1.Write a menu driven program to perform basic arithmetic operations using functions

def add(a,b):

return a+b

def subtract(a,b):

return a-b

def multiply(a,b):

return a\*b

def divide(a,b):

if b!=0:

return a/b

else:

return "Error!Division by zero"

def menu():

while True:

print("\n--Arithmetic operation--")

print("\n1.Addition")

print("\n2.Subtraction")

print("\n3.Multiplication")

print("\n4.Division")

print("\n5.Exit")

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

if choice==5:

print("Exiting the pogram")

break

elif choice in [1,2,3,4]:

num1=int(input("Enter first number"))

num2=int(input("Enter second number"))

if choice==1:

print(f"Result:{add(num1,num2)}")

if choice==2:

print(f"Result:{subtract(num1,num2)}")

if choice==3:

print(f"Result:{multiply(num1,num2)}")

if choice==4:

print(f"Result:{divide(num1,num2)}")

else:

print("Invalid choice !please try again")

menu()

2.Write a python program to find even and odd numbers from a list ,append even to even list and od to odd list

def seperate\_even\_odd(numbers):

even\_list=[]

odd\_list=[]

for num in numbers:

if num%2==0:

even\_list.append(num)

else:

odd\_list.append(num)

return even\_list,odd\_list

numbers=[10,30,23,45,42,88,95]

even\_list,odd\_list=seperate\_even\_odd(numbers)

print("Original list",numbers)

print("Even numbers",even\_list)

print("odd numbers",odd\_list)

3.Write a python function to find the count of vowels,numbers and consonants for the given user input

def count\_vowels\_number\_consonants(text):

vowels="aeiouAEIOU"

num\_vowels=0

num\_number=0

num\_consonants=0

for char in text:

if char.isdigit():

num\_number+=1

elif char.isalpha():

if char in vowels:

num\_vowels+=1

else:

num\_consonants+=1

return num\_consonants,num\_consonants,num\_number

user\_input=input("Enter a string:")

vowels,consonants,numbers= count\_vowels\_number\_consonants(user\_input)

print(f"Vowels{vowels}")

print(f"numbers{numbers}")

print(f"consonants{consonants}")

**4.Write a function to validate phoe numbers and input n names and phone numbers to store it in a dictionary and print the phone number of a particular name**

def validate\_phone\_no(phone):

return phone.isdigit() and len(phone)==10

def store\_phone\_no(n):

phone\_book={}

for i in range(n):

name=input("Enter name:").strip()

while True:

phone=input(f"Enter phone number for{name}:").strip()

if validate\_phone\_no(phone):

phone\_book[name]=phone

else:

print("Invalid phone no please enter 10 digit number")

return phone\_book

def get\_phone\_no(phone\_book,name):

return phone\_book.get(name,"Name not found in phone book")

n=int(input("How many contacts do you want to store"))

phone\_book=store\_phone\_no(n)

name\_to\_search=input("Enter the name to find the phone number").strip()

phone\_number=get\_phone\_no(phone\_book,name\_to\_search)

print(f"Phone number of {name\_to\_search}:{phone\_number}")

5.Write a program to reverse the word in a sentence and count the frequency of each character

from collections import Counter

def reverse\_words\_and\_count\_char(sentence):

words=sentence.split()

reversed\_words=' '.join(word[::-1] for word in words)

char\_frequency=Counter(sentence.replace(" "," "))

return reversed\_words,char\_frequency

sentence=input("Enter a sentence")

reversed\_sentence,char\_frequency=reverse\_words\_and\_count\_char(sentence)

print(f"Reverse words in a sentence:{reversed\_sentence}")

print("characte frequency",dict(char\_frequency))

**6.Write a program to find the keys with maximum value in a dictionary**

def find\_keys\_with\_max\_value(d):

max\_value=max(d.values)

keys\_with\_max\_value=[key for key ,value in d.items() if value==max\_value]

return keys\_with\_max\_value

my\_dict={

"a":10,

"b":20,

"c":20,

"d":5

}

keys=find\_keys\_with\_max\_value(my\_dict)

print(f"Keys wth maximum value:{keys}")

7.Find the difference between two list using comprehensions

def list\_difference (list1,list2):

return[item for item in list1 if item not in list2]

list1=[1,2,3,4,5]

list2=[4,5,6,7,8]

difference=list\_difference(list1,list2)

print("Difference:",difference)

8.Write a program to check whether entered string is anagram of each other

def are\_anagram(str1,str2):

str1=str1.replace("","").lower()

str2=str2.replace("","").lower()

return sorted(str1)==sorted(str2)

str1=input("Enter the first string")

str2=input("Enter the second string")

if are\_anagram(str1,str2):

print(f"{str1} and {str2} are anagram")

else:

print(f"{str1} and {str2} are not anagram")

9.Write a program to find all pairs of numbers a list that add up to a target(ex target value)

def find\_pairs\_with\_sum(numbers,target):

pairs=[]

for i in range(len(numbers)):

for j in range(i+1,len(numbers)):

pairs.append((numbers[i],numbers[j]))

return pairs

numbers=[2,4,3,5,1,6,3]

target=7

pairs=find\_pairs\_with\_sum(numbers,target)

print(f"Pairs that add up to{target}:{pairs}")

10.Write a python program that takes input from the user for a list of student names and this corresponding scores.The program should then convert this data into a list of tuples,where each tuples contains a students name and their score.The program should sort the list of tuples based on the scores in descending order and print the sorted list

n=int(input("Enter the number of students"))

students=[]

for i in range(n):

name=input(f"Enter name of student{i+1}")

score=float(input(f"Enter score of student{i+1}"))

students.append((name,score))

sorted\_students=sorted(students,key=lambda x:x[1],reverse=True)

print("\nSorted list of students(name,score):")

for student in sorted\_students:

print(student)

**11.Write a python program that takes the list of scores as input and performs the specified operations**

**1.calculate and print the average score of the class**

**2.Determine and print the highest score achieved in class**

**3.count and print the numbers of students who scored above average score**

scores=list(map(float,input("Enter the list of scores seperated by space").split()))

average\_score=sum(scores)/len(scores)

highest\_score=max(scores)

above\_average\_count=sum(scores>average\_score for score in scores)

print(f"\n Average score:{average\_score:.2f}")

print(f"\nHighest score:{highest\_score}")

print(f"\n Number of students above average :{above\_average\_count}")

12. Write a python program to perform a linear search on a list of integers .Input the content of list

def linear\_search(lst,target):

for index,value in enumerate(lst):

if value==target:

return index

return -1

lst=list(map(int,input("Enter numbers seperated by spaces:").split()))

target=int(input("Enter the target value to search"))

result=linear\_search(lst,target)

if result!=-1:

print(f"Target found at index{result}")

else:

print(f"Target not found in the list")

**13.Write a function to find the second largest from the list**

def second\_largest(lst):

if len(lst)<2:

return None

max\_val=second\_largest\_val=float("-inf")

for num in lst:

if num>max\_val:

maxval,second\_largest\_val=num,maxval

elif max\_val>num>second\_largest\_val:

second\_largest\_val=num

return second\_largest\_val if second\_largest\_val!=float("-inf")else None

lst=list(map(int,input("Enter the integer in the list seperated by space").split()))

result=second\_largest(lst)

print("The second largest number is:",result if result is not None else 'Not available')

**14.Write a python module called utils.py containing functions for string manipulation such as capitalize\_first\_letter,reverse\_string and count\_vowels ,in your python program demonstrate how to import and use the functions in a separate python file**

utils.py file

def capitalize\_first\_letter(s):

if not s:

return s

return s[0].upper()+s[1:]

def reverse\_string(s):

return s[::-1]

def count\_vowels\_count(s):

vowels="aeiouAEIOU"

return sum(1 for char in s if char in vowels)

main.py file

from utils import capitalize\_first\_letter,reverse\_string,count\_vowels

text="hello world"

cap\_text=capitalize\_first\_letter(text)

print("capitalized:",cap\_text)

rev\_text=reverse\_string(text)

print("Reversed:",rev\_text)

vowel\_count=count\_vowels(text)

print("Vowel count:",vowel\_count)

15.Write a python program that takes input from user for two list of integers and finds and prints the common elements between them using set

list1=input("Enter the first list of integers(seperated by spaces):").split()

list1=[int(x) for x in list1]

list2=input("Enter the second list of integers(seperated by spaces):").split()

list2=[int(x) for x in list2]

common=set(list1)&set(list2)

print("common elements:",list(common))

**16.create a bankAccount class with attributes for account holders name ,balance and account number. Implement methods to deposit,withdraw(with balance check) and check the balance**

**Demonstrate bank operations using instances**

class BankAccount:

def \_\_init\_\_(self,name,accno,balance=0):

self.name=name

self.accno=accno

self.balance=0

def deposit(self,amount):

if amount>0:

self.balance+=amount

print(f"deposited {amount} new balance:{self.balance}")

else:

print("Deposit amount must be positive")

def withdraw(self,amount):

if amount>0:

if amount<=self.balance:

self.balance-=amount

print(f"Withdrawn {amount} new balance:{self.balance}")

else:

print("Insufficient balance")

else:

print("withdraw amount must be positive")

def check\_balance(self):

print(f"current balance:{self.balance}")

acc1=BankAccount(name="Sara",accno="167738378",balance=1000)

acc1.deposit(500)

acc1.withdraw(300)

acc1.check\_balance()

acc1.withdraw(1500)

17.create a student class with name and marks attribute add methods to calculate\_average()

Return the above average of marks ,assign grade() return a grade based on the average.

class student:

def \_\_init\_\_(self,name,marks):

self.name=name

self.marks=marks

def calculate\_average(self):

if not self.marks:

return 0

return sum(self.marks)/(len(self.marks))

def assign\_grade(self):

avg=self.calculate\_average()

if 90<=avg<=100:

return 'A+'

elif 80<=avg<=90:

return 'B'

elif 70<=avg<=80:

return 'C'

elif 60<=avg<=70:

return 'D'

elif avg<40:

return 'F'

else:

return 'Pass'

student1=student(name='Ovi',marks=[87,86,98,88,78])

avg\_marks=student1.calculate\_average()

print(f"average marks{student1.name}:{avg\_marks}")

grade=student1.assign\_grade()

print(f"Grade for {student1.name}:{grade}")

18.create a vehicle class with(make and model attributes) and a method show\_info() to display them .create a car class that inherits from vehicle and adds a year attributes . create a car object to show all its details

class vehicle:

def \_\_init\_\_(self,make,model):

self.make=make

self.model=model

def show\_info(self):

print(f"Make :{self.make},Model{self.model}")

class car(vehicle):

def \_\_init\_\_(self,make,model,year):

super().\_\_init\_\_(make,model)

self.year=year

def show\_info(self):

print(f"Make {self.make },Model:{self.model},year{self.year}")

my\_car=car(make="toyota",model="corolla",year=2020)

my\_car.show\_info()

19.create a dimension class(length,breadth,height) weight class(which includes weight) and a Box class ,where Box initializes and display both dimensions and weight using super() function.

class Dimension:

def \_\_init\_\_(self,length,width,height):

self.length=length

self.width=width

self.height=height

def show\_dimensions(self):

print(f"Dimensions length:{self.length} width{self.width}height:{self.height}")

class weight:

def \_\_init\_\_(self,weight):

self.weight=weight

class Box(Dimension,weight):

def \_\_init\_\_(self,length,width,height,weight):

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

weight.\_\_init\_\_(self,weight)

def show\_info(self):

self.show\_dimensions()

self.show\_weight()

my\_box=Box(length=20,width=15,weight=25,height=10)

my\_box.show\_dimensions()

**20.write a python to create a dimension class (length,width,height) weight class which includes weight and Box class where Box initializes and display both dimensions and weight while using super() function**

class Dimension:

def \_\_init\_\_(self,length,width,height):

self.length=length

self.width=width

self.height=height

def show\_dimensions(self):

print(f"Dimensions length:{self.length} width{self.width}height:{self.height}")

class weight:

def \_\_init\_\_(self,weight):

self.weight=weight

def show\_weight(self):

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

class Box:

def \_\_init\_\_(self,length,width,weight,height):

self.dimension=Dimension(length,width,height)

self.weight=weight(weight)

def show\_info(self):

self.dimension.show\_dimensions()

self.weight.show\_weight()

my\_box=Box(length=20,width=15,weight=25)

my\_box.show\_info()

21.write a python program using classes dimensions,Box and shipping Box where dimensions stores length,width,height. Box calculates volume and shipping Box includes weight and displays the total weight

class Dimensions:

def \_\_init\_\_(self,length,width,height):

self.length=length

self.width=width

self.height=height

def show\_dimensions(self):

print(f"Dimensions length:{self.length} width{self.width}height:{self.height}")

class Box(Dimensions):

def \_\_init\_\_(self,length,width,height):

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

self.volume=self.calculate\_volume()

def calculate\_volume(self):

return self.length+self.width+self.height

def show\_volume(self):

print(f"Box volume :{self.volume}")

class ShippingBox(Box):

def \_\_init\_\_(self,length,width,height,weight):

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

self.weight=weight

def total\_weight(self):

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

shipping\_box=ShippingBox(length=10,width=5,height=8,weight=20)

shipping\_box.show\_dimensions()

shipping\_box.show\_volume()

shipping\_box.total\_weight()

22. Write a Python program to create a base class animal with the name attribute and eat() method. From this class, create three specialized classes. Mammal adds sound and speak, bird adds can fly and fly, and fish adds is freshwater and swim. Choose an inheritance approach that allows all classes to share common properties of the animal class maintaining their unique features.

class Animal:

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

self.name=name

def eat(self):

print(f"{self.name} is eating")

class Mammal(Animal):

def \_\_init\_\_(self,name,sound):

super().\_\_init\_\_(name)

self.sound=sound

def speak(self):

print(f"{self.name} says {self.sound}")

class Bird(Animal):

def \_\_init\_\_(self,name,can\_fly):

super().\_\_init\_\_(name)

self.can\_fly=can\_fly

def fly(self):

if self.can\_fly:

print(f"{self.name} is flying!")

else:

print(f"{self.name} cannot fly")

class Fish(Animal):

def \_\_init\_\_(self,name,is\_freshwater):

super().\_\_init\_\_(name)

self.is\_freshwater=is\_freshwater

def swim(self):

if self.is\_freshwater:

print(f"{self.name}is swimming in fresh water")

else:

print(f"{self.name}is swimming in salt water")

mammal=Mammal(name='Dog',sound='bark')

mammal.eat()

mammal.speak()

bird=Bird(name='Sparrow',can\_fly=True)

bird.eat()

bird.fly()

fish=Fish(name='goldfish',is\_freshwater=True)

fish.eat()

fish.swim()