**Write a program to Fibonacci series in python**

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

a=0

b=1

c=0

count=1

print("Fibonacci series:”, end=" ")

while(count<=n):

print(c,end=" ")

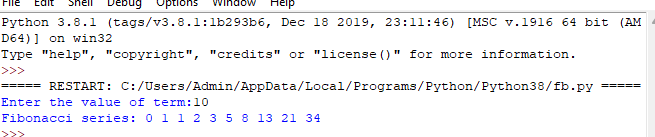
count+=1

a=b

b=c

c=a+b

**Output**:-



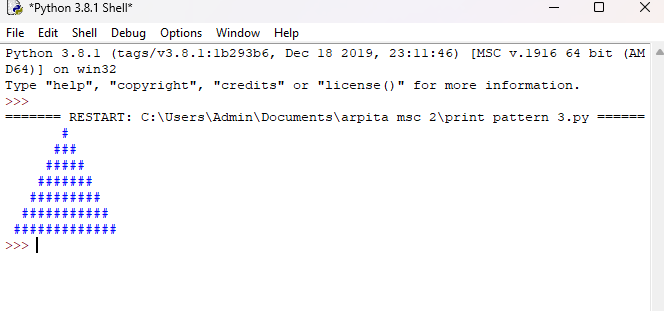
**Print Various Character Patterns(1)**

n=15

for i in range(1,n,2):

print (' ' \* ( (n-i)//2) + '#' \* i)

**Output:-**



**Print various chracter patterns.(pattern 2)**

n = 5

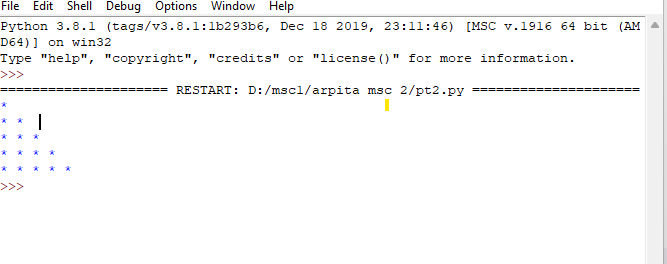
for i in range(0,n):

for j in range(0, i+1):

print("\*",end=' ')

print(" ")

**Output:-**



**Print various chracter pattern (pattern 3)**

count = 6

n = 65

for i in range(0,count):

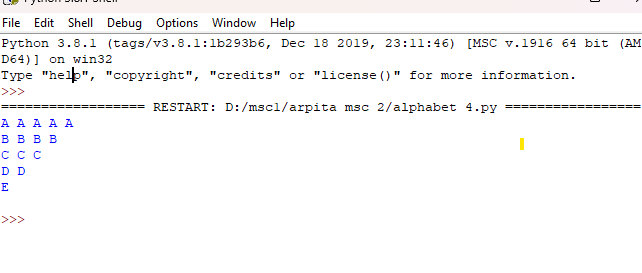
for j in range(1,count-i):

print(chr(n), end=' ')

print()

n+=1

**Output:-**



**Print various character patterns. (pattern 4)**

count = 6

n = 65

for i in range(0,count):

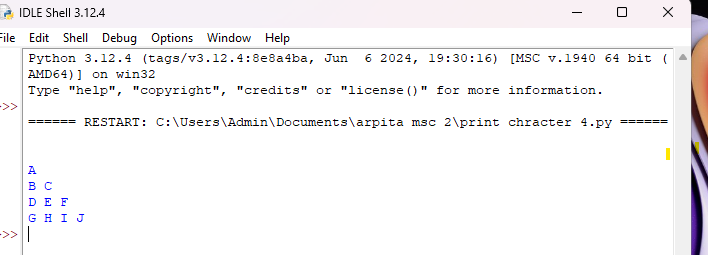
for j in range(1,i):

print(chr(n), end=' ')

n+=1

print()

**Output:-**



**Demonstrate the use of ‘if-elif-else’ student grade statements in python.**

marks = input(" Enter the percentage of marks scored : ")

marks = int(marks)

if (marks > 90):

print("congratulation your grade is A")

elif (71<marks<90):

print("congratulation your grade is B")

elif (51<marks<70):

print("congratulation your grade is c")

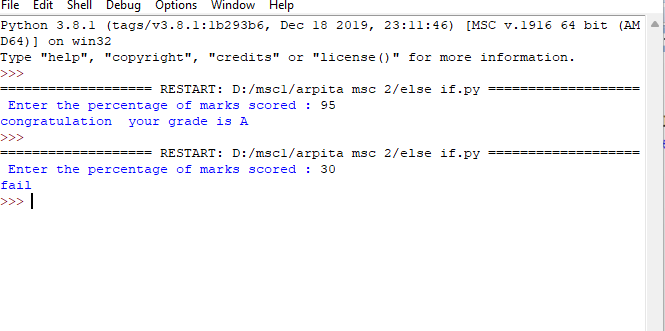
elif (41<marks<50):

print ("congratulation your grade is D")

else:

print ("fail")

**Output:-**



**Demonstrate about fundamental data types in python programming**

**(i.e.int,float,complex,bool and string)**

a,c = 10,20

b,d = 3.14,1.34

e,f = 3+4j,5+6j

h,g=" Hello Arpita ","welcome"

m,n = True,False

msg = " {} is of type {}"

print(msg.format(a, type(a)))

print(msg.format(b, type(b)))

print(msg.format(e, type(e)))

print(msg.format(m,type(n)))

print(msg.format(h, type(h)))

print()

add = " Addition of '{}' and '{}' is {}"

print(add.format(a, c, a+c))

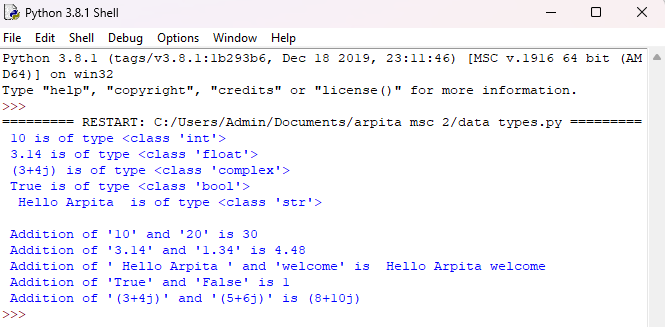
print(add.format(b, d, b+d))

print(add.format(h, g, h+g))

print(add.format(m, n, m+n))

print(add.format(e, f, e+f))

**Output:-**



**Write a program to addition of two number using command line argument.**

import sys

Add = 0

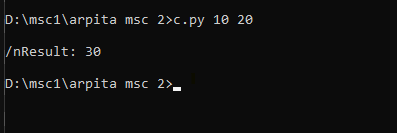
n= len(sys. argv)

for x in range(1,n):

Add +=int(sys.argv[x])

print("\n/nResult:",Add)

**Output:-**



**Demonstrate defining command line parameter using argspase module**

import sys

import argparse

users ={'Arpita':'111','Gayatri':'222','Deep':'444','Manish':'333'}

parser=argparse.ArgumentParser(description='Example script using argspace')

parser.add\_argument('-u','--user',help='User name')

parser.add\_argument('-p','--password',help='password')

args=parser.parse\_args()

if(args.user):

if args.user in users.keys():

pw=users[args.user]

if(args.password and args.password==pw):

print("you have been logged in successfully.")

else:

print("password is incorrect.")

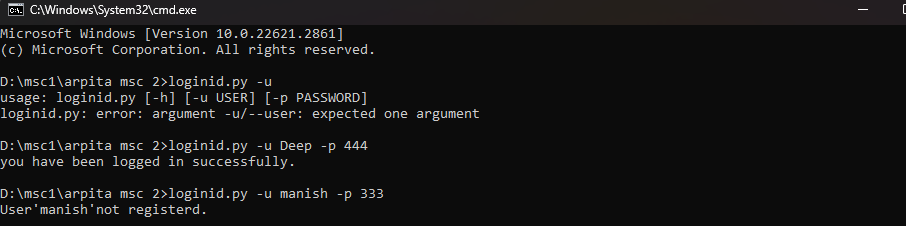
else:

print("User'{}'not registerd.".format(args.user))

else:

print("Empty user name.")

**Output :-**



**Develop a number guessing game**

import random

number=random.randint(1,10)

Attempts=10

for i in range(0,Attempts):

guess=input("Guess a number between 1 to 10:")

guess= int(guess)

if guess== number:

print("congratulation! You guess the number correctly.")

elif guess<number:

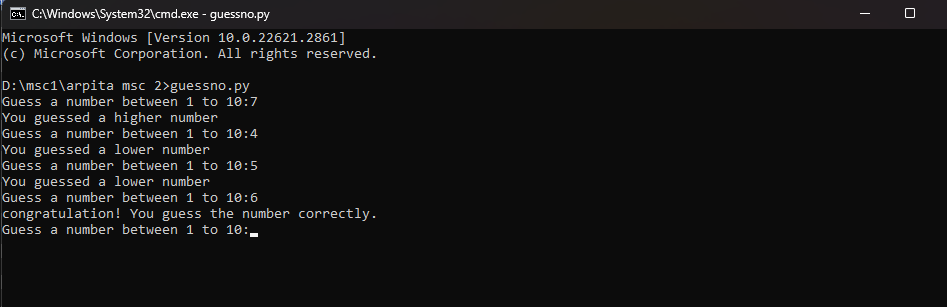
print("You guessed a lower number")

else:

print("You guessed a higher number")

print("Sorry! your attempts are over.")

**Output :-**

****

**Demonstrate the use of Arithmetic Operators**

a = float(input('Enter the first number :'))

b = float(input('Enter the second number:'))

# Addition

print ('Sum: ', a + b)

# subtraction

print ('Subtraction: ', a - b)

# multiplication

print ('Multiplication: ', a \* b)

# division

print ('Division: ', a / b)

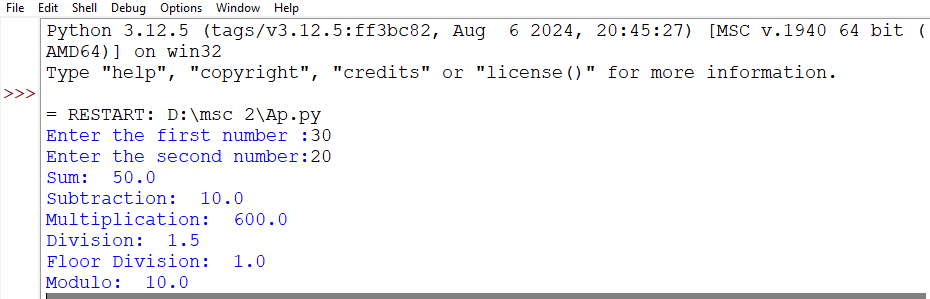
# floor division

print ('Floor Division: ', a // b)

# modulo

print ('Modulo: ', a % b)

**Output:-**

****

**Demonstrate the use of relational Operator in python**

**#Define the some variable**

a= float(input(" Enter the 1st number"))

b=float(input("Enter the 2nd number"))

**#Greater than**

print(f"a>b:{a>b}")

**#Greater than or equal to**

print(f"a>=b: {a>=b}")

**#Less than**

print (f"a<b:{a<b}")

**#Less than or equal to**

print(f"a<=b: {a<=b}")

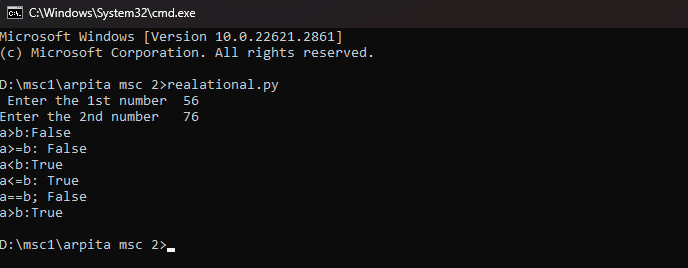
**#Equal to**

print(f"a==b; {a==b}")

**#Not Equal to**

print(f"a>b:{a!=b}")

**Output:-**

****

**Demonstrate the use of Bit Wise Operator in Python**

**# Define two numbers**

a = 10 # Binary: 1010

b = 4 # Binary: 100

**# Bitwise AND**

and\_result = a & b

print(f"Bitwise AND of {a} and {b} is {and\_result} (Binary: {bin(and\_result)})")

**# Bitwise OR**

or\_result = a | b

print(f"Bitwise OR of {a} and {b} is {or\_result} (Binary: {bin(or\_result)})")

**# Bitwise XOR**

xor\_result = a ^ b

print(f"Bitwise XOR of {a} and {b} is {xor\_result} (Binary: {bin(xor\_result)})")

**# Bitwise NOT**

not\_result = ~a

print(f"Bitwise NOT of {a} is {not\_result} (Binary: {bin(not\_result)})")

**# Bitwise Left Shift**

left\_shift\_result = a << 2

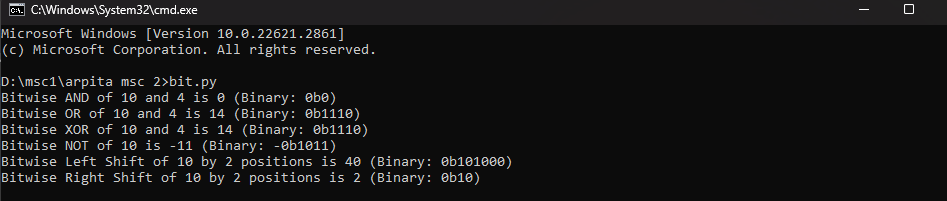
print(f"Bitwise Left Shift of {a} by 2 positions is {left\_shift\_result} (Binary: {bin(left\_shift\_result)})")

**# Bitwise Right Shift**

right\_shift\_result = a >> 2

print(f"Bitwise Right Shift of {a} by 2 positions is {right\_shift\_result} (Binary: {bin(right\_shift\_result)})")

**Output**:-



**Demonstrate the use of logical operator.**

a =float(input("Enter the 1st number"))

b =float(input("Enter the 2nd number"))

# Check the AND conditions

if a > 5 and b > 15:

print("Both conditions are True.")

else:

print("At least one condition is False.")

# Check the OR conditions

if a > 15 or b < 10:

print("At least one condition is True")

else:

print("Both conditions are False")

# Check the NOT conditions

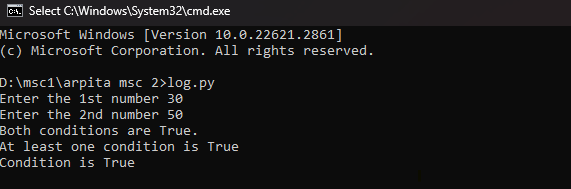
if not a > 15:

print("Condition is False")

else:

print("Condition is True")

**Output:-**



**Demonstrate the string formatting using index based and keyword based**

**insertion**

Position-based string formatting

def positin(name,mobile,roll):

return"My name is {}. My Mobile number is {} and my Roll number is {} .".format(name,mobile,roll)

#Index-Based Formatting

def index(name,mobile,roll):

return"My name is {2}. My Mobile number is {0} and my Roll number is {1} .".format(mobile,roll,name)

#keyword-Based String Formatting

def keyword(mobile,roll,name):

return"My name is {key1} , My Mobile number is {key2} and my Roll number is {key3} .".format(key1=name,key2=mobile,key3=roll)

#Example Usage

name1,mobile1,roll1="Arpita",7777777777,110

print(index(name1,mobile1,roll1))

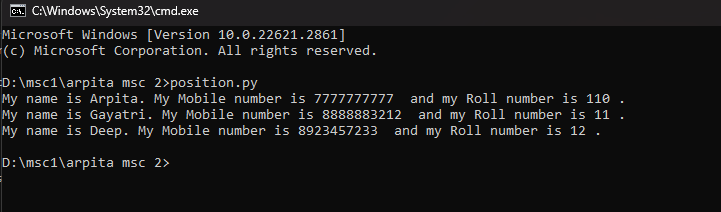
name2,mobile2,roll2="Gayatri",8888883212,11

print(index(name2,mobile2,roll2))

name1,mobile1,roll1="Deep",8923457233,12

print(index(name1,mobile1,roll1))

**Output:-**



**Demonstrate the use of slicing operation on a list.**

#Define a list of fruists

fruits=["apple","banana","cherry","date","date","fig","grape","kiwi"]

#Print the original list

print("Original list of fruits:")

print(fruits)

#Slicing examples

#Get the first three fruits

first\_fruits=fruits[:3]

print("\nFirst three fruits:", first\_fruits)

#Get the last three fruits

last\_fruits = fruits[-3:]

print("Last three fruits:",last\_fruits)

#Get fruits from index 2 to 5

middle\_fruits=fruits[2:6]

print("Middle Fruits from index 2 to 5:",middle\_fruits)

#GEt every second fruit

every\_second\_fruit=fruits[::2]

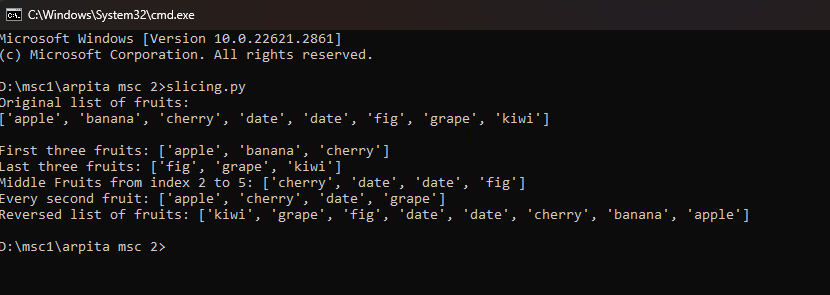
print("Every second fruit:",every\_second\_fruit)

#Reverse the list of fruits

reversed\_fruits=fruits[::-1]

print("Reversed list of fruits:",reversed\_fruits)

**Output :-**



**Demonstrate the use of index(),append(),insert(),extend() and remove()methodon list data type.**

**#list of fruits**

fruits= ["apple","banana","cherry","fig"]

fruitss =["kivi","papaya","guava"]

**#print original list**

print("Original list of fruits:")

print(fruits)

**# index method**

x= fruits.index("cherry")

print("Cherry is at index{}.".format(x))

**# append method**

fruits.append("papaya")

print("after append the element in the list:")

print(fruits)

**#insert method**

fruits.insert(1, "orange")

print("orange is inserted at index 1.")

print(fruits)

**# extend method**

fruits.extend(fruitss)

print("after extend the element in list:")

print(fruits)

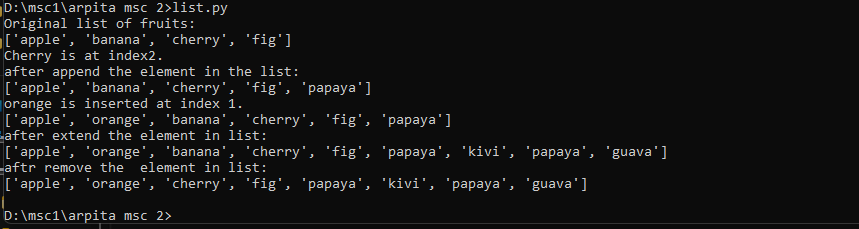
**#remove method**

fruits.remove("banana")

print("aftr remove the element in list:")

print(fruits)

**Output:-**



**Demonstrate the use of index()and count() method on tuple data type**

**# Define a list**

number=[ 1,2,3,4,5,6,7,8,9,5]

fruits= ["apple","banana","cherry","grapes"]

**#print the original list**

print("oringinal list of number:")

print(number)

print("original list of fruits:")

print(fruits)

**#find the position of the value "banana" using index method**

x=fruits.index("banana")

print("banana is at index{}.".format(x))

# **find the position of value of "3" using index method**

x1=number.index(3)

print("3 is at index{}.".format(x1))

**#count the occureance of 5**

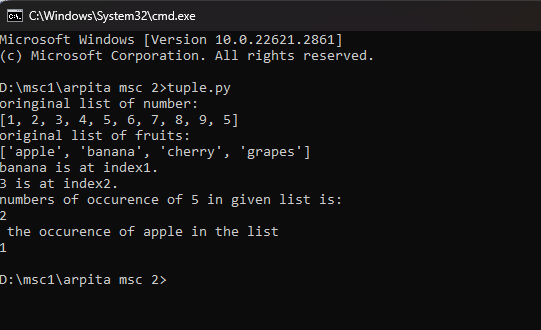
print ("numbers of occurence of 5 in given list is:")

print(number.count(5))

print(" the occurence of apple in the list")

print(fruits.count("apple"))

**Output:-**



**Demonstrate the use of add(),discard(),difference(),intersection()and union method on set data types**

**#List of student**

student={"ram","arpita", "deep", "gayatri"}

student1={" arpita", "ram", "Rudra" "pihu" }

**#Print original list**

print("original list of student:")

print(student)

print("original list of student1:")

print(student1)

**#Add method()**

student1.add("sham")

print("after adding the element in list:")

print(student1)

**#Discard method()**

student .discard("deep")

print("after discard deep the given list is:")

print(student)

**#Difference method()**

x=student.difference(student1)

print("the difference list is:")

print(x)

**#Intersection method()**

y=student.intersection(student1)

print("after applying the intersection method the given list is:")

print(y)

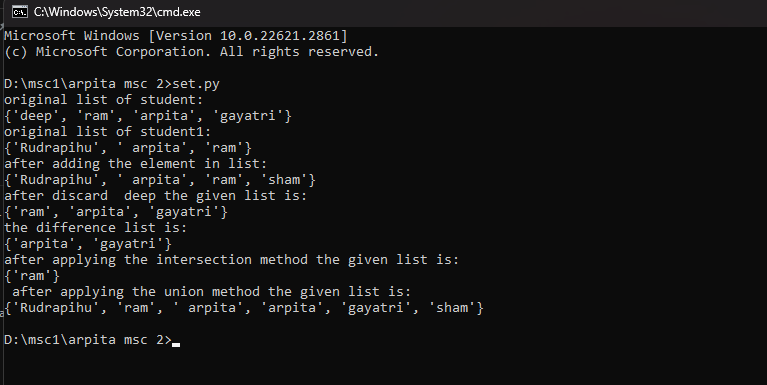
**#Union method()**

z= student.union(student1)

print(" after applying the union method the given list is:")

print(z)

**Output:-**



**Demonstrate the use of get(), setdefult(), pop(), update() methods on dictionary data types**

d = {'Name': 'Ram', 'Age': '19', 'Country': 'India'}

d2 = {'Name': 'Neha', 'Age': '22'}

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

**#print original dictionary**

print("original dictionary is:")

print(d)

print(car)

print(d2)

**#Get method**

print ("name on the list:")

print(d.get('Name'))

print("genter on the list:")

print(d.get('Gender'))

**#Pop method**

d.pop("Age")

print("After pop the item in the list:")

print(d)

**#update item**

d.update(d2)

print("After update the item in list:")

print(d)

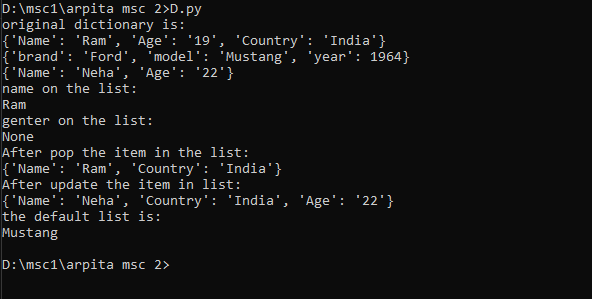
**#set default method**

x = car.setdefault("model", "Bronco")

print("the default list is:")

print(x)

**Output:-**



**Count the Words in a given sentence**

def count\_words(sentence):

word\_count=0

start=0

for i in range(len(sentence)):

if sentence[i]==' 'or i==len(sentence)-1:

if i==len(sentence)-1 and sentence[i]!=' ':

end=i+1

else:

end=i

word=sentence[start:end]

if word:

word\_count+=1

start=i+1

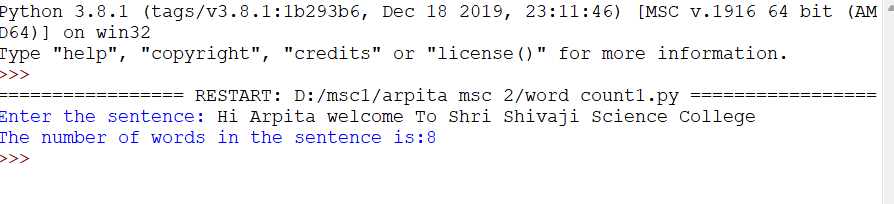
return word\_count

my\_sentence=input("Enter the sentence:")

word\_count=count\_words(my\_sentence)

print(f"The number of words in the sentence is:{word\_count}")

**Output:-**



**Make the first letter of every word in the given sentence to capital case.**

def capitalize\_and\_count(sentence):

#spilt the sentence into words

words=sentence.split()

#Initialize an empty list to hold capitalized words

capitalized\_words=[]

#Count the number of words

word\_count=0

for word in words:

#Capitalize the first letter and append to the list

capitalized\_words.append(word.capitalize())

#Increment the word count

word\_count+=1

#Join the capitalized words back into sentence

result=' '.join(capitalized\_words)

return result,word\_count

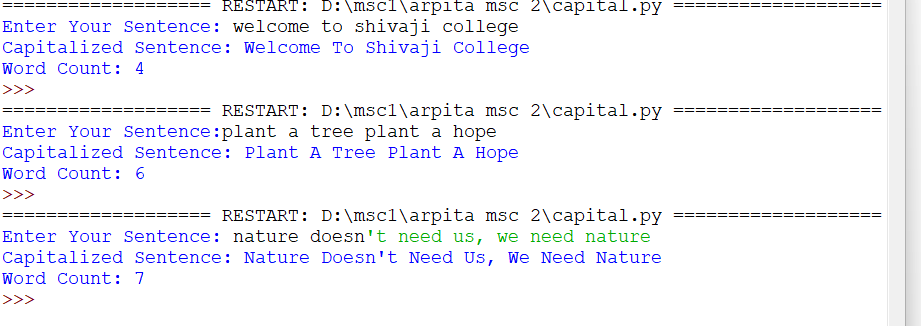
my\_sentence=input("Enter Your Sentence:")

output\_sentence,count=capitalize\_and\_count(my\_sentence)

print("Capitalized Sentence:",output\_sentence)

print("Word Count:",count)

**Output:-**



**Develop a calculator class with following mathematical operation :Addition, substraction, multiplication, division, maximum, minimum**

class List\_calculator:

def add(self,a,b):

return a+b

def substract(self,a,b):

return a-b

def multiply(self,a,b):

return a\*b

def divide(self,a,b):

if b==0:

return"Error:Division by zero"

return a/b

def maximum(self,a,b):

return max(a,b)

def minimum(self,a,b):

return min(a,b)

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

list1=[15,25,35]

list2=[10,20,30]

calc=List\_calculator()

for a,b in zip(list1 ,list2):

print(f"Calculating for:{a} and {b}")

print("Addition:",calc.add(a,b))

print("Substarction:", calc.substract(a,b))

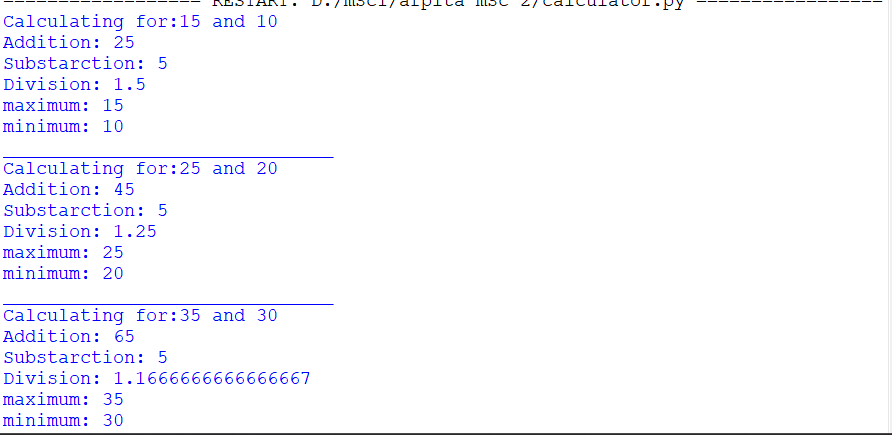
print("Division:",calc.divide(a,b))

print("maximum:",calc.maximum(a,b))

print("minimum:",calc.minimum(a,b))

print("\_"\*30)

Output:-



**Write and Read /append text into the file and display**

#File handling in python

#Function to write in the file

def write(line,append):

append = "w"

if(append): mode ="a"

file =open("E:\ARC\PRAC\FILE.txt",mode)

file.write(line)

file.close()

#Function to read the file

def read(num):

count=0

file=open("E:\ARC\PRAC\FILE.txt","r")

if(num<1):

print(file.read())

else:

line=file.readline()

count+=1

while(line and count<=num):

print(line)

line=file.readline()

count+=1

file.close()

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

print("1.write\n2.read\n3.Exit")

while(True):

c=int(input("Enter operation choice"))

if(c==1):

str=input("Enter the text to write")

write(str,True)

elif(c==2):

print("## Reading a file ## ")

n=int(input("How many lines do you want to read ? "))

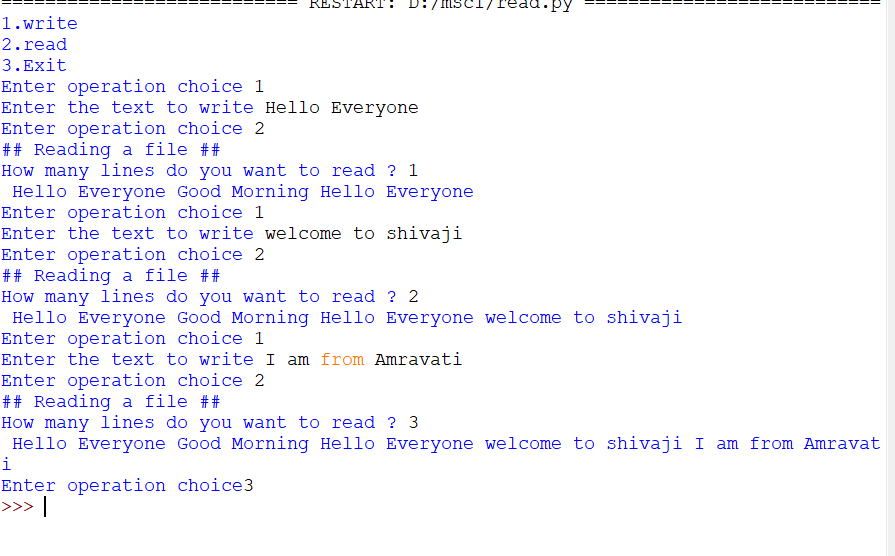
read(n)

elif(c==3):break

else:

print("Wrong choice enterd")

**Output:-**



**Demonstate Error Handling in python using try…except…else and try…finally**.

fruits =["Apple","Banana","Mango","Blueberry","Cherry"]

print("Original List Of fruit")

print(fruits)

def my\_fruit(Index):

try:

fruit=fruits[Index]

except IndexError:

print(f"Error:Index{Index}is out of range for the fruit list:")

else:

print(f"The fruit at Index {Index}is:{fruit}")

finally:

print(f"I have always show fruit.")

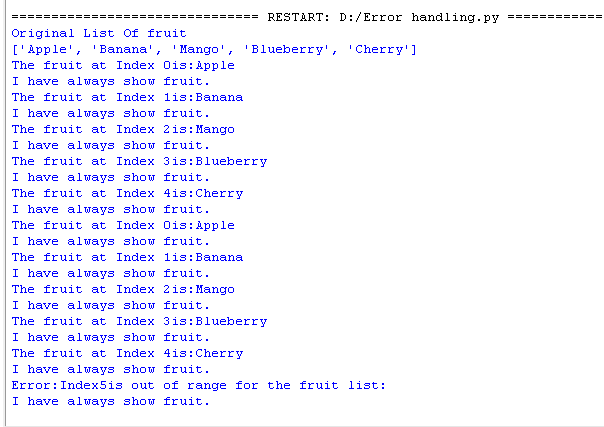
for i in range(len(fruits)):

my\_fruit(i)

for i in range(len(fruits)+1):

my\_fruit(i)

**Output:-**



**Demonstrate the use of Multithreading for printing tables.**

**CODE:**

import threading

import time

def print\_args(\*args):

queueLock.acquire() # Acquire lock to ensure sequential printing

for num in args:

print("Table of {}:".format(num))

for i in range(1, 11):

print(i \* num)

print() # Separate tables for readability

time.sleep(1)

queueLock.release() # Release the lock after finishing

queueLock = threading.Lock()

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

# Create and start threads

thread1 = threading.Thread(target=print\_args, args=(5, 6))

thread2 = threading.Thread(target=print\_args, args=(7, 8, 9))

thread1.start()

thread2.start()

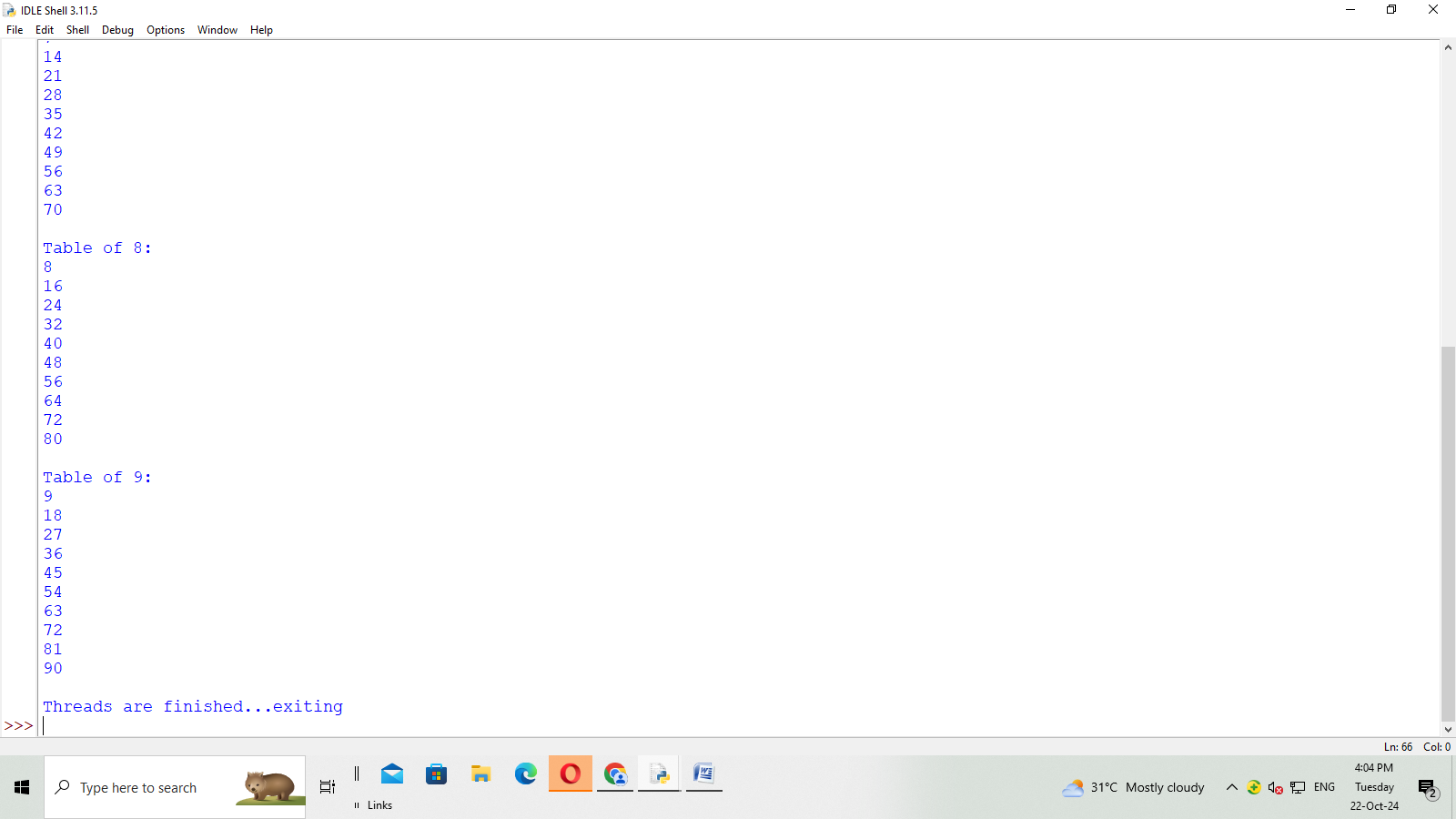
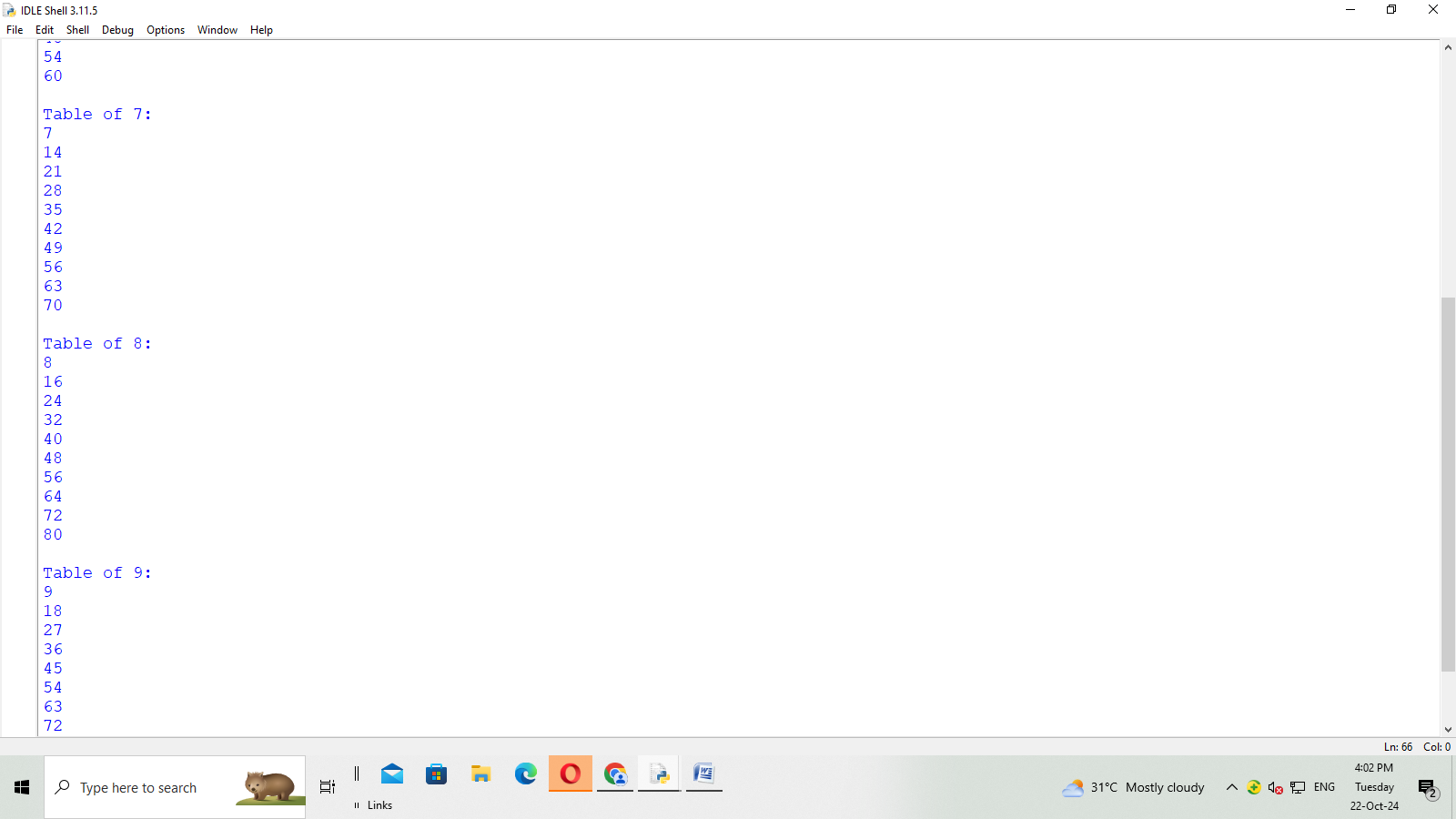
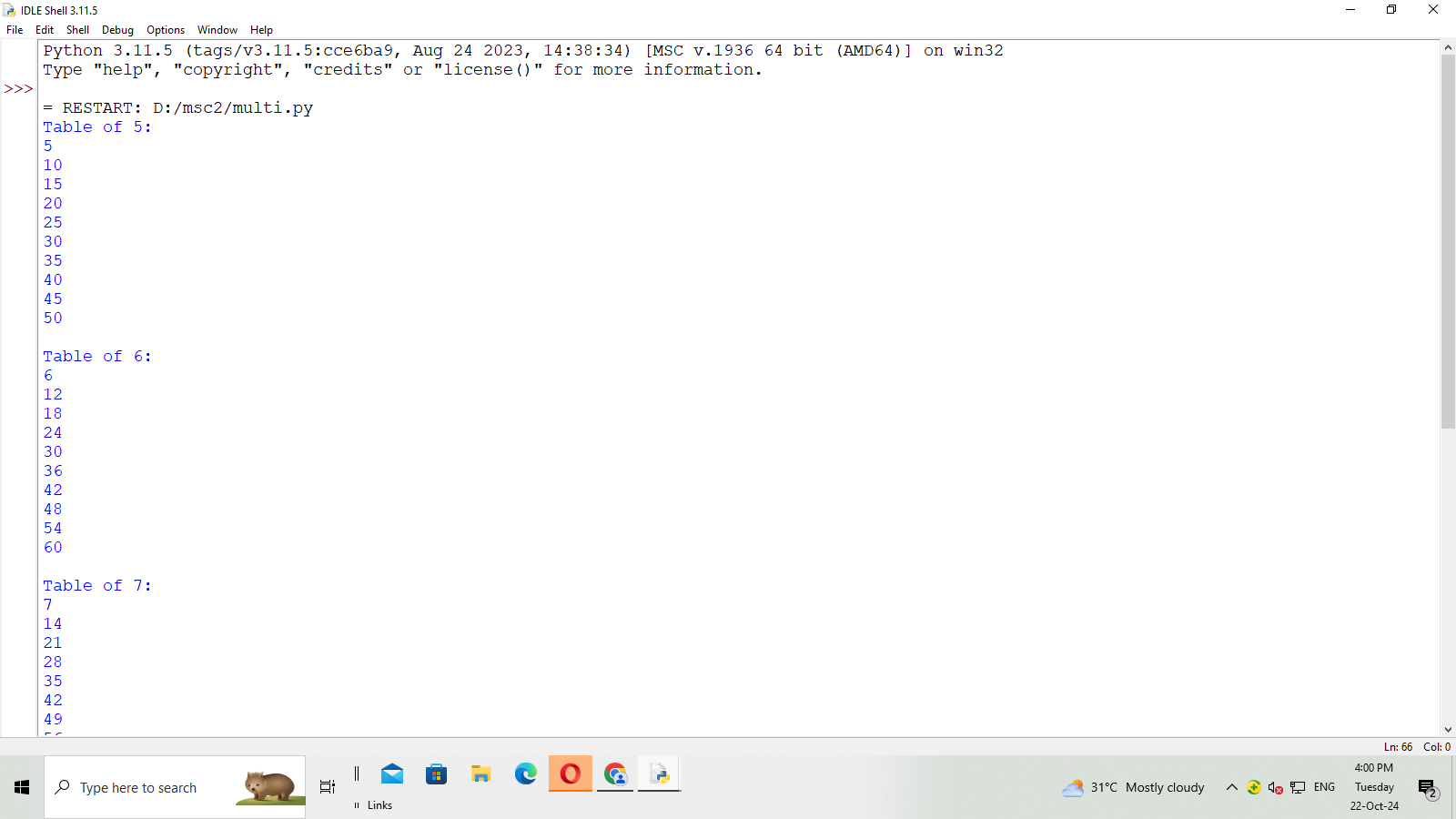
# Wait for threads to complete

thread1.join()

thread2.join()

print("Threads are finished...exiting")

**OUTPUT:**



**Create ‘Student’ table and insert few records using MySQL python Database connection**.

import mysql.connector

'''mydb = mysql.connector.connect(

host="localhost",

user="root",

password=""

)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE pydatabase")'''

mydb = mysql.connector.connect(

host="localhost",

user="root",

password="",

database="pydatabase"

)

mycursor = mydb.cursor()

**# Creating a Table**

mycursor.execute("CREATE TABLE student10(name VARCHAR(255), rollno INTEGER, \

address VARCHAR(255))")

**#Inserting the records**

sql = "INSERT INTO student10 (name, rollno, address) VALUES (%s,%s,%s)"

val = [

('Arpita', 101, 'Dabha'),

('Bhagyashri', 102, 'Partwada'),

('Nisha', 103, 'Badnera'),

('Shravani', 104,'Katsur'),

('Ekata', 105,'Yavatmal')]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")

**#Display the file from Table**

sql = "SELECT \* FROM student10";

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

**Output:-**

