1.Write a python program to find the largest of 2 numbers.

a=int(input("Enter the first num:"))

b=int(input("Enter the second num:"))

if(a>b):

print(a,"is greater than",b);

else:

print(b,"is greater than",a);

OUTPUT:

Enter the first num:4

Enter the second num:3

4 is greater than 3

-----------------------------------------------------------------------------------------

2.Write a python program to check whether the given number is a palindrome.

a=int(input("Enter a num:"))

rev=0

value=a

while(a!=0):

rem=a%10

rev=(rev\*10)+rem

a=a//10

print(rev)

if(rev==value):

print(rev,"is a palindrome!")

else:

print(rev,"is not a palindrome!")

OUTPUT:

Enter a num:232

232

232 is a palindrome!

-----------------------------------------------------------------------------------------

3.Write a python program to display Student marks card in proper format.

regno=int(input("Enter the student register no:"))

name=input("Enter the student name:")

m1=int(input("Enter the marks in maths:"))

m2=int(input("Enter the marks in english:"))

m3=int(input("Enter the marks in physics:"))

total=m1+m2+m3

avg=total/3

if(m1<40 or m2<40 or m3<40):

res="FAIL"

elif(avg>=90):

res="DISTINCTION"

elif(avg>=80):

res="FIRST CLASS"

elif(avg>=70):

res="SECOND CLASS"

elif(avg>=60):

res="THIRD CLASS"

else:

res="PASS"

print("-------------------STUDENT MARKS DETAILS------------------------")

print("RegNo\tName\tMaths\tEnglish\tPhysics\tTotal\tAverage\tResult")

print(regno," ",name," ",m1," ",m2," ",m3," ",total," ",avg," ",res)

Enter the student register no:101

Enter the student name:Ria

Enter the marks in maths:99

Enter the marks in english:99

Enter the marks in physics:99

-------------------STUDENT MARKS DETAILS------------------------

RegNo Name Maths English Physics Total Average Result

101 Ria 99 99 99 297 99.0 DISTINCTION

-----------------------------------------------------------------------------------------

4.Write a python program to print Prime numbers from 2 to 100.

for num in range(2,100):

count=0

for i in range(1,num+1):

if(num%i==0):

count+=1

if(count<=2):

print(num,end=" ")

OUTPUT:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

-----------------------------------------------------------------------------------------

5.Write a python program to print Perfect numbers.

n=int(input("Enter a number:"))

sum=0

for i in range(1,n):

if(n%1==0):

sum+=i

if(sum==n):

print(n,"is a perfect number")

else:

print(n,"is not a perfect number")

OUTPUT:

Enter a number:6

6 is a perfect number

------------------------------------------------------------------------------------------

6.Write a python program to generate n Fibonacci numbers. Store that in a list and display.

n=int(input("Enter a number:"))

f1=0

f2=1

f3=1

li=[]

li.append(f1)

li.append(f2)

if(n>1):

for i in range(1,n):

f3=f1+f2

f1=f2

f2=f3

li.append(f3)

print(li)

OUTPUT

Enter a number:5

[0, 1, 1, 2, 3, 5]

-----------------------------------------------------------------------------------------------

7.Write a python program to Remove duplicates from a list.

li=[]

n=int(input("Enter a number:"))

print("Enter the numbers along with duplicates:")

for i in range(n):

i=int(input())

li.append(i)

s1=set(li)

print("Before:",li)

l2=list(s1)

print("After:",l2)

OUTPUT:

Enter a number:5

Enter the numbers along with duplicates:

3

3

4

2

1

Before: [3, 3, 4, 2, 1]

After: [1, 2, 3, 4]

----------------------------------------------------------------------------------

8.Write a python program to satisfy the below business requirements:

1. List number of students who enrolled in python course

2. List number of students who enrolled in java course only

3. List number of students who enrolled in python and java course

4. List number and name of students who enrolled in python and java course

5. List number and name of students who enrolled in either python or java course but not both

6.List number and name of students who enrolled in either python or java course

python\_students=["Sia","Ria","Raj","Rahul","Dev","Tia","Naina"]

java\_students=["Ria","Nisha","Raj","Dev","Samay","Naina","Aditi"]

python\_enrolled\_count = len(python\_students)

print(f"Number of students enrolled in Python course: {python\_enrolled\_count}")

java\_only\_students = [student for student in java\_students if student not in python\_students]

java\_only\_count = len(java\_only\_students)

print(f"Number of students enrolled in Java course only: {java\_only\_count}")

both\_courses\_students = [student for student in python\_students if student in java\_students]

both\_courses\_count = len(both\_courses\_students)

print(f"Number of students enrolled in both Python and Java courses: {both\_courses\_count}")

print(f"Students enrolled in both Python and Java courses: {', '.join(both\_courses\_students)}")

either\_but\_not\_both\_students = [student for student in (python\_students + java\_students) if student not in both\_courses\_students]

either\_but\_not\_both\_count = len(either\_but\_not\_both\_students)

print(f"Number of students enrolled in either Python or Java but not both: {either\_but\_not\_both\_count}")

print(f"Students enrolled in either Python or Java but not both: {', '.join(either\_but\_not\_both\_students)}")

either\_course\_students = list(set(python\_students + java\_students)) # Remove duplicates using set

either\_course\_count = len(either\_course\_students)

print(f"Number of students enrolled in either Python or Java course: {either\_course\_count}")

print(f"Students enrolled in either Python or Java course: {', '.join(either\_course\_students)}")

---------------------------------------------------------------------------------

9.Write a python program for the given scenarion.

At an airport a traveller is allowed entry into the flight only if he clears the following checks:

a) Baggage Check

b) Immigration Check

c) Security Check

def chk\_baggage(baggage\_amt):

if baggage\_amt>=0 and baggage\_amt<=40:

return True

else:

return False

def chk\_immigration(exp\_year):

if exp\_year>=2001 and exp\_year<=2025:

return True;

else:

return False

def chk\_noc(noc\_status):

if noc\_status=="Yes":

return True

else:

return False

baggage\_amt=int(input("Enter the baggage amount:"))

exp\_year=int(input("Enter the year:"))

noc=input("Enter the noc status:")

if chk\_baggage(baggage\_amt) and chk\_immigration(exp\_year) and chk\_noc(noc\_status):

print("Allowed to board")

else:

print("Not allowed")

OUTPUT:

Enter the baggage amount:34

Enter the year:2020

Enter the noc status:yes

Allowed to board

----------------------------------------------------------------------------------

10.Write a function program to perform Binary search from the list. If the element is found,its location should be returned by the method,otherwise it should return -1.

def binsearch(list1,key):

low=0

high=len(list1)-1

while(low<=high):

mid=(low+high)//2

if list1[mid]==key:

return mid

elif list1[mid]<key:

low=mid+1

else:

high=mid-1

return -1

n=int(input("Enter the no.of elements:"))

list1=[]

for i in range(n):

element=int(input(f"Enter the elements{i+1}:"))

list1.append(element)

list1=sorted(list1)

print("After sorting:",list1)

key=int(input("Enter the search element:"))

res=binsearch(list1,key)

if res==-1:

print("Element not found")

else:

print("Element found at index:",res)

OUTPUT:

Enter the no.of elements:5

Enter the elements1:5

Enter the elements2:3

Enter the elements3:1

Enter the elements4:2

Enter the elements5:4

After sorting: [1, 2, 3, 4, 5]

Enter the search element:4

Element found at index: 3

----------------------------------------------------------------------------

11.Write a python program to find the Factorial and Fibonacciof a number using recursion.

def fact(n):

if n==0 or n==1:

return 1

return n\*fact(n-1)

def fib(n):

if n==0:

return 0

elif(n==1):

return 1

return fib(n-1)+fib(n-2)

n=int(input("Enter a number:"))

print(f"Factorial of {n} is {fact(n)}")

print(f"Fibonacci series of {n} is")

for i in range(n):

print(fib(i),end=" ")

OUTPUT:

Enter a number:5

Factorial of 5 is 120

Fibonacci series of 5 is

0 1 1 2 3

-----------------------------------------------------------------------------

12. Write a python program to reverse string and check whether it is Palindrome using Recursion.

def reverse(text):

if(len(text))<=1:

return text

return reverse(text[1:])+text[0]

text=input("Enter the string:")

res=reverse(text)

print("Reversed String:",res)

if res==text:

print("Palindrome")

else:

print("Not a palindrome")

OUTPUT:

Enter the string:maam

Reversed String: maam

Palindrome

--------------------------------------------------------------------

13.Write a python program to sort the elements in the array using Bubble sort technique and display the elements in descending order.

def bubbsort(arr):

n=len(arr)

for i in range(n):

for j in range (0,n-i-1):

if(arr[j]<arr[j+1]):

arr[j],arr[j+1]=arr[j+1],arr[j]

arr=list(map(int,input("Enter the array elements:").split()))

print(arr)

bubbsort(arr)

print("Array in descending order:",arr)

OUTPUT:

Enter the array elements:2 5 1 9 7 3

[2, 5, 1, 9, 7, 3]

Array in descending order: [9, 7, 5, 3, 2, 1]

----------------------------------------------------------------------------

14.Consider the list of courses opted by a student "John" and available electives as a part of Student Management System.

courses=("Python Programming","RDBMS","Web Technology","Software Engg")

electives=("Business Intelligence","Big data Analysis")

Write a python program to satisfy the following business requirements:

a) List the no.of courses opted by student John.

b) List all the courses opted by student John.

c) John is also interested in elective courses mentioned above. Print the updated tuple including electives.

d) Check whether John is allowed to change his course from SE to Computer Network.

courses=("Python Programming","RDBMS","Web Technology","Software Engg")

electives=("Business Intelligence","Big data Analysis")

no\_of\_courses=len(courses)

print("No,of Coursers:",no\_of\_courses)

print("All courses opted by John:\n",tuple(courses))

print("All courses opted by John including electives:\n",tuple(courses+electives))

OUTPUT:

No,of Coursers: 4

All courses opted by John:

('Python Programming', 'RDBMS', 'Web Technology', 'Software Engg')

All courses opted by John including electives:

('Python Programming', 'RDBMS', 'Web Technology', 'Software Engg', 'Business Intelligence', 'Big data Analysis')

---------------------------------------------------------------------------------------------------------------------

15.Write a python script to generate and print a dictionary that contains a number(between 1 and n)in the form(x,x\*x).

n=int(input("Enter a number:"))

s\_dict={x:x\*x for x in range(1,n+1)}

print(f"Dictionary that contains number and their square roots from 1 to {n}\n:{s\_dict}")

OUTPUT:

Enter a number:8

Dictionary that contains number and their square roots from 1 to 8

:{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

---------------------------------------------------------------------------------------

16.Write a python program to Map 2 lists into dictionary.

n1=list(input("Enter list 1:"))

n2=list(input("Enter list 2:"))

res=zip(n1,n2)

print(dict(res))

OUTPUT:

Enter list 1:12345

Enter list 2:chari

{'1': 'c', '2': 'h', '3': 'a', '4': 'r', '5': 'i'}

------------------------------------------------------------------------------------------------

17.COnsider 10 student name and marks in one subject. Calculate the total and average of marks and find the top 3 scorers and also the average marks scored by all the students.

stud={"Sia":75,"Tina":80,"Rahul":54,"Raj":45,"Naina":99}

total\_m=sum(stud.values())

avg\_m=total\_m/len(stud)

print(f"Average marks is:{avg\_m:.2f}")

sorted\_stud=sorted(stud.items(),key=lambda x:x[1],reverse=True)

top3\_scrers=sorted\_stud[:3]

top3\_dict=dict(top3\_scrers)

print("Top 3 scorers:",top3\_dict)

OUTPUT:

Average marks is:70.60

Top 3 scorers: {'Naina': 99, 'Tina': 80, 'Sia': 75}

-------------------------------------------------------------------------------

18.Write a python program to check whether the given sets are Subset or Superset of each other.

set1=set(input("Enter set 1:"))

set2=set(input("Enter set 2:"))

print("Set 1 is:",set1)

print("Set 2 is:",set2)

if set1.issubset(set2):

print("Set 1 is a sub set of Set 2!")

elif set1.issuperset(set2):

print("Set 1 is a super set of Set 2!")

else:

print("Neither a subset nor a super set")

OUTPUT:

Enter set 1:123456

Enter set 2:3456

Set 1 is: {'4', '5', '1', '2', '3', '6'}

Set 2 is: {'4', '5', '3', '6'}

Set 1 is a super set of Set 2!

----------------------------------------------------------------------------------------

19.Write a python program to perform the following:

a) Reverse in descending order,union in ascending order,intersection in ascending order using the input present in the file.

b) Print the output as well as save the file in a new file with file name as 'output\_program\_<<program number>>\_<<register number>>\_<<year>>\_<<month>>\_<<date>>.txt'.

c) Output of reverse,union,intersection should be printed in a newline.

d) Copy the program file from the existing file destination to location whereyour input and output file is present.

import shutil

from datetime import datetime

def read\_number(file):

with open(file,"r")as f:

return list(map(int,f.read().split()))

def save\_output(output\_file,reverse\_desc,union\_asc,intersection\_asc):

with open(output\_file."w")as f:

f.write(f"Reverse in descending order:{reverse\_desc}")

f.write(f"Union in ascending order:{union\_asc}")

f.write(f"Intersection in ascending order:{intersection\_asc}")

def main():

input\_file="digits.txt"

set1=set(read\_number(input\_file))

print(set1)

set1={2,3,4}

print(set2)

reverse\_desc=sorted(set1,reverse=True)

union\_asc=sorted(set1.union(set2))

intersection\_asc=sorted(set(set1).intersection(set2))

now=datetime.now()

output\_file=f"output\_program\_187\_{now.year}\_{now.month}\_{now.day}.txt"\

print(f"Reverse:",reverse\_desc)

print("Union:",union\_asc)

print("Intersection:",intersection\_asc)

save\_output(output\_file,reverse\_desc,union\_asc,intersection\_asc)

if \_\_name\_\_=="\_\_main\_\_":

main()

Text File Name: digits.txt

9 5 2 8

2 3 4

---------------------------------------------------------------------------------------------

20. There is a file with several text messages.Each message is in its own line.Write a python program to count the no.of lines in the file and total no.of words contained in those messages.Assume the messages contain only alphabets and numbers.

def count\_line(file\_path):

try:

line\_count=0

word\_count=0

with open("file\_path",'r') as file:

for line in file:

line=line.strip()

if line:

line\_count+=1

words=line.split()

word\_count+=len(words)

print(f"Total no of lines:{line\_count}")

print(f"Total no.of words:{word\_count}")

except FileNotFounderror:

print(f"Error:The file{file\_path}does not exist")

except Exception as e:

print(f"An error occuredd:{e}")

file\_path="abc.txt"

count\_line(file\_path)

Text fule name:abc.txt:-

hello world

good

morning

OUTPUT:

Total number of lines: 3

Total number of words: 4

------------------------------------------------------------------------------------------

21.Write a class Distance with instance variables feet and inches.Include necessary methods.Test the class.

class Distance:

def \_\_init\_\_(self,feet,inches):

self.feet=feet

self.inches=inches

def add(self):

self.feet=self.feet+self.inches//12

self.inches=self.inches%12

print(self.feet,"Feet,",self.inches,"Inches")

f=int(input("Enter feet:"))

i=int(input("Entert inches:"))

d1=Distance(f,i)

d1.add()

OUTPUT:

Enter feet:13

Entert inches:18

14 Feet, 6 Inches

-------------------------------------------------------------------------------------------

22.Write a python program to illustrate multilevel inheritance Box(length,breadth,height)as the super class,BoxWeight(weight) and BoxShipment(cost) as the sub classes. Illustrate the use of super keywords,constructor and assign the value not zero.

class Box:

def \_\_init\_\_(self,length=1,breadth=1,height=1):

self.length=length

self.breadth=breadth

self.height=height

def display\_dimensions(self):

print(f"Dimensions(l\*b\*h):{self.length}\*{self.breadth}\*{self.height}")

class BoxWeight(Box):

def \_\_init\_\_(self,length=1,breadth=1,height=1,weight=1):

super().\_\_init\_\_(length,breadth,height)

self.weight= weight

def display\_weight(self):

print(f"Weight:{self.weight}kg")

class BoxShipment(BoxWeight):

def \_\_init\_\_(self,length=1,breadth=1,height=1,weight=1,cost=1):

super().\_\_init\_\_(length,breadth,height,weight)

self.cost=cost

def display\_shipment(self):

super().display\_dimensions()

super().display\_weight()

print(f"Cost of shipment:{self.cost}dollars")

bship=BoxShipment(10,5,8,20,100)

bship.display\_shipment()

OUTPUT:

Dimensions(l\*b\*h):10\*5\*8

Weight:20kg

Cost of shipment:100dollars

--------------------------------------------------------------------------------------

23.Create an abstract class Relation with abstract method to implement the basic relational operators(==,<>,>=)on two integers. Define class Numbers with two data fields(N1,N2) which extends class Relation and illustrtate in the main class.

from abc import ABC,abstractmethod

class relation(ABC):

@abstractmethod

def is\_greater(self,a:int,b:int)->bool:

pass

@abstractmethod

def is\_less(self,a:int,b:int)->bool:

pass

@abstractmethod

def is\_equal(self,a:int,b:int)->bool:

pass

class Number(relation):

def \_\_init\_\_(self,n1:int,n2:int):

self.n1=n1

self.n2=n2

def is\_greater(self,a:int,b:int)->bool:

return a>b

def is\_less(self,a:int,b:int)->bool:

return a<b

def is\_equal(self,a:int,b:int)->bool:

return a==b

def compare(self):

print(f"{self.n1}>{self.n2}:{self.is\_greater(self.n1,self.n2)}")

print(f"{self.n1}<{self.n2}:{self.is\_less(self.n1,self.n2)}")

print(f"{self.n1}=={self.n2}:{self.is\_equal(self.n1,self.n2)}")

def main():

n1=int(input("Enter first number:"))

n2=int(input("Enter the second number:"))

num=Number(n1,n2)

num.compare()

if \_\_name\_\_=="\_\_main\_\_":

main()

OUTPUT:Enter first number:34

Enter the second number:27

34>27:True

34<27:False

34==27:False

--------------------------------------------------------------------------------

24.Consider the scenario from Student Management System and perform the following:

a) Accept student\_id and validate whether it contains only digits.

b) If student\_id is valid, accept student\_name from the yser and validate whether it contains only alphabets.

c) If student\_name is valid, accept fee\_amount paud by the student

\* Decimal point is optional in fee\_amount.

\* Only 2 digits are allowed after decimal point.

d) If invalid data is entered in any of the above steps,display appropriate error messages. Else create an email\_id for student as student\_name@ABC.com. Where ABC is the name of the college. Assume there are no duplicate names.

e) Perform above validation using regular expression and print details of students: student\_id,student\_name,fee\_amount,amount,email\_id.

import re

def student\_ms():

try:

sid=input("Enter student is:")

if not re.fullmatch(r'\d+',sid):

print("Student id must contain only alphabets")

return

sname="Enter student name:"

if not re.fullmatch(r'[a-z z-a]+',sname):

print("Student name must contain only alphabets.")

return

fees=input("Enter fee amount upto 2 decimal place:")

if not re.fullmatch(r'\d+(\.\d{1,2})?',fees):

print("Fees must be valid number upto 2 decimal place:")

return

collegename="aloysius"

emailid="{sname.replace('',").lower()}@{collegename}.ac.in"

print(f"Student id:{sid}")

print(f"Student name:{sname}")

print(f"Student fees:{fees}")

print(s"Student email:{emailid}")

except Exception as e:

print(f"An error occured:{e}")

student\_ms()

-------------------------------------------------------------------------------

25.Write a python program to add few customer details into the database and retrieve the information and print it in systematic manner.

import sqlite3

def create\_table():

conn=sqlite3.connect('customers.db')

cursor=conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS customers

(id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,email TEXT,phone TEXT)''')

conn.commit()

conn.close()

def add\_customer(name,email,phone):

conn=sqlite3.connect('customers.db')

cursor=conn.cursor()

cursor.execute("INSERT INTO customers(name,email,phone)VALUES(?,?,?)",(name,email,phone))

conn.commit()

conn.close()

def retrieve\_customer():

conn=sqlite3.connect('customers.db')

cursor=conn.cursor()

cursor.execute("SELECT\*FROM customers")

customers=cursor.fetchall()

conn.close()

return customers

create\_table()

add\_customer("Alice","alice@gamal.com","457877788")

add\_customer("Raj","raj@exam.com","7800954323")

customers=retrieve\_customer()

for customer in customers:

print(f"ID:{customer[0]},Name:{customer[1]},Email:{customer[2]},Phone:{customer[3]}")

OUTPUT:

ID:1,Name:Alice,Email:alice@gamal.com,Phone:457877788

ID:2,Name:Raj,Email:raj@exam.com,Phone:7800954323

-------------------------------------------------------------------------

26.Write a python program to create a Bank Application. Use all the properties needed to complete the application. Ask on request from the user what information they need to check. Last five transactions has to be kept in DB. User can retrieve the data lika account details, personal details, last five transactions,etc.

import sqlite3

class BankAccount:

def \_\_init\_\_(self,db="bank.db"):

self.conn=sqlite3.connect(db)

cursor=self.conn.cursor()

cursor.execute("DROP TABLE IF EXISTS accounts")

self.conn.execute("CREATE TABLE accounts(accno INTEGER PRIMARY KEY,name TEXT,balance REAL,address TEXT,phone TEXT);")

cursor.execute("DROP TABLE IF EXISTS transactions")

self.conn.execute("CREATE TABLE transactions(id INTEGER PRIMARY KEY AUTOINCREMENT,accno INTEGER,details TEXT);")

def execute(self,query,params=(),fetch=False):

cursor=self.conn.cursor()

if params:

cursor.execute(query,params)

else:

cursor.execute(query)

self.conn.commit()

if fetch:

return cursor.fetchall()

def create\_account(self,accno,name,balance,address,phone):

self.execute("INSERT INTO accounts VALUES(?,?,?,?,?)",(accno,name,balance,address,phone))

print("Account Created")

def deposite(self,accno,amount):

self.execute("UPDATE accounts SET balance=balance+? WHERE accno=?",(amount,accno))

self.execute("INSERT INTO transactions(accno,details)VALUES(?,?)",(accno,f"Deposited:{amount}"))

print("Amount Credited")

def withdraw(self,accno,amount):

balance=self.execute("SELECT balance FROM accounts WHERE accno=?",(accno),True)[0][0]

if balance>=amount:

self.execute("UPDATE accounts SET balance=balance-? WHERE accno=?",(amount,accno))

self.execute("INSERT INTO transactions(accno,details)VALUES(?,?)",(accno,f"Withdraw:{amount}"))

print("Amount Debited")

else:

print("Insufficient Balance")

def show\_details(self,accno,personal=False):

query="SELECT\*FROM accounts WHERE accno=?"if not personal else"SELECT name,addess,phone FROM accounts WHERE accno=?"

row=self.execute(query,(accno,),True)

for r in row:

print(row)

def show\_transactions(self,accno):

transactions=self.execute("SELECT details FROM transactions WHERE accno=? ORDER BY id DESC LIMIT 5",(accno),True)

for t in transactions:

print(t[0])

def close(self):

self.conn.close()

def main():

bank=BankAccount()

while True:

print("\n1.Create Account\n2.Deposite\n3.Withdraw\n4.Account Details\n5.Personal Details\n6.Transactions\n7.Exit")

ch=int(input("Enter your choice:"))

if ch==1:

bank.create\_account(int(input("Accno:")),input("Name:"),float(input("Balance:")),input("Address:"),input("Phone:"))

elif ch==2:

bank.deposite(int(input("Accno:")),float(input("Amount:")))

elif ch==3:

bank.withdraw(int(input("Accno:")),float(input("Amount:")))

elif ch==4:

bank.show\_details(int(input("Accno:")))

elif ch==5:

bank.show\_details(int(input("Accno:")),personal=True)

elif ch==6:

bank.show\_transactions(int(input("Accno:")))

elif ch==7:

bank.close()

break

if \_\_name\_\_=="\_\_main\_\_":

main()

------------------------------------------------------------------------

27. Write a python program to demonstrate the concept of built in Exception and User defined Exception.

class zeroDivisionError(Exception):

pass

def divide\_num(num1,num2):

if(num2==0):

raise zeroDivisionError("Cannot be divided by 0")

else:

return num1/num2

try:

result=divide\_num(10,2)

print(f"Result:{result}")

except zeroDivisionError as e:

print(e)

try:

result=divide\_num(10,2)

print(f"Result:{result}")

except zeroDivisionError as e:

print(e)

OUTPUT:

Result:5.0

Cannot be divided by 0

-------------------------------------------------------------------------------------