Ishaa Abdul

119A1025 SE C

**Experiment No.2**

**Aim:** WAP creating functions, classes and objects using python. Demonstrate inheritance and exception handling.

**Code:**

1. **Creating employee class and creating records.**

class emp(object):

  #define constructor

  def \_\_init\_\_(self,eid=101):

    self.eid=eid

  def get\_details(self):

    self.name=input("Enter name of emp-")

    self.sal=int(input("Enter salary-"))

    self.des=input("Enter designation-")

  def put\_details(self):

    print("Employee details are:",self.eid,self.name,self.sal,self.des)

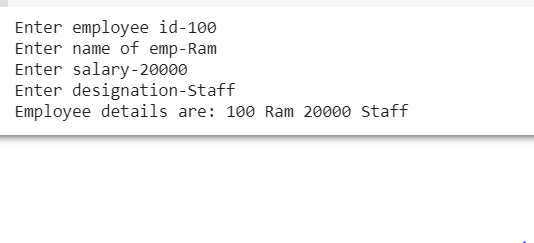
id=int(input("Enter employee id-"))

e1=emp(id)

e1.get\_details()

e1.put\_details()

**Output:**

****

1. **Program showing operator overloading.**

class fraction(object):

  def \_\_init\_\_(self,n=0,d=1):

    self.n=n

    self.d=d

  #magic addition self=f1(n,d) and other=f2(n,d)

  def \_\_add\_\_(self,other):

    n1=(self.n\*other.d)+(other.n\*self.d)

    d1=self.d\*other.d

    return n1,d1

  def \_\_sub\_\_(self,other):

    n1=(self.n\*other.d)-(other.n\*self.d)

    d1=self.d\*other.d

    return n1,d1

  def \_\_mul\_\_(self,other):

    n1=(self.n\*other.n)

    d1=(self.d\*other.d)

    return n1,d1

  def \_\_floordiv\_\_(self,other):

    n1=self.n\*other.d

    d1=self.d\*other.n

    return n1,d1

def main():

  f1=fraction(5,1)

  f2=fraction(2,3)

  print("\nAddition of fractions")

  f3=f1+f2

  print(f3)

  print("\nSubtraction of fractions")

  f3=f1-f2

  print(f3)

  print("\nMultiplication of fractions")

  f3=f1\*f2

  print(f3)

  print("\nDivision of fractions")

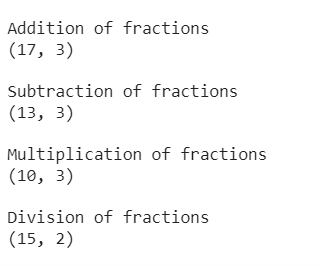
  f3=f1//f2

  print(f3)

if \_\_name\_\_=="\_\_main\_\_":

  main()

**Output:**



1. **Program to demonstrate single inheritance between class Student and Teacher.**

class Teacher(object):

  def \_\_init\_\_(self,id,name,address):

    self.id=id

    self.name=name

    self.address=address

  def display(self):

    print("Teachers details are-",self.id,self.name,self.address)

class Student(Teacher):

  def \_\_init\_\_(self,id,name,address,marks\_sub):

    super().\_\_init\_\_(id,name,address)

    self.marks=marks\_sub

  def display1(self):

    super().display()

    print("Subject marks ",self.marks)

def main():

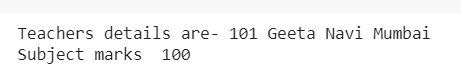
  obj=Student(101,'Geeta','Navi Mumbai',100)

  obj.display1()

if \_\_name\_\_=="\_\_main\_\_":

  main()

**Output:**



1. **Defining an abstract class with abstract methods**

from abc import ABC, abstractmethod

class Shape(object):

  @abstractmethod

  def area(self):

    pass

class Triangle(Shape):

  def get\_dimension(self):

    self.l=int(input("Enter length(triangle)-"))

    self.b=int(input("Enter breadth(triangle)-"))

  @abstractmethod

  def area(self):

    self.a=0.5\*self.l\*self.b

  def display(self):

    print("Area of triangle-",self.a)

class Rectangle(Shape):

  def get\_dimension(self):

    self.l=int(input("Enter length(rectangle)-"))

    self.b=int(input("Enter breadth(rectangle)-"))

  @abstractmethod

  def area(self):

    self.a=self.l\*self.b

  def display(self):

    print("Area of rectangle-",self.a)

t=Triangle()

t.get\_dimension()

t.area()

t.display()

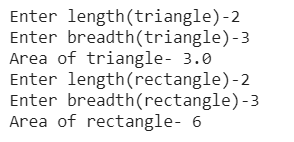
t=Rectangle()

t.get\_dimension()

t.area()

t.display()

**Ouput:**



1. **Program to demonstrate exception handling.**

class MyException(Exception):

  def \_\_init\_\_(self,arg):

    self.msg=arg

def check(id,name,age):

  if(age<20):

    raise MyException('Underage to create bank account')

  else:

    print("Person with id {},name {} and age{}".format(id,name,age))

def main():

  id=int(input("Enter id-"))

  name=input("Enter name-")

  age=int(input("Enter age-"))

  try:

    check(id,name,age)

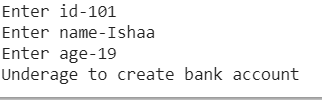
  except MyException as e:

    print(e)

if \_\_name\_\_=="\_\_main\_\_":

  main()

**Output:**



**Conclusion:** We have successfully implemented the concept of inheritance, exception handling, functions and classes.