Example
1

```
from abc import ABC, abstractmethod
class Animal(ABC):
    @abstractmethod
    def eat1(self):
        pass
    @abstractmethod
    def eat2(self):
        pass

class Tiger(Animal):
       def eat1(self):
              print("Tiger implementation ...")
class lion(Tiger):
       def eat2(self):
              print("lion implementation ...")
t = lion()
t.eat1()
t.eat2()
# Output: Tiger implementation ...
# lion implementation ...


#   Example 2
from abc import ABC, abstractmethod
class AbstractClassExample(ABC):
    def __init__(self,value):
        self.value=value
        super().__init__()
    @abstractmethod
    def do_something(self):
        pass
class DoAdd(AbstractClassExample):
    def do_something(self):
        return self.value+42
class DoMul(AbstractClassExample):
    def do_something(self):
         return self.value*42
x=DoAdd(10)
y=DoMul(10)
print(x.do_something())
print(y.do_something())
# Output-- 52 & 420
```

```
# Example 3
def status(age):
    if age<0:
        raise ValueError("Only positive integrs are allowed")
    if age>22:
        print('eligible for mrg')
    else:
        print('not eligible for mrg, try again after sometime')
try:
    num=int(input('enter your age: '))
    status(num)
except ValueError:
    print('Only positive integrs are allowed ........')
finally:
    print('finally block')
#  Output--
# enter your age: 20
# not eligible for mrg, try again after sometime
# finally block
# enter your age: 23
# eligible for mrg
# finally block

#  Example 4
class NegativeAgeException(RuntimeError):
    def __init__(self,age):
        super().__init__()
        self.age=age
    def status(age):
        if age<0:
            raise NegativeAgeException('only positive integers are allowed')
        if age>22:
            print('eligible for mrg')
        else:
            print('not eligible for mrg.......')
try:
    num=int(input('enter your age:'))
    status(num)
except NegativeAgeException:
    print('only positive integers are allowed')
except:
    print('something is wrong')

# Example 5
```

```
class TooYoungException(Exception):
    def __init__(self,age):
        self.age=age
class TooOldException(Exception):
    def __init__(self,age):
        self.age=age
try:

    age=int(input('enter age: '))
    if age<18:
        raise TooYoungException('please wait sometime')
    elif age>65:
        raise TooOldException('your age is old')
    else:
        print('we will find one girl soon')
except TooYoungException as e:
    print('please wait....')
except TooOldException as e:
    print('you are old...')
# Output-- enter age: 20
# we will find one girl soon

#  Example 6
try:
    print('outer try block')
    n=int(input('enter a number: '))
    print(10/n)
    try:
        print('inner try')
        print('anu'+'preet')
    except TypeError:
        print('Hello')
    else:
        print('inner no exception')
except ArithmeticError:
        print(10/5)
else:
        print('outer no exception')
finally: print('finally block')
# Output--enter a number: 2
# 5.0
# inner try
# anupreet
# inner no exception
# outer no exception
# finally block
```

```
#  Example 7
class Person(object):
    def __init__(self,first,last):
        self.firstname=first
        self.lastname=last
    def Name(self):
        return self.firstname+""+self.lastname
class Employee(Person):
    def __init__(self,first,last,staffnum):
        super().__init__(first,last)
        Person.__init__(self,first,last)
        self.staffnumber=staffnum
    def GetEmployee(self):
        return self.Name()+","+self.staffnumber
x=Person("komal","addanki")
y=Employee("komal","addanki","111")
print(x.Name())
print(y.GetEmployee())
# Output -- komaladdanki
# komaladdanki,111

# Example 8
class Person:
    def __init__(self,first,last):
        self.firstname=first
        self.lastname=last
    def __str__(self):
        return self.firstname+""+self.lastname
class Employee(Person):
    def __init__(self,first,last,id):
        super().__init__(first,last)
        self.id=id
    def __str__(self):
        return super().__str__()+""+self.id
x=Person("kamalpreet","gurpreet")
y=Employee("kamalpreet","gurpreet","111")
print(x)
print(y)
# Output--kamalpreetgurpreet
# kamalpreetgurpreet111

#  Exapmle 9
class Vehicle:
    def __del__(self):
        print("vehicle object destroyed")
```

```
        print(10/0)
v=Vehicle()
del v
# Output--    print(10/0) ZeroDivisionError: division by zero

# Exapmle 10
from unicodedata import name
class customer:
    def __init__(self,name,bal=0.0):
        self.name=name
        self.bal=bal
    def deposit(self,amount):
        self.bal=self.bal+amount
    def withdraw(self,amount):
        if amount>self.bal:
            raise RuntimeError('withdraw amount is more than balance')
        else:
            self.bal=self.bal-amount
    def remaining(self):
        return self.bal
c=customer("komal",10000)
damt=int(input('enter amount to deposit'))
c.deposit(damt)
amt=int(input('enter amount to withdraw'))
c.withdraw(amt)
print(c.remaining())
# Output-- enter amount to deposit20000
# enter amount to withdraw5100
# 24900
```