INDIRAGANDHINATIONALOPENUNIVERSITY

Regional Centre Delhi-3

Format for Assignment Submission

For Term End Exam June/December- JUNE **(Year)** 2025

(Please read the instructions given below carefully before submitting assignments)

1. **Name of the Student** : Ashu Chamrasiya
2. **Enrollment Number** : 2501321326
3. **Programme Code** : MCA-NEW
4. **Course Code** (Use this format course-wise separately) : MCS-201
5. **Study Centre Code** : LSC-38046
6. **Name of the Study Centre With complete address** : RAJDHANI COLLEGE RAJA GARDEN NEW DELHI
7. **Mobile Number** : 8448713694
8. **E-mail ID** : Intermezzobrilliance@gmail.com
9. **Details if this same assignment has been submitted anywhere else also** : No
10. **Above information is cross checked and it is correct: Yes/No** ✓

Date of Submission: 01-06-2025

Ashu

(Signature of the student)

A. Important Instructions:-

1. Please do not send any assignment at any email of the Regional Centre, it will not be considered.
2. Please avoid duplicacy. Do not re-submit the same assignment anywhere else or by any other means.
3. About the mode of submission of assignments, pl wait for instructions from IGNOU Hqtrs. As soon as we shall come to know, we will share it with all.
4. Please do not use plastic covers. Use plain A4 size pages for assignments for uniformity and better management with this cover page format on each assignment.
5. Please write your name and enrollment no. at the bottom of each page of your assignment.
6. Please retain a photocopy set of assignment submitted with you for record(may be asked to submit at later stage) and also keep the assignment submission receipt in safe custody.
7. If assignment awards are not updated in your Grade Card **within next 09 months**, please write to us at **rcdelhi3@ignou.ac.in** giving your complete details and attaching the proof of assignment submission.
8. Assignment Question Paper can be downloaded from: https://webservices.ignou.ac.in/assignments/

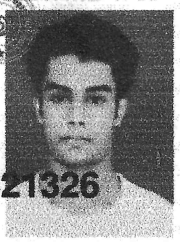
B. Compulsory sequence of the Assignment Set:

1. Duly Filled in Assignment Submission Cover Page (This Format Page).
2. Copy of IGNOU Identity Card.
3. Print out of valid/applicable assignment question paper.
4. Handwritten Assignment, written on both the sides of page (preferably plain A4 size).

ignou
THE PEOPLE'S
UNIVERSITY

इंदिरा गांधी राष्ट्रीय मुक्त विश्वविद्यालय
मैदान गढ़ी, नई दिल्ली - 110068
Indira Gandhi National Open University
Maidan Garhi,New Delhi - 110068

IGNOU - Student Identity Card



2501321326

Enrolment Number :	2501321326
RC Code :	38: DELHI 3 Naraina
Name of the Programme :	MCA_NEW : Master of Computer Applications
Name :	ASHU CHAURASIYA
Father's Name :	VIJAY
Address :	Flat No. 270,G, F Block-14, Pocket 13, Sector-20 RNORTH WEST
Pin Code :	110086

Instructions :
1. This card should be produced on demand at the Study Center, Examination Center or any other Establishment of IGNOU to use its facilities.
2. The facilities would be available only relating to the Programme/course for which the student is registered.
3. This ID Card is generated online. Students are advised to take a color print of this ID Card and get it laminated.
4. The student details can be cross checked with the QR Code at www.ignou.ac.in

Registrar
Student Registration Division

Course Code	:	**MCS-201**
Course Title	:	**Programming in C and PYTHON**
Assignment Number	:	**PGDCA_NEW(I)/201/Assign/2025**
Maximum Marks	:	**100**
Weightage	:	**30%**
Last Date of Submission	:	**30th April 2025 (for January Session)**

There are ten questions in this assignment which carries 80 marks. Each question carries 8 marks. Rest 20 marks are for viva-voce. Answer all the questions from both the sections i.e. Section A and Section B. You may use illustrations and diagrams to enhance the explanations. Include the screen layouts also along with your assignment responses. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

SECTION-A (C-Programming)

Q1: Write an algorithm, draw a flow chart and write its corresponding C program to convert a Binary decimal number to its equivalent Decimal number. **(8 Marks)**

Q2: Write an algorithm and use the concept of Structures to write the program in C, to generate Progress-Report of students of a class X of the school for all its 4 terms (the class is of 20 students). Assumptions can be made wherever necessary. **(8 Marks)**

Q3: Write a C program to generate the following pattern: **(8 Marks)**

```
*
* *
* * *
* * * *
* * * * *
```

Q4: Write a C program to perform the following operation on matrices D = A * (B + C), where A, B and C are matrices of (3 X 3) size and D is the resultant matrix. **(8 Marks)**

Q5: Use the concept of File Handling, to Write a program in C, to collect a list of N numbers in a file, and separate the even and odd numbers from the given list of N numbers, and put them in two separate files namely even_file and odd_file, respectively. **(8 Marks)**

SECTION-B (PYTHON-Programming)

Q6: Write Python code to perform the following: **(8 Marks)**

(i) Copy content of file first.txt to second.txt
(ii) Reading a file
(iii) Writing into a file
(iv) Appending into a file

Q7: Write an algorithm to find the slope of a line segment whose endpoint coordinates are

$(x1, y1)$ and $(x2, y2)$. The algorithm gives output whether the slope is positive, negative or zero. Transform your algorithm into Python program. **(8 Marks)**

Note: Slope of line segment = $(y2 - y1)/(x2-x1)$.

Q8: Write a programme in Python to create a package named Volume and create 3 module in it named – Cube, Cuboid and Sphere each having a function to calculate Volume of Cube, Cuboid and Sphere respectively. Import the module in separate location and use the functions. Assumptions can be made wherever necessary. Support your programme with suitable comments to improve readability. **(8 Marks)**

Q9: Write a program in Python to perform following: **(8 Marks)**

- To find square root of numbers in a list using lambda function.
- To display first n lines from a file, where n is given by user.
- To display size of a file in bytes
- To display frequency of each word in a file.

Q10: What are Co-routines? How Co-routines differ from threads? How Co-routines support cooperative multi-tasking in python? Compare Subroutines and Co-routines. **(8 Marks)**

Enrollment No. 2501321326

Question no. 1 : Write an algorithm, draw a flow chart and write its corresponding C program to convert a binary decimal number to its equivalent Decimal number.

Answer:

Algorithm to Convert Binary to Decimal

Start.

Initialize decimalNum = 0, base = 1.

Read binaryNum.

Loop while binaryNum > 0:

remainder = binaryNum % 10

decimalNum += remainder * base

binaryNum /= 10

base *= 2

Print decimalNum.

Stop

C Program to Convert Binary to Decimal

C

```
#include <stdio.h>
int main() {
    long long binaryNum;
    int decimalNum = 0, remainder, base = 1;
    printf("Enter a binary number: ");
    scanf("%lld", &binaryNum);
```

```
    while (binaryNum > 0) {
        remainder = binaryNum % 10;
        decimalNum += remainder * base;
        binaryNum /= 10;
        base *= 2;
    }
    printf("Equivalent decimal number: %d\n", decimalNum);
    return 0;
}
```

Question no. 2: Write an algorithm and use the concept of Structures to write the program in C, to generate Progress-Report of students of a class X of the school for all its 4 terms (the class is of 20 students). Assumptions can be made wherever necessary.

Answer:

Algorithm to Generate Student Progress Report

Start.

Define Student structure: studentID, name[50], marks[4][5], totalMarks[4], percentage[4].

Declare classX[20] of Student type.

For each student (0 to 19):

Read ID, Name.

For each term (0 to 3):

Initialize totalMarks[term] = 0.

For each subject (0 to 4):

Read mark.

Add mark to totalMarks[term].

Calculate percentage[term] = (float)totalMarks[term] / 500 * 100 .(Assuming 5 subjects, max 100 marks each, total 500).

For each student (0 to 19):

Print ID, Name.

For each term (0 to 3):

Print Term #, Total Marks, Percentage.

Stop.

C Program to Generate Progress Report using Structures

C

```
#include <stdio.h>
#include <string.h>
struct Student {
    int studentID;
      char name[50];
    int marks[4][5];
    int totalMarks[4];
    float percentage[4];
};
int main() {
    const int NUM_STUDENTS = 20;
```

```
const int NUM_TERMS = 4;
const int NUM_SUBJECTS = 5;
struct Student classX[NUM_STUDENTS];
printf("--- Enter Student Data and Marks ---\n");
for (int i = 0; i < NUM_STUDENTS; i++) {
    printf("\nEnter details for Student %d:\n", i + 1);
    printf("Student ID: ");
    scanf("%d", &classX[i].studentID);
    getchar();
    printf("Name: ");
    fgets(classX[i].name, sizeof(classX[i].name), stdin);
    classX[i].name[strcspn(classX[i].name, "\n")] = 0;
    for (int j = 0; j < NUM_TERMS; j++) {
        classX[i].totalMarks[j] = 0;
        printf("Enter marks for %s for Term %d (out of
100 per subject):\n", classX[i].name, j + 1);
        for (int k = 0; k < NUM_SUBJECTS; k++) {
            printf("  Subject %d: ", k + 1);
            scanf("%d", &classX[i].marks[j][k]);
            classX[i].totalMarks[j] += classX[i]
marks[j][k];
        }
        classX[i].percentage[j] = (float)classX[i].totalMarks[j]
```

```
/ (NUM_SUBJECTS * 100) * 100;
        }
    }
    printf("\n\n--- Class X Progress Report ---\n");
    for (int i = 0; i < NUM_STUDENTS; i++) {
        printf("\n----\n");
        printf("Student ID: %d\n", classX[i].studentID);
        printf("Name: %s\n", classX[i].name);
        for (int j = 0; j < NUM_TERMS; j++) {
            printf("Term %d:\n", j + 1);
            printf("  Total Marks: %d / %d\n", classX[i]
totalMarks[j], NUM_SUBJECTS * 100);
            printf("  Percentage: %.2f%%\n", classX[i]
percentage[j]);
            printf("\n");
        }
    }
    return 0;
}
```

There are many way to create such program, and a very general code has been given making assumptions.

Question no. 3: Write a C program to generate the following pattern:

*

* *

* * *

* * * *

* * * * *

Answer:

C Program to Generate Pattern

C

```
#include <stdio.h>
int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

Question no. 4: Write a C program to perform the following operation on matrices D = A * (B + C), where A, B and C are matrices of (3 X 3) size and D is the resultant matrix.

Answer:

C Program for Matrix Operations D = A * (B + C)

```
#include <stdio.h>

#define SIZE 3

void readMatrix(int matrix[SIZE][SIZE], char name) {
    printf("Enter elements for Matrix %c:\n", name);
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }
}

void printMatrix(int matrix[SIZE][SIZE], char name) {
    printf("\nMatrix %c:\n", name);
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%5d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void addMatrices(int mat1[SIZE][SIZE], int mat2[SIZE][SIZE], int
result[SIZE][SIZE]) {
```

```
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}
void multiplyMatrices(int mat1[SIZE][SIZE], int mat2[SIZE][SIZE], int
result[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            result[i][j] = 0;
            for (int k = 0; k < SIZE; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
int main() {
    int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE];
    int sumBC[SIZE][SIZE];
    int D[SIZE][SIZE];
    readMatrix(A, A);
    readMatrix(B, B);
```

```
    readMatrix(C, C);
    addMatrices(B, C, sumBC);
    printf("\n--- Intermediate Result (B + C) ---");
    printMatrix(sumBC, 5);
    multiplyMatrices(A, sumBC, D);
    printf("\n--- Final Result D = A * (B + C) ---");
    printMatrix(D, D);
    return 0;
}
```

Question no. 5: Use the concept of File Handling, to Write a program in C, to collect a list of N numbers in a file, and separate the even and odd numbers from the given list of N numbers, and put them in two separate files namely even_file and odd_file, respectively.

Answer:

Algorithm for Separating Even/Odd Numbers in Files

Start.

Declare file pointers: input_file, even_file, odd_file.

Declare variables: N (number of elements), num (current number).

Open input.txt in write mode ("w").

Prompt user for N numbers.

For i from 0 to N-1:

Read num.

Write num to input_file.

Close input_file.

Open input.txt in read mode ("r").

Open even_file.txt in write mode ("w").

Open odd_file.txt in write mode ("w").

Loop while fscanf successfully reads a number from input_file:

If num % 2 == 0: write num to even_file.

Else: write num to odd_file.

Close all files.

Stop.

C Program for Separating Even/Odd Numbers in Files

C

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main() {
    FILE *input_file, *even_file, *odd_file;
    int N, num;
    // Create input.txt and populate it with N numbers
    input_file = fopen("input.txt", "w");
    if (input_file == NULL) {
        perror("Error creating input.txt");
        return 1;
    }
    printf("Enter the number of integers (N): ");
```

```
scanf("%d", &N);
printf("Enter %d integers: \n", N);
for (int i = 0 ; i < N; i++) {
  scanf("%d", &num);
  fprintf(input_file, "%d\n", num);
}
fclose(input_file);
printf("Numbers written to input.txt\n");
// Open files for reading and writing even/odd numbers
input_file = fopen("input.txt", "r");
even_file = fopen("even_file.txt", "w");
odd_file = fopen("odd_file.txt", "w");
if (input_file == NULL || even_file == NULL || odd_file == NULL) {
    perror("Error opening files");
  return 1;
}
// Read from input_file and separate numbers
while (fscanf(input_file, "%d", &num) == 1) {
  if (num % 2 == 0) {
    fprintf(even_file, "%d\n", num);
  } else {
    fprintf(odd_file, "%d\n", num);
  }
```

```
    }
    printf("Numbers separated into even_file.txt and odd_file.txt\n");
    // Close all files
     fclose(input_file);
    fclose(even_file);
    fclose(odd_file);
      return 0;
}
```

Question no. 6: Write Python code to perform the following: (i) Copy content of file first.txt to second.txt (ii) Reading a file (iii) Writing into a file (iv) Appending into a file

Answer:

Python Code for File Operations

Python

```
# (i) Copy content of file first.txt to second.txt
try:
    with open("first.txt", "w") as f_first:
        f_first.write("This is the content of first.txt\nLine 2 from first.txt")
    with open("first.txt", "r") as f_in:
        content = f_in.read()
    with open("second.txt", "w") as f_out:
        f_out.write(content)
```

```
    print("Content copied from first.txt to second.txt")
except FileNotFoundError:
    print("Error: first.txt not found for copying.")
# (ii) Reading a file (e.g., second.txt)
try:
    with open("second.txt", "r") as f_read:
        print("\n--- Reading second.txt ---")
        print(f_read.read())
except FileNotFoundError:
    print("Error: second.txt not found for reading.")
# (iii) Writing into a file (overwrites existing content)
with open("output.txt", "w") as f_write:
    f_write.write("This is new content written to output.txt.\n")
    f_write.write("This line overwrites previous content.\n")
print("\nNew content written to output.txt")
# (iv) Appending into a file
with open("output.txt", "a") as f_append:
    f_append.write("This line is appended to output.txt.\n")
    f_append.write("Another appended line.\n")
print("Content appended to output.txt")
# Verify appended content by reading output.txt
try:
    with open("output.txt", "r") as f_verify:
```

```
        print("\n--- Verifying output.txt after append ---")
        print(f_verify.read())
except FileNotFoundError:
    print("Error: output.txt not found for verification.")
```

Question no. 7: Write an algorithm to find the slope of a line segment whose endpoint coordinates are $(x1, y1)$ and $(x2, y2)$. The algorithm gives output whether the slope is positive, negative or zero. Transform your algorithm into Python program.

Answer:

Algorithm to Find Slope and Its Sign

Start.

Input x1, y1, x2, y2.

Check for vertical line: If x2 - x1 == 0:

If y2 - y1 == 0: Print "Slope is undefined (points are identical)."

Else: Print "Slope is undefined (vertical line)."

Go to step 7.

Calculate slope: slope = (y2 - y1) / (x2 - x1).

Determine slope sign:

If slope > 0: Print "Slope is positive."

Else if slope < 0: Print "Slope is negative."

Else (slope == 0): Print "Slope is zero."

Print "Calculated Slope:", slope.

Stop.

Python Program for Slope Calculation

Python

```
def calculate_slope_and_sign(x1, y1, x2, y2):
    """
    Calculates the slope of a line segment and determines if it's
positive, negative, or zero.
    Handles vertical lines.
    """
    delta_x = x2 - x1
    delta_y = y2 - y1
    if delta_x == 0:
        if delta_y == 0:
            print("Slope is undefined (points are identical)
")
        else:
            print("Slope is undefined (vertical line)")
        return None  # Return None for undefined slope
    else:
        slope = delta_y / delta_x
        print(f"Calculated slope: {slope}")
        if slope > 0:
            print("Slope is positive")
        elif slope < 0:
```

```
        print("Slope is negative.")
    else:
        print("Slope is zero.")
    return slope

# Tryouts
print("--- Test Case 1: Positive Slope ---")
calculate_slope_and_sign(1, 2, 3, 4) # Slope = (4-2)/(3-1) = 2/2 = 1
print("\n--- Test Case 2: Negative Slope ---")
calculate_slope_and_sign(1, 4, 3, 2) # Slope = (2-4)/(3-1) = -2/2 = -1
print("\n--- Test Case 3: Zero Slope (Horizontal Line) ---")
calculate_slope_and_sign(1, 2, 5, 2) # Slope = (2-2)/(5-1) = 0/4 = 0
print("\n--- Test Case 4: Undefined Slope (Vertical Line) ---")
calculate_slope_and_sign(2, 1, 2, 5) # Slope = (5-1)/(2-2) = 4/0
(Undefined)
print("\n--- Test Case 5: Undefined Slope (Identical Points) ---")
calculate_slope_and_sign(2, 2, 2, 2) # Slope = (2-2)/(2-2) = 0/0
(Undefined)
```

Question no. 8: Write a programme in Python to create a package named Volume and create 3 module in it named — Cube, Cuboid and Sphere each having a function to calculate Volume of Cube, Cuboid and Sphere respectively. Import the module in separate location and use the functions. Assumptions can be made wherever necessary. Support your programme with suitable comments to improve readability.

Answer:

Python Program for Volume Package

Directory Structure:

my_project/

main_program.py

Volume/

__init__.py

Cube.py

Cuboid.py

Sphere.py

Content of Volume/__init__.py (Empty or can list modules explicitly):

Python

```
# This file makes Volume a Python package.
# It can be empty, or you can import specific functions/modules
# from its sub-modules to make them directly accessible via Volume
function_name
# For this example, we'll keep it simple and just make it a
package.
```

Content of Volume/Cube.py:

Python

```
# Volume/Cube.py
def calculate_cube_volume(side):
    """
```

```
    Calculates the volume of a cube.

    Args:
        side (float or int): The length of one side of the
cube.

    Returns:
        float: The volume of the cube.
    """
    return side ** 3

if __name__ == "__main__":
    # Example usage when running this module directly
    print(f"Volume of cube with side 5: {calculate_cube_volume(5)}")
```

Content of volume/cuboid.py:

```
Python
# volume/cuboid.py
def calculate_cuboid_volume(length, width, height):
    """
    Calculates the volume of a cuboid.

    Args:
        length (float or int): The length of the cuboid.
        width (float or int): The width of the cuboid.
        height (float or int): The height of the cuboid.

    Returns:
        float: The volume of the cuboid.
```

```
    """
    return length * width * height

if __name__ == "__main__":
    # Example usage when running this module directly
    print(f"Volume of cuboid 2x3x4: {calculate_cuboid_volume(2, 3, 4)}")
```

Content of Volume/Sphere.py:

```
Python
# Volume/Sphere.py
import math
def calculate_sphere_volume(radius):
    """
    Calculates the volume of a sphere.
    Args:
        radius (float or int): The radius of the sphere.
    Returns:
        float: The volume of the sphere.
    """
    return (4/3) * math.pi * (radius ** 3)

if __name__ == "__main__":
    # Example usage when running this module directly
    print(f"Volume of sphere with radius 3: {calculate_sphere_volume(3):.2f}")
```

Content of main_program.py (separate location, importing modules):

Python

```
# main_program.py
# This script demonstrates how to import and use functions from the
Volume' package.
# Option 1: Import specific functions from modules
from Volume.Cube import calculate_cube_volume
from Volume.Cuboid import calculate_cuboid_volume
from Volume.Sphere import calculate_sphere_volume
print("--- Calculating Volumes using imported functions ---")
# Use the imported functions
cube_vol = calculate_cube_volume(7)
print(f"Volume of Cube (side 7): {cube_vol}")
cuboid_vol = calculate_cuboid_volume(4, 5, 6)
print(f"Volume of Cuboid (4x5x6): {cuboid_vol}")
sphere_vol = calculate_sphere_volume(3.5)
print(f"Volume of Sphere (radius 3.5): {sphere_vol:.2f}")
# Option 2: Import the entire module (less common for specific functions)
# import Volume.Cube
# import Volume.Cuboid
# import Volume.Sphere
# print("\n--- Calculating Volumes using module objects (Alternative) ---")
# cube_vol_alt = Volume.Cube.calculate_cube_volume(2)
# print(f"Volume of Cube (side 2): {cube_vol_alt}")
```

cuboid_vol_alt = VolumeCuboid.calculate_cuboid_volume(), 2, 3)

print(f"Volume of Cuboid () x2x3): {cuboid_vol_alt}")

sphere_vol_alt = VolumeSphere.calculate_sphere_volume())

print(f"Volume of Sphere (radius)): {sphere_vol_alt: .2f}")

Question no. 9: Write a program in Python to perform following:

To find square root of numbers in a list using lambda function.

To display first n lines from a file, where n is given by user.

To display size of a file in bytes

To display frequency of each word in a file.

Answer:

Python Program for Various Operations

Python

```
import math
import os # For file size
# --- 1. Find square root of numbers in a list using lambda function
---
numbers = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# Use map with a lambda function to apply sqrt to each element
square_roots = list(map(lambda x: math.sqrt(x), numbers))
print(f"Original numbers: {numbers}")
print(f"Square roots (using lambda): {square_roots}")
# --- 2. Display first n lines from a file ---
# Create a dummy file for demonstration
```

```
with open("sample_lines.txt", "w") as f:
    f.write("Line 1: Hello world\n")
    f.write("Line 2: Python programming\n")
    f.write("Line 3: File handling example\n")
    f.write("Line 4: More lines follow\n")
    f.write("Line 5: End of sample file\n")
file_name_lines = "sample_lines.txt"
try:
    n_lines = int(input("\nEnter number of lines to display from
sample_lines.txt: "))
    with open(file_name_lines, "r") as f:
        for i, line in enumerate(f):
            if i < n_lines:
                print(line.strip()) # strip() removes
  newline characters
            else:
                break
except FileNotFoundError:
    print(f"Error: {file_name_lines} not found.")
except ValueError:
    print("Invalid input for number of lines. Please enter an integer
.")
# --- 3. Display size of a file in bytes ---
```

```
file_name_size = "sample_lines.txt" # Using the previously created file
try:
    file_size = os.path.getsize(file_name_size)
    print(f"\nSize of {file_name_size}: {file_size} bytes")
except FileNotFoundError:
    print(f"Error: {file_name_size} not found to check size.")
# --- 4. Display frequency of each word in a file ---
# Create another dummy file for word frequency
with open("word_freq_sample.txt", "w") as f:
    f.write("Python is a great language. Python programming is fun.\n")
    f.write("Learning Python is rewarding. Is Python easy?")
file_name_freq = "word_freq_sample.txt"
word_frequency = {}
try:
    with open(file_name_freq, "r") as f:
        text = f.read().lower() # Read content and convert to
lowercase
        # Remove punctuation and split into words
        words = text.replace('.', '').replace(',', '').replace('?',
'').split()
        for word in words:
            word_frequency[word] = word_frequency.get(word, 0)
+ 1
```

```
        print(f"\nWord frequency in {file_name_freq}: ")
        for word, count in word_frequency.items():
            print(f"{word}: {count}")
except FileNotFoundError:
    print(f"Error: {file_name_freq} not found for word frequency analysis
")
```

Question no. 10: What are Co-routines? How Co-routines differ from threads? How Co-routines support cooperative multi-tasking in python? Compare Subroutines and Co-routines.

Answer:

Co-routines, Threads, and Multi-tasking

Co-routines: Co-routines are functions that can be paused and resumed. They allow for cooperative multitasking, meaning the execution of a function can be suspended at certain points (using yield or await in Python) and later resumed from where it left off. They manage their own yielding of control, typically within a single thread.

Co-routines vs. Threads:

When contrasting coroutines and threads, the fundamental difference lies in their approach to concurrency and how they achieve it. Threads represent a more traditional, operating-system-level form of concurrency. Each thread is an independent execution unit managed by the OS scheduler, meaning the OS decides when to switch between threads (pre-emptive multitasking). This allows true parallelism on multi-core processors,

with overhead: creating and managing threads is relatively expensive in terms of memory and CPU cycles, and they introduce the complexity of race conditions and locks for shared data, as multiple threads can genuinely execute simultaneously. In contrast, coroutines offer a lightweight, user-space form of concurrency. They are functions whose execution can be paused and resumed at specific points, allowing for cooperative multitasking. The key here is "cooperative": coroutines explicitly yield control back to a scheduler (often an event loop) when they encounter a blocking operation (like I/O), allowing another coroutine to run. This means only one coroutine runs at a time within a single thread, eliminating the need for complex locks for shared data (unless explicitly using threading primitives within coroutines). Coroutines are significantly cheaper to create and switch between than threads, making them ideal for I/O-bound tasks where many operations spend most of their time waiting, but they don't provide true parallelism for CPU-bound tasks.

How Co-routines Support Cooperative Multi-tasking in Python: In Python, co-routines (implemented using async def and await, or older yield for generators) support cooperative multitasking by explicitly ceding control. When a co-routine encounters an await or yield expression, it pauses its execution and returns control to the event loop. The event loop can then pick another co-routine that is ready to run. This allows multiple "tasks" to make progress concurrently within a single thread, without the overhead of context switching between OS threads. This is particularly

I/O-bound operations, where tasks spend most of their time waiting.

Subroutines vs Co routines

When distinguishing between coroutines and subroutines, the core difference lies in their control flow and state management capabilities. A subroutine, which is what a typical function or method in most programming languages is, executes sequentially from its entry point to its exit point. It follows a LIFO (Last-In, First-Out) execution model, always returning control to its caller once its task is complete, and its local state is generally destroyed upon completion. In essence, it has a single entry point and a single exit point. Conversely, a coroutine is a more generalized form of a subroutine that allows for multiple entry and exit points. A coroutine can be paused at any point within its execution (using constructs like yield in Python), returning control to its caller, but crucially, it retains its state (local variables, execution context) when it yields. When resumed, it continues execution from precisely where it left off, rather than starting from the beginning. This "pause-and-resume" capability makes coroutines suitable for cooperative multitasking, allowing them to manage complex, non-blocking operations, whereas subroutines are inherently blocking and sequential