-*-*-*
```

In [2]:

```
'''5.Create a class named as Student to store the name and marks in
three subjects. Use List to store the marks.

a.Write an instance method called compute to compute total marks and average marks of a stude
b.Write a method called display to display student information.'''

#name of student and marks in three subject
class Student:
    def __init__(self):
        self.name=input("Enter Name: ")
        self.marks=[]
    def getDetail(self):
        for i in range(3):
            m=int(input("Enter the marks of %s in subject %d :"%(self.name,i+1)))
            self.marks.append(m)
        print("\n")
    def compute(self):
        m=self.marks
        self.total_marks=m[0]+m[1]+m[2]
        self.average_marks=self.total_marks/len(m)

    def display(self):
        print("*"*20,"STUDENT DETAIL","*"*20)
        print("Name Of Student: ",self.name)
        print("Total Marks: ",self.total_marks)
        print("Average Marks: ",self.average_marks)

s=Student()
s.getDetail()
s.compute()
s.display()
```

```
Enter Name: Gaurav Singh
Enter the marks of Gaurav Singh in subject 1 :90
Enter the marks of Gaurav Singh in subject 2 :92
Enter the marks of Gaurav Singh in subject 3 :94


******************** STUDENT DETAIL ********************
Name Of Student:  Gaurav Singh
```

```
Total Marks: 276
Average Marks: 92.0
```

In [3]:

```
'''6. Create a class Employee that keeps a track of the number of employees in
   an organization and also stores their name, designation and salary details.
a. Write a method called getdata to take input (name, designation, salary) from user.
b. Write a method called average to find average salary of all the employees in the organization.
c. Write a method called display to print all the information of an employee'''

class Employee:
    count = 0

    #def __init__(self,name,designation,salary):  # If we use we will get Type Error
    def __init__(self,name=None,designation=None,salary=None,mobileno=None):
        self.name = name
        self.designation = designation
        self.salary = salary
        self.mobileno=mobileno
        Employee.count = Employee.count + 1

    def getdata(self):
        self.name = input("Enter Name of Employee: ")
        self.designation = input("Enter Designation : ")
        self.salary = int(input("Enter Salary : "))
        self.mobileno=input("Enter Contact Number: ")
        print("\n")

    @staticmethod
    def average(li):
        sum = 0
        for obj in li:
            sum = sum + obj.salary
        return sum/Employee.count

    def display(self):
        print("-*"*15)
        print("Name Of Employee :",self.name)
        print("Designation :",self.designation)
        print("Salary :",self.salary)
        print("Contact Number :",self.mobileno)
        print("-*"*15)
        print("\n")

myemplist = []
```

```
n = int(input("Enter number of employees: "))
print("\n")
for i in range(n):
    E1 = Employee()
    E1.getdata()
    myemplist.append(E1)
    E1.display()
print("Total Number of Employee: ",Employee.count)
print("Average Salary :",Employee.average(myemplist))
```

```
Enter number of employees: 2


Enter Name of Employee: Gaurav Singh
Enter Designation : AI
Enter Salary : 45000
Enter Contact Number: 9968975055


-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Name Of Employee : Gaurav Singh
Designation : AI
Salary : 45000
Contact Number : 9968975055
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


Enter Name of Employee: Ninja
Enter Designation : ML
Enter Salary : 67000
Enter Contact Number: 9873214565


-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Name Of Employee : Ninja
Designation : ML
Salary : 67000
Contact Number : 9873214565
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


Total Number of Employee:  2
Average Salary : 56000.0
```

In [4]:

```
'''7. Create a Python class named Circle constructed by a radius.
   Use a class variable to define the value of constant PI.
a. Write two methods to be named as area and circum to compute the area
   and the perimeter of a circle respectively by using class variable PI.
b. Write a method called display to print area and perimeter. '''

class Circle:
    PI=3.14159

    def __init__(self,radius=None):      #Constructed by radius
        self.radius=float(input("Enter length of radius in CM: "))

    def circum(self):
        return 2*Circle.PI*self.radius

    def area(self):
        return Circle.PI*self.radius*self.radius

    def display(self):
        print("The Area of Circle is=",self.area(),"sq cm")
        print("The Circumference of Circle is=",self.circum(),"cm")

#Creating object of Circle class
c=Circle()
#Calling the display Method
c.display()
```

```
Enter length of radius in CM: 7
The Area of Circle is= 153.93791 sq cm
The Circumference of Circle is= 43.98226 cm
```

In [5]:

```
'''8. Create a class called String that stores a string and all its status details such as
   number of uppercase letters,lowercase letters, vowels ,consonants and space in instance variab
a. Write methods named as count_uppercase, count_lowercase, count_vowels,
   count_consonants and count_space to count corresponding values.
b. Write a method called display to print string along with all the values computed by methods in

class String:

    def __init__(self):
        self.vowels=0
        self.consonants=0
        self.spaces=0
        self.uppercase=0
        self.lowercase=0
        self.string=str(input("Enter a String: "))
        print("\n")

    def count_vowels(self):
        for i in self.string:
            if i in ('a','e','i','o','u','A','E','I','O','U'):
                self.vowels+=1



    def count_consonants(self):
        for i in self.string:
            if i not in ('a','e','i','o','u','A','E','I','O','U',' '):
                self.consonants+=1

    def count_uppercase(self):
        for i in self.string:
            if i.isupper():
                self.uppercase+=1

    def count_lowercase(self):
        for i in self.string:
            if i.islower():
                self.lowercase+=1

    def count_spaces(self):
        for i in self.string:
            if i==' ':
```

```
            self.spaces+=1

    def compute(self):
        self.count_vowels()
        self.count_consonants()
        self.count_spaces()
        self.count_uppercase()
        self.count_lowercase()

    def display(self):
        print("*-"*20)
        print('Vowels=',self.vowels)
        print('Consonants=',self.consonants)
        print('Spaces=',self.spaces)
        print('Uppercase=',self.uppercase)
        print('Lowercase=',self.lowercase)
        print('*-'*10,"Have A Nice Day","*-"*10)
s=String()
s.compute()
s.display()
```

```
Enter a String: Artificial Intelligence


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
Vowels= 10
Consonants= 12
Spaces= 1
Uppercase= 2
Lowercase= 20
*-*-*-*-*-*-*-*-*-*- Have A Nice Day *-*-*-*-*-*-*-*-*-*-
```

In [6]:

```
'''9. Write a program that has a class called Fraction with attributes
  numerator and denominator.
a. Write a method called getdata to enter the values of the attributes.
b. Write a method show to print the fraction in simplified form.'''

class Fraction:
    def get_data(self):
        self.__num=int(input("Numenator: "))
        self.__deno=int(input("Denomenator: "))
        if self.__deno==0:
            print("ZeroDivisonEror")
            exit()

    def __GCD(self,a,b):
        if b==0:
            return a
        else:
            return self.__GCD(a,a%b)

    def __simplify(self):
        common_divisor=self.__GCD(self.__num,self.__deno)
        self.__num/=common_divisor
        self.__deno/=common_divisor

    def show_data(self):
        self.__simplify()
        print("Simplified Form=",self.__num,"/",self.__deno)
f=Fraction()
f.get_data()
f.show_data()
```

```
Numenator: 27
Denomenator: 123
Simplified Form= 1.0 / 4.555555555555555
```

In [7]:

```
'''10. Write a program that has a class Numbers with a list as an instance variable.
 a. Write a method called insert_element that takes values from user.
 b. Write a class method called find_max to find and print largest value in the list.'''

class Number():
    def __init__(self):
        self.mylist=[]

    def insert_element(self):
        value=int(input("How Many Value do you want to enter: "))
        for i in range(value):
            values=int(input("Enter values: "))
            self.mylist.append(values)

    def find_max(self):

        max=self.mylist[1]
        for i in self.mylist:
            if (i>max):
                max=i
        print("The Maxium Value in the list is=",max)

#Creating Object of Number class
n=Number()
#calling the insert element method
n.insert_element()
#Now calling the findmax method
n.find_max()
```

```
How Many Value do you want to enter: 3
Enter values: 10
Enter values: 15
Enter values: 20
The Maxium Value in the list is= 20
```

In [8]:

```
'''11. Write a program that has a class Point with attributes x and y.
 a. Write a method called midpoint that returns a midpoint of a line joining two points.
 b. Write a method called length that returns the length of a line joining two points.'''

class Point():

    def __init__(self):
        self.x1=int(input("Enter the value of x1: "))
        self.y1=int(input("Enter the value of y1: "))
        self.x2=int(input("Enter the value of x2: "))
        self.y2=int(input("Enter the value of y2: "))

    def midself(self):
        return (self.x1+self.x2)/2,(self.y1+self.y2)/2

    def length(self):
        return (((self.x2-self.x1)**2)+((self.y2-self.y1)**2))**0.5

P=Point()
print("The Mid self of line is=",P.midself())
print("The Length of line is= %.2f" %P.length(),"unit")
```

```
Enter the value of x1: 5
Enter the value of y1: 2
Enter the value of x2: 7
Enter the value of y2: 7
The Mid self of line is= (6.0, 4.5)
The Length of line is= 5.39 unit
```

In [9]:

```
'''12.Create a class called Complex. Write a menu driven program to read, display,
add and subtract two complex numbers by creating corresponding instance methods.'''

class Complex():

    def read(self):
        self.real1=int(input("Enter real part 1: "))
        self.comp1=int(input("Enter comp part 1: "))
        self.real2=int(input("Enter real part 2: "))
        self.comp2=int(input("Enter comp part 2: "))

    def display(self):
        print("First Complex number C1=",str(self.real1)+"+"+str(self.comp1)+"i")
        print("Second Complex number C2=",str(self.real2)+"+"+str(self.comp2)+"i")
        print("\n")

    def add(self):
        real_sum=self.real1+self.real2
        comp_sum=self.comp1+self.comp2
        print("The Sum of Complex Number is :",str(real_sum)+"+"+str(comp_sum)+"i")
        print("\n")
    def subtract(self):
        real_sub=self.real1-self.real2
        comp_sub=self.comp1-self.comp2
        if comp_sub<0:
            print("The Subtraction of Complex Number is :",str(real_sub)+str(comp_sub)+"i")
        else:
            print("The Subtraction of Complex Number is :",str(real_sub)+"+"+str(comp_sub)+"i")
        print("\n")

C=Complex()
while True:
    print("* "*17)
    print('* 1. Read Complex Number        *')
    print('* 2. Display Complex Number     *')
    print('* 3. Add Two Complex Number     *')
    print('* 4. Subtract Two Complex Number *')
    print(" *"*17)
    choice=int(input("Enter Your choice from above or Press any key to terminate: "))
    print("\n")
    if choice==1:
```

```
            C.read()
        elif choice==2:
            C.display()
        elif choice==3:
            C.add()
        elif choice==4:
            C.subtract()
        else:
            print("Oops! It is an invalid input. Logging Off....")
            break
```

```
* * * * * * * * * * * * * * * * * *
* 1. Read Complex Number        *
* 2. Display Complex Number     *
* 3. Add Two Complex Number     *
* 4. Subtract Two Complex Number *
 * * * * * * * * * * * * * * * * *
Enter Your choice from above or Press any key to terminate: 1


Enter real part 1: 4
Enter comp part 1: 6
Enter real part 2: 8
Enter comp part 2: 10
* * * * * * * * * * * * * * * * * *
* 1. Read Complex Number        *
* 2. Display Complex Number     *
* 3. Add Two Complex Number     *
* 4. Subtract Two Complex Number *
 * * * * * * * * * * * * * * * * *
Enter Your choice from above or Press any key to terminate: 5


Oops! It is an invalid input. Logging Off....
```

In [10]:

```
'''13.Write a Program to illustrate the use of
__str__(), __repr__(), __new__,  __doc__, __dict__, __name__ and __bases__ methods.'''

class Builtin():
    '''I am the documentation of class'''

    def __new__(cls,var1,var2):   #Use of static method __new__()
        print("__new__() magic method is called")
        inst=object.__new__(cls)
        return inst

    def __init__(self,var1,var2):    #Use of __init__ and __new__() method
        print("__init__() magic method is called")
        self.var1=var1
        self.var2=var2

    def __repr__(self):
        return "var1=%s,var2=%s"%(self.var1,self.var2)

    def __str__(self):
        return "var1 is %s and var2 is %s"%(self.var1,self.var2)

B=Builtin(10,20)
print("I am __str__() method",B)   #This will call __str__() method.
print("I am __repr__() method",[B])  #This will call __repr__() method.

print("Hello, i am __doc__ method Called by object.__doc__: ",B.__doc__)
print("Hello, i am __dict__ method Called by object.__dict__: ",B.__dict__)
print("Hello, i am __name__ method Called by class.__name__: ",Builtin.__name__)
print("Hello, i am __bases__ method Called by class.__bases__: ",Builtin.__bases__)
```

```
__new__() magic method is called
__init__() magic method is called
I am __str__() method var1 is 10 and var2 is 20
I am __repr__() method [var1=10,var2=20]
Hello, i am __doc__ method Called by object.__doc__:  I am the documentation of class
Hello, i am __dict__ method Called by object.__dict__:  {'var1': 10, 'var2': 20}
Hello, i am __name__ method Called by class.__name__:  Builtin
Hello, i am __bases__ method Called by class.__bases__:  (<class 'object'>,)
```

In [11]:

```
'''14. Create a BankAccount class. Your class should support the following methods:
a. __init__(self, account_no)
b. deposit (self, account_no, amount)
c. withdraw (self, account_no, amount)
d. get_balance (self, account_no)'''

class BankAccount():
    def __init__(self,account_no):
        self.account_no=account_no
        self.Balance=0
        print("New Account Created Successfully")

    def deposit(self,account_no,amount):
        self.Balance+=amount
        print("Your Updated Account Balace After deposit is=INR",self.Balance,"/-")

    def withdraw(self,account_no,amount):
        if (amount>self.Balance):
            print("You Don't have sufficient Balance")
            print("You have only INR %d /- rupees left in your Account"%self.Balance)
        else:
            self.Balance-=amount
            print('New Balance after withdrawl is=Rs',self.Balance)

    def get_balance(self,account_no):
        print("Current Account balance is=Rs",self.Balance)

Bank=BankAccount(3014522574210)
Bank.deposit(3014522574210,151500)
Bank.withdraw(3014522574210,51500)
Bank.get_balance(3014522574210)
```

```
New Account Created Successfully
Your Updated Account Balace After deposit is=INR 151500 /-
New Balance after withdrawl is=Rs 100000
Current Account balance is=Rs 100000
```

In [12]:

```
'''15. Write a program to illustrate the use of following built-in methods:
 a. hasattr(obj,attr)
 b. getattr(object, attribute_name [, default])
 c. setattr(object, name, value)
 d. delattr(class_name, name)'''

class BuiltinMethod():

    def __init__(self,var):
        self.var=var

    def Display(self):
        print("Var is = ",self.var)

obj=BuiltinMethod(10)
obj.Display()
#Using hasattr(object,attr) method
print("Check if object has attribute var.....",hasattr(obj,'var'))
#Using getattr(object,attributename) method
getattr(obj,'var')
#Using setattr(object,name,value) method
setattr(obj,'var',50)
print("After setting value,var is = ",obj.var)
#We can also change the name in setattr
setattr(obj,'count',50)
print("New Variable count is created and the value is = ",obj.count)
#Using delattr(class_name,name) method
delattr(obj,'var')
# print("After deleting,var is = ",obj.var) #It will give AttributeError
```

```
Var is =  10
Check if object has attribute var..... True
After setting value,var is =  50
New Variable count is created and the value is =  50
```

In [13]:

```
'''16. Write a program to create class Employee. Display the personal information and
salary details of 5 employees using single inheritance.'''
class Employee():
    empCount=0
    def __init__(self,name,deg,dep,contact,sal):
        self.name=name
        self.deg=deg
        self.dep=dep
        self.contact=contact
        self.sal=sal
        Employee.empCount+=1

class Information(Employee):
    def personal(self):
        print("Personal information of Employee "+str(Employee.empCount))
        print("*"*20)
        print("Name of Employee: ",self.name)
        print("Degination: ",self.deg)
        print("Department: ",self.dep)
        print("Contact: ",self.contact)
        print("Salary :",self.sal)
        print("*"*20,"\n")


i1=Information("Gaurav Singh","AI","EM","9968975055",75000)
i1.personal()
i2=Information("Aman Singh","AS","Account","9478486572",60000)
i2.personal()
i3=Information("Ninja ","GD","Product","9756348574",55000)
i3.personal()
i4=Information("Aarav Chauhan","IW","Lancing","6979893210",65000)
i4.personal()
i5=Information("Amrit Gupta","MD","Service","9945178756",75000)
i5.personal()
print("There are total "+str(Employee.empCount)+" Employee")
```

```
Personal information of Employee 1
**********
Name of Employee:  Gaurav Singh
Degination:  AI
Department:  EM
Contact:  9968975055
Salary : 75000
**********

Personal information of Employee 2
**********
Name of Employee:  Aman Singh
Degination:  AS
Department:  Account
Contact:  9478486572
Salary : 60000
**********

Personal information of Employee 3
**********
Name of Employee:  Ninja
Degination:  GD
Department:  Product
Contact:  9756348574
Salary : 55000
**********

Personal information of Employee 4
**********
Name of Employee:  Aarav Chauhan
Degination:  IW
Department:  Lancing
Contact:  6979893210
Salary : 65000
**********

Personal information of Employee 5
**********
Name of Employee:  Amrit Gupta
Degination:  MD
Department:  Service
Contact:  9945178756
Salary : 75000
**********

There are total 5 Employee
```

In [14]:

```
'''17. WAP that extends the class Employee. Derive two classes
Manager and Team Leader from Employee class.
Display all the details of the employee working under a particular Manager and Team Leader. '''
class Employee():
    empCount=0
    def __init__(self,name=None,age=None,exp=None,dept=None,contact=None,qual=None):
        self.name=input("Enter name of Employ: ")
        self.age=int(input("Enter Age of Employee: "))
        self.exp=int(input("Enter experience of Employee in Year: "))
        self.dept=input("Enter department of Employee: ")
        self.contact=input("Enter Contact Number: ")
        self.qual=input("Enter Highest Qualification: ")
        Employee.empCount+=1
    def display(self):
        print("Name of Employee : ",self.name)
        print("Age : ",self.age)
        print("Experince: ",self.exp)
        print("Department: ",self.dept)
        print("Contact: ",self.contact)
        print("Qualification: ",self.qual)
        print("\n")

class Manager(Employee):
    def __init__(self,name1=None):
        self.name1=input("Enter the name of Manager: ")
        Employee.__init__(self)
    def displayData(self):
        print("Detail of Employee Working Under Manager Mr "+self.name1+" are : ")
        print("**"*18)
        super().display()
        print("**"*18,"\n")

class Team_Leader(Employee):
    def __init__(self,name2=None):
        self.name2=input("Enter the name of Team_leader: ")
        Employee.__init__(self)



    def DisplayData(self):
        print("Detail of Employee Working Under Team_Leader Mr "+self.name2+" are : ")
        print("**"*18)
```

```
        super().display()
        print("**"*18,"\n")

t=Team_Leader()
t.DisplayData()
m=Manager()
m.displayData()
print("The Total number of Employee is: ",Employee.empCount)
```

```
Enter the name of Team_leader: Gaurav Singh
Enter name of Employ: Ninja
Enter Age of Employee: 26
Enter experience of Employee in Year: 2
Enter department of Employee: AI
Enter Contact Number: 12345678910
Enter Highest Qualification: BTech
Detail of Employee Working Under Team_Leader Mr Gaurav Singh are :
************************************
Name of Employee :  Ninja
Age :  26
Experince:  2
Department:  AI
Contact:  12345678910
Qualification:  BTech


************************************

Enter the name of Manager: JBL
Enter name of Employ: Tesla
Enter Age of Employee: 30
Enter experience of Employee in Year: 5
Enter department of Employee: ML
Enter Contact Number: 10987654321
Enter Highest Qualification: MTech
Detail of Employee Working Under Manager Mr JBL are :
************************************
Name of Employee :  Tesla
Age :  30
Experince:  5
Department:  ML
Contact:  10987654321
Qualification:  MTech


************************************

The Total number of Employee is:  2
```

In [15]:

```
'''18.  Write a program that has a class Point.
Define another class Location which has two objects (Location and destination)
of class Point. Also, define a function in Location that prints the reflection on the y-axis'''

class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def get(self):
        return self.x,self.y
class Location(Point):
    def __init__(self,x1,y1,x2,y2):
        self.Location=Point(x1,y1)
        self.Destination=Point(x2,y2)

    def show(self):
        print("Location = ",self.Location.get())
        print("Destination = ",self.Destination.get())

    def Reflection(self):
        self.Destination.x=-self.Destination.x
        print("Reflection = ",self.Destination.x,self.Destination.y)
L=Location(1,2,3,4)
L.show()
L.Reflection()
```

```
Location =  (1, 2)
Destination =  (3, 4)
Reflection =  -3 4
```

In [16]:

```
'''19. WAP that create a class Student having attribute as name and age and
Marks class inheriting Students class with its own attributes marks1,
marks2 and marks3 as marks in 3 subjects. Also, define the class Result
that inherits the  Marks class with its own attribute total.
Every class has its own display() method to display the corresponding details.
Use __init__() and super() to implement the above classes.'''
class Student():
    def __init__(self,name=None,age=None):
        self.name=input("Enter Name of student: ")
        self.age=int(input("Enter age: "))
    def display(self):
        print("Name of Student : ",self.name)
        print("Age : ",self.age)


class Marks(Student):

    def get_marks(self,m1=None,m2=None,m3=None):
        self.m1=int(input("Enter the marks in Subject 1 = "))
        self.m2=int(input("Enter the marks in Subject 2 = "))
        self.m3=int(input("Enter the marks in Subject 3 = "))

    def display(self):
        super().display()
        print("Marks in Subject 1 = ",self.m1)
        print("Marks in Subject 2 = ",self.m2)
        print("Marks in Subject 3 = ",self.m3)

class Result(Marks):
    def display(self,total=None,avg=None):
        self.total=self.m1+self.m2+self.m3
        self.avg=(self.total)/3
        print("**"*20)
        super().display()
        print("Total Marks Obtained By Student is = ",self.total)
        print("Average Marks Of Student is = ",self.avg)
        print("**"*20)


r=Result()
r.get_marks()
```

```
r.display()
```

```
Enter Name of student: Gaurav Singh
Enter age: 20
Enter the marks in Subject 1 = 95
Enter the marks in Subject 2 = 90
Enter the marks in Subject 3 = 92
****************************************
Name of Student :  Gaurav Singh
Age :  20
Marks in Subject 1 =  95
Marks in Subject 2 =  90
Marks in Subject 3 =  92
Total Marks Obtained By Student is =  277
Average Marks Of Student is =  92.33333333333333
****************************************
```

In [17]:

```
'''20. Write a program that create a class Distance with members km and metres.
Derive classes School and office which store the distance
from your house to school and office along with other details.'''

class Distance():
    def __init__(self,km=None,metre=None,speed=None,time=None,choice=None):
        self.speed = int(input("Enter speed in km/h = "))
        self.time = float(input("Enter time in hour = "))
        self.km = self.speed*self.time
        self.metre = ((5/18)*self.speed)*(self.time*3600)

    def display(self):
        print(round(self.km,3),"KM or",round(self.metre,3),"metre")


class School(Distance):
    def __init__(self):
        print("Calculating Distance of School from House")
        super().__init__()
    def display(self):
        print("The Distance of School from House is : ",end='')
        super().display()
        print("**"*30)

class Office(Distance):
    def __init__(self):
        print("Calculating Distance of Office: ")
        super().__init__()
    def display(self):
        print("The Distance of Office from House is: ",end='')
        super().display()
        print("**"*30)

o=Office()
o.display()
s=School()
s.display()
```

```
Calculating Distance of Office:
Enter speed in km/h = 62
Enter time in hour = 0.5
```

```
The Distance of Office from House is: 31.0 KM or 31000.0 metre
************************************************************
Calculating Distance of School from House
Enter speed in km/h = 76
Enter time in hour = 0.42
The Distance of School from House is : 31.92 KM or 31920.0 metre
************************************************************
```

In [18]:

```
'''21. Write a program to create an abstract class Vehicle.
Derive three classes Car, Motorcycle and Truck from it.
Define appropriate methods and print the details of vehicle'''

class Vehicle:
    def vehicle_number(self):
        raise NotImplementedError()
    def fuel_type(self):
        raise NotImplementedError()
    def color(self):
        raise NotImplementedError()
    def no_of_wheels(self):
        raise NotImplementedError()
    def cc(self):
        raise NotImplementedError()

class Car(Vehicle):
    def vehicle_number(self):
        return "BR06-4200"
    def fuel_type(self):
        return "Petrol"
    def color(self):
        return "White"
    def no_of_wheels(self):
        return "Four"
    def cc(self):
        return "1390cc"


class Motorcycle(Vehicle):
    def vehicle_number(self):
        return "BR06-0420"
    def fuel_type(self):
        return "Petrol"
    def color(self):
        return "white"
    def no_of_wheels(self):
        return "Two"
    def cc(self):
        return "360cc"
```

```
class Truck(Vehicle):
    def vehicle_number(self):
        return "BR06-2569"
    def fuel_type(self):
        return "Diesel"
    def color(self):
        return "Orange"
    def no_of_wheels(self):
        return "Tweleve"
    def cc(self):
        return "5005cc"

Jaguar=Car()
print("Car--------->>",Jaguar.vehicle_number(),Jaguar.fuel_type(),Jaguar.color(),Jaguar.no_of_w
print("\n")
Bullet=Motorcycle()
print("Motorcycle-->>",Bullet.vehicle_number(),Bullet.fuel_type(),Bullet.color(),Bullet.no_of_whe
print("\n")
Turbo=Truck()
print("Truck------->>",Turbo.vehicle_number(),Turbo.fuel_type(),Turbo.color(),Turbo.no_of_whe
```

```
Car---------->> BR06-4200 Petrol White Four 1390cc


Motorcycle-->> BR06-0420 Petrol white Two 360cc


Truck------->> BR06-2569 Diesel Orange Tweleve 5005cc
```

In [19]:

```
'''22.Write a program that has a class Polygon.
Derive two classes Rectangle and triangle from polygon and write
methods to get the details of their dimensionsand  hence calculate the area'''

class Polygon():
    def get_data(self):
        raise NotImplementedError()
    def area(self):
        raise NotImplementedError()

class Rectangle(Polygon):
    def get_data(self):
        self.length=float(input("Length of Rectnagle   = "))
        self.breadth=float(input("Breadth of Rectabgle = "))
    def area(self):
        return self.length*self.breadth

class Triangle(Polygon):
    def get_data(self):
        self.a=float(input("Enter side a = "))
        self.b=float(input("Enter side b = "))
        self.c=float(input("Enter side c = "))
        self.s=(self.a+self.b+self.c)/2

    def area(self):
        return (self.s*(self.s-self.a)*(self.s-self.b)*(self.s-self.c))**0.5

R=Rectangle()
R.get_data()
print("The Area of Rectangle is =",R.area())
print("**"*30)
T=Triangle()
T.get_data()
print("The Area of Triangle is =",T.area())
```

```
Length of Rectnagle   = 10
Breadth of Rectabgle = 8
The Area of Rectangle is = 80.0
************************************************************
Enter side a = 5
Enter side b = 6
```

```
Enter side c = 8
The Area of Triangle is = 14.981238266578634
```

In [20]:

```
'''23. Write a program that extends the class Shape to calculate
the area of a circle and a cone .(use super to inherit base class methods'''

class Shape():
    PI=3.14
    def __init__(self,radius=None):
        self.radius=int(input("Enter Radius = "))

class Circle(Shape):
    def area(self):
        return Shape.PI*self.radius*self.radius

class Cone(Shape):
    def get_data(self):
        self.s=int(input("Enter slant height = "))
    def area(self):
        return Shape.PI*self.radius*(self.radius+self.s)

C=Circle()
print("The Area of the circle is = ",C.area())
print("**"*30)
cone=Cone()
cone.get_data()
print("The Ara of the cone is = ",cone.area())
```

```
Enter Radius = 7
The Area of the circle is =  153.86
************************************************************
Enter Radius = 15
Enter slant height = 8
The Ara of the cone is =  1083.3
```

In [21]:

```
'''24. Write a program to demonstrate hybrid inheritance and show MRO for each class'''

class School():
    def func1(self):
        print("In School")
class Student1(School):
    def func2(self):
        print("In student 1")
class Student2(School):
    def func3(self):
        print("In student 2")
class Student3(Student1,Student2):
    def func4(self):
        print("In Student 3")

print("Showing MRO for each class")
print(School.mro())
print(Student1.mro())
print(Student2.mro())
print(Student3.mro())

#demonstration of Hybrid Inheritence
S=Student3()
S.func1()
S.func2()
S.func3()
S.func4()
```

```
Showing MRO for each class
[<class '__main__.School'>, <class 'object'>]
[<class '__main__.Student1'>, <class '__main__.School'>, <class 'object'>]
[<class '__main__.Student2'>, <class '__main__.School'>, <class 'object'>]
[<class '__main__.Student3'>, <class '__main__.Student1'>, <class '__main__.Student2'>, <class '__main__.School'>, <class 'objec
In School
In student 1
In student 2
In Student 3
```

In [22]:

```
'''25. Write a program to overload + operator to multiply to
fraction object of fraction class which contain two instance
variable numerator and denominator.Also, define the instance
method  simplify() to simplify the fraction objects'''

class Fraction():
    def __init__(self):
        self.num=0
        self.deno=1  #Since denominator cant be zero
    def get(self):
        self.num=int(input("Enter the numenator = "))
        self.deno=int(input("Enter the denomenator = "))
    def simplify(self):   #to simplyfy the fraction object
        common_divisor=Fraction.GCD(self.num,self.deno)
        self.num=self.num//common_divisor
        self.deno=self.deno//common_divisor
    @staticmethod
    def GCD(num,deno):
        if deno==0:
            return num
        else:
            return Fraction.GCD(deno,num%deno)

    def __add__(self,F):
        Temp=Fraction()
        Temp.num=(self.num*F.deno)+(F.num*self.deno)
        Temp.deno=self.deno*F.deno
        return Temp
    def display(self):
        self.simplify()
        return str(self.num)+"/"+str(self.deno)

F1=Fraction()
F1.get()
F2=Fraction()
F2.get()
F3=Fraction()
F3=F1+F2
print("Resultant Fraction is = ",F3.display())
```

```
Enter the numenator = 27
```

```
Enter the denomenator = 123
Enter the numenator = 35
Enter the denomenator = 128
Resultant Fraction is =  2587/5248
```

In [23]:

```
'''26. Write a program to compare two-person object
based on their age by overloading > operator.'''

print("Hello User ! A person is 18 year 8 month and 8 days old, Hence Enter the value Accordingl
print("Enter the Date of birth to compare AGE from the foramt DD/MM/YYYY")
print("**"*30)
class Person():
    def __init__(self):
        self.d=self.m=self.y=0

    def get(self):
        self.d = int(input("Enter the Day = "))
        self.m = int(input("Enter the Month = "))
        self.y = int(input("Enter the Year = "))
    def __gt__(self,P):    #__gt__ > and __It__ <
        Flag = False
        if self.y>P.y:
            if self.m>P.m:
                if self.d>P.d:
                    Flag = True
        return Flag

P1=Person()
P1.get()
print("**"*20)
P2=Person()
P2.get()
print("**"*20)
print("P1 > P2",P1>P2)
```

```
Hello User ! A person is 18 year 8 month and 8 days old, Hence Enter the value Accordingly
Enter the Date of birth to compare AGE from the foramt DD/MM/YYYY
************************************************************
Enter the Day = 20
Enter the Month = 11
Enter the Year = 20
****************************************
Enter the Day = 10
Enter the Month = 10
Enter the Year = 19
****************************************
P1 > P2 True
```

In [24]:

```
'''27. Write a program to overload in Operator'''
class Popular():
    def __init__(self):
        self.max_popularity = {'Python':100,"Java":90,"Ruby":70,"c++":95,"Perl":50}
    def __contains__(self,lan):
        if lan in self.max_popularity:
            return True
        else:
            return False
    def __getitem__(self,lan):
        return self.max_popularity[lan]
    def __str__(self):
        return "The Dictionary has name of Language and its popularity percentage alloted to them

P=Popular()
print(str(P))
lan = input("Enter the language for which you want to know popularity : ")
if lan in P:
    print("The popularity of Language",lan,"is = ",P[lan])
```

```
The Dictionary has name of Language and its popularity percentage alloted to them
Enter the language for which you want to know popularity : Cpp
```

In [25]:

```
'''28. WAP to create a Complex class having real and imaginary as it attributes.
Overload the +,-,/,* and += operators for objects of Complex class'''
class Complex():
    def __init__(self):
        self.real = 0
        self.imag = 0
    def setValue(self,real,imag):
        self.real = real
        self.imag = imag
    #Overloading the + Operataor
    def __add__(self,C):
        Temp=Complex()
        Temp.real = self.real + C.real
        Temp.imag = self.imag + C.imag
        print("(",Temp.real,'+',Temp.imag,'i)')
    #Overloading the - Operataor
    def __sub__(self,C):
        Temp=Complex()
        Temp.real = self.real - C.real
        Temp.imag = self.imag - C.imag
        print("(",Temp.real,'-',Temp.imag,'i)')
    #Overloading the / Operataor
    def __truediv__(self,C):
        Temp=Complex()
        Temp.real = self.real / C.real
        Temp.imag = self.imag / C.imag
        print("(",round(Temp.real,2),'/',Temp.imag,'i)')
    #Overloading the * Operataor
    def __mul__(self,C):
        Temp=Complex()
        Temp.real = self.real * C.real
        Temp.imag = self.imag * C.imag
        print("(",Temp.real,'*',Temp.imag,'i*i)')
    #Overloading the += Operataor
    def __iadd__(self,C):
        Temp=Complex()
        self.real += C.real
        self.imag += self.imag + C.imag
        print("(",self.real,'+',self.imag,'i)')
    def __repr__(self):
        return self.real,self.imag
```

```
C1 = Complex()
C1.setValue(1,2)
C2 = Complex()
C2.setValue(3,4)
C3 = Complex()
C3 = C1+C2
C4 = Complex()
C4 = C1-C2
C5 = Complex()
C5 = C1/C2
C6 = Complex()
C6 = C1*C2
C1 += C2
```

```
( 4 + 6 i)
( -2 - -2 i)
( 0.33 / 0.5 i)
( 3 * 8 i*i)
( 4 + 8 i)
```

In [26]:

```
'''29. Write a program to inspect the object using type() ,id(), isinstance(), issubclass()
and callable() built-in function.'''

#Defining the parent class
class Vehicles:
    # Constructor
    def __init__(self):
        pass

# Defining Child class
class Car(Vehicles):
    # Constructor
    def __init__(self):
        Vehicles.__init__(self)
        print("Inspecting the object.....")

C=Car()

#Inspecting the object using type()
print('Inspecting using type() ---->>> ',type(C))
#Inspecting the object using id()
print('Inspecting using id() ---->>> ',id(C))
#Inspecting the object using isinstance()
print('Inspecting using isinstance() ---->>> ',isinstance(C,Car))
#Inspecting the object using issubclass()
print('Inspecting using issubclass() ---->>> ',issubclass(Car,Vehicles))
#Inspecting the object using callable()
print('Inspecting using callable() ---->>> ',callable(C))
```

```
Inspecting the object.....
Inspecting using type() ---->>> <class '__main__.Car'>
Inspecting using id() ---->>> 2227243603856
Inspecting using isinstance() ---->>> True
Inspecting using issubclass() ---->>> True
Inspecting using callable() ---->>> False
```

In [27]:

```
'''30. WAP to inspect the program code using the functions of inspect module.'''

#1. isclass()
import inspect
class A(object):
    pass
print(inspect.isclass(A))

#2. ismodule()
import numpy
print(inspect.ismodule(numpy))

#3. isfunction()
def fun(a):
    return 2*a
print(inspect.isfunction(fun))

#4. ismethod()
import collections
print(inspect.ismethod(collections.Counter))

#5. getmro()
class A(object):
    pass
class B(A):
    pass
class C(B):
    pass
print(inspect.getmro(C))

#6. getmembers()
import inspect
import math
print(inspect.getmembers(math))

#7. sinnature()
import collections
print(inspect.signature(collections.Counter))

#8. stack()
def Fibonacci(n):
```

```
    if n < 0:
        return 0

    elif n == 0:
        return 0

    elif n == 1:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)
Fibonacci(12)
print(inspect.stack())

#9. getsource()
def fun(a,b):
    return a*b
print(inspect.getsource(fun))

#10. getdoc()
import inspect
from tkinter import *
root = Tk()
root.title('SHIVESH SHIVAM')
print(inspect.getdoc(root))
```

```
True
True
True
False
(<class '__main__.C'>, <class '__main__.B'>, <class '__main__.A'>, <class 'object'>)
[('__doc__', 'This module provides access to the mathematical functions\ndefined by the C standard.'), ('__loader__', <class '_f
er'>), ('__name__', 'math'), ('__package__', ''), ('__spec__', ModuleSpec(name='math', loader=<class '_frozen_importlib.BuiltinI
('acos', <built-in function acos>), ('acosh', <built-in function acosh>), ('asin', <built-in function asin>), ('asinh', <built-in function asin
atan>), ('atan2', <built-in function atan2>), ('atanh', <built-in function atanh>), ('ceil', <built-in function ceil>), ('comb', <built-in fun
t-in function copysign>), ('cos', <built-in function cos>), ('cosh', <built-in function cosh>), ('degrees', <built-in function degrees>), ('
('e', 2.718281828459045), ('erf', <built-in function erf>), ('erfc', <built-in function erfc>), ('exp', <built-in function exp>), ('expm1
('fabs', <built-in function fabs>), ('factorial', <built-in function factorial>), ('floor', <built-in function floor>), ('fmod', <built-in fun
n function frexp>), ('fsum', <built-in function fsum>), ('gamma', <built-in function gamma>), ('gcd', <built-in function gcd>), ('hypot'
nf', inf), ('isclose', <built-in function isclose>), ('isfinite', <built-in function isfinite>), ('isinf', <built-in function isinf>), ('isnan', <bu
<built-in function isqrt>), ('ldexp', <built-in function ldexp>), ('lgamma', <built-in function lgamma>), ('log', <built-in function log>), ('
0>), ('log1p', <built-in function log1p>), ('log2', <built-in function log2>), ('modf', <built-in function modf>), ('nan', nan), ('perm', <bui
41592653589793), ('pow', <built-in function pow>), ('prod', <built-in function prod>), ('radians', <built-in function radians>), ('rema
inder>), ('sin', <built-in function sin>), ('sinh', <built-in function sinh>), ('sqrt', <built-in function sqrt>), ('tan', <built-in function tan
anh>), ('tau', 6.283185307179586), ('trunc', <built-in function trunc>)]
(iterable=None, /, **kwds)
[FrameInfo(frame=<frame at 0x00000206920308B0, file '<ipython-input-27-15119bb24955>', line 53, code <module>, filename='
955>', lineno=53, function='<module>', code_context=['print(inspect.stack())\n'], index=0), FrameInfo(frame=<frame at 0x000002
s\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\interactiveshell.py', line 3437, code run_code>, filename='C:\\Users
te-packages\\IPython\\core\\interactiveshell.py', lineno=3437, function='run_code', code_context=['                exec(code_ob
```

er_ns)\n'], index=0), FrameInfo(frame=<frame at 0x00000206903A69B0, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packag
veshell.py', line 3357, code run_ast_nodes>, filename='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\inter
function='run_ast_nodes', code_context=['   if (await self.run_code(code, result, async_=asy)):\n'], index=0), FrameIn
02069163DAB0, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\interactiveshell.py', line 3165, code r
='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\interactiveshell.py', lineno=3165, function='run_cell_asyn
has_raised = await self.run_ast_nodes(code_ast.body, cell_name,\n'], index=0), FrameInfo(frame=<frame at 0x0000020691F472
naconda3\\lib\\site-packages\\IPython\\core\\async_helpers.py', line 68, code _pseudo_sync_runner>, filename='C:\\Users\\th
ackages\\IPython\\core\\async_helpers.py', lineno=68, function='_pseudo_sync_runner', code_context=['   coro.send(None)\
me=<frame at 0x00000206904B8CD0, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\interactiveshel
ll>, filename='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core\\interactiveshell.py', lineno=2940, function='_
return runner(coro)\n'], index=0), FrameInfo(frame=<frame at 0x0000020692015240, file 'C:\\Users\\thats\\anaconda3\\lib\\
re\\interactiveshell.py', line 2894, code run_cell>, filename='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\IPython\\core
2894, function='run_cell', code_context=['   result = self._run_cell(\n'], index=0), FrameInfo(frame=<frame at 0x0000020
\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\zmqshell.py', line 536, code run_cell>, filename='C:\\Users\\thats\\anacon
ykernel\\zmqshell.py', lineno=536, function='run_cell', code_context=['   return super(ZMQInteractiveShell, self).run_cell(*a
FrameInfo(frame=<frame at 0x00000206915C5AB0, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\ipkern
ute>, filename='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\ipkernel.py', lineno=306, function='do_execute',
= shell.run_cell(code, store_history=store_history, silent=silent)\n'], index=0), FrameInfo(frame=<frame at 0x00000206917B940
ppData\\Roaming\\Python\\Python38\\site-packages\\tornado\\gen.py', line 234, code wrapper>, filename='C:\\Users\\thats\\
\\Python38\\site-packages\\tornado\\gen.py', lineno=234, function='wrapper', code_context=['   yielded = ctx_run(ne
meInfo(frame=<frame at 0x00000206904B9810, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\kernelbas
request>, filename='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\kernelbase.py', lineno=543, function='execut
self.do_execute(\n'], index=0), FrameInfo(frame=<frame at 0x00000206917B9B20, file 'C:\\Users\\thats\\AppData\\Roaming'
ckages\\tornado\\gen.py', line 234, code wrapper>, filename='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-
lineno=234, function='wrapper', code_context=['   yielded = ctx_run(next, result)\n'], index=0), FrameInfo(frame=<fra
0, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\kernelbase.py', line 268, code dispatch_shell>, filename='
3\\lib\\site-packages\\ipykernel\\kernelbase.py', lineno=268, function='dispatch_shell', code_context=['   yield gen.may
ents, msg))\n'], index=0), FrameInfo(frame=<frame at 0x00000206917BB6C0, file 'C:\\Users\\thats\\AppData\\Roaming\\Pyth
s\\tornado\\gen.py', line 234, code wrapper>, filename='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packa
=234, function='wrapper', code_context=['   yielded = ctx_run(next, result)\n'], index=0), FrameInfo(frame=<frame at
'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel\\kernelbase.py', line 365, code process_one>, filename='C:\\User
\\site-packages\\ipykernel\\kernelbase.py', lineno=365, function='process_one', code_context=['   yield gen.maybe_future(di
FrameInfo(frame=<frame at 0x00000206920BD440, file 'C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-pack
775, code run>, filename='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packages\\tornado\\gen.py', lineno=
text=['   yielded = self.gen.send(value)\n'], index=0), FrameInfo(frame=<frame at 0x0000020691FFEC10, file 'C:\\U
ming\\Python\\Python38\\site-packages\\tornado\\gen.py', line 814, code inner>, filename='C:\\Users\\thats\\AppData\\Roam
e-packages\\tornado\\gen.py', lineno=814, function='inner', code_context=['   self.ctx_run(self.run)\n'], index=0), Frame
00206916BD9A0, file 'C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packages\\tornado\\ioloop.py', line 741
me='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packages\\tornado\\ioloop.py', lineno=741, function='_run
ret = callback()\n'], index=0), FrameInfo(frame=<frame at 0x0000020692002780, file 'C:\\Users\\thats\\AppData\\Roaming\\
kages\\tornado\\ioloop.py', line 688, code <lambda>>, filename='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site
py', lineno=688, function='<lambda>', code_context=['   lambda f: self._run_callback(functools.partial(callback, future))\n
e=<frame at 0x00000206916BB8A0, file 'C:\\Users\\thats\\anaconda3\\lib\\asyncio\\events.py', line 81, code _run>, filename=
a3\\lib\\asyncio\\events.py', lineno=81, function='_run', code_context=['   self._context.run(self._callback, *self._args)\n'
=<frame at 0x00000206904A1A00, file 'C:\\Users\\thats\\anaconda3\\lib\\asyncio\\base_events.py', line 1859, code _run_on
ats\\anaconda3\\lib\\asyncio\\base_events.py', lineno=1859, function='_run_once', code_context=['   handle._run()\n'],
<frame at 0x0000020691FFA7C0, file 'C:\\Users\\thats\\anaconda3\\lib\\asyncio\\base_events.py', line 570, code run_forever
ts\\anaconda3\\lib\\asyncio\\base_events.py', lineno=570, function='run_forever', code_context=['   self._run_once()\
me=<frame at 0x0000020691FFA5E0, file 'C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packages\\tornado
199, code start>, filename='C:\\Users\\thats\\AppData\\Roaming\\Python\\Python38\\site-packages\\tornado\\platform\\asy
='start', code_context=['   self.asyncio_loop.run_forever()\n'], index=0), FrameInfo(frame=<frame at 0x0000020691D78A
naconda3\\lib\\site-packages\\ipykernel\\kernelapp.py', line 612, code start>, filename='C:\\Users\\thats\\anaconda3\\lib\\si
elapp.py', lineno=612, function='start', code_context=['   self.io_loop.start()\n'], index=0), FrameInfo(frame=<frame at 0
'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\traitlets\\config\\application.py', line 845, code launch_instance>, filename
da3\\lib\\site-packages\\traitlets\\config\\application.py', lineno=845, function='launch_instance', code_context=['   app.sta
o(frame=<frame at 0x000002068DF73800, file 'C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel_launcher.py', line
='C:\\Users\\thats\\anaconda3\\lib\\site-packages\\ipykernel_launcher.py', lineno=16, function='<module>', code_context=['
\n'], index=0), FrameInfo(frame=<frame at 0x000002068D7D6010, file 'C:\\Users\\thats\\anaconda3\\lib\\runpy.py', line 87,
='C:\\Users\\thats\\anaconda3\\lib\\runpy.py', lineno=87, function='_run_code', code_context=['   exec(code, run_globals)\n']
=<frame at 0x000002068D7606B0, file 'C:\\Users\\thats\\anaconda3\\lib\\runpy.py', line 194, code _run_module_as_main>, fil
naconda3\\lib\\runpy.py', lineno=194, function='_run_module_as_main', code_context=['   return _run_code(code, main_globals,

```
def fun(a,b):
    return a*b
```

```
Toplevel widget of Tk which represents mostly the main window
of an application. It has an associated Tcl interpreter.
```

In [28]:

```
'''31. Write a program to create a new list containing
the first letters of every element in an already existing list.'''

A=input("Enter String : ").split()
mylist=[str(i) for i in A]
def first_letter(mylist):
    for i in mylist:
        return i[0]

newlist=list(map(first_letter,mylist))

print("Old List = ",mylist)
print("New List = ",newlist)
```

```
Enter String : Artificial Intelligence
Old List =  ['Artificial', 'Intelligence']
New List =  ['A', 'I']
```

In [29]:

```
'''32 .Write a program using reduce() function to calculate
the sum of first 10 natural numbers'''

from functools import reduce
def add(x,y):
    return x+y
natural_num=[int(i) for i in range(1,11)]
print("First 10 Natural Numbers are----->>>>")
print(natural_num)

result=reduce(add,natural_num)
print("The Sum of first 10 Natural Number is ---->>>> ",result)
```

```
First 10 Natural Numbers are----->>>>
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The Sum of first 10 Natural Number is ---->>>>  55
```

In [30]:

```
'''33. Write a program that convert a list of temperatures in Celsius
into Fahrenheit using map() function.'''

mytemp=input("Enter temprature(s) in Celcius seperated by space : ").split()
myTemp=[int(i) for i in mytemp]

def toFarenheit(temp):
    return (temp*(9/5))+32

tempF=list(map(toFarenheit,myTemp))

print("The temperature(s) Entered By User in Celcius----->>>>>")
print(myTemp)
print("The temperature(s) converted in Farenheit are ------>>>>>>")
print(tempF)
```

```
Enter temprature(s) in Celcius seperated by space : 10 15 20 25 30 35 40
The temperature(s) Entered By User in Celcius----->>>>>
[10, 15, 20, 25, 30, 35, 40]
The temperature(s) converted in Farenheit are ------>>>>>>
[50.0, 59.0, 68.0, 77.0, 86.0, 95.0, 104.0]
```

In [31]:

```
'''34. Write a program that creates an iterator to print squares of numbers'''
class Square():
    def __iter__(self):
        self.num=int(input("Enter a Number : "))
        return self
    def __next__(self):
        x=self.num
        self.num=x**2
        return self.num

myclass=Square()
myiter=iter(myclass)
print("The Square of entered Number Generated by Iterator---->>>>",next(myiter))
```

```
Enter a Number : 10
The Square of entered Number Generated by Iterator---->>>> 100
```

In [32]:

```
'''35. Write a program that create a custom iterator to create even numbers.'''
class Even():
    def __iter__(self):
        self.num=0
        return self
    def __next__(self):
        num=self.num
        self.num+=2
        return num

e=Even()
i=iter(e)
enter=int(input("Enter a Number of terms : "))
for x in i:
    if x>enter:
        break
    else:
        print(x)
print("Operate next(i) to create even number iterator")
```

```
Enter a Number of terms : 10
0
2
4
6
8
10
Operate next(i) to create even number iterator
```

In [33]:

```
'''36. Write a program to create a generator that starts counting
from 0 and raise an exception when counter is equal to 10.'''

import time
def counting():
    num=0
    print("Counting Begins ---->>>> ")
    while True:
        yield num
        num=num+1


for i in counting():
    if i==10:
        raise StopIteration
    else:
        print(i)
        time.sleep(0.5)
```

```
Counting Begins ---->>>>
0
1
2
3
4
5
6
7
8
9


---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-33-c99182291b37> in <module>
     13 for i in counting():
     14     if i==10:
---> 15         raise StopIteration
     16     else:
     17         print(i)

StopIteration:
```

In [34]:

```
'''37. Write a program to create a generator to print the Fibonacci number.'''

def Fibonacci(terms):
    x,y=0,1
    print(x,end=' ')
    for _ in range(terms):
        x,y=y,x+y
        yield x
for i in Fibonacci(10):
    print(i,end=' ')
```

```
0 1 1 2 3 5 8 13 21 34 55
```

In [38]:

```
'''38. Write a program to create an arithmetic calculator using tkinter'''

import tkinter as tk
from functools import partial
root=tk.Tk()
root.title("Arithmetic Calculator")
root.geometry("600x500")
def findsum(l3,num1,num2):
    n1=int(num1.get())
    n2=int(num2.get())
    n3=n1+n2
    l3.config(fg="white",bg="black",height=3,width=20,text="Sum=%d"%n3)

def findsub(l3,num1,num2):
    n1=int(num1.get())
    n2=int(num2.get())
    n3=n1-n2
    l3.config(fg="white",bg="black",height=3,width=20,text="Difference=%d"%n3)

def findmul(l3,num1,num2):
    n1=int(num1.get())
    n2=int(num2.get())
    n3=n1*n2
    l3.config(fg="red",bg="yellow",height=3,width=20,text="Product=%d"%n3)

def findmod(l3,num1,num2):
    n1=int(num1.get())
    n2=int(num2.get())
    n3=n1%n2
    l3.config(fg="yellow",bg="blue",height=3,width=20,text="Modulus=%d"%n3)

def finddiv(l3,num1,num2):
    n1=int(num1.get())
    n2=int(num2.get())
    n3=n1/n2
    l3.config(fg="yellow",bg="blue",height=3,width=20,text="Division=%d"%n3)



#Labels
l1=tk.Label(root,text="Enter First Number",fg='red').place(x=20,y=60)
l2=tk.Label(root,text="Enter Second Number",fg='red').place(x=20,y=120)
```

```
#TextField
number1=tk.StringVar()
number2=tk.StringVar()  #use to hold value of textfiled,initially 0
t1=tk.Entry(root,textvariable=number1,bg='yellow',fg='black').place(x=200,y=60)
t2=tk.Entry(root,textvariable=number2,bg='yellow',fg='black').place(x=200,y=120)

labelre=tk.Label(root)
labelre.place(x=200,y=150)
findsum=partial(findsum,labelre,number1,number2)   #partialFunction
findsub=partial(findsub,labelre,number1,number2)   #partialFunction
findmul=partial(findmul,labelre,number1,number2)   #partialFunction
findmod=partial(findmod,labelre,number1,number2)   #partialFunction
finddiv=partial(finddiv,labelre,number1,number2)   #partialFunction
#button

button=tk.Button(root,text="ADD",command=findsum,bg='orange').place(x=50,y=300)
button=tk.Button(root,text="SUB",command=findsub,bg='yellow').place(x=150,y=300)
button=tk.Button(root,text="MUL",command=findmul,bg='green').place(x=250,y=300)
button=tk.Button(root,text="DIV",command=finddiv,bg='cyan').place(x=50,y=350)
button=tk.Button(root,text="MOD",command=findmod,bg='pink').place(x=150,y=350)
button=tk.Button(root,text="EXIT",command=root.destroy,bg='red').place(x=250,y=350)

root.mainloop()
```

In [40]:

```
'''39. Write a program to draw colored shapes (line, rectangle, oval) on canvas.'''
import tkinter as tk
top=tk.Tk()
top.title("Pattern-Design")
canvas=tk.Canvas(top,bg='white',height=700,width=1400)

#Designing with rectangle
canvas.create_rectangle(300,600,1100,50,outline='black',fill='red',width=7)
#creating Lines
canvas.create_line(1120,300,1400,300,fill='black',width=10)
#creating oval
canvas.create_oval(0,50,290,590,outline='black',fill='cyan',width=7)
#Creating TextBox
canvas.create_text(700,20,text='Developed By: Gaurav Singh',fill='black',font='bold')
canvas.create_text(140,650,text='OVAL',fill='black',font='bold')
canvas.create_text(700,650,text='RECTANGLE',fill='black',font='bold')
canvas.create_text(1260,350,text='LINE',fill='black',font='bold')

canvas.pack()
top.mainloop()
```

In [41]:

```
'''40. Write a program to create a window that
disappears automatically after 5 seconds.'''

import tkinter as tk
root = tk.Tk()
root.title("Automatically Disappear")
root.geometry("700x500")

tk.Label(root, text="It will Disapper in 5 Second",bg='black',fg='orange').place(x=250,y=150)

root.after(5000, lambda: root.destroy())     # time in ms 5 sec=5000

root.mainloop()
```

In [42]:

```
'''41. Write a program to create a button and a label inside the frame
widget. Button should change the color upon hovering over the button and label
should disappear on clicking the button.'''

import tkinter as tk
class Test():
  def __init__(self):
     self.root = tk.Tk()
     self.root.title("Question_41")
     self.root.geometry("700x600")
     self.label=tk.Label(self.root,
                         text = "i will disappead if you click this button")
     self.buttonForget = tk.Button(self.root,fg='orange',bg='black',
                        text = 'Click me to disable Label',
                        command=lambda:self.label.destroy())

     self.buttonForget.pack()
     self.buttonForget.place(x=260,y=280)

     self.label.pack()
     self.label.place(x=240,y=220)
     self.root.mainloop()

app = Test()
```

In [43]:

```
'''42. Write a program to create radio-buttons
(Male, Female, and Transgender) and a label.
Default selection should be on Female and
the label must display the current selection made by user'''

from IPython.display import display
import ipywidgets as widgets
y=widgets.Label()
x=widgets.RadioButtons(
    options=['Male','Female','Transgender'],
    value='Female',
    description='Gender',
    disabled=False
)
display(x,y)
pythonlink=widgets.link((x,'value'),(y,'value'))
```

Gender
- (●) Male
- ( ) Female
- ( ) Transgender

Male

In [44]:

```
'''43. Write a program to display a menu on the menu bar.'''
from tkinter import *
from tkinter.ttk import *
from time import strftime

# creating tkinter window
root = Tk()
root.title('Menu Bar Implementation')
root.geometry("600x500")

# Creating Menubar
menubar = Menu(root)

# Adding File Menu and commands
file = Menu(menubar, tearoff = 0)
menubar.add_cascade(label ='File', menu = file)
file.add_command(label ='New File        Ctrl+N', command = None)
file.add_command(label ='Open...         Ctrl+O', command = None)
file.add_command(label ='Save            Ctrl+S', command = None)
file.add_command(label ='Save As         Ctrl+Shift+S', command = None)
file.add_separator()
file.add_command(label ='Print         Ctrl+P', command = None)
file.add_command(label ='Recent Files...', command = None)
file.add_separator()
file.add_command(label ='Exit', command = root.destroy)
file.add_separator()
file.add_separator()
file.add_command(label ='Created By Gaurav', command = None)

# Adding Edit Menu and commands
edit = Menu(menubar, tearoff = 0)
menubar.add_cascade(label ='Edit', menu = edit)
edit.add_command(label ='Cut', command = None)
edit.add_command(label ='Copy', command = None)
edit.add_command(label ='Paste', command = None)
edit.add_command(label ='Select All', command = None)
edit.add_separator()
edit.add_command(label ='Find...', command = None)
edit.add_command(label ='Find again', command = None)

# Adding Search Menu and commands
```

```
search = Menu(menubar, tearoff = 0)
menubar.add_cascade(label ='Search', menu = search)

# Adding Help Menu
help_ = Menu(menubar, tearoff = 0)
menubar.add_cascade(label ='Help', menu = help_)
help_.add_command(label ='Tk Help', command = None)
help_.add_command(label ='Demo', command = None)
help_.add_separator()
help_.add_command(label ='About Tk', command = None)

# Adding name Menu and commands
user = Menu(menubar, tearoff = 0)
menubar.add_cascade(label ='Contact Gaurav', menu = user)
user.add_command(label = "contact",command=None)
user.add_command(label = "mail",command=None)
user.add_command(label = "terminate",command=root.destroy)



# display Menu
root.config(menu = menubar)
mainloop()
```

In [45]:

```
'''44. Write a NumPy program to create an array of (3, 4)
shape, multiply every element value by 3 and display the new array.'''
import numpy as np
arr1=np.arange(1,13)
arr1=arr1.reshape(3,4)
print("Array of shape (3,4)--->>>\n",arr1)
print("\n")
newarr=arr1*3
print("Array after multiplying every element by e --->>>>\n",newarr)
```

```
Array of shape (3,4)--->>>
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]


Array after multiplying every element by e --->>>>
 [[ 3  6  9 12]
 [15 18 21 24]
 [27 30 33 36]]
```

In [46]:

```
'''45. Write a NumPy program to compute the multiplication of two given matrixes.'''

import numpy as np
mat1 = [[1, 0, 1], [0, 1, 1],[1,1,1]]
mat2 = [[1, 2], [3, 4],[5,6]]
#Showing origianl Matrix
print("original matrix:")
print(mat1)
print(mat2)
result = np.dot(mat1,mat2)
print("Result of the said matrix multiplication:")
print(result)
```

```
original matrix:
[[1, 0, 1], [0, 1, 1], [1, 1, 1]]
[[1, 2], [3, 4], [5, 6]]
Result of the said matrix multiplication:
[[ 6  8]
 [ 8 10]
 [ 9 12]]
```

In [47]:

```
'''46. Write a Program to create a series from a list, numpy array and dict'''

import pandas as pd
import numpy as np

#sample List
mylist = ['Dream','ii','Do','it']
list2Series = pd.Series(mylist,index=['a','b','c','d'])
print("Creating Series from list : ")
print("Original List : ",mylist)
print(list2Series)
print('---'*10+">>>>>>>>>>"+'---'*10)
#Sample Numpy Array
myarr = np.array(['T','E','S','L','A','O'])
arr2Series = pd.Series(myarr)
print("Creating Series from Numpy Array : ")
print("Original Array : ",myarr)
print(arr2Series)
print('---'*10+">>>>>>>>>>"+'---'*10)
#Sample Dictionary
mydict= {'Russia' : 1, 'Canada' : 2, 'US' : 3, 'China' : 4, 'Brazil' : 5, 'India' :7}
dict2series = pd.Series(mydict,index=['a','b','c','d','e','f'])
print("Creating Series from Dictionary : ")
print("Original Dictionary : ",mydict)
print(dict2series)
print('---'*10+">>>>>>>>>>"+'---'*10)
print('---'*10+">>>>>>>>>>"+'---'*10)
```

```
Creating Series from list :
Original List :  ['Dream', 'ii', 'Do', 'it']
a   Dream
b      ii
c      Do
d      it
dtype: object
------------------------------>>>>>>>>>>------------------------------
Creating Series from Numpy Array :
Original Array :  ['T' 'E' 'S' 'L' 'A' 'O']
0   T
1   E
2   S
3   L
4   A
5   O
```

```
dtype: object
------------------------------>>>>>>>>>>------------------------------
Creating Series from Dictionary :
Original Dictionary :  {'Russia': 1, 'Canada': 2, 'US': 3, 'China': 4, 'Brazil': 5, 'India': 7}
a   NaN
b   NaN
c   NaN
d   NaN
e   NaN
f   NaN
dtype: float64
------------------------------>>>>>>>>>>------------------------------
------------------------------>>>>>>>>>>------------------------------
```

In [49]:

```
'''47. Write a Program to convert a numpy array to a dataframe of given shape.'''
import pandas as pd
import numpy as np
myarr = np.array([[10,20,30,20,10],
                  [40,50,60,50,40],
                  [70,80,90,80,70]])
#Converting into Dataframe
df = pd.DataFrame(data = myarr)
print(df)
```

```
   0  1  2  3  4
0 10 20 30 20 10
1 40 50 60 50 40
2 70 80 90 80 70
```

In [50]:

```
'''48.Write a program to count number of missing values in each column.'''
import pandas as pd
import numpy as np
def main():

   # List of Tuples
   students = [ ('jack', np.NaN, 'Sydeny' , 'Australia') ,
            ('Riti', np.NaN, 'Delhi' , 'India' ) ,
            ('Vikas', 31, np.NaN , 'India' ) ,
            ('Neelu', 32, 'Bangalore' , 'India' ) ,
            ('John', 16, 'New York' , 'US') ,
            ('John' , 11, np.NaN, np.NaN ) ,
           (np.NaN , np.NaN, np.NaN, np.NaN )
            ]

   #Create a DataFrame object
   dfObj = pd.DataFrame(students, columns = ['Name' , 'Age', 'City' , 'Country'])
   print("Original Dataframe" , dfObj, sep='\n')
   print("Check NaN in Dataframe" , dfObj.isnull(), sep='\n')

   print("***Count all NaN in a DataFrame (both columns & Rows)***")

   print("Total NaN in Dataframe" , dfObj.isnull().sum().sum(), sep='\n')

   print("***Count NaN in each column of a DataFrame***")

   print("Nan in each columns" , dfObj.isnull().sum(), sep='\n')

   print("***Count NaN in each row of a DataFrame***")

   for i in range(len(dfObj.index)) :
      print("Nan in row ", i , " : " ,  dfObj.iloc[i].isnull().sum())


if __name__ == '__main__':
   main()
```

```
Original Dataframe
   Name  Age     City   Country
0  jack  NaN   Sydeny  Australia
1  Riti  NaN    Delhi     India
2  Vikas 31.0     NaN     India
```

```
3  Neelu  32.0  Bangalore     India
4   John  16.0   New York        US
5   John  11.0        NaN       NaN
6    NaN   NaN        NaN       NaN
Check NaN in Dataframe
    Name    Age   City  Country
0  False   True  False    False
1  False   True  False    False
2  False  False   True    False
3  False  False  False    False
4  False  False  False    False
5  False  False   True     True
6   True   True   True     True
***Count all NaN in a DataFrame (both columns & Rows)***
Total NaN in Dataframe
9
***Count NaN in each column of a DataFrame***
Nan in each columns
Name       1
Age        3
City       3
Country    2
dtype: int64
***Count NaN in each row of a DataFrame***
Nan in row  0  :  1
Nan in row  1  :  1
Nan in row  2  :  1
Nan in row  3  :  0
Nan in row  4  :  0
Nan in row  5  :  2
Nan in row  6  :  4
```

In [54]:

```
'''49. Write a program to replace missing values in a column of a
dataframe by the mean value of that column'''
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,70012,np.nan,70013],
'purch_amt':[150.5,np.nan,65.26,110.5,948.5,np.nan,5760,1983.43,np.nan,250.45, 75.29,3045.6],
'sale_amt':[10.5,20.65,np.nan,11.5,98.5,np.nan,57,19.43,np.nan,25.45, 75.29,35.6],
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17','2012-09-10','2012-07-27','2012-(
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,np.nan]})
print("Original Orders DataFrame:")
print(df)
print("Using median in purch_amt to replace NaN:")
df['purch_amt'].fillna(df['purch_amt'].median(), inplace=True)
print(df)
print("Using mean to replace NaN:")
df['sale_amt'].fillna(int(df['sale_amt'].mean()), inplace=True)
print(df)
```

```
Original Orders DataFrame:
    ord_no  purch_amt  sale_amt   ord_date  customer_id  salesman_id
0  70001.0     150.50     10.50 2012-10-05       3002       5002.0
1      NaN        NaN     20.65 2012-09-10       3001       5003.0
2  70002.0      65.26        NaN        NaN       3001       5001.0
3  70004.0     110.50     11.50 2012-08-17       3003          NaN
4      NaN     948.50     98.50 2012-09-10       3002       5002.0
5  70005.0        NaN        NaN 2012-07-27       3001       5001.0
6      NaN    5760.00     57.00 2012-09-10       3001       5001.0
7  70010.0    1983.43     19.43 2012-10-10       3004          NaN
8  70003.0        NaN        NaN 2012-10-10       3003       5003.0
9  70012.0     250.45     25.45 2012-06-27       3002       5002.0
10     NaN      75.29     75.29 2012-08-17       3001       5003.0
11 70013.0    3045.60     35.60 2012-04-25       3001          NaN
Using median in purch_amt to replace NaN:
    ord_no  purch_amt  sale_amt   ord_date  customer_id  salesman_id
0  70001.0     150.50     10.50 2012-10-05       3002       5002.0
1      NaN     250.45     20.65 2012-09-10       3001       5003.0
2  70002.0      65.26        NaN        NaN       3001       5001.0
3  70004.0     110.50     11.50 2012-08-17       3003          NaN
4      NaN     948.50     98.50 2012-09-10       3002       5002.0
5  70005.0     250.45        NaN 2012-07-27       3001       5001.0
6      NaN    5760.00     57.00 2012-09-10       3001       5001.0
7  70010.0    1983.43     19.43 2012-10-10       3004          NaN
8  70003.0     250.45        NaN 2012-10-10       3003       5003.0
```

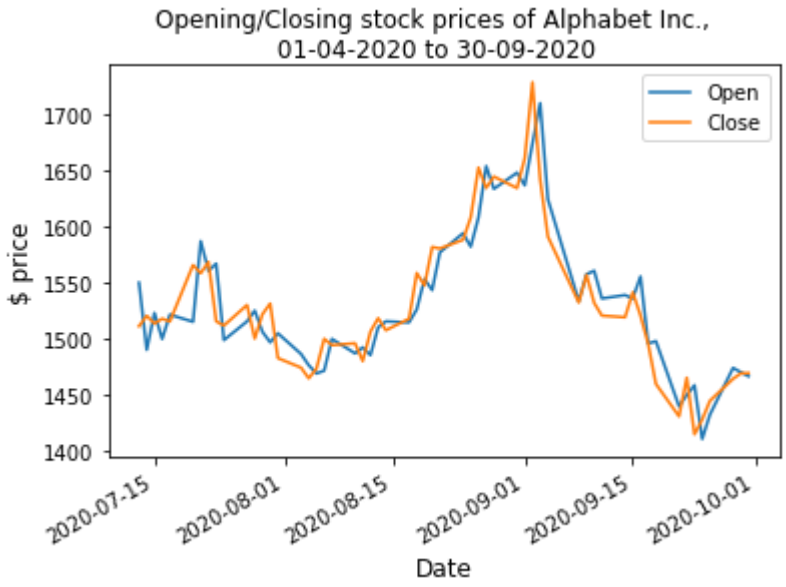
```
9  70012.0    250.45    25.45  2012-06-27       3002      5002.0
10     NaN     75.29    75.29  2012-08-17       3001      5003.0
11 70013.0   3045.60    35.60  2012-04-25       3001         NaN
Using mean to replace NaN:
    ord_no  purch_amt  sale_amt   ord_date  customer_id  salesman_id
0  70001.0     150.50    10.50  2012-10-05       3002      5002.0
1      NaN     250.45    20.65  2012-09-10       3001      5003.0
2  70002.0      65.26    39.00         NaN       3001      5001.0
3  70004.0     110.50    11.50  2012-08-17       3003         NaN
4      NaN     948.50    98.50  2012-09-10       3002      5002.0
5  70005.0     250.45    39.00  2012-07-27       3001      5001.0
6      NaN    5760.00    57.00  2012-09-10       3001      5001.0
7  70010.0    1983.43    19.43  2012-10-10       3004         NaN
8  70003.0     250.45    39.00  2012-10-10       3003      5003.0
9  70012.0     250.45    25.45  2012-06-27       3002      5002.0
10     NaN     75.29    75.29  2012-08-17       3001      5003.0
11 70013.0    3045.60    35.60  2012-04-25       3001         NaN
```

In [55]:

```
'''50. Write a Pandas program to create a line plot of the opening,
closing stock prices of Alphabet Inc. between two specific dates.
Use the alphabet_stock_data.csv file to extract data.'''

import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("alphabet_stock_data.csv")
start_date = pd.to_datetime('2020-4-1')
end_date = pd.to_datetime('2020-09-30')
df['Date'] = pd.to_datetime(df['Date'])
new_df = (df['Date']>= start_date) & (df['Date']<= end_date)
df2 = df.loc[new_df]
plt.figure(figsize=(10,10))
df2.plot(x='Date', y=['Open', 'Close']);
plt.suptitle('Opening/Closing stock prices of Alphabet Inc.,\n 01-04-2020 to 30-09-2020', font
plt.xlabel("Date",fontsize=12, color='black')
plt.ylabel("$ price", fontsize=12, color='black')
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



In [ ]:

```
'''*****Thank You.*****'''
```