1. **Write a Python code to print the sum of natural numbers using recursive functions.**

**Program:**

 **def sumNnum(n):**

 **if (n == 0):**

 **return 0**

 **return n + sumNnum (n - 1)**

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

**print(sumNnum(n))**

**Output:**

**Enter a natural number: 10**

**55**

1. **Write a python code to display the sum of cubes of digits of a number.**

**Program:**

**def sumCdigits(number):**

 **sumCubes = 0**

 **while number > 0:**

 **digit = number % 10**

 **sumCubes += digit \*\* 3**

 **number //= 10**

 **return sumCubes**

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

**sumCubes = sumCdigits(number)**

**print(f"The sum of the cubes of the digits of {number} is {sumCubes}.")**

**Output:**

**Enter a number: 123**

**The sum of the cubes of the digits of 123 is 36.**

1. **Write a python code that takes two lists and returns True if they have at least one common member.**

**Program:**

**def commonMem(list1, list2):**

 **for element in list1:**

 **if element in list2:**

 **return True**

 **return False**

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

**list2 = [6, 7, 8, 9, 10]**

**print(commonMem(list1, list2))**

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

**list2 = [1, 6, 7, 8, 9]**

**print(commonMem(list1, list2))**

**Output:**

**False**

**True**

1. **Write a python program to combine the content of two file and store it in a single list and display that list.**

**Program: (Create new files abc.txt and xyz.txt) and in abc.txt type Hello world and in xyz.txt type Easy to learn and save**

**def combine\_files(file1, file2):**

 **return [line for line in open(file1, "r")] + [line for line in open(file2, "r")]**

**combined\_list = combine\_files("abc.txt", "xyz.txt")**

**print(combined\_list)**

**Output:**

**['Hello world’, 'Easy to learn']**

1. **Implement the concept of multiple inheritance using python.**

**Program:**

**class base\_class:**

 **def s1(self):**

 **print("Base class")**

**class st1:**

 **def s2(self):**

 **print("Derived Class 1")**

**class st2(base\_class, st1):**

 **def s3(self):**

 **print("Derived Class 2")**

**obj=st2()**

**obj.s1()**

**obj.s2()**

**obj.s3()**

**Output:**

**Base class**

**Derived Class 1**

**Derived Class 2**

1. **Write a python code to check whether the entered number is Armstrong or not.**

**Program:**

**def armnum(num):**

 **sum=0**

 **temp = num**

 **while temp > 0:**

 **digit = temp % 10**

 **sum += digit \*\* 3**

 **temp //= 10**

 **if num == sum:**

 **print(num, " is an Armstrong Number")**

 **else:**

 **print(num, " is not an Armstrong Number")**

**num=int(input("Enter any number: "))**

**armnum(num)**

**Output:**

**Enter any number: 153**

**153 is an Armstrong Number**

1. **Design a class complex for adding two complex number and also show the use of constructor.**

**Program:**

**class Complex:**

 **def \_\_init\_\_(self, real, imaginary):**

 **self.real = real**

 **self.imaginary = imaginary**

 **def \_\_add\_\_(self, other):**

 **return Complex(self.real + other.real, self.imaginary + other.imaginary)**

 **def \_\_str\_\_(self):**

 **return f"{self.real} + {self.imaginary}j"**

**z1 = Complex(37, 31)**

**z2 = Complex(34, 5)**

**result = z1 + z2**

**print(result)**

**Output:**

**71 + 36j**

1. **Write a python program to take a character from user and search that character in the file. If the character is present then print total count of that character in the file or else display the message “no such character”.**

**Program:**

**def search\_character\_in\_file(character, file\_path):**

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

 **count = 0**

 **for line in f:**

 **for c in line:**

 **if c == character:**

 **count += 1**

 **if count > 0:**

 **return count**

 **else:**

 **return "no such character"**

**character = input("Enter a character: ")**

**file\_path = input("Enter the path to the file: ")**

**result = search\_character\_in\_file(character, file\_path)**

**print(result)**

**Output:**

**Enter a character: a**

**Enter the path to the file: abc.txt**

**1**

1. **Write a Python program to sum all the items in a dictionary.**

**Program:**

**d = { 't1': 10, 't2': 20, 't3': 30}**

**print(d)**

**print("sum: ", sum(d.values()))**

**Output:**

**{'t1': 10, 't2': 20, 't3': 30}**

**sum: 60**

1. **Write a Python program to sort (ascending and descending) a dictionary by value.**

**Program:**

**import operator**

**d = {1:22, 3:13, 4:20, 2:37, 0:27}**

**t = sorted(d.items(), key = operator.itemgetter(0))**

**print("In ascending order by value: ", t)**

**t = sorted(d.items(), key = operator.itemgetter(0), reverse = True)**

**print("In descending order by value: ", t)**

**Output:**

**In ascending order by value: [(0, 27), (1, 22), (2, 37), (3, 13), (4, 20)]**

**In descending order by value: [(4, 20), (3, 13), (2, 37), (1, 22), (0, 27)]**

1. **Write a Python program to merge the content of two files into one file.**

**Program: (two txt files should be created for executing the program name must be abc.txt and xyz.txt)**

**def merge\_files(file1, file2, output\_file):**

 **with open(file1, "r") as f1, open(file2, "r") as f2, open(output\_file, "w") as f3:**

 **for line in f1:**

 **f3.write(line)**

 **for line in f2:**

 **f3.write(line)**

**file1 = input("Enter the path to the first file: ")**

**file2 = input("Enter the path to the second file: ")**

**output\_file = input("Enter the path to the output file: ")**

**merge\_files(file1, file2, output\_file)**

**print("The files have been merged successfully.")**

**Output:**

**Enter the path to the first file: abc.txt**

**Enter the path to the second file: xyz.txt**

**Enter the path to the output file: qwerty.txt**

**The files have been merged successfully.**

1. **Write a python program to clone or copy a list. (Without using any built-in function).**

**Program:**

**L1 = [1,2,3,4,5,6]**

**L2 = list(L1)**

**print("L1: ", L1)**

**print("L2: ", L2)**

**Output:**

**L1: [1, 2, 3, 4, 5, 6]**

**L2: [1, 2, 3, 4, 5, 6]**

1. **Write a python program to accept an Integer list from user and print the sum of all the odd numbers from that list.**

**Program:**

**def sum\_of\_odd\_numbers(list1):**

 **sum\_of\_odd\_numbers = 0**

 **for number in list1:**

 **if number % 2 != 0:**

 **sum\_of\_odd\_numbers += number**

 **return sum\_of\_odd\_numbers**

**list1 = []**

**while True:**

 **number = input("Enter an integer: ")**

 **if number == "":**

 **break**

 **list1.append(int(number))**

**sum\_of\_odd\_numbers = sum\_of\_odd\_numbers(list1)**

**print(f"The sum of all the odd numbers in the list is {sum\_of\_odd\_numbers}.")**

**Output:**

**Enter an integer: 5**

**Enter an integer: 31**

**Enter an integer: 34**

**Enter an integer: 37**

**Enter an integer: 1**

**Enter an integer:**

**The sum of all the odd numbers in the list is 74.**

1. **Write a python code to display the odd and even numbers separately from a list. (All elements in the list should be taken from user).**

**Program:**

**def get\_integer\_list\_from\_user():**

 **integer\_list = []**

 **while True:**

 **integer = input("Enter an integer: ")**

 **if integer == "":**

 **break**

 **integer\_list.append(int(integer))**

 **return integer\_list**

**def display\_odd\_and\_even\_numbers\_separately(integer\_list):**

 **odd\_numbers = []**

 **even\_numbers = []**

 **for integer in integer\_list:**

 **if integer % 2 != 0:**

 **odd\_numbers.append(integer)**

 **else:**

 **even\_numbers.append(integer)**

 **print("Odd numbers:")**

 **for odd\_number in odd\_numbers:**

 **print(odd\_number)**

 **print("Even numbers:")**

 **for even\_number in even\_numbers:**

 **print(even\_number)**

**integer\_list = get\_integer\_list\_from\_user()**

**display\_odd\_and\_even\_numbers\_separately(integer\_list)**

**Output:**

**Enter an integer: 5**

**Enter an integer: 34**

**Enter an integer:**

**Odd numbers:5**

**Even numbers:34**

1. **Write a python code to print the reverse of a number.**

**Program:**

**def revnum(num):**

 **sum=0**

 **while num != 0:**

 **rem = num % 10**

 **sum = rem + sum \* 10**

 **num = num // 10**

 **print("Reverse num: ", sum)**

**num=int(input("Enter any number: "))**

**revnum(num)**

**Output:**

**Enter any number: 123**

**Reverse num: 321**

1. **Design a class that store the information of student and display the same.**

**Program:**

**class student:**

 **def info(self, studname, studaddr):**

 **print("Name:", studname, " Address: ", studaddr)**

**obj = student()**

**obj.info("Nikita", "Malad")**

**Output:**

**Name: Nikita Address: Malad**

1. **Design an employee class using python for reading and displaying the employee information, the getInfo() and displayInfo() methods will be used respectively (use Constructor).**

**Program:**

**class Employee:**

 **def \_\_init\_\_(self, name, age, salary):**

 **self.name = name**

 **self.age = age**

 **self.salary = salary**

 **def getInfo(self):**

 **return f"{self.name} is {self.age} years old and earns ${self.salary} per year."**

 **def displayInfo(self):**

 **print(self.getInfo())**

**employee = Employee("Archana Dhawade", 31, 50000)**

**employee\_info = employee.getInfo()**

**employee.displayInfo()**

**Output:**

**Archana Dhawade is 31 years old and earns $50000 per year.**

1. **Write a python program that defines a class employee. Define two subclass engineers and manager. Every class should have a method “print designation” that print designation of each employee.**

**Program:**

**class Employee:**

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

 **self.name = name**

 **def print\_designation(self):**

 **print(f"Employee: {self.name}")**

**class Engineer(Employee):**

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

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

 **self.department = department**

 **def print\_designation(self):**

 **super().print\_designation()**

 **print(f"Designation: Engineer ({self.department})")**

**class Manager(Employee):**

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

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

 **self.team = team**

 **def print\_designation(self):**

 **super().print\_designation()**

 **print(f"Designation: Manager ({self.team})")**

**engineer = Engineer("Archana Dhawade", "Software")**

**manager = Manager("Tejaswini Parab", "Sales")**

**engineer.print\_designation()**

**manager.print\_designation()**

**Output:**

**Employee: Archana Dhawade**

**Designation: Engineer (Software)**

**Employee: Tejaswini Parab**

**Designation: Manager (Sales)**

1. **Write a python code to find the factorial of a number using recursive function.**

**Program:**

**def fact(x):**

 **if x == 0 or x == 1:**

 **return 1**

 **else:**

 **return (x \* fact(x-1))**

**x=int(input("Enter a number: "))**

**print("The factorial of ",x,": ", fact(x))**

**Output:**

**Enter a number: 5**

**The factorial of 5 : 120**

1. **Implement the concept of single inheritance using python.**

**Program:**

**class st:**

 **def s1(self):**

 **print("Base Class")**

**class st1(st):**

 **def s2(self):**

 **print("Derived Class")**

**obj = st1()**

**obj.s1()**

**obj.s2()**

**Output:**

**Base Class**

**Derived Class**

1. **Design a simple Python GUI calculator.**

**Program:**

**import tkinter as tk**

**class Calculator:**

 **def \_\_init\_\_(self):**

 **self.root = tk.Tk()**

 **self.root.title("Calculator")**

 **self.display = tk.Entry(self.root)**

 **self.display.pack(fill=tk.X)**

 **self.buttons = {}**

 **for digit in range(10):**

 **self.buttons[digit] = tk.Button(self.root, text=str(digit), command=lambda digit=digit: self.on\_digit\_click(digit))**

 **self.buttons[digit].pack()**

 **self.buttons["+"] = tk.Button(self.root, text="+", command=self.on\_plus\_click)**

 **self.buttons["-"] = tk.Button(self.root, text="-", command=self.on\_minus\_click)**

 **self.buttons["\*"] = tk.Button(self.root, text="\*", command=self.on\_multiply\_click)**

 **self.buttons["/"] = tk.Button(self.root, text="/", command=self.on\_divide\_click)**

 **self.buttons["="] = tk.Button(self.root, text="=", command=self.on\_equals\_click)**

 **self.buttons["+"].pack()**

 **self.buttons["-"].pack()**

 **self.buttons["\*"].pack()**

 **self.buttons["/"].pack()**

 **self.buttons["="].pack()**

 **self.root.mainloop()**

 **def on\_digit\_click(self, digit):**

 **self.display.insert("end", str(digit))**

 **def on\_plus\_click(self):**

 **self.operand1 = float(self.display.get())**

 **self.operation = "+"**

 **self.display.delete(0, tk.END)**

 **def on\_minus\_click(self):**

 **self.operand1 = float(self.display.get())**

 **self.operation = "-"**

 **self.display.delete(0, tk.END)**

 **def on\_multiply\_click(self):**

 **self.operand1 = float(self.display.get())**

 **self.operation = "\*"**

 **self.display.delete(0, tk.END)**

 **def on\_divide\_click(self):**

 **self.operand1 = float(self.display.get())**

 **self.operation = "/"**

 **self.display.delete(0, tk.END)**

 **def on\_equals\_click(self):**

 **self.operand2 = float(self.display.get())**

 **self.result = eval(f"{self.operand1} {self.operation} {self.operand2}")**

 **self.display.delete(0, tk.END)**

 **self.display.insert("end", str(self.result))**

**calculator = Calculator()**

1. **Write a python program to accept an Integer list from user and print all the prime numbers in that list.**

**Program:**

**def get\_integer\_list\_from\_user():**

 **integer\_list = []**

 **while True:**

 **integer = input("Enter an integer: ")**

 **if integer == "":**

 **break**

 **integer\_list.append(int(integer))**

 **return integer\_list**

**def is\_prime(number):**

 **for divisor in range(2, int(number\*\*0.5) + 1):**

 **if number % divisor == 0:**

 **return False**

 **return True**

**def print\_all\_prime\_numbers\_in\_list(integer\_list):**

 **for integer in integer\_list:**

 **if is\_prime(integer):**

 **print(integer)**

**integer\_list = get\_integer\_list\_from\_user()**

**print\_all\_prime\_numbers\_in\_list(integer\_list)**

**Output:**

1. **Write a python code to check whether the entered number is prime or not.**

**Program:**

**num=int(input("enter any number:"))**

**if num>1:**

 **for i in range(2, int(num/2)+1):**

 **if (num % i) == 0:**

 **print(num, "is not a prime number")**

 **else:**

 **print(num, "is a prime number")**

 **break**

**else:**

 **print(num,"is not a prime number")**

**Output:**

**Enter any number:23**

**23 is a prime number**

1. **Consider the Tuple t=(10,5,12,11,33,100,17,7,13). Write a python code to display all the prime numbers from the tuple t.**

**Program:**

**def is\_prime(num):**

 **if num <= 1:**

 **return False**

 **if num <= 3:**

 **return True**

 **if num % 2 == 0 or num % 3 == 0:**

 **return False**

 **i = 5**

 **while i \* i <= num:**

 **if num % i == 0 or num % (i + 2) == 0:**

 **return False**

 **i += 6**

 **return True**

**t = (10, 5, 12, 11, 33, 100, 17, 7, 13)**

**prime\_numbers = [num for num in t if is\_prime(num)]**

**print("Prime numbers in the tuple:", prime\_numbers)**

**Output:**

**Prime numbers in the tuple: [5, 11, 17, 7, 13]**

1. **Implement the concept of inheritance using python.**

**Program:**

**Output:**

1. **Write a python script to concatenate following dictionaries to create a new one. Sample Dictionary: dic1={1:10, 2:20,} dic2= {3:30, 4:40} dic3={ 5:50, 6:60}.**

**Program:**

**dic1 = {1:10, 2:20}**

**dic2={3:30, 4:40}**

**dic3={5:50, 6:60}**

**dic1.update (dic 2)**

**dic1.update (dic 3)**

**print (dic1)**

**Output:**

**{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}**

1. **Implement the concept of multilevel inheritance using python.**

**Program:**

**class base\_class:**

 **def s1(self):**

 **print("Base class")**

**class st1(base\_class):**

 **def s2(self):**

 **print("Derived Class 1")**

**class st2(st1):**

 **def s3(self):**

 **print("Derived Class 2")**

**obj=st2()**

**obj.s1()**

**obj.s2()**

**obj.s3()**

**Output:**

**Base class**

**Derived Class 1**

**Derived Class 2**

1. **Define a function that computes the length of a given list or string.**

**Program:**

**def calen(n):**

 **count=0**

 **for i in n:**

 **count=count+1**

 **return count**

**print(calen([1,2,3,4])**

**print(calen(“student”)**

**Output:**

**7**

1. **Create module “Area.py” with the functions Area-Circle(). Area\_Triangle() and Area\_Rect(). Create new file use the Area\_circle(), Area\_Triangle() and Area\_Rect(), from the Area module to calculate the areas.**

**Program:**

**Area.py**

**Output:**

1. **Write a function that takes a character (i.e. a string of length 1) and returns True if it is a vowel, False otherwise.**

**Program:**

**def vchk(ch):**

 **if (ch =='a' or ch=='A' or ch=="e" or ch== 'E' or ch == ‘ i’ or ch == 'I' or ch== 'o' or ch ==**

 **'0' or ch == 'U' or ch==’u’):**

 **print (ch," is a vowel")**

 **else:**

 **print (ch, is a consonant)**

**ch= input ("Enter a single character (a-z/A-Z) only: ")**

**vchk (ch)**

**Output:**

**Enter a single character (a-z/A-Z) only:a**

**a is a vowel**

1. **Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers). 1) Write a static method called subtract, which takes two number parameters, b and c, and returns b - c.
2) Write a method called value which returns a tuple containing the values of x and y. Make this method into a property, and write a setter and a deleter for manipulating the values of x and y.**

**Program:**

**Output:**

1. **Design a database application using Python GUI that receives the following information from an Employee (EMPNO , EMPName , DeptName ,DateofBirth) The application should also display the information of all Employees once the user clicks on submit.**

**Program:**

**Output:**

1. **Design a database application using Python GUI to search the specified record of an employee using Emp\_ID from the database and display the same. Employee table(Emp\_id, emp\_name, emp\_age, dept\_name).**

**Program:**

**Output:**

1. **Design a database application using Python GUI that allows the user to add, delete and modify the Patient records Patient table ( Patient\_id , Patient\_Name , age ,address)**

**Program:**

**Output:**

1. **Design a database application using Python GUI to modify a specified record of a Customer using Cust\_id from the database and display the modified record. Customer Table(Cust\_id, Cust\_name , address, account Type)**

**Program:**

**Output:**

1. **Design a database application using Python GUI that allows the user to add, delete and modify the Bank Customer records. Customer Table (Custid,Custname,Age,Address)**

**Program:**

**Output:**

1. **Design a simple database application using Python GUI that stores the details of a bank Customer (Cust\_id, Cust\_name , address, account Type) and retrieve the same.**

**Program:**

**Output:**

1. **Design a database application using Python GUI that allows the user to add, delete and modify the employee records. Employee Table (Empid,Ename,Age,Address).**

**Program:**

**Output:**

1. **Design a database application using Python GUI to modify a specified record of a Student using Stud\_ID from the database and display the modified record. Student table(Stud\_id, Stud\_name, Stud\_address, Course\_name).**

**Program:**

**Output:**

1. **Design a database application using Python GUI to search the specified record of a Product using Pro\_ID from the database and display the same. Product table (Pro\_id, Pro\_name, Quantity).**

**Program:**

**Output:**

1. **Design a database application using Python GUI that allows the user to add, delete and modify the user login records. Login table(User\_name, User\_id, User\_Password)**

**Program:**

**Output:**

1. **Design a simple database application using Python GUI that stores the login details of user (User\_name,User\_id,User\_Password) and display the message “Record Inserted Successfully” after record insertion.**

**Program:**

**Output:**

1. **Design a simple database application using Python GUI that stores the records of a Patient and retrieve the same. Patient table ( Patient\_id , Patient\_Name , age ,address)**

**Program:**

**Output:**

1. **Design a database application using Python GUI to search the specified record of a Student using Stud\_ID from the database and display the same. Student table(Stud\_id, Stud\_name, Stud\_address, Course\_name).**

**Program:**

**Output:**

1. **Design a simple database application using Python GUI that stores the details of a Product (Pro\_id, Pro\_name, Quantity) and retrieve the same.**

**Program:**

**Output:**

1. **Design a simple database application using Python GUI that deletes the login details of a user based on userid and display the message “Record Deleted Successfully” after record deletion. Login table(User\_name, User\_id, User\_Password)**

**Program:**

**Output:**