**Q.1)Write a Python program to find number is greater or smaller from list which is named mylist. Write manual code**

**Ans:**

mylist = [10, 5, 20, 15, 8]

smallest = mylist[0]

largest = mylist[0]

for num in mylist:

if num < smallest:

smallest = num

elif num > largest:

largest = num

print("The smallest number is:", smallest)

print("The largest number is:", largest)

**Q.2) Demonstrate various operations on String and tuple. String is " Hello friends how are you", Tuple name is MyTuple**

**Ans:**

# String Operations

my\_string = "Hello friends how are you"

# Length of the string

print("Length of the string:", len(my\_string))

# Convert to uppercase

print("Uppercase:", my\_string.upper())

# Convert to lowercase

print("Lowercase:", my\_string.lower())

# Count occurrences of a substring

substring = "o"

print("Occurrences of 'o':", my\_string.count(substring))

# Split the string into a list of words

words = my\_string.split()

print("Split into words:", words)

# Joining the words with a different separator

separator = "-"

joined\_string = separator.join(words)

print("Joined with separator:", joined\_string)

# Tuple Operations

MyTuple = (1, 2, 3, 4, 5)

# Length of the tuple

print("Length of the tuple:", len(MyTuple))

# Accessing elements by index

print("Element at index 2:", MyTuple[2])

# Slicing tuple

print("Slice from index 1 to 3:", MyTuple[1:4])

# Concatenating tuples

AnotherTuple = (6, 7, 8)

concatenated\_tuple = MyTuple + AnotherTuple

print("Concatenated tuple:", concatenated\_tuple)

# Repetition of tuple

repeated\_tuple = MyTuple \* 2

print("Repeated tuple:", repeated\_tuple)

**Q.3) Create a File newfile.txt. Write four to six lines in your file and perform following, number of line in file, number of words in line, number of vowels in file**

**Ans:**

with open("newfile.txt", "w") as file:

file.write("This is the first line.\n")

file.write("Second line here.\n")

file.write("Another line.\n")

file.write("And one more line.\n")

# Count the number of lines in the file

with open("newfile.txt", "r") as file:

lines = file.readlines()

num\_lines = len(lines)

print("Number of lines in file:", num\_lines)

# Count the number of words in each line

with open("newfile.txt", "r") as file:

words\_per\_line = [len(line.split()) for line in file]

print("Number of words in each line:", words\_per\_line)

# Count the number of vowels in the file

with open("newfile.txt", "r") as file:

text = file.read().lower()

num\_vowels = sum(text.count(vowel) for vowel in "aeiou")

print("Number of vowels in file:", num\_vowels)

**Q.4) Python program to check the student’s Grades in Examination based on following information. If Percentage is >80 “Outstanding”, if Percentage>=70 and <80 “Distinction”, if percentage>=60 and <70 “First Class”, percentage >=50 and <60 “Second Class”, percentage >=40 and <50 “Pass Class”, percentage <40 “Fail**

Ans: # Taking input of total marks and obtained marks

total\_marks = int(input("Enter total marks: "))

obtained\_marks = int(input("Enter obtained marks: "))

# Calculating percentage

percentage = (obtained\_marks / total\_marks) \* 100

# Checking grade based on percentage

if percentage > 80:

grade = "Outstanding"

elif percentage >= 70:

grade = "Distinction"

elif percentage >= 60:

grade = "First Class"

elif percentage >= 50:

grade = "Second Class"

elif percentage >= 40:

grade = "Pass Class"

else:

grade = "Fail"

# Displaying the grade

print("Grade:", grade)

**Q.5) Write Python code to use array package and perform, create one-D array, two-D array, perform addition, multiplication and product of array**

**Ans:**

import array as arr

import numpy as np

# Create one-dimensional array

one\_d\_array = arr.array('i', [1, 2, 3, 4, 5])

# Create two-dimensional array

two\_d\_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Addition of arrays

array\_sum = np.add(one\_d\_array, two\_d\_array.flatten()) # Flatten two-dimensional array and add to one-dimensional array

# Multiplication of arrays

array\_product = np.multiply(one\_d\_array, two\_d\_array.flatten()) # Flatten two-dimensional array and multiply with one-dimensional array

# Product of arrays

array\_dot\_product = np.dot(one\_d\_array, two\_d\_array.flatten()) # Dot product of arrays

# Displaying the results

print("One-dimensional array:", one\_d\_array)

print("Two-dimensional array:")

print(two\_d\_array)

print("Addition of arrays:", array\_sum)

print("Multiplication of arrays:", array\_product)

print("Dot product of arrays:", array\_dot\_product)

**Q.6) Write python program to create class name Student. Enter student names, class, rollno, Marks of two subjects and display total and average marks of student**

**Ans:**

class Student:

def \_\_init\_\_(self, name, class\_name, roll\_no, marks\_subject1, marks\_subject2):

self.name = name

self.class\_name = class\_name

self.roll\_no = roll\_no

self.marks\_subject1 = marks\_subject1

self.marks\_subject2 = marks\_subject2

def calculate\_total\_marks(self):

return self.marks\_subject1 + self.marks\_subject2

def calculate\_average\_marks(self):

return (self.marks\_subject1 + self.marks\_subject2) / 2

# Input student details

name = input("Enter student name: ")

class\_name = input("Enter class: ")

roll\_no = input("Enter roll number: ")

marks\_subject1 = float(input("Enter marks for subject 1: "))

marks\_subject2 = float(input("Enter marks for subject 2: "))

# Create a student object

student = Student(name, class\_name, roll\_no, marks\_subject1, marks\_subject2)

# Calculate total and average marks

total\_marks = student.calculate\_total\_marks()

average\_marks = student.calculate\_average\_marks()

# Display results

print("\nStudent Details:")

print("Name:", student.name)

print("Class:", student.class\_name)

print("Roll No:", student.roll\_no)

print("Marks for Subject 1:", student.marks\_subject1)

print("Marks for Subject 2:", student.marks\_subject2)

print("\nTotal Marks:", total\_marks)

print("Average Marks:", average\_marks)

**Q.7) Write a python program to calculate net salary of employee based on following data. Basic salary=16500, HRA=67%, DA=137%, TA=2%, NAAC=1500, Deduction- Professional Tax=200, PF=1800, Calculate gross Salary monthly and Yearly, Calculate net salary monthly and Yearly, Calculate income tax on employee salary yearly**

**Ans:**

# Employee data

basic\_salary = 16500

hra\_rate = 67

da\_rate = 137

ta\_rate = 2

naac = 1500

professional\_tax = 200

pf = 1800

# Calculate gross salary monthly

hra = (hra\_rate / 100) \* basic\_salary

da = (da\_rate / 100) \* basic\_salary

ta = (ta\_rate / 100) \* basic\_salary

gross\_salary\_monthly = basic\_salary + hra + da + ta + naac

# Calculate gross salary yearly

gross\_salary\_yearly = 12 \* gross\_salary\_monthly

# Calculate net salary monthly

deductions = professional\_tax + pf

net\_salary\_monthly = gross\_salary\_monthly - deductions

# Calculate net salary yearly

net\_salary\_yearly = 12 \* net\_salary\_monthly

# Calculate taxable income yearly

taxable\_income = gross\_salary\_yearly - (pf \* 12)

# Calculate income tax yearly

if taxable\_income <= 250000:

tax = 0

elif 250001 <= taxable\_income <= 500000:

tax = (taxable\_income - 250000) \* 0.05

elif 500001 <= taxable\_income <= 1000000:

tax = 12500 + (taxable\_income - 500000) \* 0.2

else:

tax = 112500 + (taxable\_income - 1000000) \* 0.3

# Display results

print("Monthly Gross Salary:", gross\_salary\_monthly)

print("Yearly Gross Salary:", gross\_salary\_yearly)

print("Monthly Net Salary:", net\_salary\_monthly)

print("Yearly Net Salary:", net\_salary\_yearly)

print("Income Tax (Yearly):", tax)

**Q.8) This Python program takes input from the user in the form of space-separated integers and creates a tuple from those values. It then finds the smallest and largest elements in the tuple and prints them. (Use manual code, not built-in functions)**

**Ans:**

# Take input from the user

input\_values = input("Enter space-separated integers: ")

# Split the input string into a list of integers

values\_list = [int(num) for num in input\_values.split()]

# Create a tuple from the list

input\_tuple = tuple(values\_list)

# Initialize variables to store the smallest and largest elements

smallest = input\_tuple[0]

largest = input\_tuple[0]

# Iterate through the tuple to find the smallest and largest elements

for num in input\_tuple:

if num < smallest:

smallest = num

elif num > largest:

largest = num

# Print the smallest and largest elements

print("Smallest element:", smallest)

print("Largest element:", largest)

**Q.9) Write Python code to to create user defined function sum\_of\_digit to perform sum of digits of user input (e.g. if user input is 5678= 26)**

**Ans:**

def sum\_of\_digits(number):

# Convert the number to a string to iterate over its digits

number\_str = str(number)

# Initialize a variable to store the sum of digits

sum\_digits = 0

# Iterate over each digit in the number

for digit in number\_str:

# Convert the digit back to an integer and add it to the sum

sum\_digits += int(digit)

# Return the sum of digits

return sum\_digits

# Take input from the user

user\_input = int(input("Enter a number: "))

# Call the function and display the result

print("Sum of digits:", sum\_of\_digits(user\_input))

**Q.10) Write Python program to create parameterized constructor to take the name of emp, emp id and employee basic salary. Take salary at run time and calculate salary for 12 months**

**Ans:**

class Employee:

def \_\_init\_\_(self, name, emp\_id, basic\_salary):

self.name = name

self.emp\_id = emp\_id

self.basic\_salary = basic\_salary

def calculate\_salary\_yearly(self, monthly\_salary):

return 12 \* monthly\_salary

# Take input for employee details

name = input("Enter employee name: ")

emp\_id = input("Enter employee ID: ")

basic\_salary = float(input("Enter employee basic salary: "))

# Take input for monthly salary

monthly\_salary = float(input("Enter employee monthly salary: "))

# Create Employee object

employee = Employee(name, emp\_id, basic\_salary)

# Calculate and display yearly salary

yearly\_salary = employee.calculate\_salary\_yearly(monthly\_salary)

print("Yearly Salary for employee", employee.name, "with ID", employee.emp\_id, "is:", yearly\_salary)

**Q.11) Write Python program to take user string “Hello friends how are you, All the best for your practical examination” Use regular expression and find all words starting from ‘H’ letter, display the position of that word. Find ‘Hello” word and replace it with ‘Namaste”**

**Ans:**

import re

# User input string

user\_string = "Hello friends how are you, All the best for your practical examination"

# Find all words starting from 'H' letter and display their positions

matches = re.finditer(r'\bH\w\*', user\_string)

print("Words starting from 'H' letter and their positions:")

for match in matches:

print("Word:", match.group(), "Position:", match.start())

# Find 'Hello' word and replace it with 'Namaste'

new\_string = re.sub(r'\bHello\b', 'Namaste', user\_string)

# Display the modified string

print("\nModified string:")

print(new\_string)

**Q.12) Write Python code to check given number is odd or even using user defined function**

**Ans:**

def check\_odd\_even(number):

if number % 2 == 0:

return "Even"

else:

return "Odd"

# Take input from the user

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

# Call the function to check if the number is odd or even

result = check\_odd\_even(num)

# Display the result

print("The number", num, "is", result)

**Q.13) This Python program takes input from the user for key value pair for your dictionary. It then find the smallest and largest elements in the dictionary and prints them. (Use manual code, not built-in functions)**

**Ans:**

# Take input from the user for number of key-value pairs

num\_pairs = int(input("Enter the number of key-value pairs: "))

# Initialize an empty dictionary

my\_dict = {}

# Take input for key-value pairs from the user

for i in range(num\_pairs):

key = input("Enter key: ")

value = int(input("Enter value: "))

my\_dict[key] = value

# Initialize variables to store the smallest and largest elements

smallest\_key = None

smallest\_value = None

largest\_key = None

largest\_value = None

# Iterate through the dictionary to find the smallest and largest elements

for key, value in my\_dict.items():

if smallest\_value is None or value < smallest\_value:

smallest\_key = key

smallest\_value = value

if largest\_value is None or value > largest\_value:

largest\_key = key

largest\_value = value

# Print the smallest and largest elements

print("Smallest element:", smallest\_key, "->", smallest\_value)

print("Largest element:", largest\_key, "->", largest\_value)

**Q.14) Write recursion function to calculate factorial of given number.**

**Ans:**

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

# Take input from the user

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

# Call the recursive function to calculate the factorial

result = factorial(num)

# Display the result

print("Factorial of", num, "is:", result)

**Q.15) Write Python code read a file content from Myfile.text. Display contents of file**

**Ans:**

# Open the file for reading

with open("Myfile.txt", "r") as file:

# Read the contents of the file

file\_contents = file.read()

# Display the contents of the file

print("Contents of Myfile.txt:")

print(file\_contents)

**Q.16) Write Python code to perform addition of two integers using Lambda function**

**Ans:**

# Define the lambda function for addition

addition = lambda x, y: x + y

# Take input from the user for two integers

num1 = int(input("Enter the first integer: "))

num2 = int(input("Enter the second integer: "))

# Perform addition using the lambda function

result = addition(num1, num2)

# Display the result

print("Result of addition:", result)

**Q.17) Write Python code to create user defined package named Mypackage. Create module named Factorial to calculate factorial of user input. Import module to display the factorial of given number**

**Ans:**

# Factorial.py

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

#Mypackage

from Mypackage import Factorial

# Take input from the user

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

# Calculate the factorial using the factorial function from the Factorial module

result = Factorial.factorial(num)

# Display the result

print("Factorial of", num, "is:", result)

**Q.18) Write Python code perform following operations on list, insert new elements in list, display index of fourth element in your list, sort list elements in ascending and descending order, remove list element from list**

**Ans:**

# Define a list

my\_list = [10, 5, 20, 15, 8]

# Insert new elements in the list

my\_list.append(25) # Append an element to the end of the list

my\_list.insert(2, 12) # Insert an element at index 2

# Display index of fourth element in the list

print("Index of the fourth element:", my\_list[3])

# Sort list elements in ascending order

sorted\_list\_asc = sorted(my\_list)

# Sort list elements in descending order

sorted\_list\_desc = sorted(my\_list, reverse=True)

# Remove list element from list

element\_to\_remove = 20

if element\_to\_remove in my\_list:

my\_list.remove(element\_to\_remove)

# Display the list and sorted lists

print("Original list:", my\_list)

print("Sorted list in ascending order:", sorted\_list\_asc)

print("Sorted list in descending order:", sorted\_list\_desc)

**Q.19) Write Python code to create list of 5 subjects to store first element is name of subject, second element is internal marks out of 30, third element is End term marks out of 70. Create DataFrame named Result\_Analysis. Give labels to DataFrame as Subject, Internal Marks, End\_Term Marks. Apply basic operations on DataFrame**

**Ans:**

import pandas as pd

# Create a list of subjects with internal and end term marks

subjects\_data = [

["Mathematics", 25, 60],

["Science", 28, 65],

["English", 22, 58],

["History", 20, 55],

["Computer Science", 27, 62]

]

# Create DataFrame named Result\_Analysis

Result\_Analysis = pd.DataFrame(subjects\_data, columns=["Subject", "Internal Marks", "End Term Marks"])

# Display the DataFrame

print("Result Analysis DataFrame:")

print(Result\_Analysis)

# Basic operations on DataFrame

# Display the mean of each column

print("\nMean of each column:")

print(Result\_Analysis.mean())

# Display the maximum value of each column

print("\nMaximum value of each column:")

print(Result\_Analysis.max())

# Display the minimum value of each column

print("\nMinimum value of each column:")

print(Result\_Analysis.min())

# Display the summary statistics of the DataFrame

print("\nSummary statistics:")

print(Result\_Analysis.describe())

**Q.20) Write Python program to create Base class Employee, Derived classes HR Dept. and IT Dept. Take name Empid, and salary. Display the details of employee of both the departments.**

**Ans:**

class Employee:

def \_\_init\_\_(self, name, emp\_id, salary):

self.name = name

self.emp\_id = emp\_id

self.salary = salary

def display\_details(self):

print("Name:", self.name)

print("Emp ID:", self.emp\_id)

print("Salary:", self.salary)

class HRDept(Employee):

def \_\_init\_\_(self, name, emp\_id, salary):

super().\_\_init\_\_(name, emp\_id, salary)

self.department = "HR Department"

def display\_details(self):

super().display\_details()

print("Department:", self.department)

class ITDept(Employee):

def \_\_init\_\_(self, name, emp\_id, salary):

super().\_\_init\_\_(name, emp\_id, salary)

self.department = "IT Department"

def display\_details(self):

super().display\_details()

print("Department:", self.department)

# Input for HR Department employee

hr\_name = input("Enter HR employee name: ")

hr\_emp\_id = input("Enter HR employee ID: ")

hr\_salary = float(input("Enter HR employee salary: "))

# Input for IT Department employee

it\_name = input("Enter IT employee name: ")

it\_emp\_id = input("Enter IT employee ID: ")

it\_salary = float(input("Enter IT employee salary: "))

# Create HR Department employee object

hr\_employee = HRDept(hr\_name, hr\_emp\_id, hr\_salary)

# Create IT Department employee object

it\_employee = ITDept(it\_name, it\_emp\_id, it\_salary)

# Display details of HR Department employee

print("\nDetails of HR Department Employee:")

hr\_employee.display\_details()

# Display details of IT Department employee

print("\nDetails of IT Department Employee:")

it\_employee.display\_details()

**Q.21) Write Python code to validate email Id using regular expressions**

**Ans:**

import re

def validate\_email(email):

# Regular expression for validating email

pattern = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

# Use the match function to search for the pattern in the email

if re.match(pattern, email):

return True

else:

return False

# Take input from the user

email = input("Enter an email address: ")

# Call the function to validate the email

if validate\_email(email):

print("Valid email address")

else:

print("Invalid email address")

**Q.22) Write Python code to open a file name Python.txt. Add at least four lines in file and display all the contents on screen.**

**Ans:**

# Open the file in append mode to add content

with open("Python.txt", "a") as file:

# Add at least four lines to the file

file.write("Python is a versatile programming language.\n")

file.write("It is widely used in web development, data science, and artificial intelligence.\n")

file.write("Python has a simple and easy-to-learn syntax.\n")

file.write("It supports multiple programming paradigms including object-oriented, imperative, and functional programming.\n")

# Open the file in read mode to display contents

with open("Python.txt", "r") as file:

# Read and display all contents on the screen

contents = file.read()

print("Contents of Python.txt:")

print(contents)

**Q.23) Write Python code to demonstrate Built-in packages in Python**

# Importing built-in packages

import os # Provides operating system dependent functionality

import datetime # Provides classes for manipulating dates and times

import random # Provides functions for generating random numbers

import math # Provides mathematical functions

import json # Provides functions for encoding and decoding JSON data

import csv # Provides classes for reading and writing CSV files

import sys # Provides access to some variables used or maintained by the Python interpreter

import time # Provides functions for working with time

# Using the os package to get information about the current working directory

current\_directory = os.getcwd()

print("Current working directory:", current\_directory)

# Using the datetime package to get the current date and time

current\_date\_time = datetime.datetime.now()

print("Current date and time:", current\_date\_time)

# Using the random package to generate a random integer between 1 and 10

random\_number = random.randint(1, 10)

print("Random number:", random\_number)

# Using the math package to calculate the square root of a number

square\_root = math.sqrt(25)

print("Square root of 25:", square\_root)

# Using the json package to encode and decode JSON data

data = {"name": "John", "age": 30}

json\_data = json.dumps(data) # Encoding JSON data

print("Encoded JSON data:", json\_data)

decoded\_data = json.loads(json\_data) # Decoding JSON data

print("Decoded JSON data:", decoded\_data)

# Using the csv package to read data from a CSV file

with open("example.csv", "r") as file:

reader = csv.reader(file)

for row in reader:

print(row)

# Using the sys package to get command line arguments

print("Command line arguments:", sys.argv)

# Using the time package to get the current time in seconds since the epoch

current\_time = time.time()

print("Current time in seconds since the epoch:", current\_time)

**Q.24) Write Python code to find the given number is palindrome or not**

**Ans:**

def is\_palindrome(number):

# Convert the number to a string

num\_str = str(number)

# Reverse the string

reversed\_str = num\_str[::-1]

# Check if the original string is equal to the reversed string

if num\_str == reversed\_str:

return True

else:

return False

# Take input from the user

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

# Check if the number is a palindrome

if is\_palindrome(num):

print(num, "is a palindrome")

else:

print(num, "is not a palindrome")

**Q.25) Write Python code to overload arithmetic operators (any three).**

**Ans:**

class MyNumber:

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

self.value = value

# Overloading the addition operator

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

return self.value + other.value

# Overloading the subtraction operator

def \_\_sub\_\_(self, other):

return self.value - other.value

# Overloading the multiplication operator

def \_\_mul\_\_(self, other):

return self.value \* other.value

# Create instances of MyNumber class

num1 = MyNumber(5)

num2 = MyNumber(3)

# Perform arithmetic operations using overloaded operators

print("Addition:", num1 + num2)

print("Subtraction:", num1 - num2)

print("Multiplication:", num1 \* num2)

**Q.26) Write Python code to perform basic set operations.**

**Ans:**

# Define two sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Display the sets

print("Set 1:", set1)

print("Set 2:", set2)

# Union operation

union\_set = set1.union(set2)

print("Union of Set 1 and Set 2:", union\_set)

# Intersection operation

intersection\_set = set1.intersection(set2)

print("Intersection of Set 1 and Set 2:", intersection\_set)

# Difference operation

difference\_set = set1.difference(set2)

print("Difference of Set 1 and Set 2 (Set 1 - Set 2):", difference\_set)

# Symmetric difference operation

symmetric\_difference\_set = set1.symmetric\_difference(set2)

print("Symmetric Difference of Set 1 and Set 2:", symmetric\_difference\_set)

# Check if one set is subset of another

is\_subset = set1.issubset(set2)

print("Is Set 1 a subset of Set 2?", is\_subset)

# Check if one set is superset of another

is\_superset = set1.issuperset(set2)

print("Is Set 1 a superset of Set 2?", is\_superset)

**Q.27) Write Python code to create file named file.txt. Change the name of file..txt with file1.txt. Remove file1.txt**

**Ans:**

import os

# Create file named file.txt

with open("file.txt", "w") as file:

file.write("This is a sample file.")

# Rename file.txt to file1.txt

os.rename("file.txt", "file1.txt")

print("File renamed to file1.txt")

# Remove file1.txt

os.remove("file1.txt")

print("File file1.txt removed")

**Q.28) Write Python program to create user defined function to print number in reverse order**

**Ans:**

def print\_reverse\_number(number):

# Convert the number to a string and reverse it

reversed\_number = str(number)[::-1]

print("Number in reverse order:", reversed\_number)

# Take input from the user

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

# Call the function to print the number in reverse order

print\_reverse\_number(num)

**Q.29) Write a Python program to demonstrate multilevel inheritance. The base class name is GrandParent Create two classes which will be derived from related class which demonstrate multilevel inheritance**

**Ans:**

class GrandParent:

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

self.name = name

def display(self):

print("GrandParent:", self.name)

class Parent(GrandParent):

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

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

self.age = age

def display(self):

super().display()

print("Parent's age:", self.age)

class Child(Parent):

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

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

self.school = school

def display(self):

super().display()

print("Child's school:", self.school)

# Create instances of Child class

child = Child("John", 10, "ABC School")

# Display details using Child's display method

child.display()

**Q.30) Write program to create file. Enter some contents in your file. Use readline(), readlines() function**

**Ans:**

# Create a file and enter contents

with open("my\_file.txt", "w") as file:

file.write("Line 1: This is the first line.\n")

file.write("Line 2: This is the second line.\n")

file.write("Line 3: This is the third line.\n")

# Open the file in read mode

with open("my\_file.txt", "r") as file:

# Read one line at a time using readline()

print("Reading one line at a time using readline():")

line = file.readline()

while line:

print(line.strip()) # Strip the newline character

line = file.readline()

# Reset the file pointer to the beginning

file.seek(0)

# Read all lines at once using readlines()

print("\nReading all lines at once using readlines():")

lines = file.readlines()

for line in lines:

print(line.strip()) # Strip the newline character