## **Computer architecture outcomes 3 & 4**

****

**Name : Erya Anom**

**Course : Programming LO3 and LO4**

Table of Contents

[**Computer architecture outcomes 3 & 4** 1](#_Toc198041877)

[1.Program Design of the program 3](#_Toc198041878)

[2. Implementation of the Program 4](#_Toc198041879)

[3. Documentation and Testing 5](#_Toc198041880)

[4. Chronological Development Log 9](#_Toc198041881)

# 1.Program Design of the program

Pseudocode. The design is to be presented in pseudocode, which is a valuable tool used for outlining the steps involved in a program or algorithm. It employs straightforward language that mirrors programming concepts but isn’t bound by the syntax rules of any specific programming language. The primary purposes of pseudocode include planning and designing a program prior to writing actual code, clearly communicating logic in team settings or assessments.

FUNCTION extract\_ip(line):

Split the line into parts

IF the list has at least one part:

RETURN the first part (assumed to be the IP)

ELSE:

RETURN None

FUNCTION analyse\_logs(filename):

Initialize empty list ip\_list

TRY:

Open the file in read mode

FOR each line in the file:

Call extract\_ip(line)

IF and IP is found:

Add it to ip\_list

EXCEPT FileNotFoundError:

Print error message about missing file

RETURN ip\_list

FUNCTION find\_most\_frequent(ip\_list):

Create an empty dictionary ip\_count

FOR each IP in ip\_list:

IF IP is already in ip\_count:

Increase its count by 1

ELSE:

Set its count to 1

Find the IP with the highest count

RETURN the most frequent IP and its count

FUNCTION main:

Ask the user for a filename

Call analyse\_logs(filename) to get all Ips

IF no Ips are found:

Print a message and exit

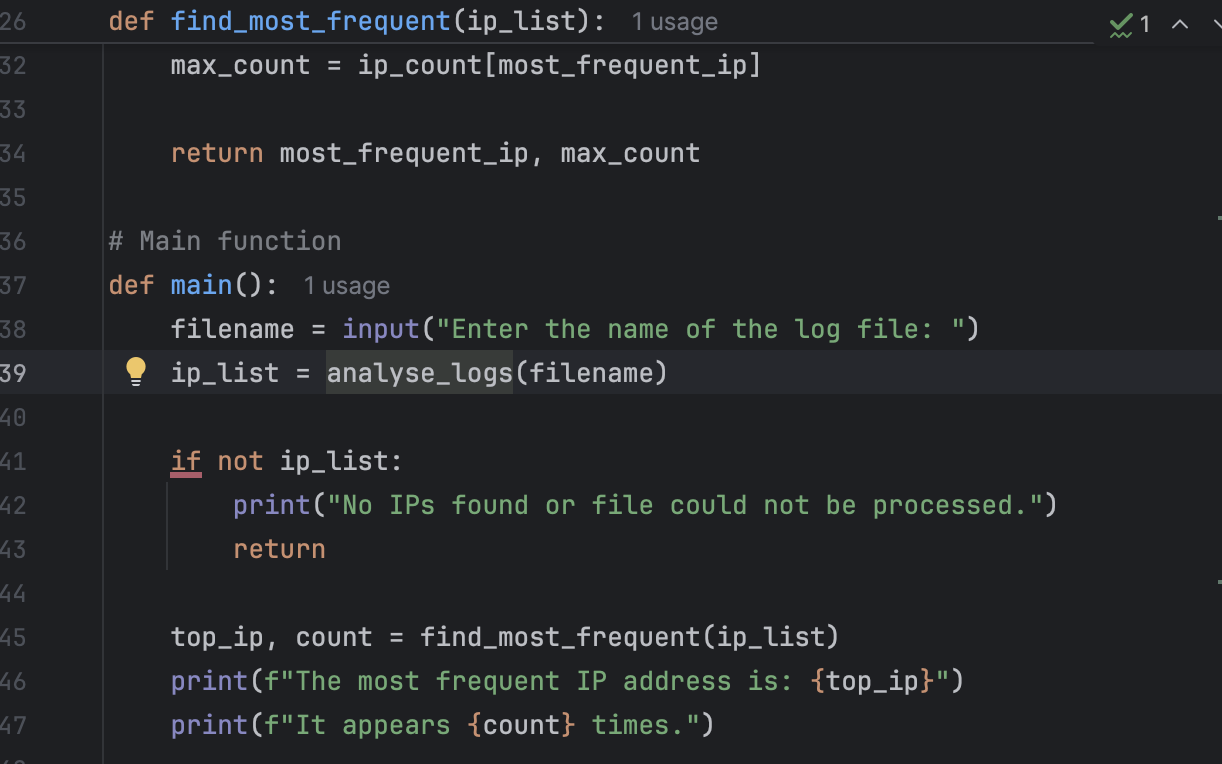
ELSE:

Call find\_most\_frequent(ip\_list)

Print the most frequent IP and how many times it appread

IF this script is being run directly:

Call the main function



To begin with, the **def** **main( )** code is the written of the function and can be translated into pseudocode. **def main ( )**  it is called defines the main function of the program, this part which runs the script has been executed.

**filename = input(“Enter the name of the log file: “)** this script to asks the user to type the name of the log file for example, **sample\_log\_1.txt**. The user’s inut is stored in variable **filename**

**ip\_list = analyse\_logs(filename)**, this line called the **analyse\_logs( )**. This function to give it the filename the user typed. That means, the program asks the user to type the name of the log file, and it is stored in the variable filename.

Furthermore, if not ip\_list, if the list is empty the file could not be read or no IP addresses were found and it will shows;

**print(“No Ips found or file could not be processed.”)**

# A screenshot of a computer program AI-generated content may be incorrect.A screenshot of a computer program AI-generated content may be incorrect.2. Implementation of the Program

This is the implementation of the program that works in Python code. This means write a Python program that matches the design that has been planned by pseudocode or flowchart, the script should successfully run and complete the task. In this case finding the most frequent IP address from the log file.

The script that already uses;

# Function to extract the IP address from a log line

# Main function

A data structure is the way of storing data so it can access or manipulate it. For common example is lists (arrays), dictionaries, sets, etc.

In this case it uses two data structure which are;

* **Ip\_list = [ ]** – this is a list which is equivalent to an array
* **Ip\_count = { }** – this is a dictionaries, which is stores data as a key value pairs IP address to count.

# 3. Documentation and Testing

A screenshot of a computer program

AI-generated content may be incorrect.**3.1. Sample\_log\_1.txt**

This screenshot displays a file name **Sample\_log\_1.txt** it indicates that the most frequently occurring IP address is 172.16.17.108, which is expected to appears 15 times in the log, and this is expectation was confirmed by the result.

A screenshot of a computer program

AI-generated content may be incorrect.**3.2. Sample\_log\_2.txt**

This screenshot displays a file name **Sample\_log\_2.txt** it indicates that the most frequently occurring IP address is 172.16.129.246, which is expected to appears 17 times in the log, and this is expectation was confirmed by the result.

A screenshot of a computer program

AI-generated content may be incorrect.**3.3. Sample\_log\_3.txt**

This screenshot displays a file name **Sample\_log\_3.txt, indicating** that the most frequently occurring IP address is 172.16.35.206, which is expected to appear 16 times in the log. This expectation was confirmed by the result.

A screenshot of a computer

AI-generated content may be incorrect.**3.4 Full\_log.txt**

This screenshot displays a file name **Full\_log.txt,** it indicates that the most frequently occurring IP address is 172.16.10.207, which is expected to appear 535 times in the log, and this expectation was confirmed by the result.

# 4. Chronological Development Log

I worked on a program to analyse log files for IP addresses over a week. On the first day, I created a basic outline. On the second day, I wrote the

**extract\_ip( )** function and tested it with a sample line.

On day three I developed the **analyse\_logs ( )** function to read log files and collected IP addresses. The next day, I added the **find\_most\_frequent ( )** function, which uses dictionaries to identify the most common IP addresses.

Day five focused on testing the program with sample files. I confirmed that it returned accurate results. On day six, I added error handling and cleaned up the code with comments.

In the last two days, I performed final tests. On day seven, I used a completed set of log data and recorded the most frequent IP addresses. And the final day, I prepared a report with screenshots of the program and submitted it.

In conclusion, The program successfully extracts and analyse IP addresses from the log files. And, it uses arrays and dictionaries when needed. All tests work correctly and provide the right results. the designed and code are well organised and include clear comments.