Institute of Computer Technology

B. Tech. Computer Science and Engineering

Semester: IV

Sub: Functional Programming

Course Code: 2CSE403

Practical Number: 4

Objective:

Q1. You're tasked with analyzing the frequency of words in a text file to identify the most commonly used words. Python programming involves opening the text file, reading its contents, and splitting the text into words. You would then use a dictionary to count the occurrences of each word. Additionally, you might filter out common stop words or punctuation to focus on meaningful terms. Finally, you would display the word frequencies or save them to another file for further analysis.

Code :

```
import string

from collections import Counter


STOP_WORDS = {"the", "is", "in", "and", "to", "of", "a", "for", "on", "with", "as", "by", "an", "it", "this",
"that"}


def word_frequency_analysis(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        text = file.read().lower()


    text = text.translate(str.maketrans('', '', string.punctuation))


    words = text.split()


    words = [word for word in words if word not in STOP_WORDS]
```

```
    word_counts = Counter(words)

    for word, count in word_counts.most_common(10):
        print(f"{word}: {count}")

word_frequency_analysis("sample.txt")
```

Output :-

```
python: 2
all: 2
extremely: 1
powerful: 1
programming: 1
language: 1
having: 1
applications: 1
web: 1
development: 1
```

Q2. You need to search through a large text file to find occurrences of specific patterns or phrases. Python programming involves opening the text file and reading its contents line by line. You would then use string manipulation functions or regular expressions to search for the desired patterns within each line. When a match is found, you can print the line or extract relevant information for further processing. This approach is useful for tasks such as searching for error messages in log files or extracting data from structured text documents.

Code :

```
import re

def extract_emails(filename):
    email_pattern = r"[a-zA-Z0-9._%+-]+@gnu\.ac\.in"
    with open(filename, "r", encoding="utf-8") as file:
        lines = file.readlines()

    emails = []
    for i, line in enumerate(lines, start=1):
```

```
        matches = re.findall(email_pattern, line)
        for email in matches:
            emails.append(f"E-mail {i} :: {email}")

    return emails

filename = "q2.txt"  # Replace with your actual filename
emails = extract_emails(filename)
for email in emails:
    print(email)
```

Output :-

```
E-mail 1 :: vishva1@gnu.ac.in
E-mail 2 :: vishva2@gnu.ac.in
E-mail 3 :: vishva3@gnu.ac.in
E-mail 4 :: vishva4@gnu.ac.in
E-mail 5 :: vishva5@gnu.ac.in
E-mail 6 :: vishva6@gnu.ac.in
E-mail 7 :: vishva7@gnu.ac.in
E-mail 8 :: vishva8@gnu.ac.in
E-mail 9 :: vishva9@gnu.ac.in
E-mail 10 :: vishva10@gnu.ac.in
```

Q3. You are working on a project to develop a personal diary application. The application should allow users to create, view, and update diary entries, with each entry stored in a separate text file on the user's device. As part of the development process, you need to implement functionality to handle file operations, including reading existing diary entries, creating new entries, and appending additional content to existing entries. How would you design and implement the file handling aspects of this diary application to ensure seamless interaction with diary entries stored as text files?

Code :

```
import os

DIARY_FOLDER = "diary_entries"

# Ensure diary folder exists
```

```
os.makedirs(DIARY_FOLDER, exist_ok=True)

def create_diary_entry():
    date = input("Enter date (YYYY-MM-DD): ")
    filename = os.path.join(DIARY_FOLDER, f"{date}.txt")

    content = input("Write your diary entry:\n")

    with open(filename, 'w', encoding='utf-8') as file:
        file.write(content)

    print(f"Diary entry saved as {filename}")

def view_diary_entry():
    date = input("Enter date to view (YYYY-MM-DD): ")
    filename = os.path.join(DIARY_FOLDER, f"{date}.txt")

    if os.path.exists(filename):
        with open(filename, 'r', encoding='utf-8') as file:
            print("\nDiary Entry:")
            print(file.read())
    else:
        print("No entry found for this date.")

def append_diary_entry():
    date = input("Enter date to append to (YYYY-MM-DD): ")
    filename = os.path.join(DIARY_FOLDER, f"{date}.txt")

    if os.path.exists(filename):
        content = input("Enter additional content:\n")
        with open(filename, 'a', encoding='utf-8') as file:
```

```
            file.write("\n" + content)
        print("Entry updated.")
    else:
        print("No entry found. Creating a new one.")
        create_diary_entry()

while True:
    choice = input("\nChoose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: ")
    if choice == "1":
        create_diary_entry()
    elif choice == "2":
        view_diary_entry()
    elif choice == "3":
        append_diary_entry()
    elif choice == "4":
        break
    else:
        print("Invalid choice. Please try again.")
```

Output :-

```
Choose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: 1
Enter date (YYYY-MM-DD): 2025-02-27
Write your diary entry:
hellow test
Diary entry saved as diary_entries\2025-02-27.txt

Choose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: 2
Enter date to view (YYYY-MM-DD): 2025-02-27

Diary Entry:
hellow test

Choose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: 3
Enter date to append to (YYYY-MM-DD): 2025-02-27
Enter additional content:
wtsup
Entry updated.

Choose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: 2
Enter date to view (YYYY-MM-DD): 2025-02-27

Diary Entry:
hellow test
wtsup

Choose an option: (1) Create Entry, (2) View Entry, (3) Append Entry, (4) Exit: 4
```