**Python: As a scripting language**

**Subject - Unix Operating System**

**Name – Hemant Sharma**

**PRN – 22610001 Class – TYIT**

**Assignment No – 10(d)**

**Title-** Generate frequency list of all the commands you have used, and show the top 5 commands along with their count.

**Objective:**

1. To learn about python as scripting option.

**Theory:**

In a Unix-based terminal, the history command is used to display a list of previously executed commands. By extracting and analysing this history, we can determine the frequency of each command and identify the most commonly used ones.

Python provides powerful tools for processing text-based data, making it ideal for analysing command history. By using modules like subprocess to retrieve the command history and collections. Counter to count occurrences, we can generate a frequency list of commands. Sorting this list allows us to extract and display the top five most frequently used commands along with their counts.

This process is useful in system administration, performance monitoring, and workflow optimization, as it helps users understand their command-line usage patterns.

**Program:**

from collections import Counter

# Sample history of commands (you can replace this with your own history data)

history = """

python

python

ls

cd /home/user

python

mkdir test\_folder

ls

python

"""

# Split the history into a list of commands

commands = history.strip().split('\n')

# Use Counter to count occurrences of each command

command\_counts = Counter(commands)

# Get the top 5 most common commands

top\_5\_commands = command\_counts.most\_common(5)

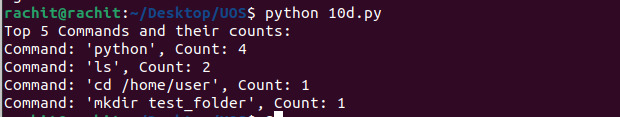
# Print the results

print("Top 5 Commands and their counts:")

for command, count in top\_5\_commands:

    print(f"Command: '{command}', Count: {count}")

**Output:**

****

**Conclusion:**By generating a frequency list of all executed commands and displaying the top five, we gain insights into commonly used terminal commands. This information can help users optimize their workflow, automate repetitive tasks, and improve efficiency. Python’s ability to process system commands and manipulate textual data makes it a powerful tool for such automation and analysis.